A számítástudomány alapjai

Gráfbejárások és legrövidebb utak

2024. szeptember 24.

Munkaterv

Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.

Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.

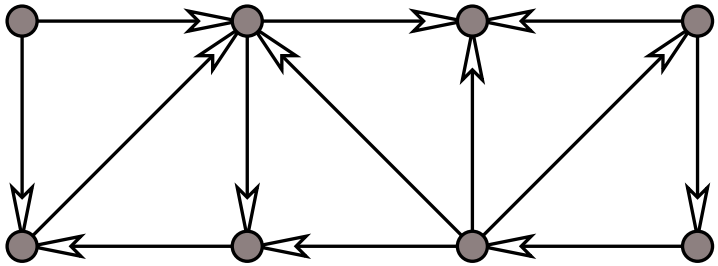
Munkaterv

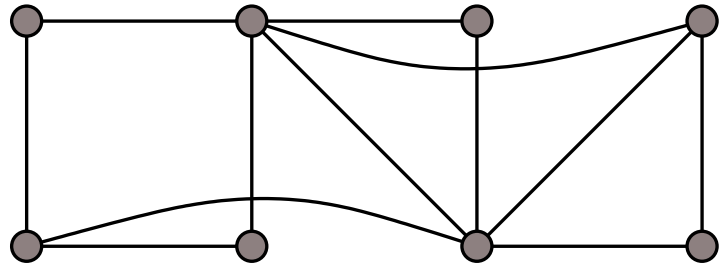
- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.
- Ennek során (többek között) egy újfajta módszert látunk a feszítőfa (ill. feszítő erdő) keresésére.

Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.
- Ennek során (többek között) egy újfajta módszert látunk a feszítőfa (ill. feszítő erdő) keresésére.
- Ezért a továbbiakban irányított gráfokkal foglalkozunk: irányítatlan gráf esetén arra az irányított gráfra gondolunk, amit az irányítatlanból az élek oda-vissza irányításával kapunk. Ezzel a módszerrel az irányított gráfra kapott eredmények az irányítatlan esetre is könnyen átültethetők.

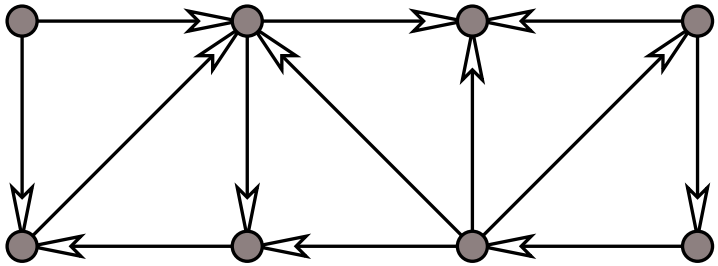
Általános gráfbejárás

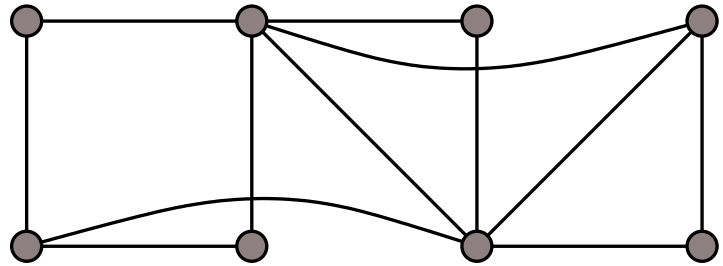




A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

Általános gráfbejárás

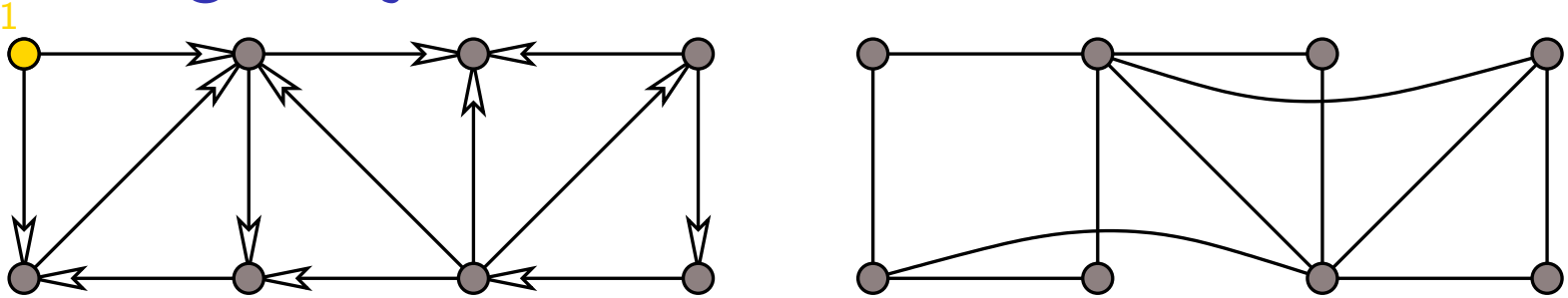




A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

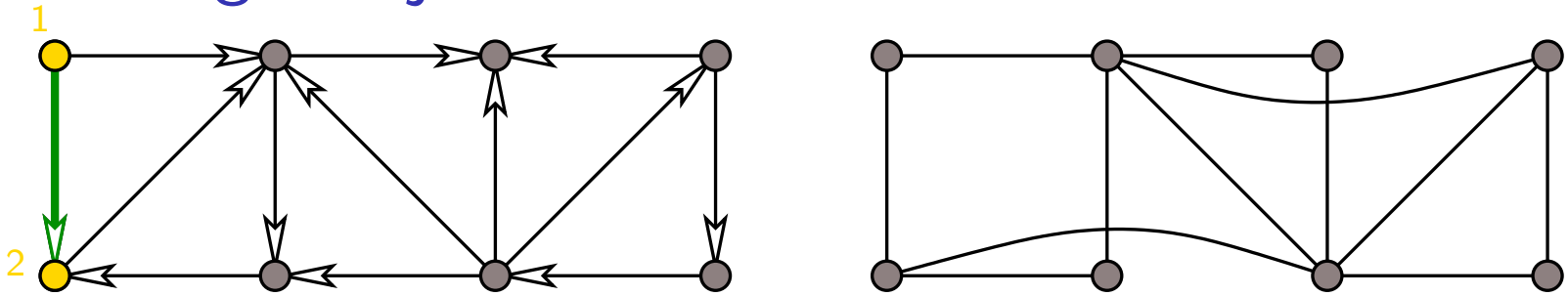
Általános gráfbejárás



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

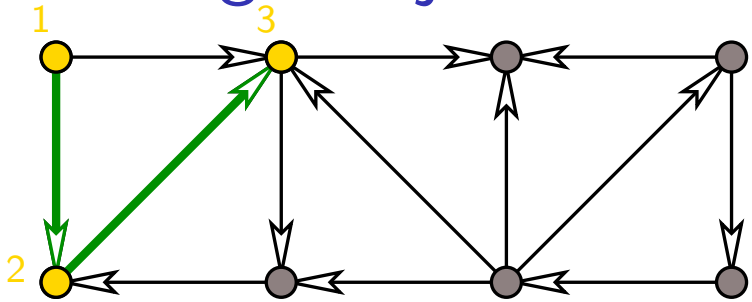
Általános gráfbejárás

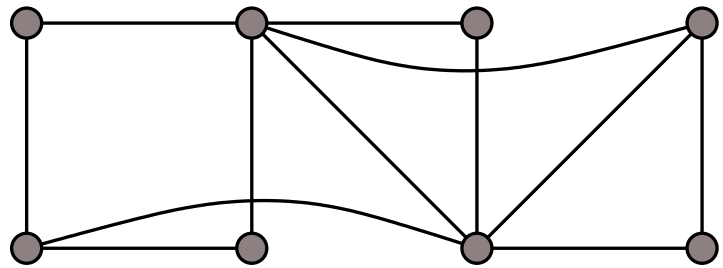


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

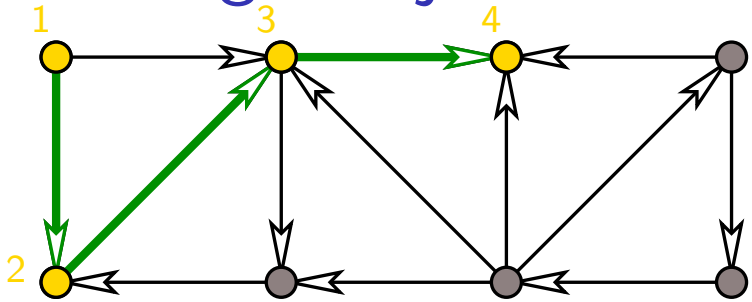


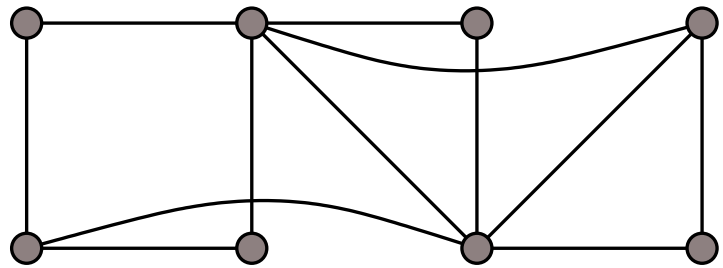


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

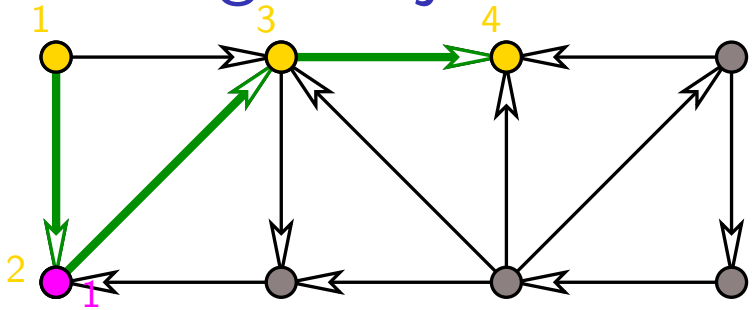


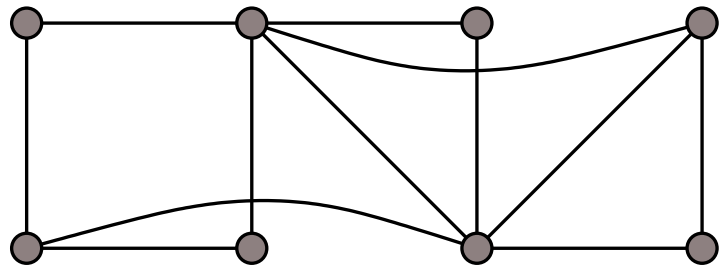


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

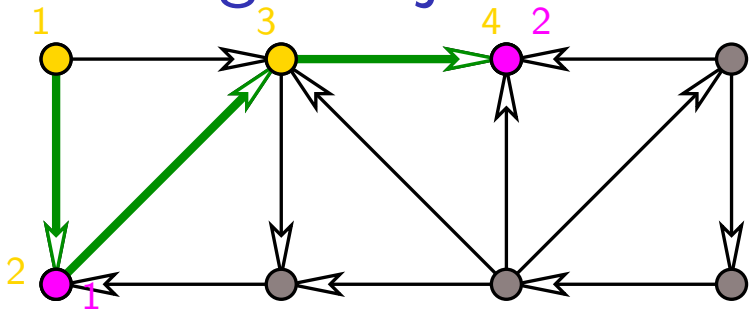


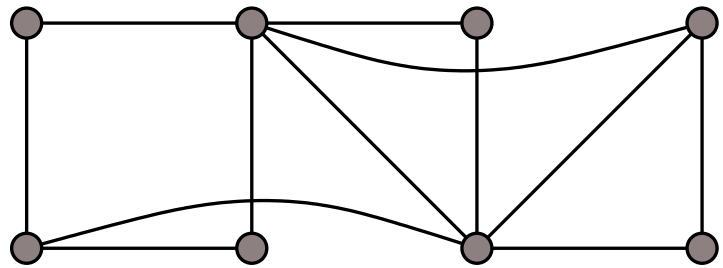


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

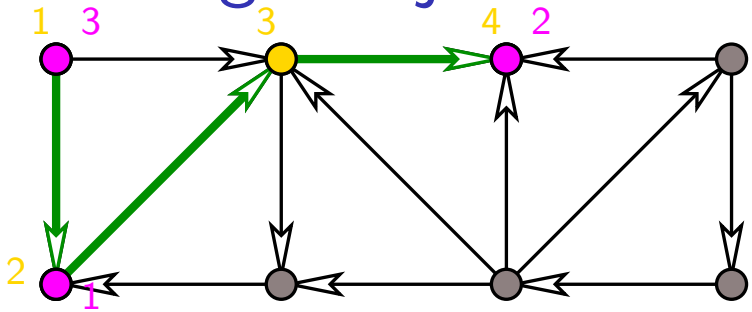


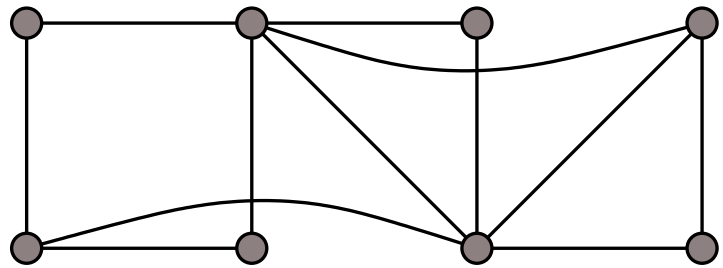


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

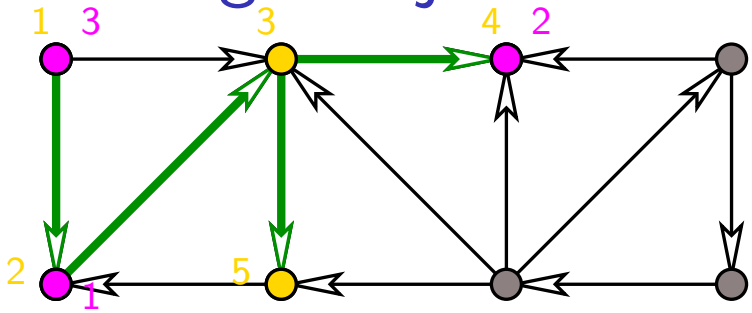


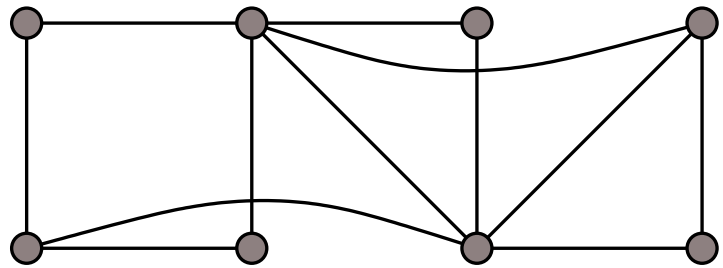


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

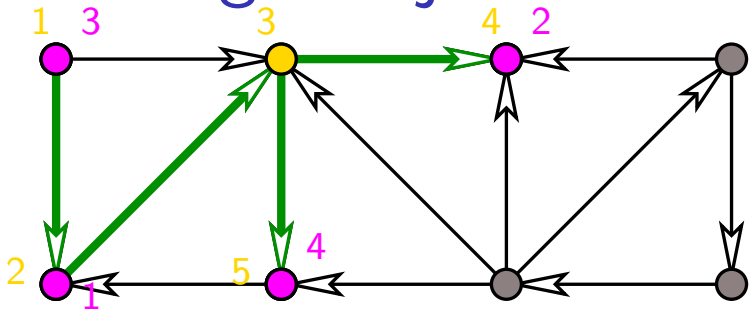


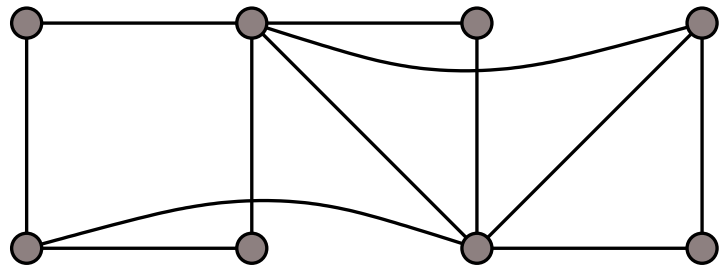


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

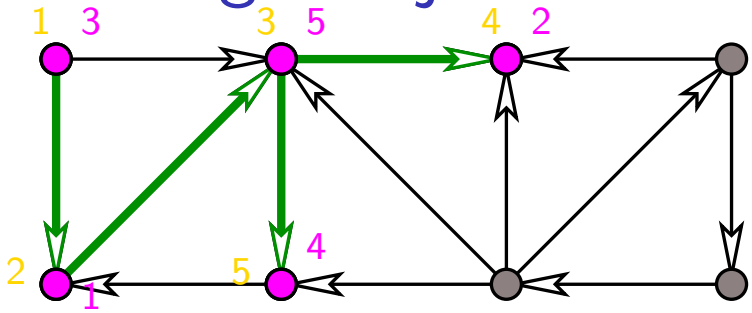


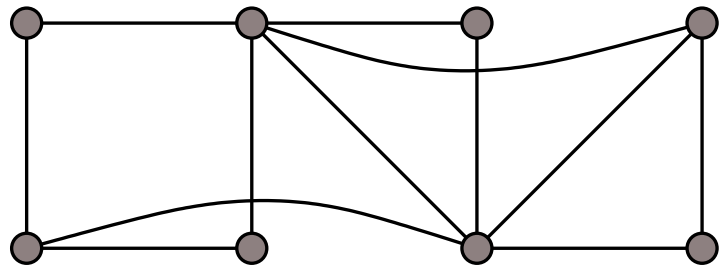


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

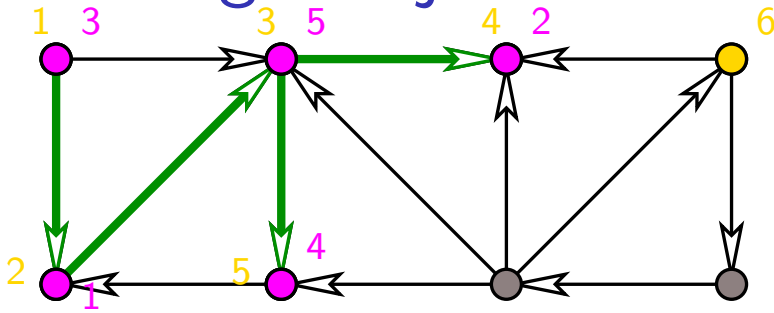


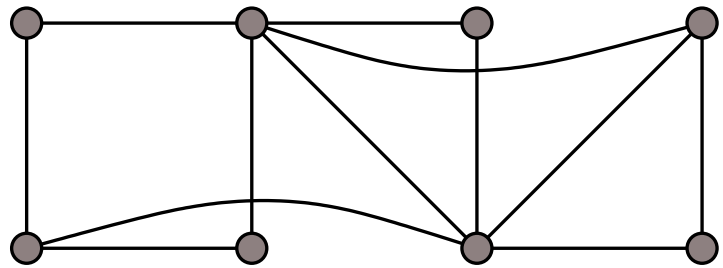


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

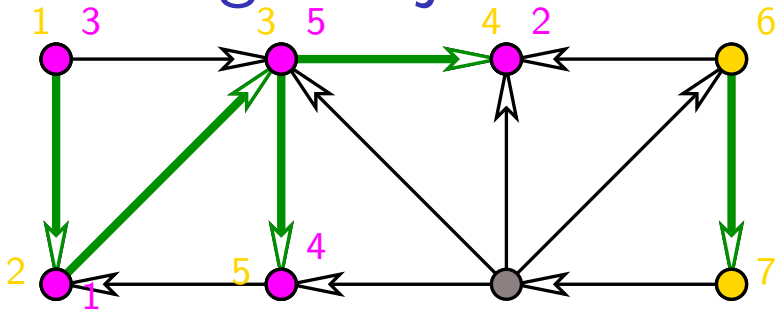


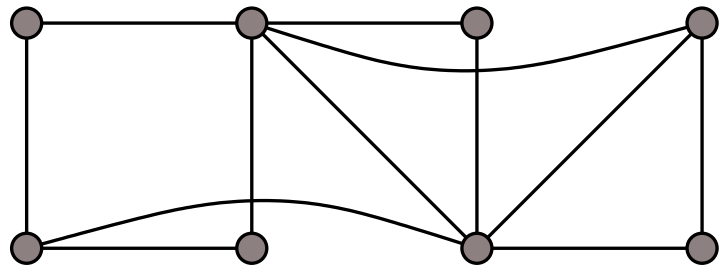


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

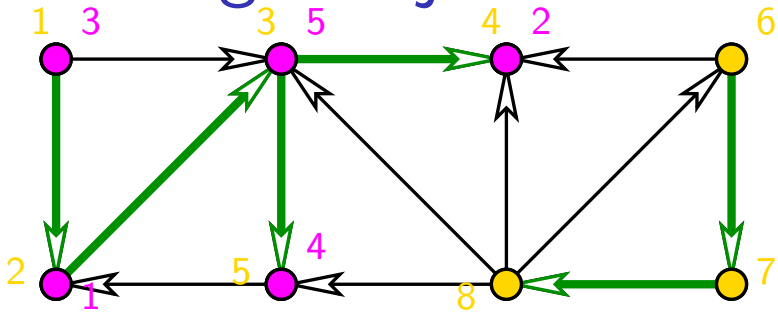


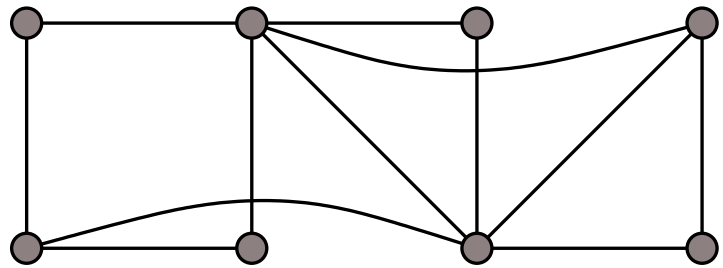


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

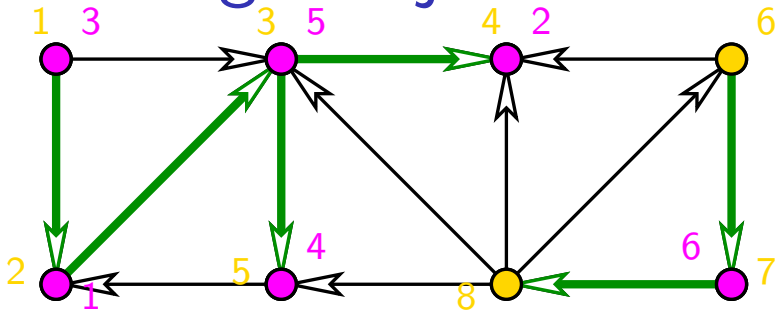


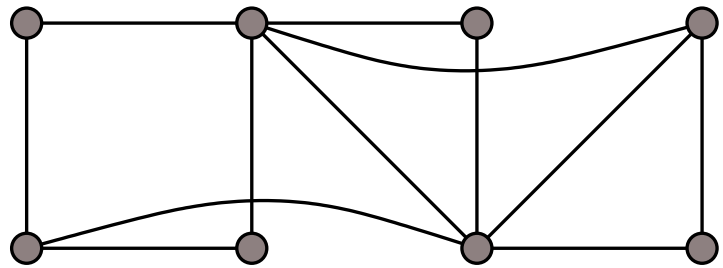


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

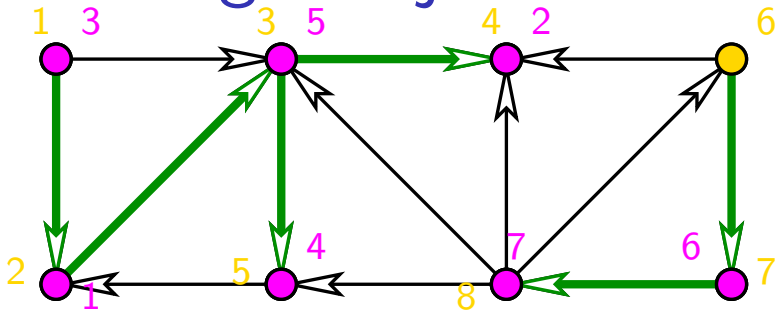


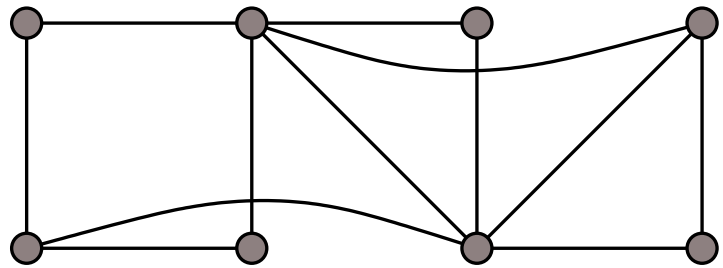


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

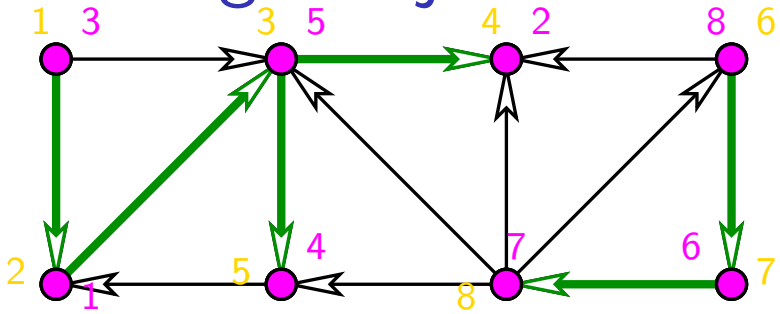


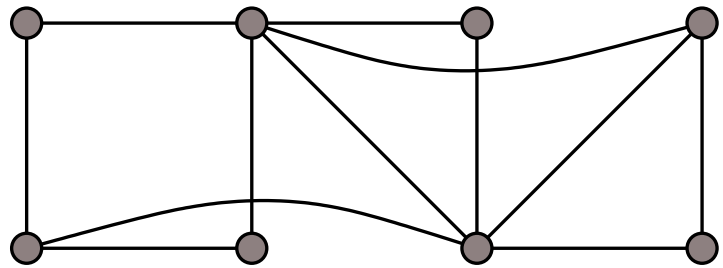


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás

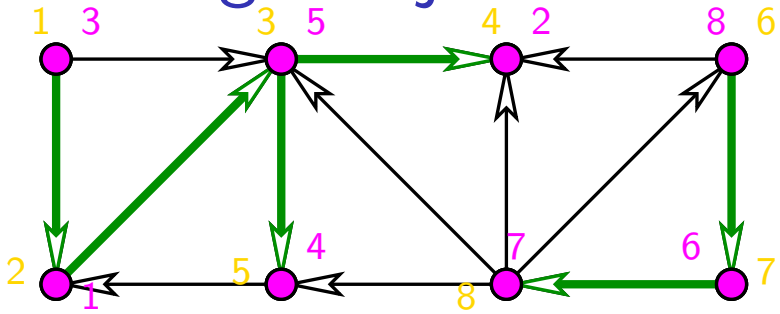


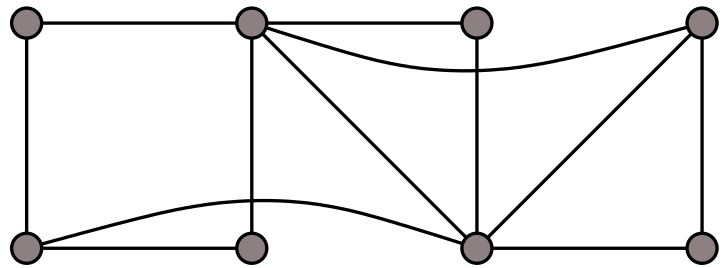


A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Általános gráfbejárás



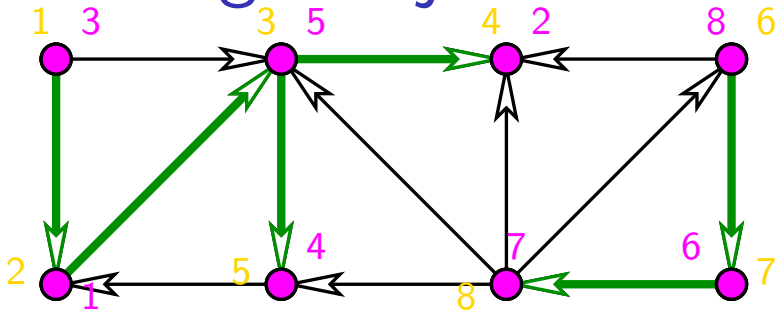


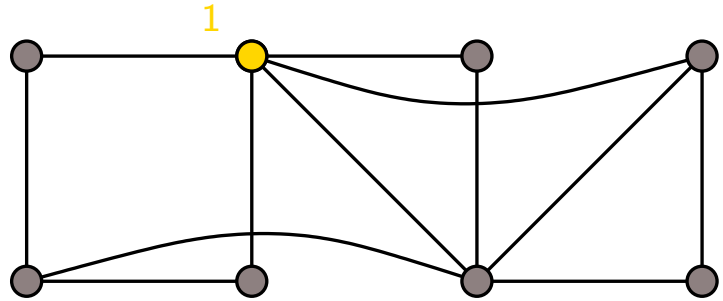
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



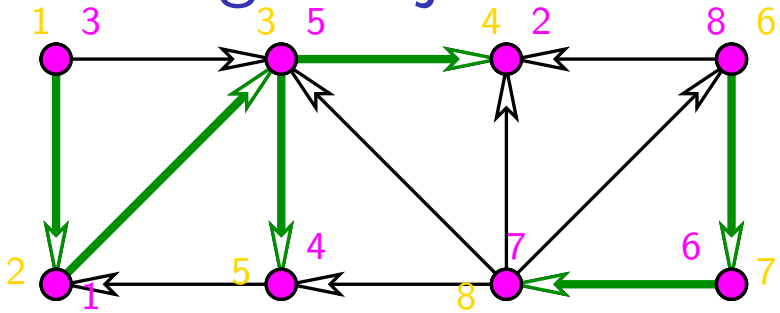


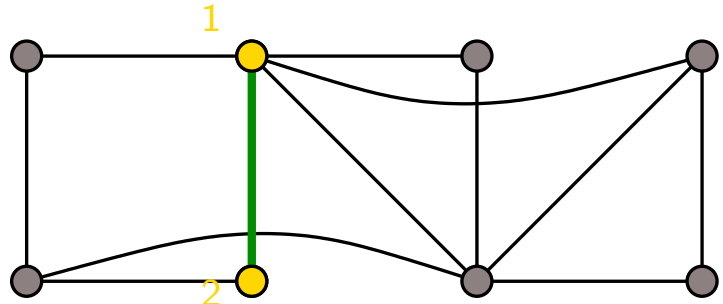
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



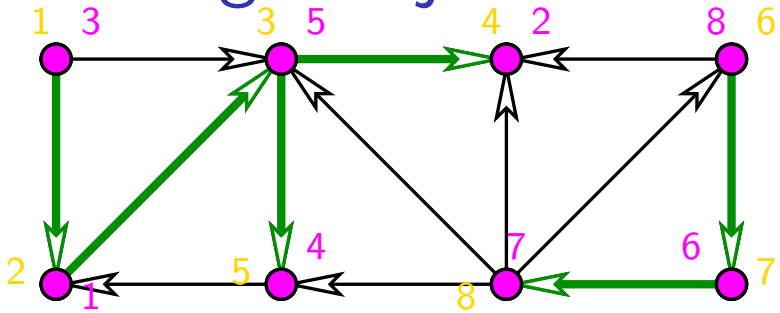


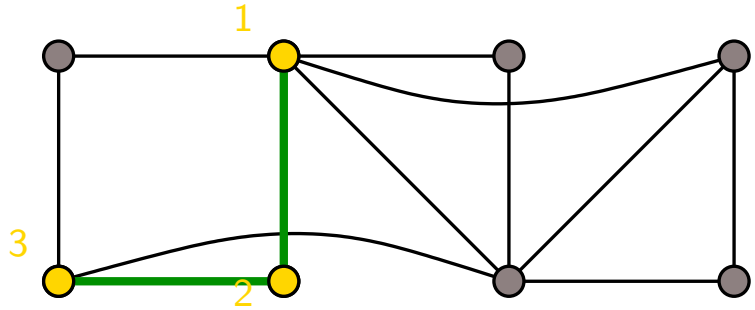
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



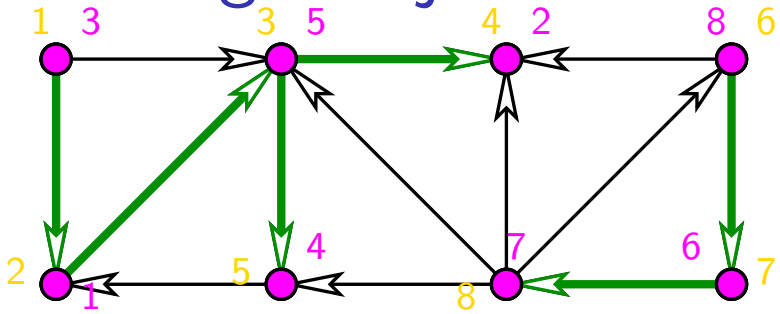


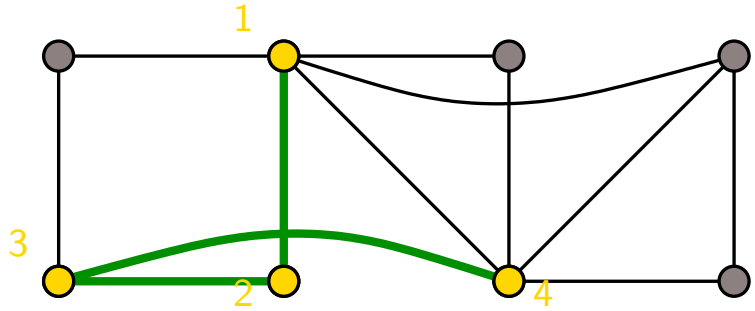
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



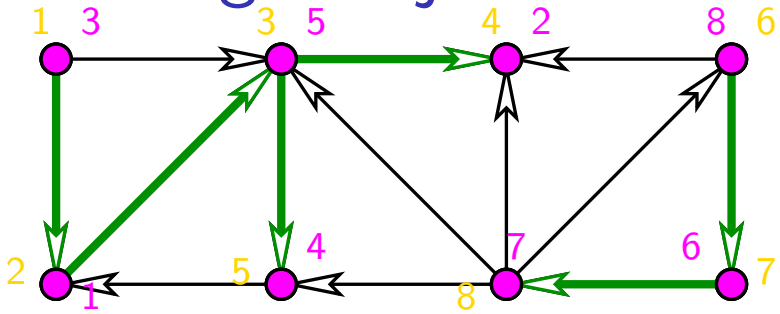


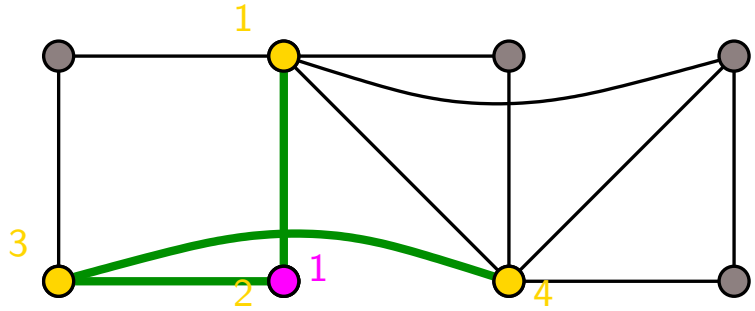
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



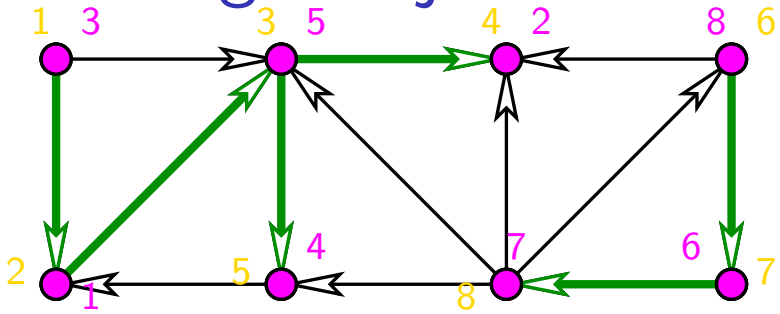


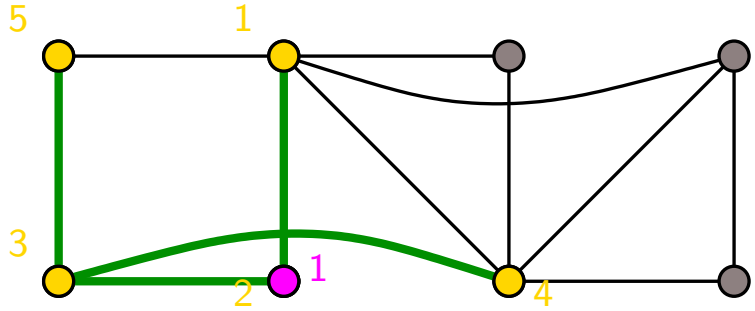
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



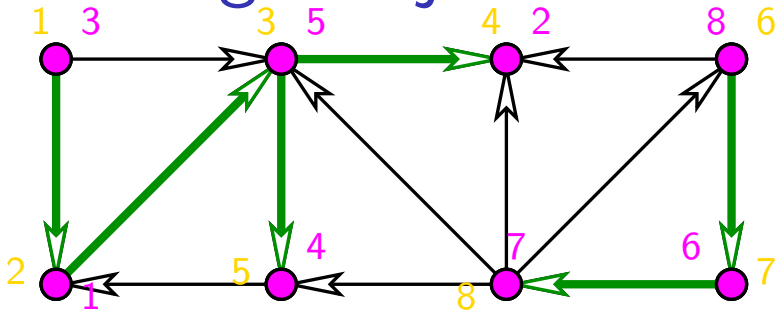


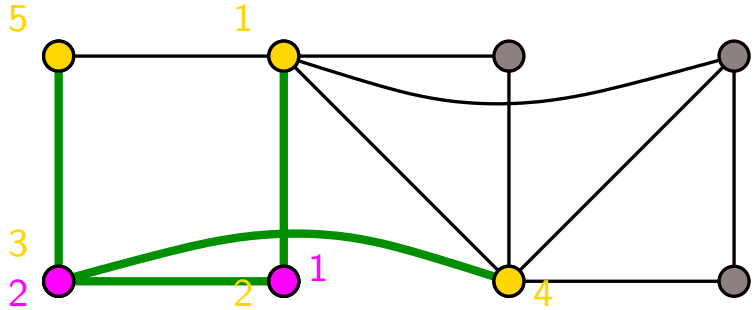
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\to$ elért $\to$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



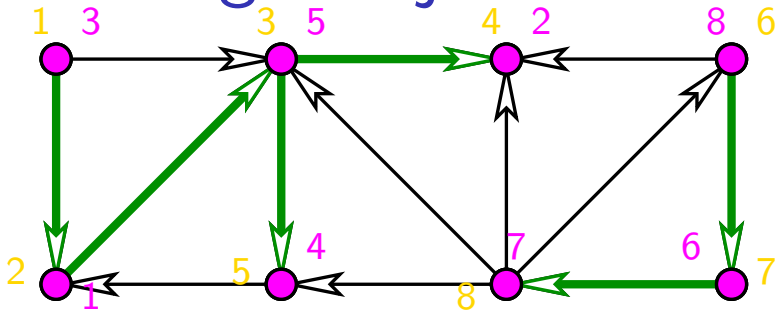


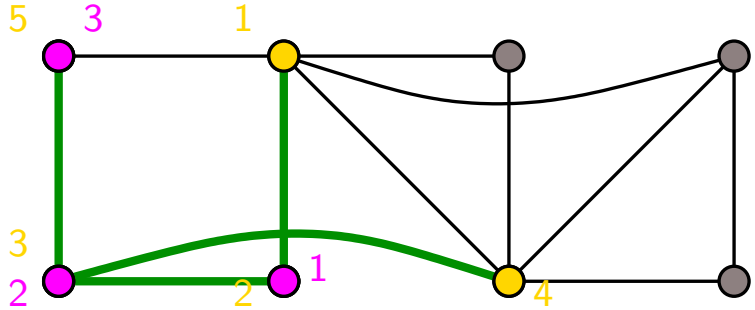
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\to$ elért $\to$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



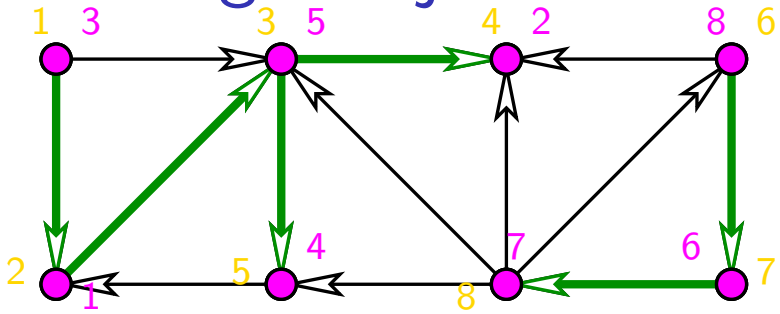


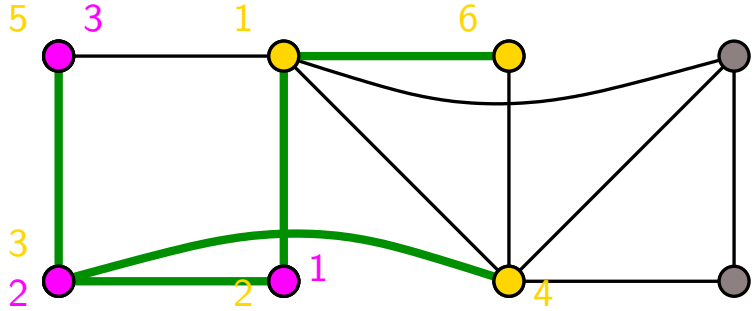
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



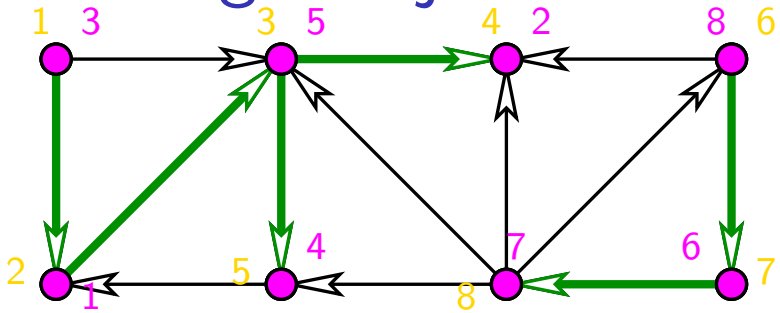


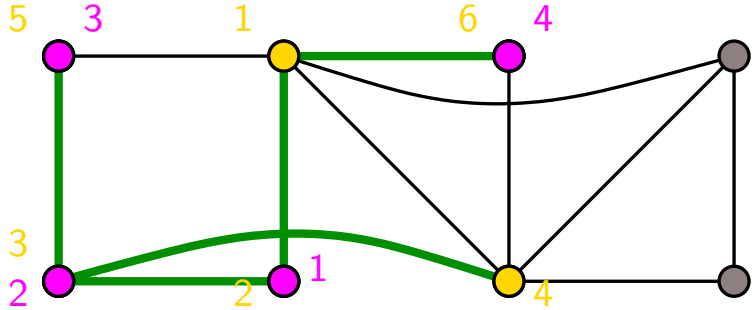
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



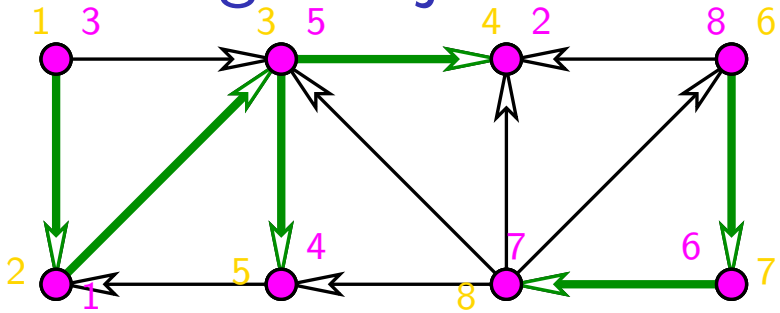


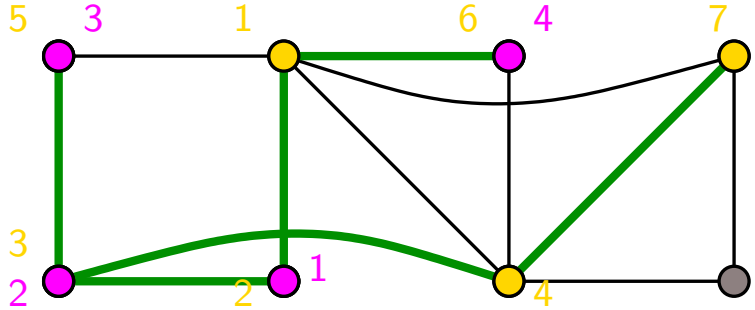
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



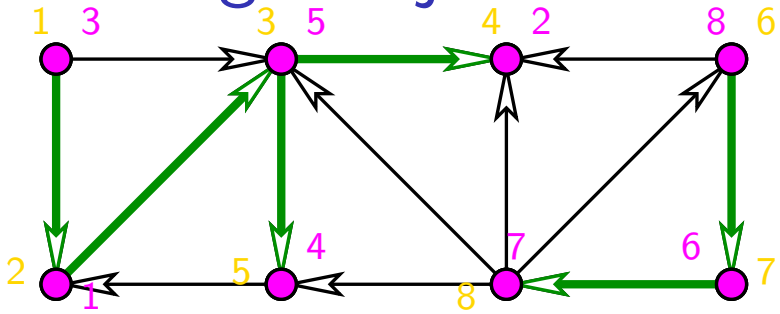


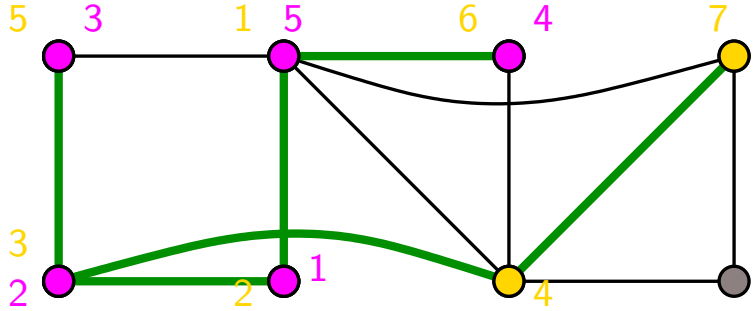
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



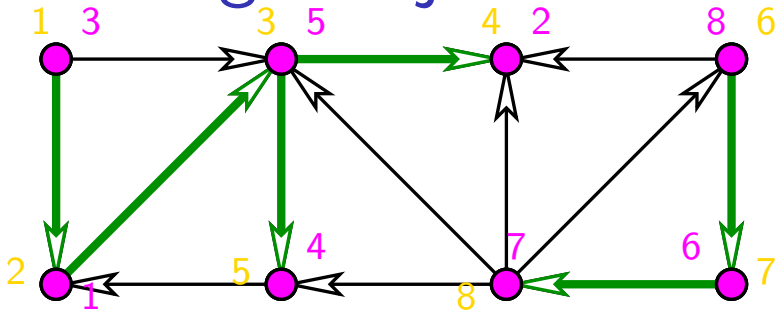


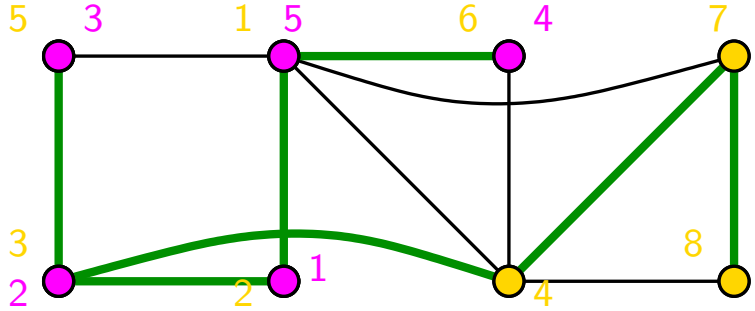
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



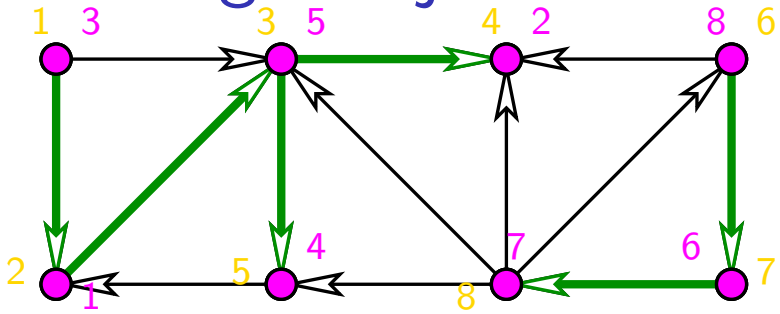


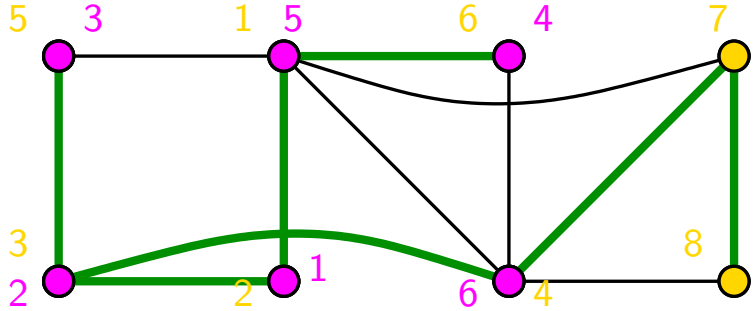
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



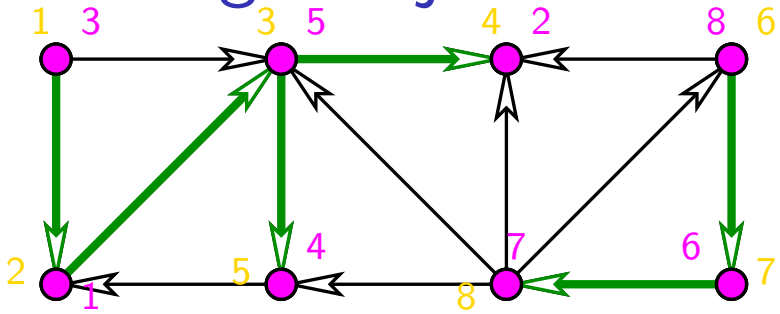


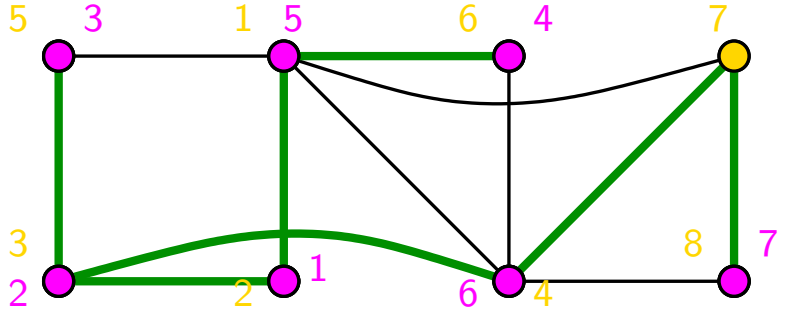
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



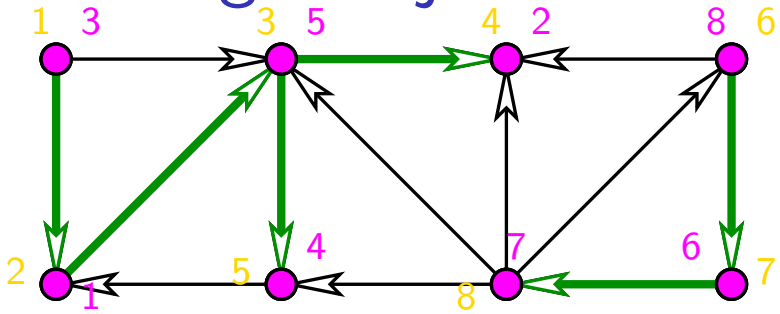


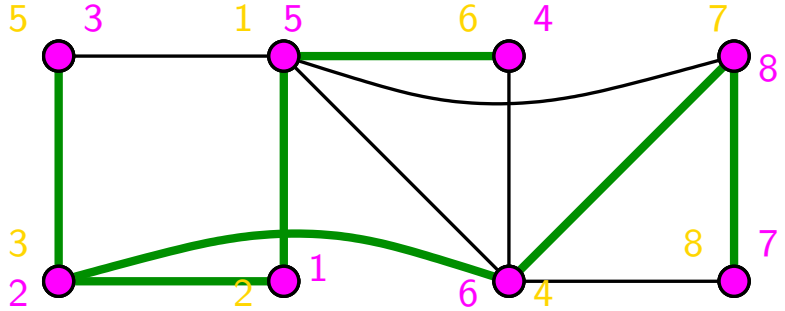
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\rightarrow$ elért $\rightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás



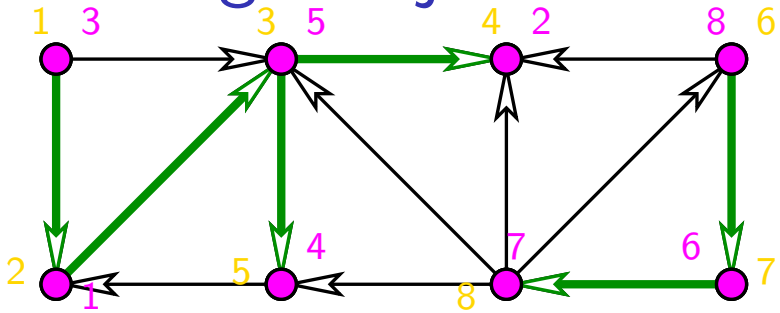


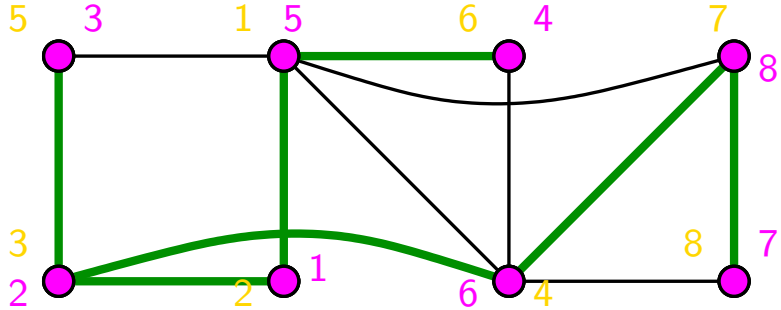
A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.
   - **(1a)** Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.
   - **(1b)** Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs.
   - **(2a)** Ha van eléretlen u csúcs, akkor u-t elértté tesszük.
   - **(2b)** Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett): END.

Nézzük meg egy irányítatlan gráf bejárását is.

Általános gráfbejárás

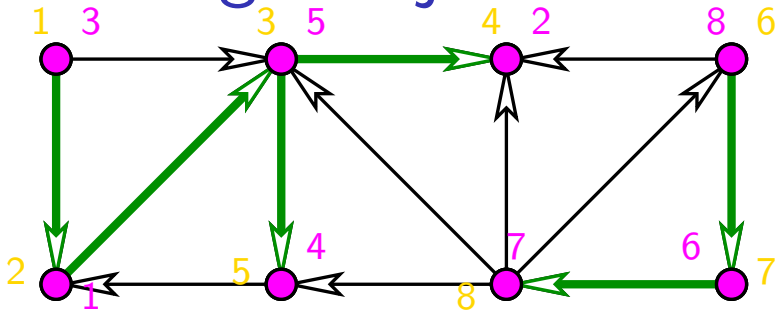


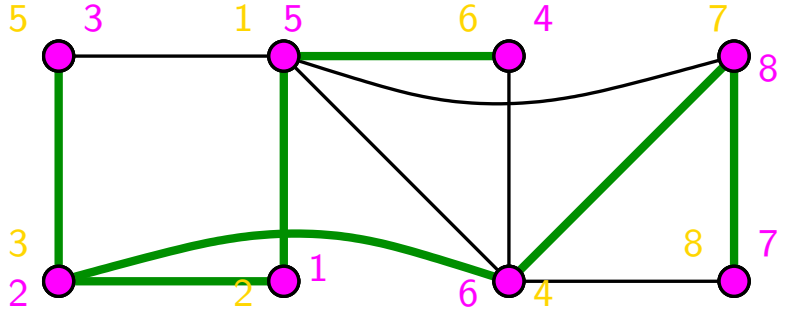


Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás

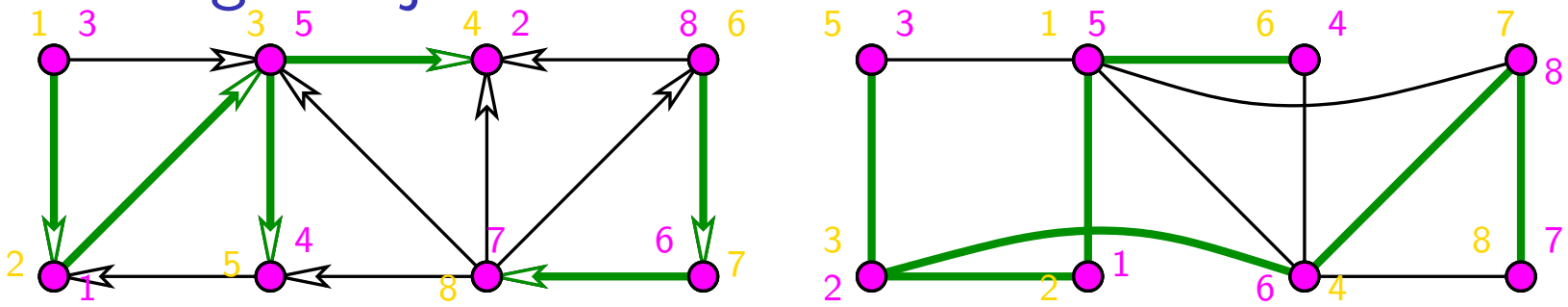




Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).
Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

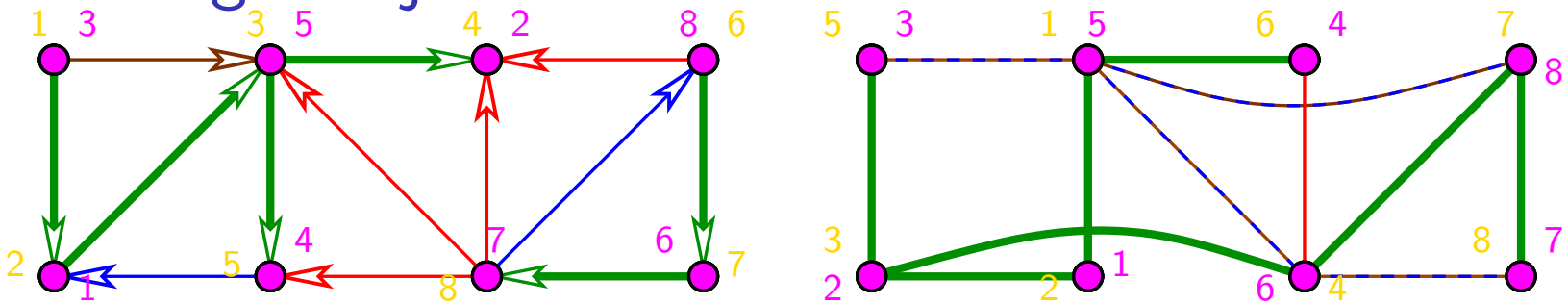
uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

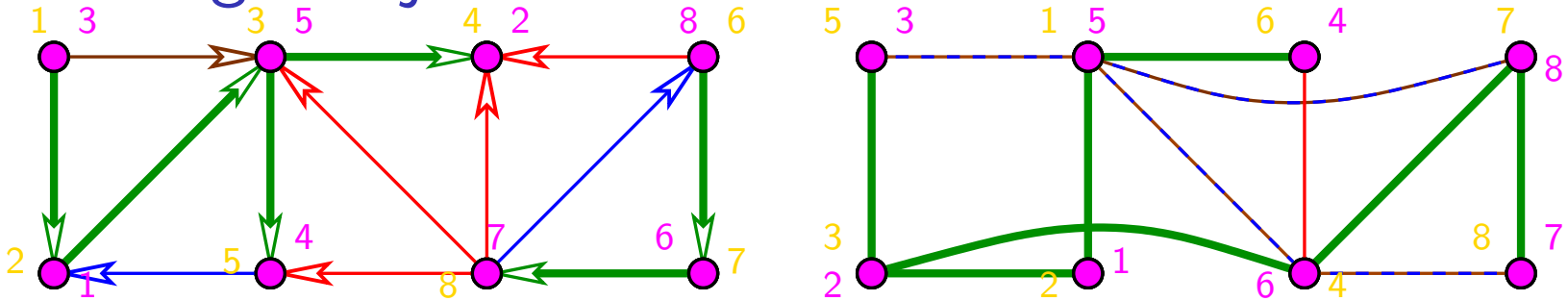
uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

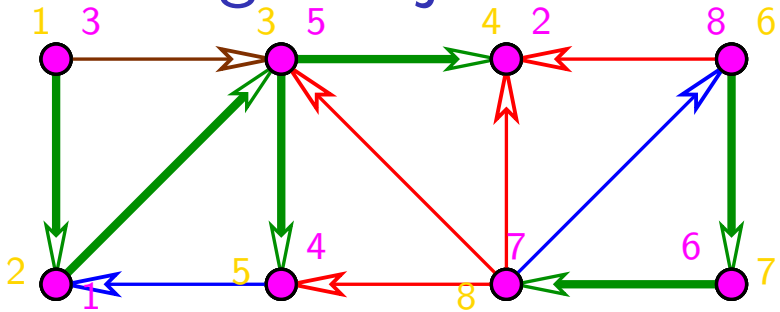
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

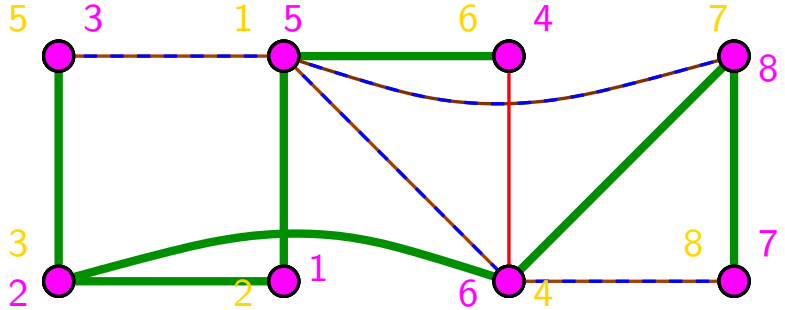
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

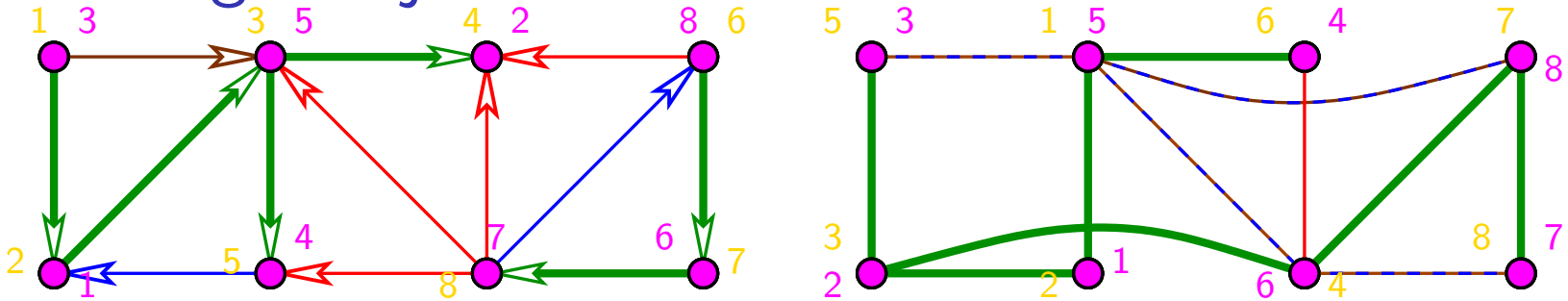
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

(A bejárás fája valójában egy gyökereiből kifelé irányított erdő.)

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

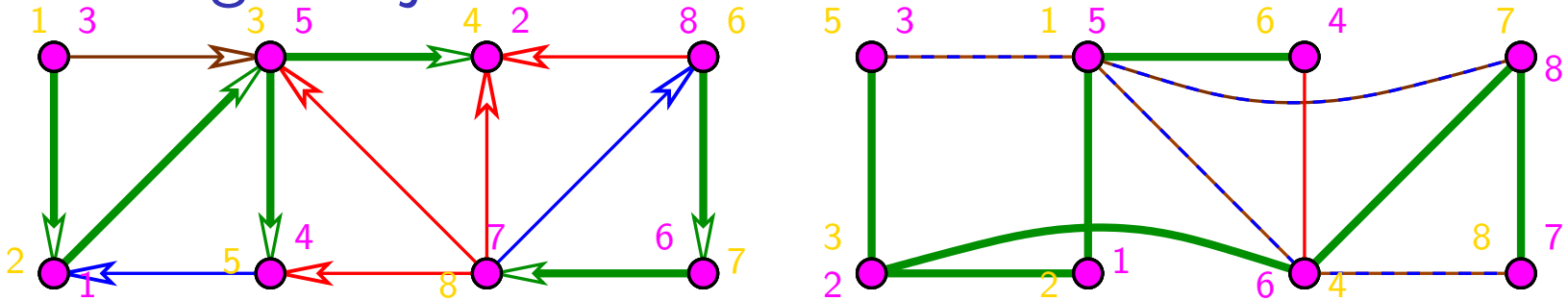
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

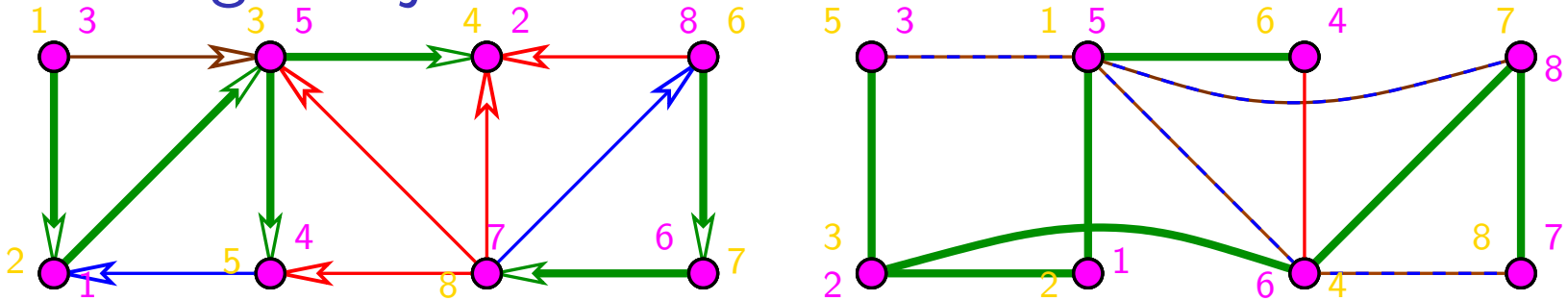
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

Megf: Irányítatlan esetben az előreél és a visszaél ugyanazt jelenti.

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

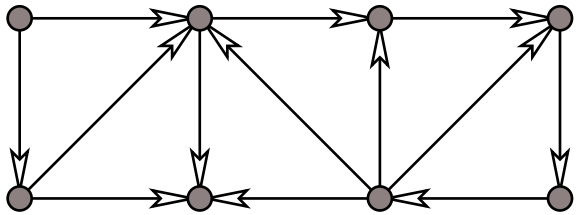
Általános gráfbejárás



Input: $G = (V, E)$ (ir/ir.tatlan) gráf, (esetleg $r \in V$ gyökér[1]).
Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.
(2) Az élek osztályozása:
faél: Olyan él, ami mentén egy csúcs elértté vált.
uv **előreél:** nem faél, de u-ból v-be faélekből irányított út vezet.
uv **visszaél:** v-ből u-ba faélekből irányított út vezet.
keresztél: minden más él (u és v közt nincs leszármazási viszony).
(3) A **bejárás fája**: a faélek alkotta részgráf.
Megf: Irányítatlan esetben az előreél és a visszaél ugyanazt jelenti.
Terminológia: Ha a bejárás fájában u-ból v-be irányított út vezet, akkor u a v őse és v az u leszármazottja. A faél és az előreél tehát ősből leszármazottba, a visszaél pedig leszármazottból ősbe vezet.

[1] A gyökér már kezdetben elért állapotú, így kivétel az általános szabály alól.

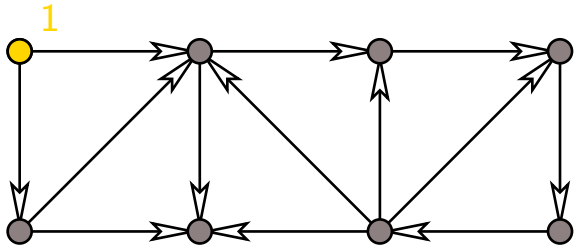
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

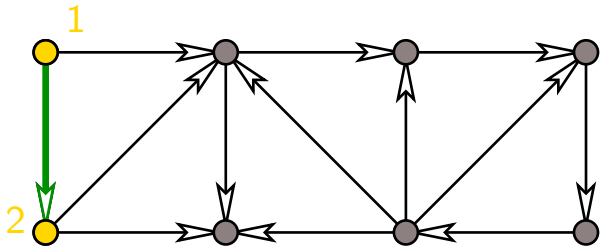
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

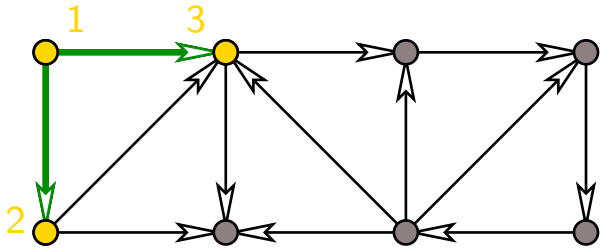
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

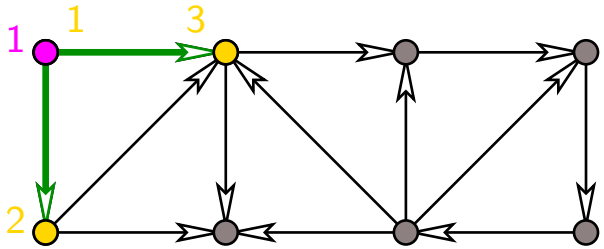
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

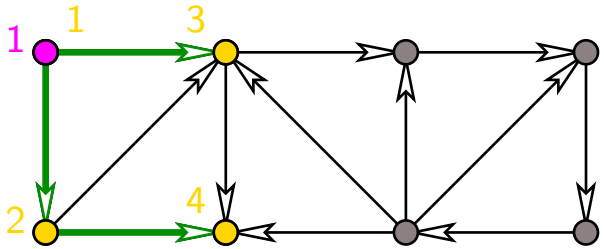
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

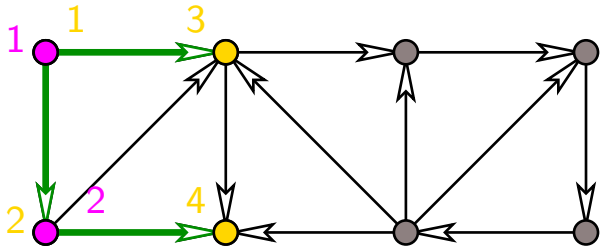
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

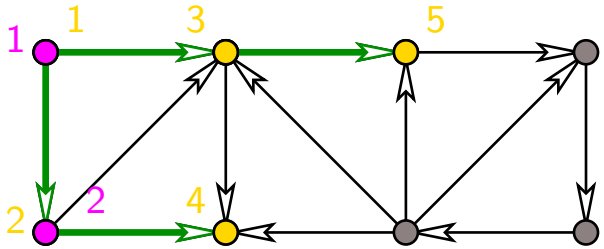
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

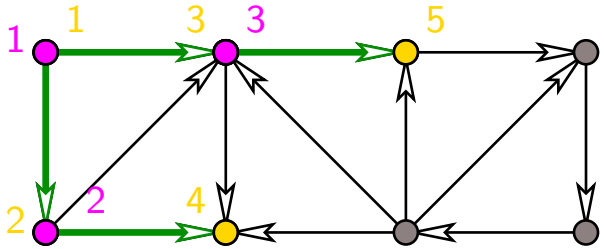
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

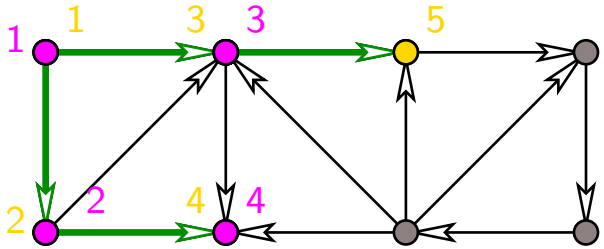
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

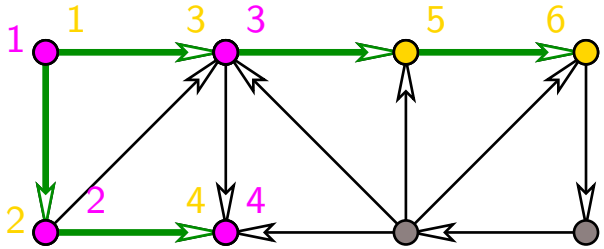
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

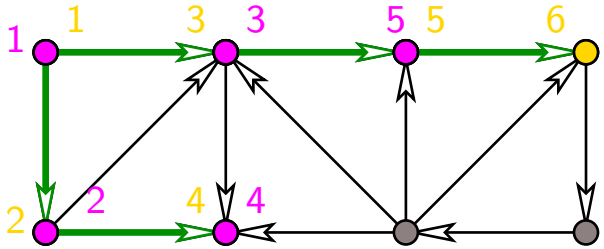
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

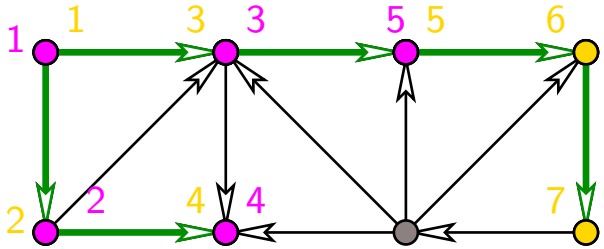
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

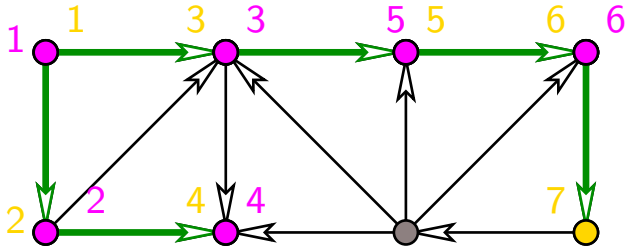
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

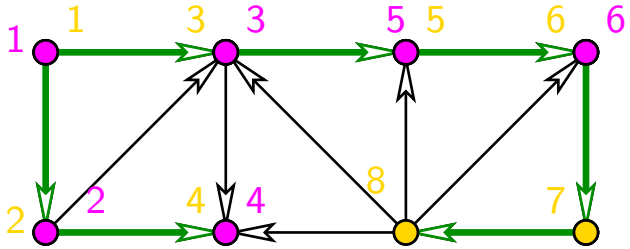
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

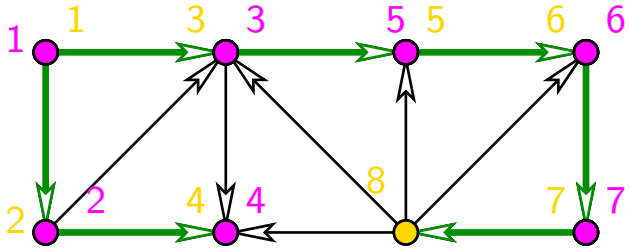
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

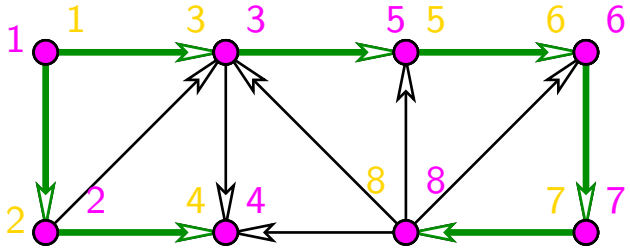
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

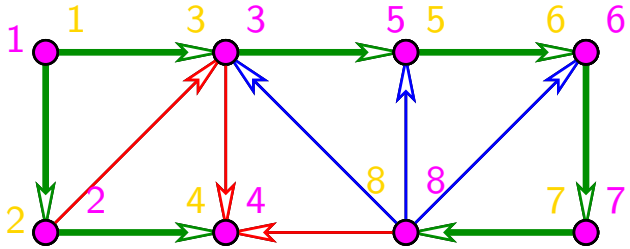
A BFS és tulajdonságai



Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

Nézzük meg egy **irányított** gráf BFS bejárását.

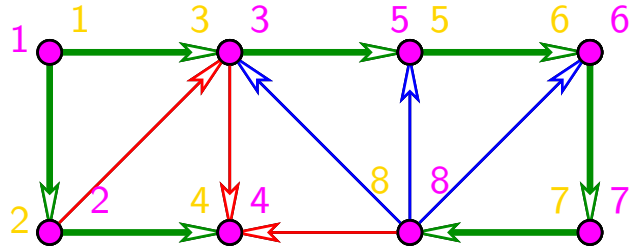
A BFS és tulajdonságai



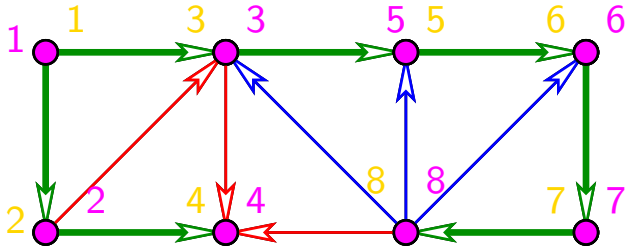
Szélességi bejárás (BFS) szabálya: Olyan bejárás, ahol az $\boxed{1.}$ esetben mindig a legkorábban elért u csúcsot választjuk.

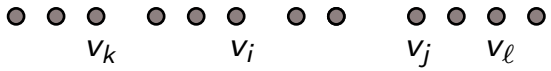
Nézzük meg egy **irányított** gráf BFS bejárását.

A BFS és tulajdonságai



A BFS és tulajdonságai

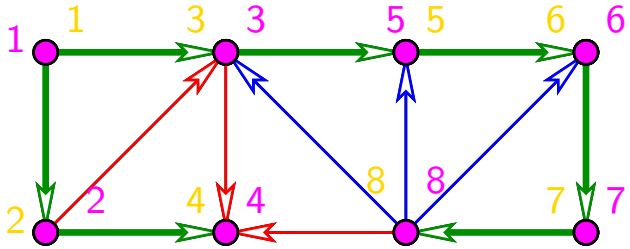


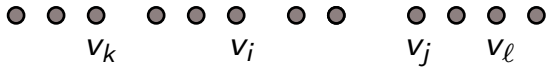


Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

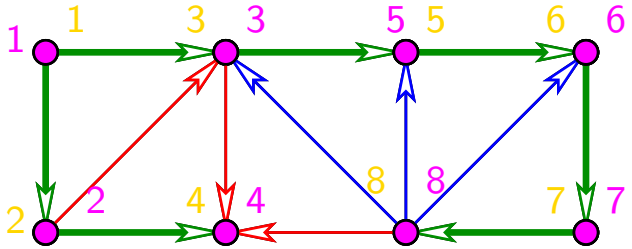
A BFS és tulajdonságai

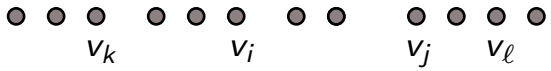




Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.
Biz: A v_i befejezésének pillanatában v_i minden gyereke elért, de v_j-nek még egy gyereke sem az. Ezért v_j gyerekeit a v_i csúcs befejezése után érjük el, majd ezt követően fejezzük be v_j-t. □

A BFS és tulajdonságai

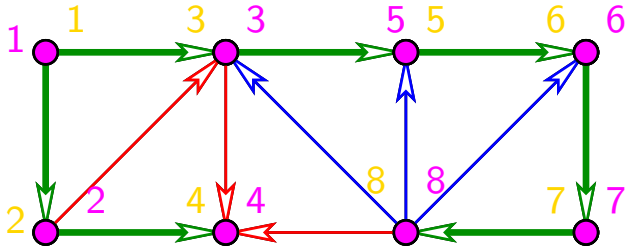


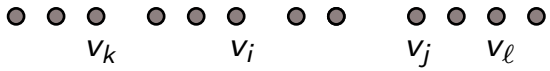


Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

A BFS és tulajdonságai



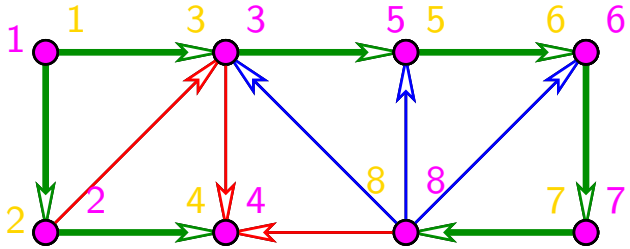


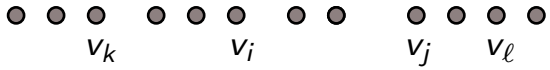
Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

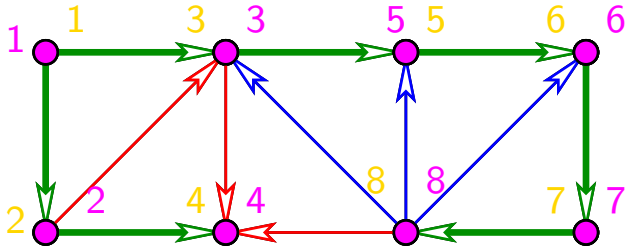
A BFS és tulajdonságai

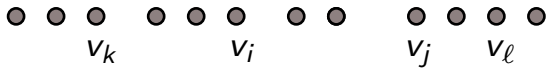




Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.
(2) **Az elérési és befejezési sorrend megegyezik.**
Biz: Ha v_i-t korábban érjük el, mint v_j-t, akkor (1) miatt v_i-t korábban is fejezzük be v_j-nél. Ezért bármely két csúcs sorrendje ugyanaz az elérési sorrendben mint befejezési sorrendben. Tehát az elérési sorrendnek meg kell egyeznie a befejezési sorrenddel. □

A BFS és tulajdonságai



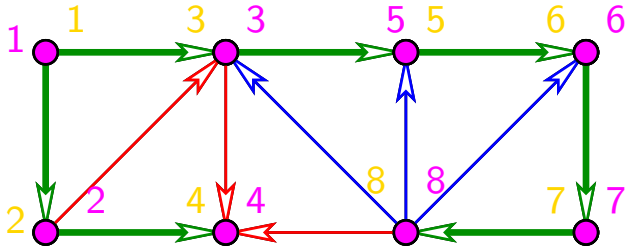


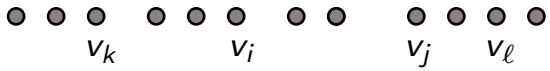
Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

A BFS és tulajdonságai





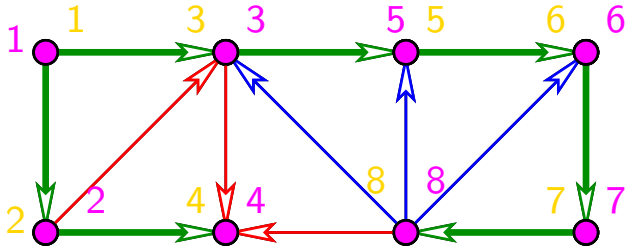
Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

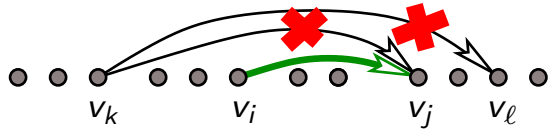
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

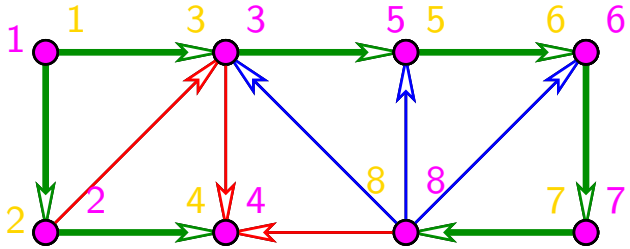
A BFS és tulajdonságai

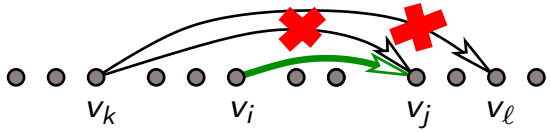




Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.
(2) **Az elérési és befejezési sorrend megegyezik.**
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.
Biz: Ha $v_k v_\ell \in E(G)$, akkor v_ℓ szülője v_k vagy egy v_k-t megelőző csúcs. (1) miatt v_j szülője sem következhet v_k után, vagyis v_i nem lehet v_j szülője. □

A BFS és tulajdonságai





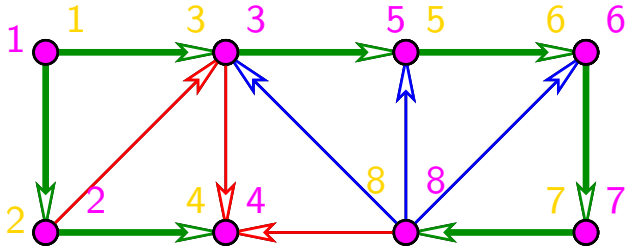
Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

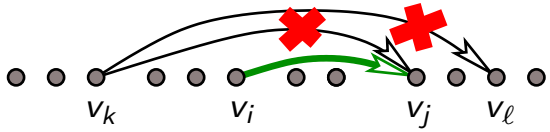
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

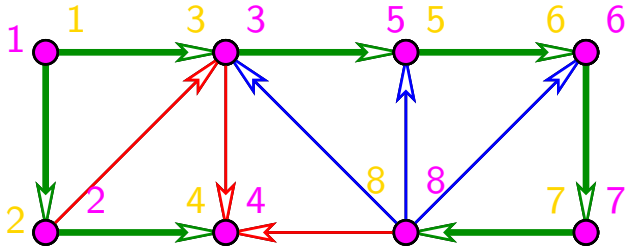
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

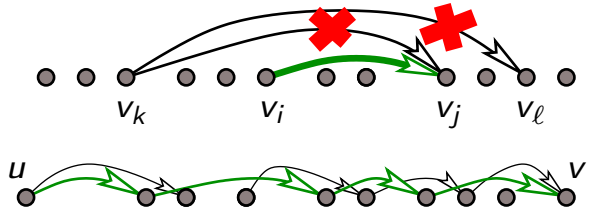
(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

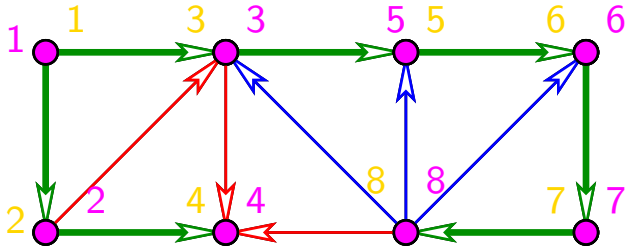
(2) **Az elérési és befejezési sorrend megegyezik.**

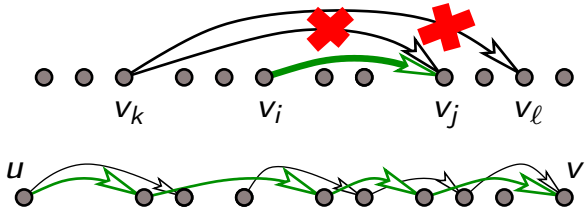
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

Biz: Ha P' egy G-beli uv-út, akkor P' egyetlen éle sem ugorhat át P-beli élt. Ezért P' utolsó éle nem kezdődhet korábban P utolsó élénél. Hasonló igaz P' utolsó előtti, stb éleire. Így v-ből nem lehet u-ba visszajutni P élszámánál kevesebb élen. □

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

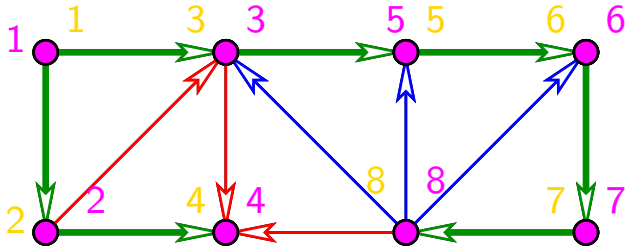
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

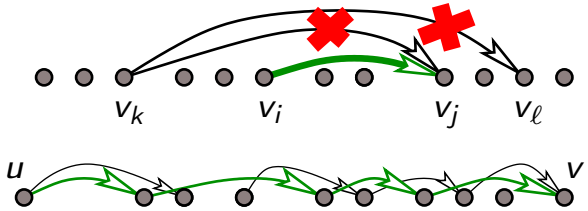
(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

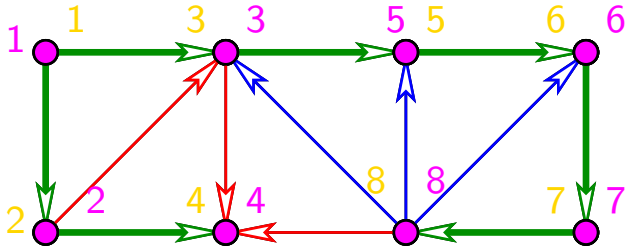
(2) **Az elérési és befejezési sorrend megegyezik.**

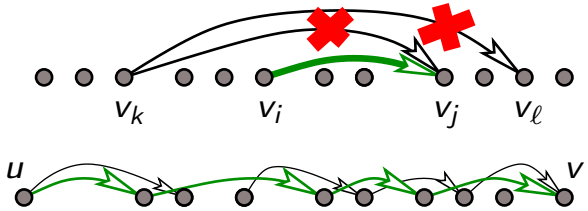
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

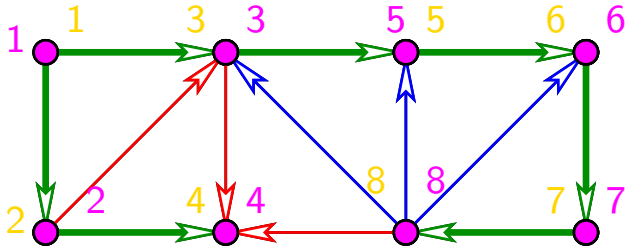
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

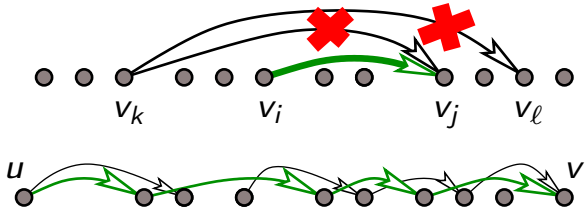
(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

Biz: Közvetlenül adódik (5)-ből.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

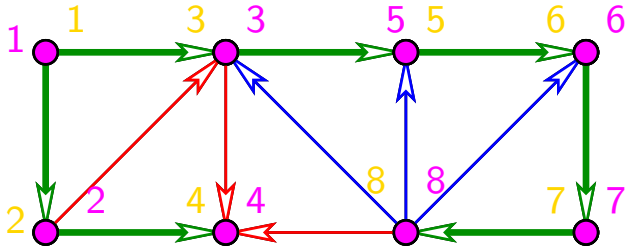
(2) **Az elérési és befejezési sorrend megegyezik.**

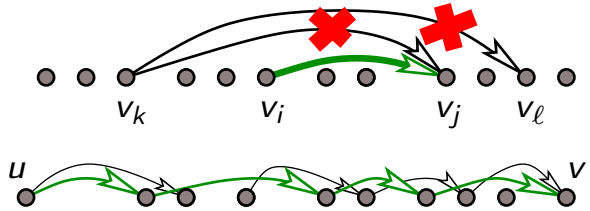
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

A BFS és tulajdonságai





Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

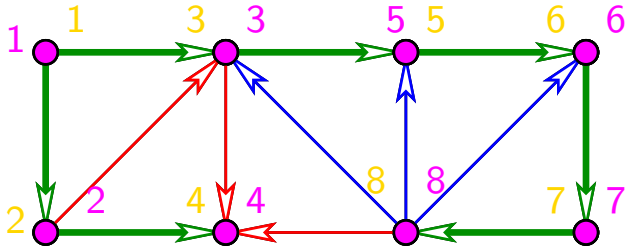
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

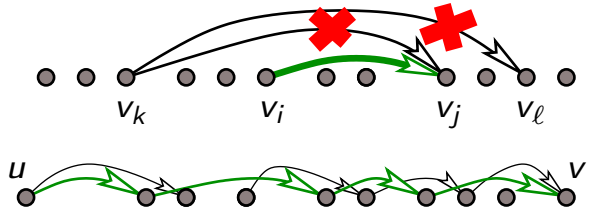
(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

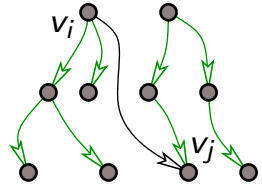
(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

(6) Minden él legfeljebb egy szintet lép lefelé a BFS-fában, így **nincs előreél.** (Irányítatlan esetben csak faél és keresztél van.)

A BFS és tulajdonságai







Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

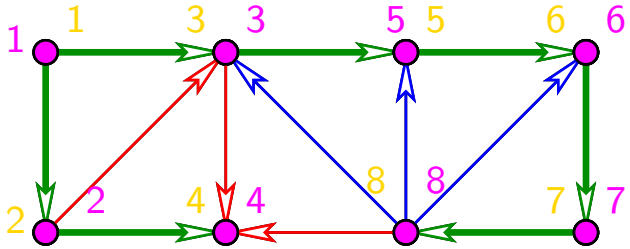
(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

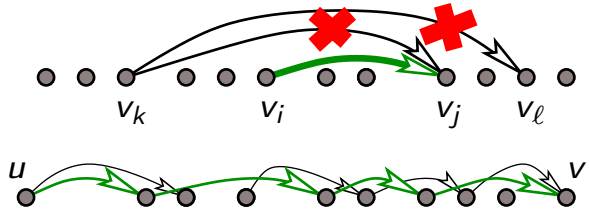
(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

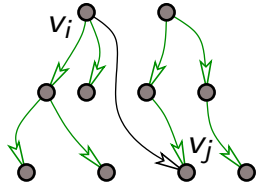
(6) Minden él legfeljebb egy szintet lép lefelé a BFS-fában, így **nincs előreél.** (Irányítatlan esetben csak faél és keresztél van.)

Biz: Ha $v_i v_j$ legalább két szintet lép lefelé, akkor v_1-ből v_j-be nem a BFS-fabeli lenne a legrövidebb út. Ez ellentmond (5)-nek.

A BFS és tulajdonságai







Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

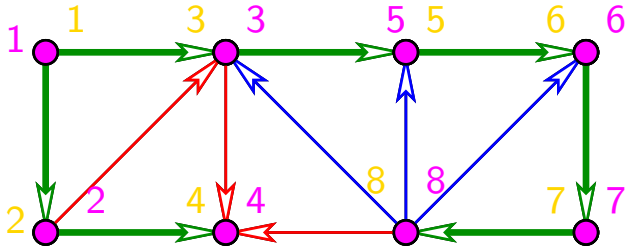
(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

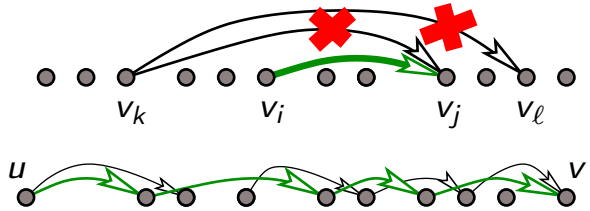
(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

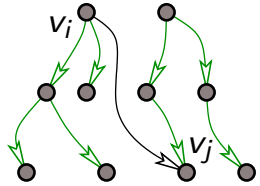
(6) Minden él legfeljebb egy szintet lép lefelé a BFS-fában, így **nincs előreél.** (Irányítatlan esetben csak faél és keresztél van.)

Biz:

A BFS és tulajdonságai







Állítás: Tfh $G = (V, E)$ **BFS bejárása után** a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei az elérési sorrendben megelőzik v_j gyerekeit.

(2) **Az elérési és befejezési sorrend megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) Ha P a BFS-fa uv-útja, akkor P legrövidebb uv-út G-ben is.

(5) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

(6) Minden él legfeljebb egy szintet lép lefelé a BFS-fában, így **nincs előreél.** (Irányítatlan esetben csak faél és keresztél van.)

Biz: Ha pedig lenne előreél, akkor az legalább két szintet lépne lefelé. Most láttuk, hogy ez lehetetlen. □

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Megj: Az ℓ hosszfüggvény konzervatív tulajdonsága azt jelenti, hogy G-ben nincs negatív összhosszúságú irányított kör.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma.
Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.

Triviális (r, ℓ)-felső becslés:

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.
Triviális (r, ℓ)-felső becslés: $f(v) = \begin{cases} 0 & v = r \\ \infty & v \neq r \end{cases}$.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma.
Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.
Triviális (r, ℓ)-felső becslés: $f(v) = \begin{cases} 0 & v = r \\ \infty & v \neq r \end{cases}$.
Pontos (r, ℓ)-felső becslés:

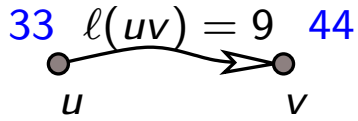
Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\tilde{\ell}(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\tilde{\ell}(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.

Az ℓ hosszfv konzervatív, ha $\tilde{\ell}(C) \geq 0$ a G bármely C (ir) körére.

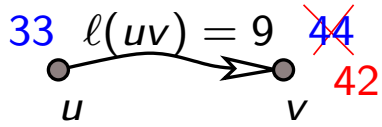
Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\tilde{\ell}(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.
Triviális (r, ℓ)-felső becslés: $f(v) = \begin{cases} 0 & v = r \\ \infty & v \neq r \end{cases}$.
Pontos (r, ℓ)-felső becslés: $f(v) = dist_\ell(r, v)\ \forall v \in V(G)$.

Az élmenti javítás



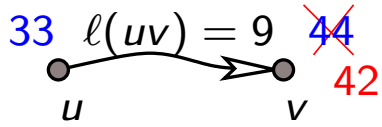
Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Az élmenti javítás



Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Az élmenti javítás

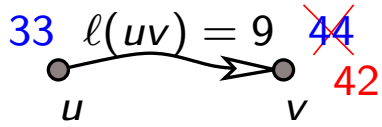


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.

Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

Az élmenti javítás



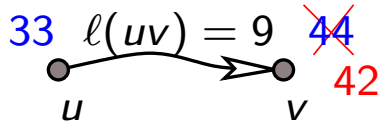
Def: Tfh f egy (r,ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.

Ekkor (1) Az f (r,ℓ)-fb élmenti javítása mindig (r,ℓ)-fb-t ad.

Megj: A fenti Megfigyelés teljesüléséhez nem szükséges megkívánni az ℓ hosszfüggvény nemnegativitását: már az is elég, ha ℓ konzervatív.

Az élmenti javítás

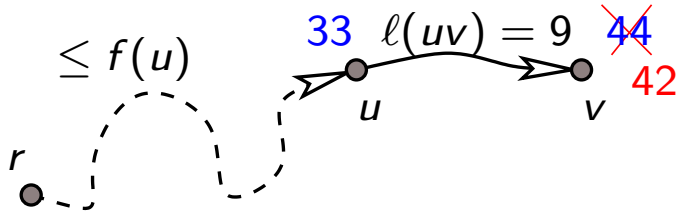


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.

Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

Az élmenti javítás

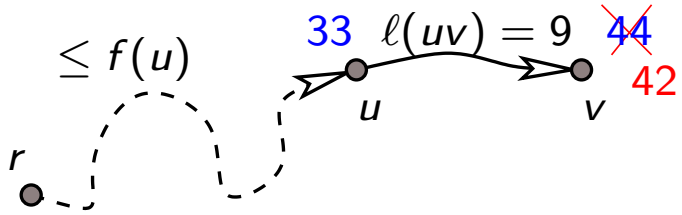


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

Biz: Azt kell megmutatni, hogy van olyan rv-út, aminek a hossza legfeljebb $f(u) + \ell(uv)$. Ha egy legrövidebb ru-utat kiegészítünk az uv éllel, akkor olyan rv-élsorozatot kapunk, aminek az összhossza $dist_\ell(r, u) + \ell(uv) \leq f(u) + \ell(uv)$. ,,Könnyen" látható, hogy az élhosszfv konzervativitása miatt ha van x összhosszúságú rv-élsorozat, akkor van legfeljebb x összhosszúságú rv-út is. Ezek szerint van legfeljebb $f(u) + \ell(u, v)$ hosszúságú uv-út is, azaz az uv élmenti javítás után szintén (r, ℓ)-fb-t kapunk. □

Az élmenti javítás

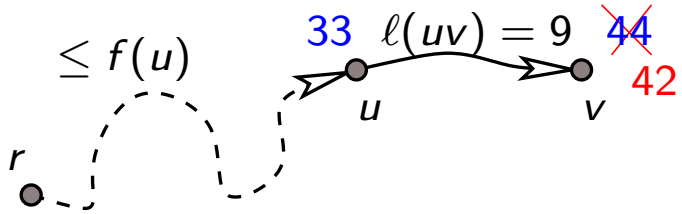


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.

Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

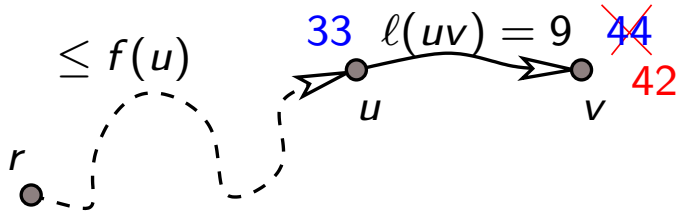
Az élmenti javítás



Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

Az élmenti javítás

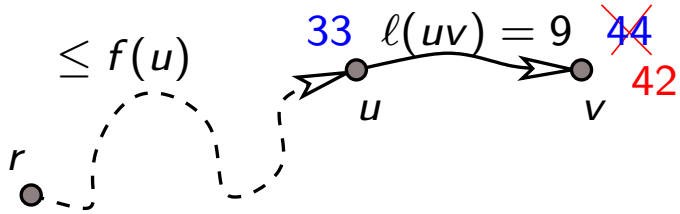


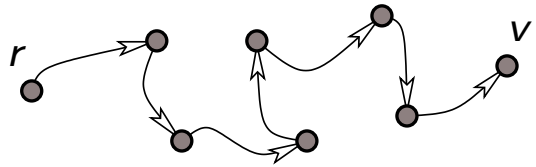
Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$. Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad. (2) f (r, ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

Biz: $\Rightarrow$: Ha f pontos, akkor biztosan nincs rajta érdmei élmenti javítás: ha volna, akkor egy felső becslés a pontos érték alá csökkenne, így az élmenti javítás eredménye nem lenne (r, ℓ)-fb.

Az élmenti javítás



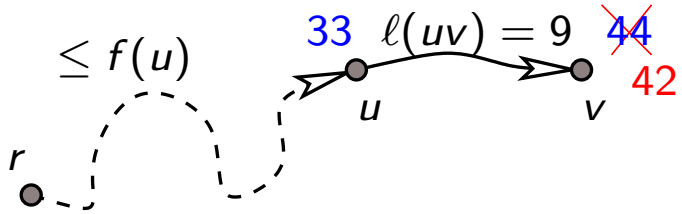


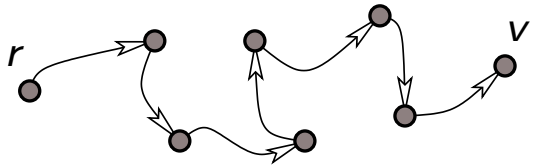
Def: Tfh f egy (r,ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u)+\ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r,ℓ)-fb élmenti javítása mindig (r,ℓ)-fb-t ad.
(2) f (r,ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

Biz: $\Rightarrow$: Ha f pontos, akkor biztosan nincs rajta érdmei élmenti javítás: ha volna, akkor egy felső becslés a pontos érték alá csökkenne, így az élmenti javítás eredménye nem lenne (r,ℓ)-fb.
$\Leftarrow$: Legyen $v \in V(G)$ tetsz, és legyen P egy legrövidebb rv-út. A P egyik éle mentén sincs érdemi élmenti javítás, ezért P első, második, harmadik stb csúcsára pontos a felső becslés, azaz $f(u) = dist_\ell(r,u)$ teljesül ezen u csúcsokra. Ez igaz P utolsó csúcsára, a tetszőlegesen választott v-re is. □

Az élmenti javítás

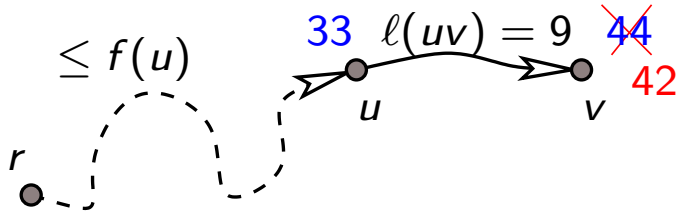


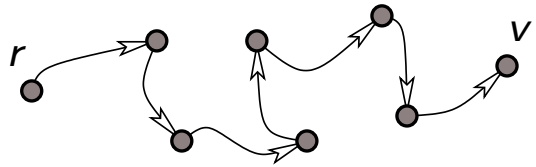


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

Az élmenti javítás





Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

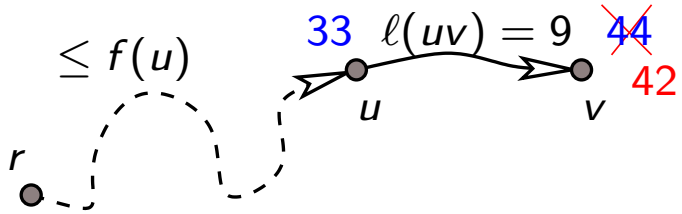
Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

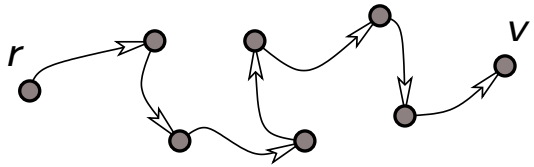
Köv: Adott G, nemnegatív ℓ hosszfüggvény és $r \in V(G)$ gyökér esetén ha kiindulunk a triviális (r, ℓ)-fb-ből, és addig végzünk élmenti javításokat, amíg lehet, akkor a végén megkapjuk minden csúcs r-től való távolságát.

Kérdés: Milyen sorrendben végezzül az élmenti javításokat, ha garantáltan gyorsan szeretnénk végeznünk a feladattal?

Bemelegítés Fontos spec. eset: nemnegatív élhosszok.

Az élmenti javítás

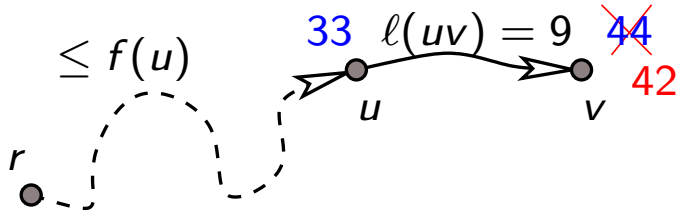


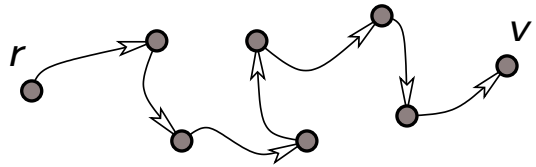


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

Az élmenti javítás



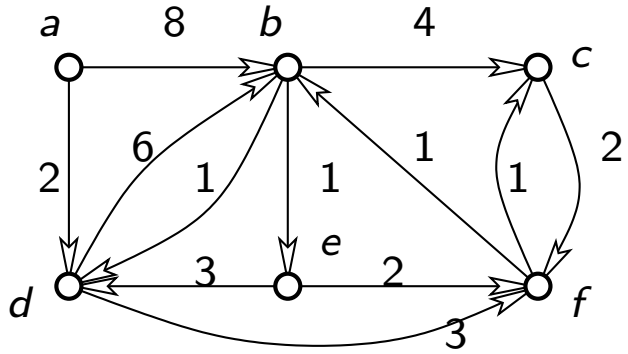


Def: Tfh f egy (r,ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u)+\ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv nemnegatív és $f(r) = 0$.
Ekkor (1) Az f (r,ℓ)-fb élmenti javítása mindig (r,ℓ)-fb-t ad.
(2) f (r,ℓ)-fb (pontos) $\iff$ (f-en nincs érdemi élmenti javítás).

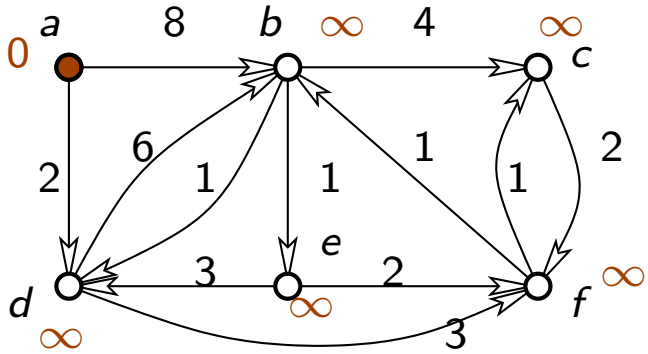
Dijkstra-algoritmus: Input: $G = (V,E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r,v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r,ℓ)-fb.
Az i-dik fázis ($i = 1,2,\ldots,|V|$):
1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.
Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:
1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.
Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



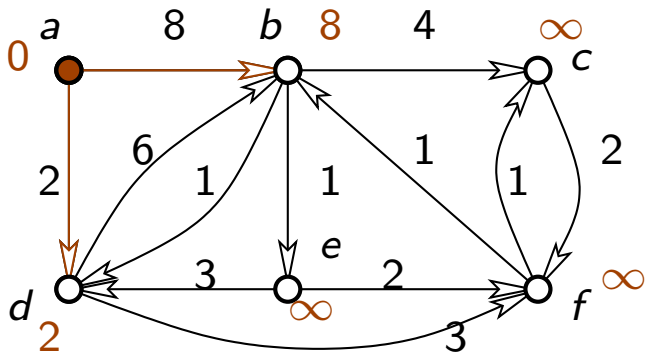
a	b	c	d	e	f
0	∞	∞	∞	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



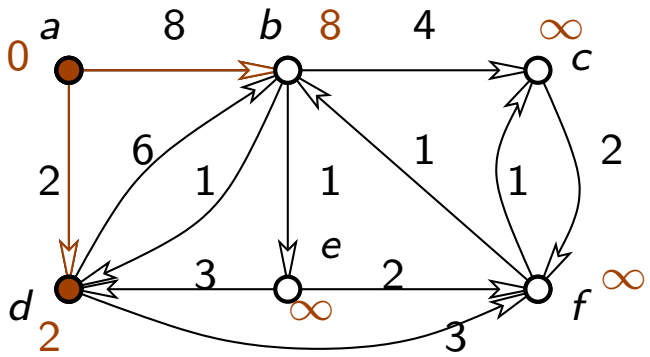
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



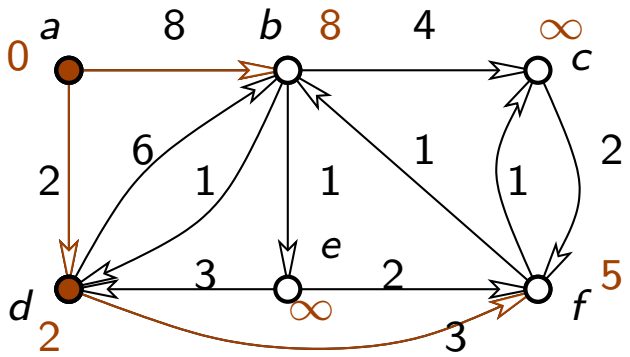
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



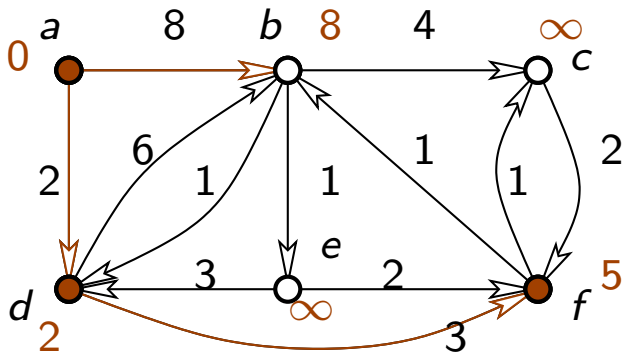
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



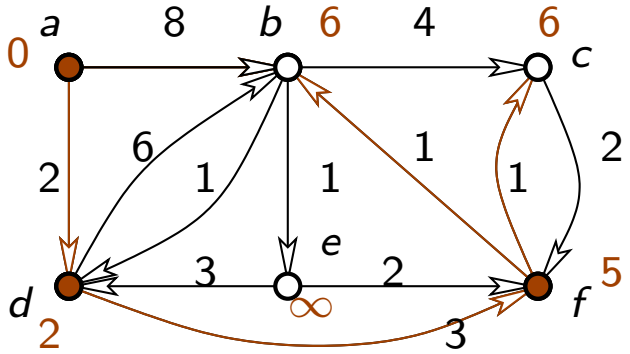
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



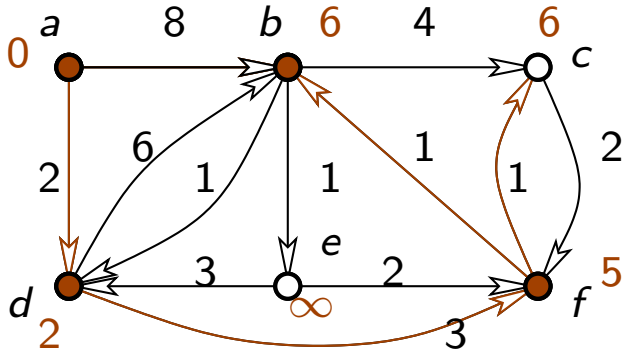
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



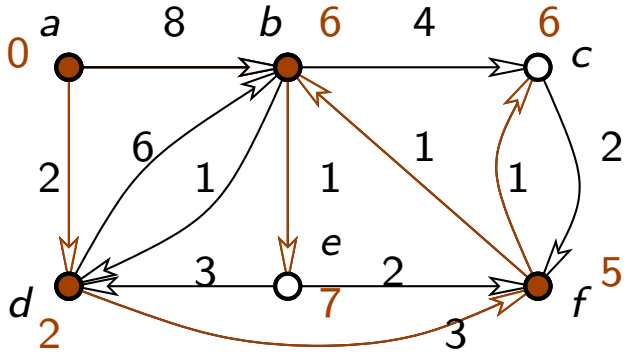
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



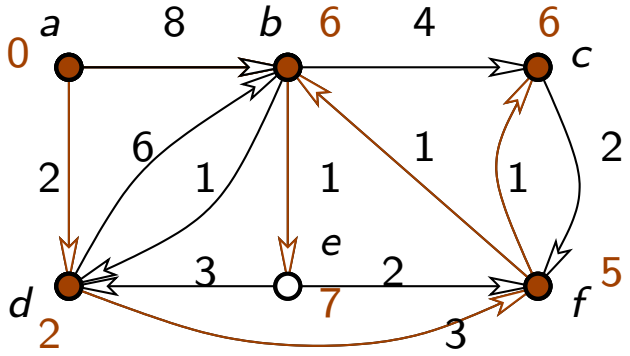
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



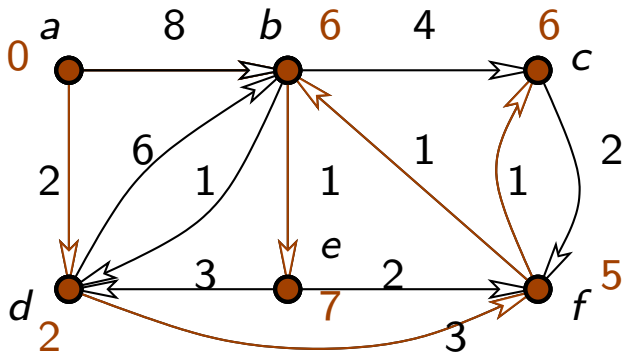
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



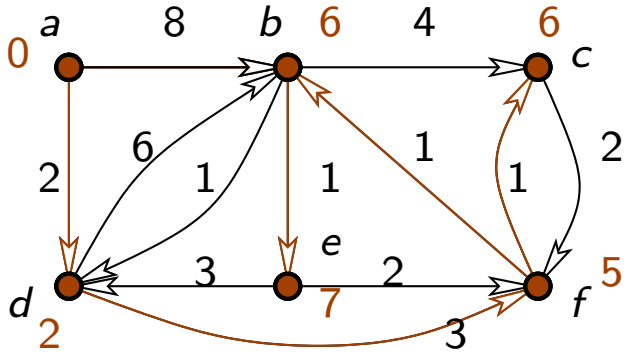
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



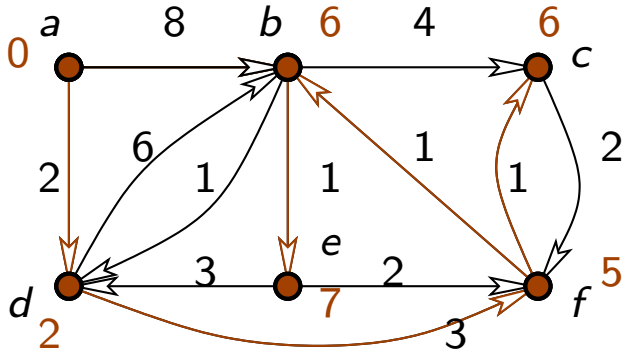
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet út r-ből v-be megjelölt éleken is, és ennek hossza megegyezik $f_{|V|}(v)$-vel.

Dijkstra, egy példán



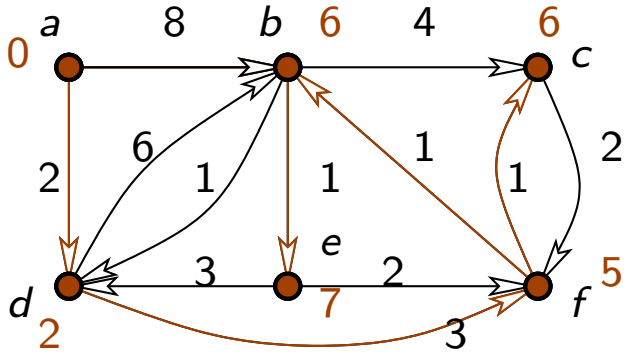
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet út r-ből v-be megjelölt éleken is, és ennek hossza megegyezik $f_{|V|}(v)$-vel.
Biz: $f_{|V|}(r) = 0$, és a megjelölt élek mentén haladva az $f_{|V|}$ érték az élhosszal növekszik. □

Dijkstra, egy példán



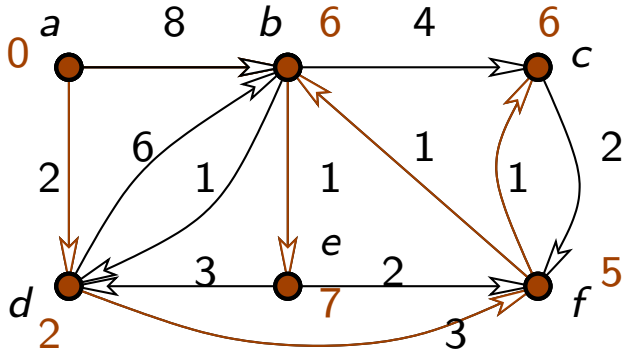
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet út r-ből v-be megjelölt éleken is, és ennek hossza megegyezik $f_{|V|}(v)$-vel.

Dijkstra, egy példán



a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Megf: Ha v-be vezet megjelölt él, akkor vezet út r-ből v-be megjelölt éleken is, és ennek hossza megegyezik $f_{|V|}(v)$-vel.

Köv: Ha a Dijkstra-algoritmus helyes, akkor az algoritmus végén a megjelölt élek egy legrövidebb utak fáját alkotják r gyökérrel.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
Biz: Az i-dik fázisban $f_i(u_i) \leq f_i(u_{i+1})$ teljesült az u_i választása miatt. Ezek után $f_i(u_i)$ már nem változott: $f_{|V|}(u_i) = f_i(u_i)$. Ugyan $f_i(u_{i+1})$ még csökkenhetett, de csak az $u_i u_{i+1}$ él mentén történt javítás miatt, hiszen az $(i+1)$-dik fázisban u_{i+1} bekerült az U_{i+1} halmazba, és a hozzá tartozó (r, ℓ)-fb már nem csökken tovább. Mivel $\ell(u_i u_{i+1}) > 0$, ezért
$f_{i+1}(u_{i+1}) = \min\{f_i(u_{i+1}), f_i(u_i) + \ell(u_i u_{i+1})\} \geq f_i(u_i)$. Tehát
$f_{|V|}(u_i) = f_i(u_i) \leq f_{i+1}(u_{i+1}) = f_{|V|}(u_{i+1})$ □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Biz: Legyen $u_iu_j \in E(G)$ a G egy tetsz. éle. Ha $i > j$, akkor (2) miatt $f_{|V|}(u_i) \geq f_{|V|}(u_j)$, ezért az u_iu_j mentén történő javítás nem tudja $f_{|V|}(u_j)$-t csökkenteni, hisz $\ell(u_iu_j)$ pozitív.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Biz: Legyen $u_iu_j \in E(G)$ a G egy tetsz. éle. Ha $i > j$, akkor (2) miatt $f_{|V|}(u_i) \geq f_{|V|}(u_j)$, ezért az u_iu_j mentén történő javítás nem tudja $f_{|V|}(u_j)$-t csökkenteni, hisz $\ell(u_iu_j)$ pozitív.
Ha pedig $i < j$, akkor az i-dik fázisban megtörtént az u_iu_j mentén történő javítás, és ezt követően $f(u_i)$ nem változott, azaz $f_{|V|}(u_i) = f_i(u_i)$. Az $f(u_j)$ becslés viszont tovább csökkenhetett a későbbi élmenti javítások során: $f_{|V|}(u_j) \leq f_i(u_j)$. Ezért az u_iu_j él mentén sem az i-dik fázisban, sem később nem lehetséges érdemi javítás. □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Biz: A Dijkstra-algoritmus az f_0 triviális (r, ℓ)-fb-ből indul ki, és élmenti javításokat alkalmaz. Így minden f_i (speciálisan $f_{|V|}$ is) (r, ℓ)-fb lesz. A fenti (3)-as megfigyelés miatt $f_{|V|}$-n nem végezhető érdemi élmenti javítás. Ezért egy korábbi (2)-es megfigyelés miatt $f_{|V|}$ pontos (r, ℓ)-fb, azaz $f_{|V|}(v) = dist_\ell(r, v)$ $\forall v \in V(G)$. □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$ esetén.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$.
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

„Lépésszámanalízis”: Ha a G gráfnak n csúcsa és m éle van, akkor a Dijkstra-algoritmus n-szer keresi meg egy legfeljebb n elemből álló számhalmaz minimumát. Ez összességében legfeljebb $konst \cdot n^2$ lépést igényel. Ezen kívül legfeljebb m élmenti javítás történik, ami $konst' \cdot m$ lépés. Összességében tehát legfeljebb $konst'' \cdot (n^2 + m)$ lépésre van szükség, az algoritmus hatékony.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

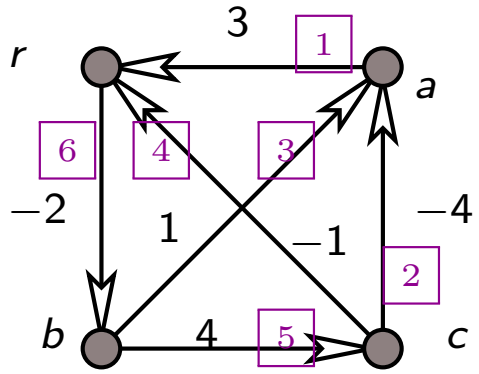
Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1



	r	a	b	c	
f_0	0	∞	∞	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

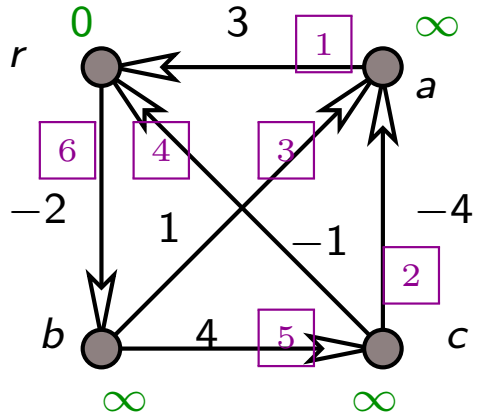
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

1. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

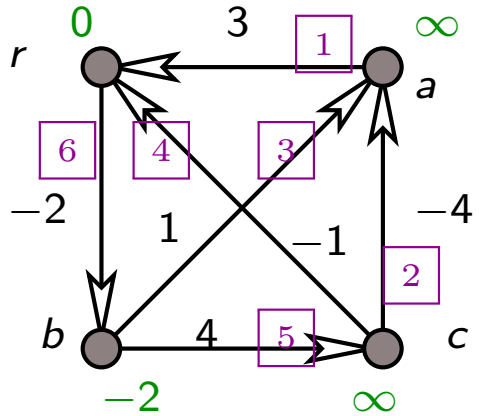
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

1. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

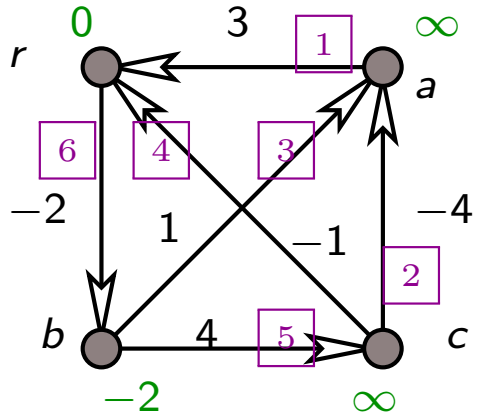
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

2. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

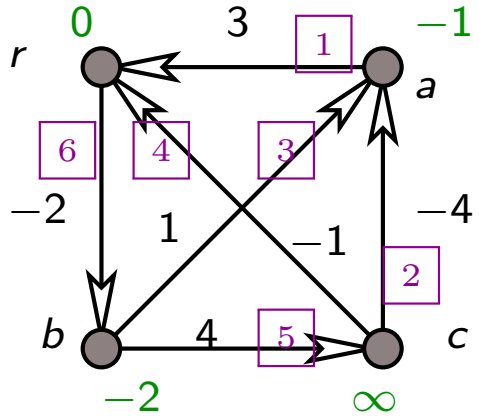
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

2. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

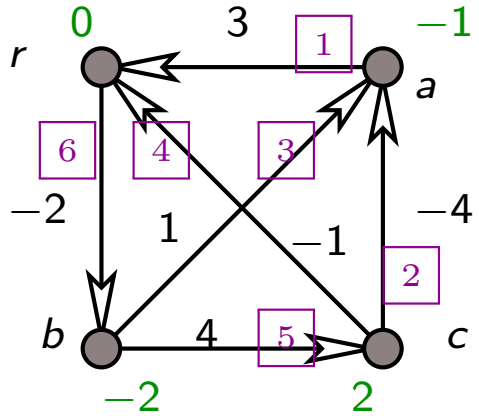
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

2. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

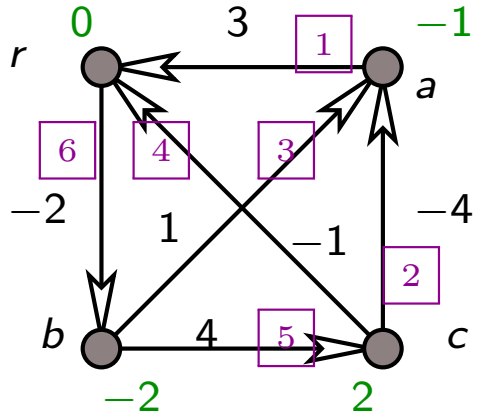
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

2. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

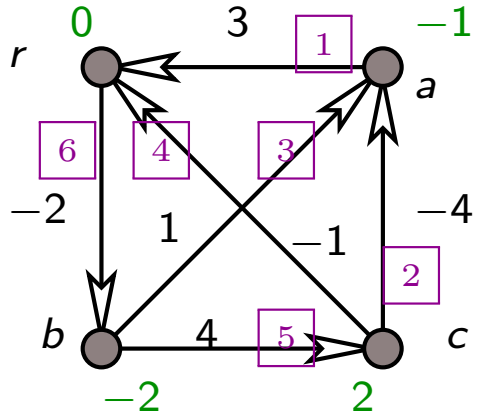
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

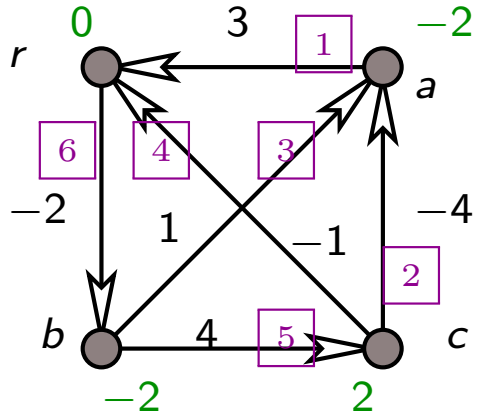
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

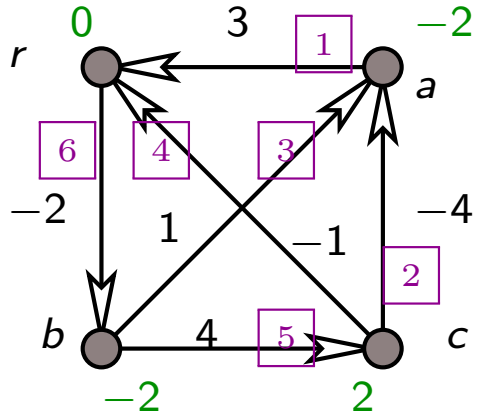
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

A Dijkstra-algoritmus negatív élhosszok mellett hibás eredményt adhat.
Azonban konzervatív hosszfüggvény esetén is igaz, hogy

- (r, ℓ)-fb élmenti javítása (r, ℓ)-fb-t eredményez, ill.
- ha egy (r, ℓ)-fb-en nem végezhető érdemi émj, akkor pontos.

Konzervatív hosszfv esetén is hasonló a stratégiát követünk:

Addig végzünk érdemi émj-okat a triviális (r, ℓ)-fb-en, míg lehet.

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

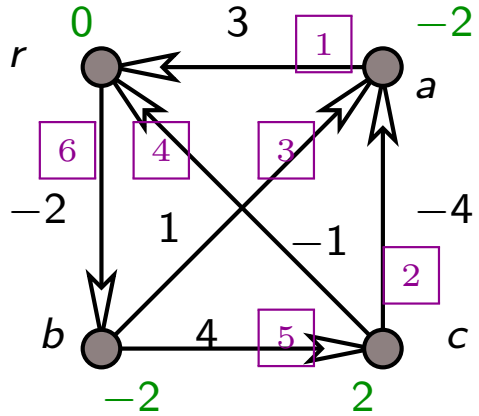
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	
f_3	0	-2	-2	2	

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

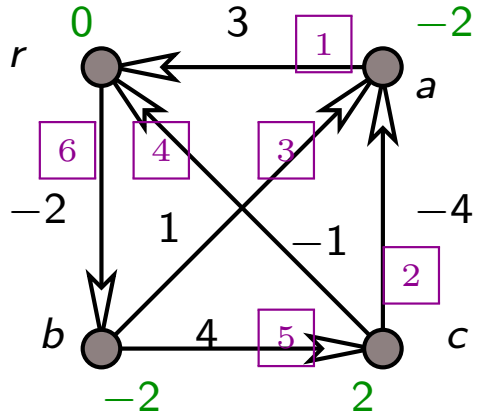
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	
f_3	0	-2	-2	2	

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

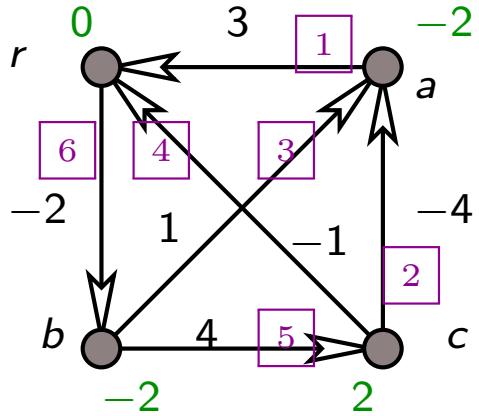
f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

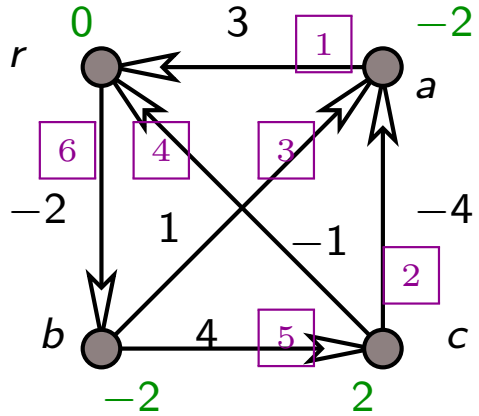
OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Biz: $f_1(v) = dist_\ell(r, v)$ ha $\exists \leq 1$-élű legrövidebb rv-út.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	
f_3	0	-2	-2	2	

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

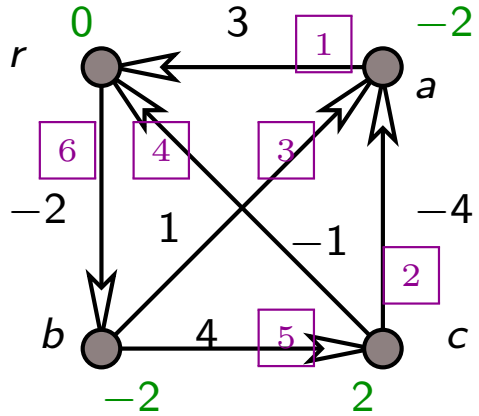
Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Biz: $f_1(v) = dist_\ell(r, v)$ ha $\exists \leq 1$-élű legrövidebb rv-út.

$f_2(v) = dist_\ell(r, v)$ ha $\exists \leq 2$-élű legrövidebb rv-út.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

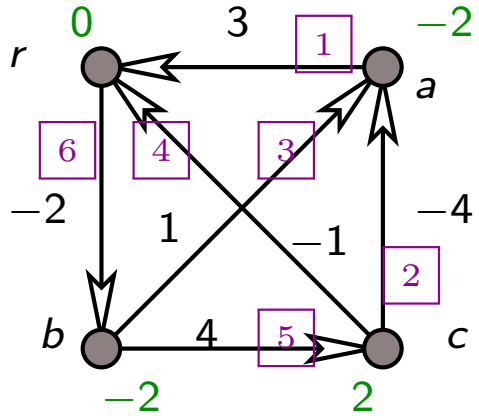
Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Biz: $f_1(v) = dist_\ell(r, v)$ ha $\exists \leq 1$-élű legrövidebb rv-út.

$f_2(v) = dist_\ell(r, v)$ ha $\exists \leq 2$-élű legrövidebb rv-út. s.í.t

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

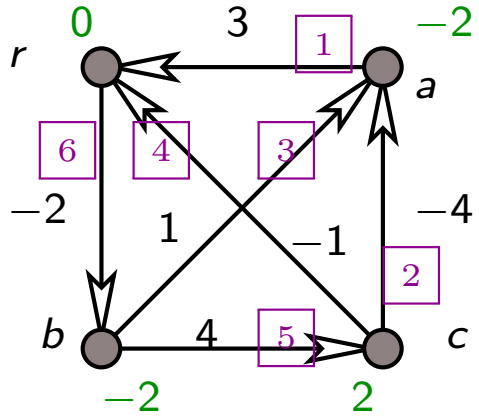
Biz: $f_1(v) = dist_\ell(r, v)$ ha $\exists \leq 1$-élű legrövidebb rv-út.

$f_2(v) = dist_\ell(r, v)$ ha $\exists \leq 2$-élű legrövidebb rv-út. s.í.t

$f_{n-1}(v) = dist_\ell(r, v)$ ha $\exists \leq (n-1)$-élű legrövidebb rv-út.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Biz: $f_1(v) = dist_\ell(r, v)$ ha $\exists \leq 1$-élű legrövidebb rv-út.

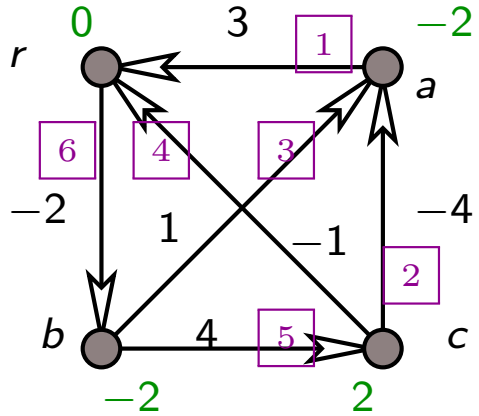
$f_2(v) = dist_\ell(r, v)$ ha $\exists \leq 2$-élű legrövidebb rv-út. s.í.t

$f_{n-1}(v) = dist_\ell(r, v)$ ha $\exists \leq (n-1)$-élű legrövidebb rv-út.

Tehát $f_{n-1}(v) = dist_\ell(r, v)$ $\forall v \in V$. $\square$

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c	
f_0	0	∞	∞	∞	
f_1	0	∞	-2	∞	
f_2	0	-1	-2	2	
f_3	0	-2	-2	2	

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

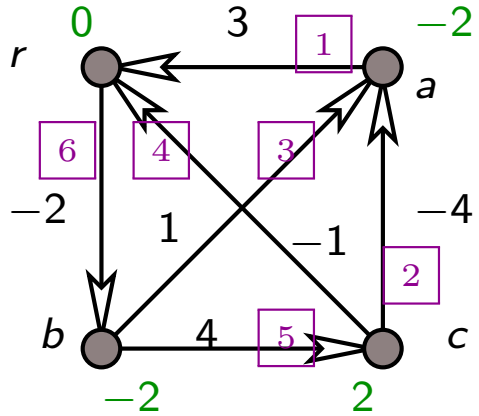
f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

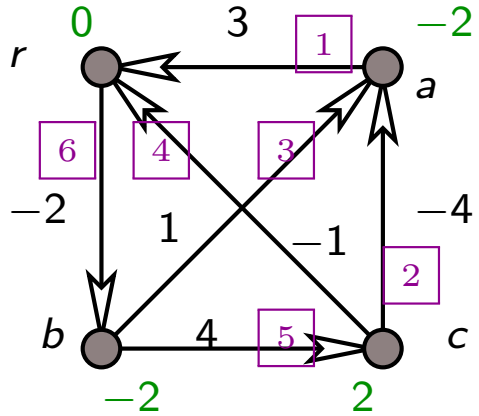
OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Megf: Ha $f_i = f_{i-1}$, akkor a Ford-algoritmust az i-dik fázis után be lehet fejezni, hisz onnantól nincs több érdemi élmenti javítás, így $f_{n-1} = f_i$.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

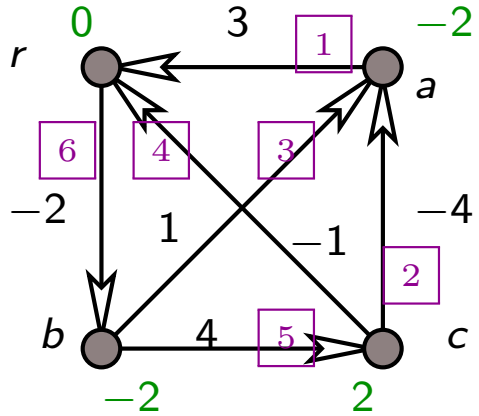
Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Megf: Ha $f_i = f_{i-1}$, akkor a Ford-algoritmust az i-dik fázis után be lehet fejezni, hisz onnantól nincs több érdemi élmenti javítás, így $f_{n-1} = f_i$.

Megj: Az $f_{n-1}(v)$-t beállító élek legrövidebb utak fáját alkotják.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$, $r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

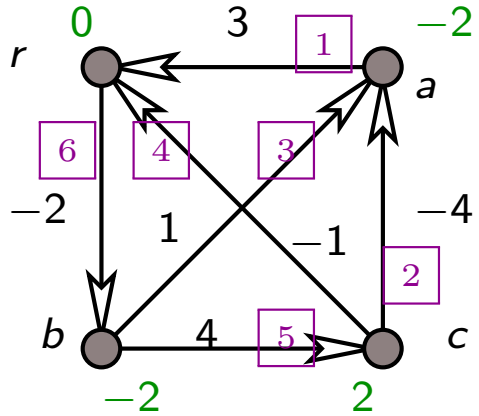
Megf: Ha $f_i = f_{i-1}$, akkor a Ford-algoritmust az i-dik fázis után be lehet fejezni, hisz onnantól nincs több érdemi élmenti javítás, így $f_{n-1} = f_i$.

Megj: Az $f_{n-1}(v)$-t beállító élek legrövidebb utak fáját alkotják.

Biz: A Dijkstra esethez hasonló. Tetsz. v csúcsból visszafelé követve a végső értékeket beállító éleket, egy $f_{n-1}(v)$ hosszúságú rv-utat találunk. □

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

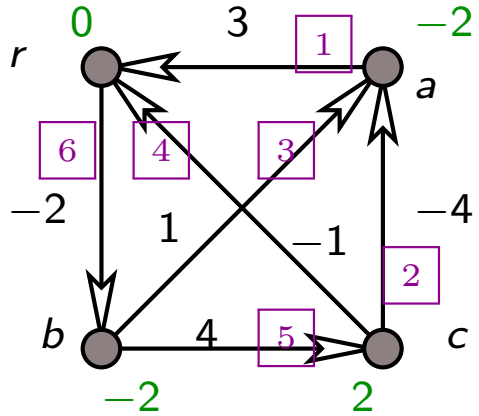
Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Megf: Ha $f_i = f_{i-1}$, akkor a Ford-algoritmust az i-dik fázis után be lehet fejezni, hisz onnantól nincs több érdemi élmenti javítás, így $f_{n-1} = f_i$.

Megj: Az $f_{n-1}(v)$-t beállító élek legrövidebb utak fáját alkotják.

Legrövidebb utak konzervatív hosszfüggvény esetén 1

3. fázis



	r	a	b	c
f_0	0	∞	∞	∞
f_1	0	∞	-2	∞
f_2	0	-1	-2	2
f_3	0	-2	-2	2

Ford-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}$,$r \in V$.

Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: f_0 a triv. (r, ℓ)-fb, $|V| = n$, $E = \{e_1, e_2, \ldots, e_m\}$. Az i-dik fázis $i = 1, 2, \ldots, n-1$-re az alábbi.

f_i-t f_{i-1}-ből kapjuk, az $e_1, \ldots, e_m$ élmenti javítások után.

OUTPUT: $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

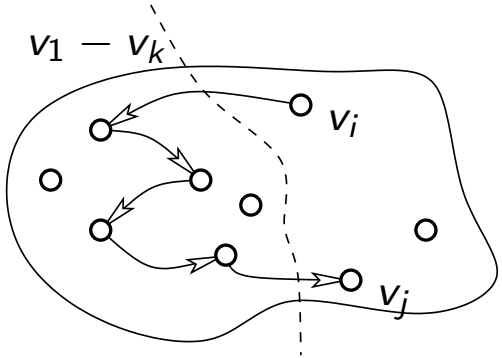
Állítás: Ha ℓ konzervatív, akkor $dist_\ell(r, v) = f_{n-1}(v)$ $\forall v \in V$.

Megf: Ha $f_i = f_{i-1}$, akkor a Ford-algoritmust az i-dik fázis után be lehet fejezni, hisz onnantól nincs több érdemi élmenti javítás, így $f_{n-1} = f_i$.

Megj: Az $f_{n-1}(v)$-t beállító élek legrövidebb utak fáját alkotják.

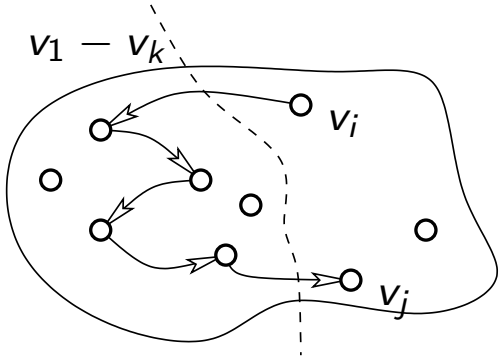
„Lépésszámanalízis": Ha a $|V(G)| = n$ és $|E(G)| = m$, akkor minden fázisban $\leq m$ émj, ami $konst \cdot m$ lépés. Ez összesen $\leq konst \cdot (n-1) \cdot m \leq konst \cdot n^3$ lépés, az algoritmus hatékony.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

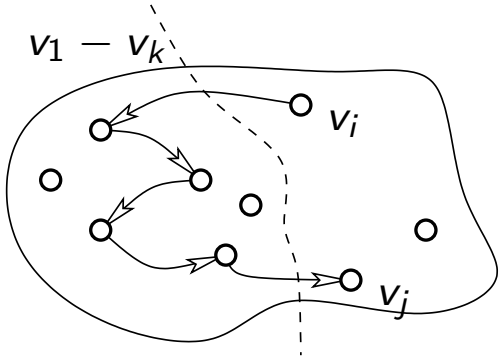
Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$.

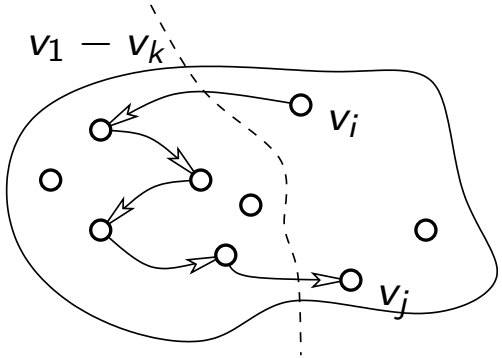
Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i, i) = 0$,

Legrövidebb utak konzervatív hosszfüggvény esetén 2

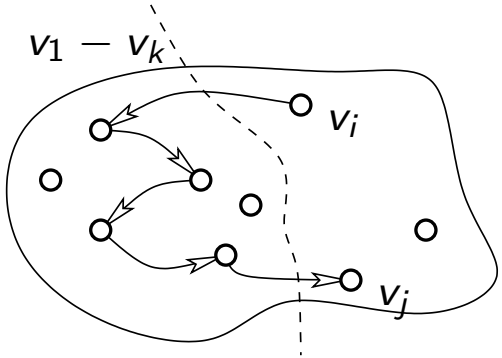


Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i, i) = 0$,

$v_i v_j \in E \Rightarrow d^{(0)}(i, j) = \ell(v_i v_j)$,

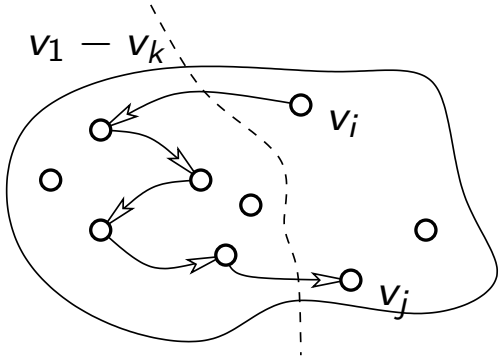
Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i,j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i,j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i,i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i,j) = \ell(v_i v_j)$, különben $d^{(0)}(i,j) = \infty$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2

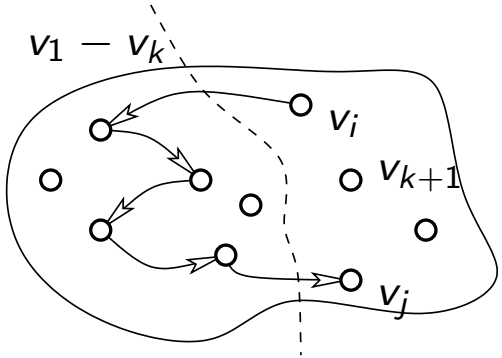


Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i, i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i, j) = \ell(v_i v_j)$, különben $d^{(0)}(i, j) = \infty$.

(3) Ha ℓ konzervatív, akkor tetsz. i, j ill. $k \leq n$ esetén $d^{(k+1)}(i, j) = \min\{d^{(k)}(i, j), d^{(k)}(i, k + 1) + d^{(k)}(k + 1, j)\}$ teljesül.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i,j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

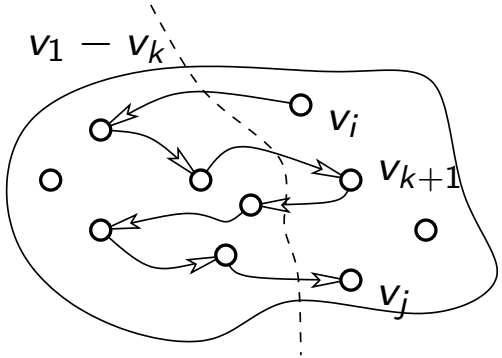
Megf: (1) $d^{(n)}(i,j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i,i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i,j) = \ell(v_i v_j)$, különben $d^{(0)}(i,j) = \infty$.

(3) Ha ℓ konzervatív, akkor tetsz. i, j ill. $k \leq n$ esetén $d^{(k+1)}(i,j) = \min\{d^{(k)}(i,j), d^{(k)}(i,k+1) + d^{(k)}(k+1,j)\}$ teljesül.

Biz: Tekintsünk egy $d^{(k+1)}(i,j)$-t meghatározó P utat.

I. eset: $v_{k+1} \notin P$. Ekkor $d^{(k+1)}(i,j) = d^{(k)}(i,j)$, és $d^{(k+1)}(i,j) \leq d^{(k)}(i,k+1) + d^{(k)}(k+1,j)$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i,j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i,j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i,i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i,j) = \ell(v_i v_j)$, különben $d^{(0)}(i,j) = \infty$.

(3) Ha ℓ konzervatív, akkor tetsz. i, j ill. $k \leq n$ esetén $d^{(k+1)}(i,j) = \min\{d^{(k)}(i,j), d^{(k)}(i,k+1) + d^{(k)}(k+1,j)\}$ teljesül.

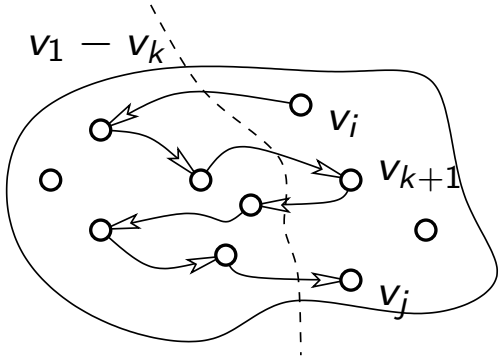
Biz: Tekintsünk egy $d^{(k+1)}(i,j)$-t meghatározó P utat.

I. eset: $v_{k+1} \notin P$. Ekkor $d^{(k+1)}(i,j) = d^{(k)}(i,j)$, és $d^{(k+1)}(i,j) \leq d^{(k)}(i,k+1) + d^{(k)}(k+1,j)$.

II. eset: $v_{k+1} \in P$. Ekkor $d^{(k+1)}(i,j) \leq d^{(k)}(i,j)$, és $d^{(k+1)}(i,j) = d^{(k)}(i,k+1) + d^{(k)}(k+1,j)$.

Mindkét esetben helyes a képlet. □

Legrövidebb utak konzervatív hosszfüggvény esetén 2

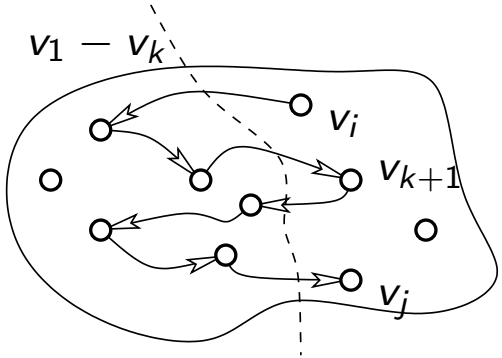


Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i, i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i, j) = \ell(v_i v_j)$, különben $d^{(0)}(i, j) = \infty$.

(3) Ha ℓ konzervatív, akkor tetsz. i, j ill. $k \leq n$ esetén $d^{(k+1)}(i, j) = \min\{d^{(k)}(i, j), d^{(k)}(i, k + 1) + d^{(k)}(k + 1, j)\}$ teljesül.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



Tfh $G = (V, E)$, $\ell : E \to \mathbb{R}$ és $V = \{v_1, v_2, \ldots, v_n\}$. Jelölje $d^{(k)}(i, j)$ a legrövidebb olyan $v_i v_j$-út hosszát, aminek belső csúcsai csak $v_1, v_2, \ldots, v_k$ lehetnek.

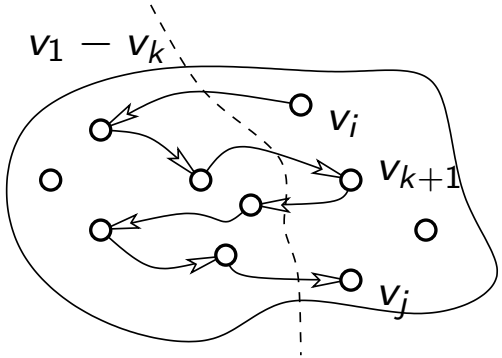
Megf: (1) $d^{(n)}(i, j) = dist_\ell(v_i, v_j)$. (2) $d^{(0)}(i, i) = 0$, $v_i v_j \in E \Rightarrow d^{(0)}(i, j) = \ell(v_i v_j)$, különben $d^{(0)}(i, j) = \infty$.

(3) Ha ℓ konzervatív, akkor tetsz. i, j ill. $k \leq n$ esetén $d^{(k+1)}(i, j) = \min\{d^{(k)}(i, j), d^{(k)}(i, k+1) + d^{(k)}(k+1, j)\}$ teljesül.

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.

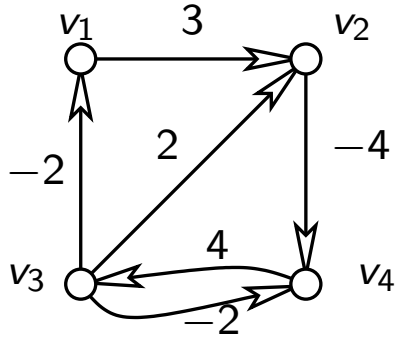
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



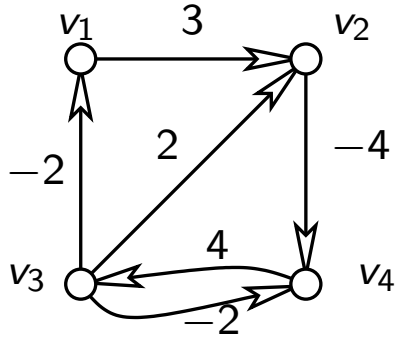
Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

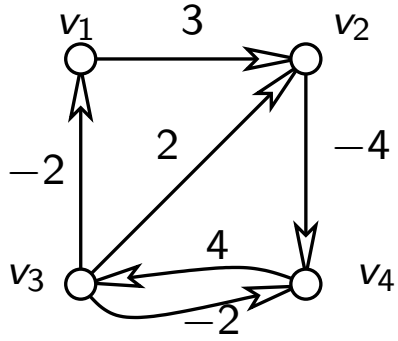
Legrövidebb utak konzervatív hosszfüggvény esetén 2



$d^{(0)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	∞
v_2	∞	0	∞	-4
v_3	-2	2	0	-2
v_4	∞	∞	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2

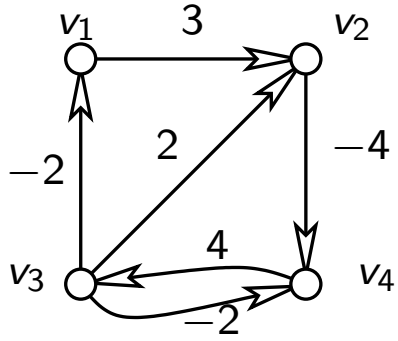


$d^{(0)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	∞
v_2	∞	0	∞	-4
v_3	-2	2	0	-2
v_4	∞	∞	4	0

$d^{(1)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	∞
v_2	∞	0	∞	-4
v_3	-2	1	0	-2
v_4	∞	∞	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

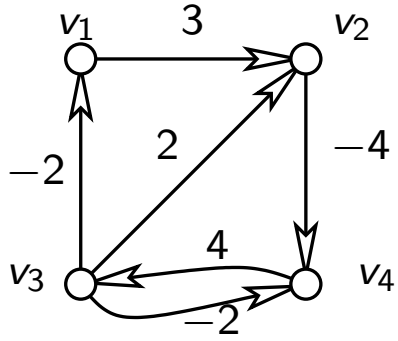
Legrövidebb utak konzervatív hosszfüggvény esetén 2



$d^{(1)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	∞
v_2	∞	0	∞	-4
v_3	-2	1	0	-2
v_4	∞	∞	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2

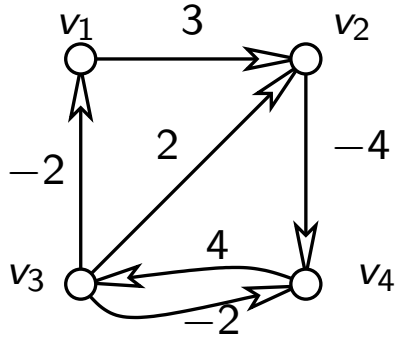


$d^{(1)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	∞
v_2	∞	0	∞	-4
v_3	-2	1	0	-2
v_4	∞	∞	4	0

$d^{(2)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	∞	∞	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

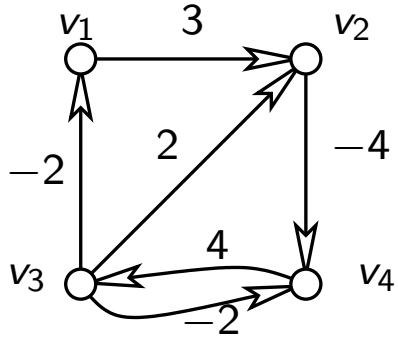
Legrövidebb utak konzervatív hosszfüggvény esetén 2



$d^{(2)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	∞	∞	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2

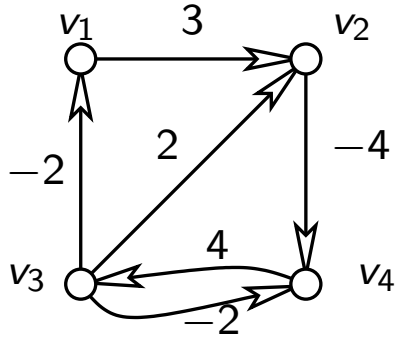


$d^{(2)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	∞	∞	4	0

$d^{(3)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

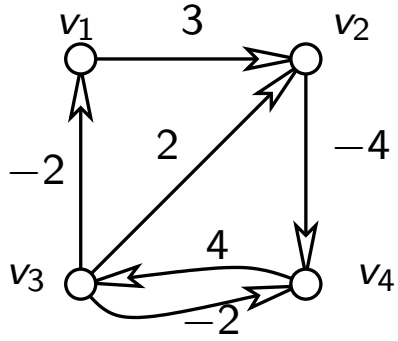
Legrövidebb utak konzervatív hosszfüggvény esetén 2



$d^{(3)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2

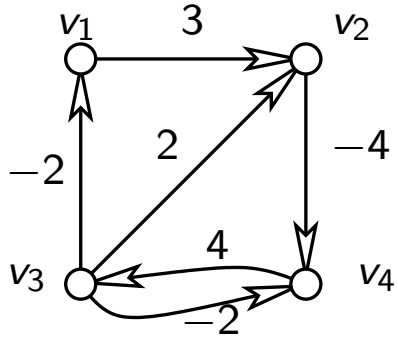


$d^{(3)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	−1
v_2	∞	0	∞	−4
v_3	−2	1	0	−3
v_4	2	5	4	0

$d^{(4)}$	v_1	v_2	v_3	v_4
v_1	0	3	3	−1
v_2	−2	0	0	−4
v_3	−2	1	0	−3
v_4	2	5	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



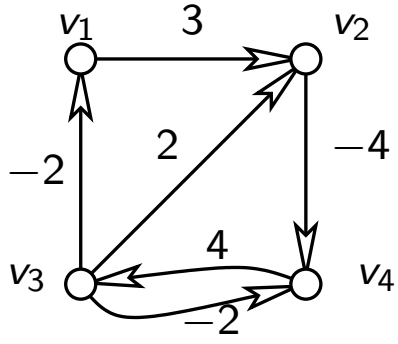
$d^{(3)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

$d^{(4)}$	v_1	v_2	v_3	v_4
v_1	0	3	3	-1
v_2	-2	0	0	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

„Lépésszámanalízis”: A $d^{(0)}$ felírása $konst \cdot n^2$ lépés. Minden fázis $konst' \cdot n^2$. Mivel összesen n fázis van, a lépésszám legfeljebb $konst'' \cdot n^3$ lépés, az algoritmus hatékony.

Legrövidebb utak konzervatív hosszfüggvény esetén 2



$d^{(3)}$	v_1	v_2	v_3	v_4
v_1	0	3	∞	-1
v_2	∞	0	∞	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

$d^{(4)}$	v_1	v_2	v_3	v_4
v_1	0	3	3	-1
v_2	-2	0	0	-4
v_3	-2	1	0	-3
v_4	2	5	4	0

Floyd-algoritmus: Input: $G = (V, E)$, konzervatív $\ell : E \to \mathbb{R}$.
Output: $dist_\ell(u, v)$ $\forall u, v \in V$ Működés: $d^{(0)}$ felírása (2) alapján. Az i-dik fázis: $d^{(i-1)}$-ből meghatározzuk $d^{(i)}$-t (3) alapján. OUTPUT: $d^{(n)}(u, v) = dist_\ell(u, v)$ $\forall u, v \in V$.

„Lépésszámanalízis”: A $d^{(0)}$ felírása $konst \cdot n^2$ lépés. Minden fázis $konst' \cdot n^2$. Mivel összesen n fázis van, a lépésszám legfeljebb $konst'' \cdot n^3$ lépés, az algoritmus hatékony.

Ford vs Floyd: Konzervatív hosszfüggvényre működnek helyesen.
Mindkét algoritmus észreveszi, ha ℓ nem konzervatív. (!!)
Legrövidebb utak is találhatók ezekkel az algoritmusokkal: Ford a gyökérből bárhová, Floyd pedig bmely két csúcs között. (!!)
A Ford ritka gráfokra jelentősen olcsóbb, sok él esetén a Floyd nem sokkal drágább.

Mit tanultunk ma?

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)
- Legrövidebb út keresése (r, ℓ)-fb élmenti javításával, Dijkstra

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)
- Legrövidebb út keresése (r, ℓ)-fb élmenti javításával, Dijkstra
- Ford- és Floyd-algoritmus konzervatív hosszfv esetén

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)
- Legrövidebb út keresése (r, ℓ)-fb élmenti javításával, Dijkstra
- Ford- és Floyd-algoritmus konzervatív hosszfv esetén

Köszönöm a figyelmet!

Kínzó kérdések

Kínzó kérdések

- Mi a tehergépkocsi?

Kínzó kérdések

- Mi a tehergépkocsi?
- 1/1975. (II. 5.) KPM–BM együttes rendelet a közúti közlekedés szabályairól
  Tehergépkocsi: a személygépkocsit, az autóbuszt, a trolibuszt és a vontatót kivéve minden gépkocsi.

Kínzó kérdések

- Mi a tehergépkocsi?
- 1/1975. (II. 5.) KPM–BM együttes rendelet a közúti közlekedés szabályairól
  Tehergépkocsi: a személygépkocsit, az autóbuszt, a trolibuszt és a vontatót kivéve minden gépkocsi.
- Hogyan lehet a BFS algoritmust felhasználni adott r gyökér esetén az összes $dist_\ell(r, v)$ távolság meghatározására, ha minden $\ell(e)$ élhossz $1, 2$ vagy 3?

Kínzó kérdések

- Mi a tehergépkocsi?
- 1/1975. (II. 5.) KPM–BM együttes rendelet a közúti közlekedés szabályairól
  Tehergépkocsi: a személygépkocsit, az autóbuszt, a trolibuszt és a vontatót kivéve minden gépkocsi.
- Hogyan lehet a BFS algoritmust felhasználni adott r gyökér esetén az összes $dist_\ell(r, v)$ távolság meghatározására, ha minden $\ell(e)$ élhossz $1, 2$ vagy 3?
- Tervezzünk csavaranyákból és cukorspárgából olyan gravitációs elven működő mechanikus számítógépet, ami alkalmas az inputként megadott, nemnegatív élhosszokkal rendelkező irányítatlan gráf tetszőleges gyökérpontjából a többi csúcs távolságának a meghatározására.