Algoritmuselmélet

Dijkstra algoritmusa

Katona Gyula Y. / Vizer Máté

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Legrövidebb utak élsúlyozott gráfban

Volt már korábban:
Legrövidebb élsorozatok egy adott pontból, ha nincs negatív kör:

Input: $G = (V, E, c)$, (súlyozott gráf), $s \in V$
(irányított vagy irányítatlan gráf is lehet)

Output: A legrövidebb (legkisebb súlyú) élsorozat és annak hossza s-ből a gráf minden más pontjába.

- Ha minden súly 1 $\implies$ BFS
  Lépésszám: $O(n^2)$ mátrixszal, $O(n+m)$ éllistával
- Ha a gráf DAG $\implies$ Dinamikus programozás
  Lépésszám: $O(n^2)$ mátrixszal, $O(n+m)$ éllistával
- Súlyozott gráf, ha lehet negatív él is $\implies$ Bellman-Ford algoritmus lesz BSZ2-n
  Lépésszám: $O(n^3)$ mátrixszal, $O(n(n+m))$ éllistával

A legrövidebb utak problémája

Most legyen minden súly nemnegatív, azaz $c(e) \geq 0$ minden $e \in E$ esetén. Így nem lehetnek negatív körök.

Erre a speciális esetre Dijkstra algoritmusa az általános esetnél hatékonyabban megoldja a feladatot.

Edsger Wybe Dijkstra (1930–2002)
holland matematikus, informatikus.
Nevének kiejtése: Dájksztra
1959-ben publikálta algoritmusát.

Mi most az irányított esetet tárgyaljuk, az irányítatlan eset lényegében ugyanúgy működik.

A legrövidebb utak irányított gráfban

Legrövidebb $u \rightsquigarrow v$ út: egy olyan $u \rightsquigarrow v$ út, melynek a hossza minimális a G-beli $u \rightsquigarrow v$ utak között.
u és v csúcsok (G-beli) $d(u, v)$ *távolsága*:
— 0, ha $u = v$;
— ∞, ha nincs $u \rightsquigarrow v$ út
— egyébként pedig a legrövidebb $u \rightsquigarrow v$ út hossza.

Vigyázat, irányított gráfban u és v nem felcserélhető: ha az egyik csúcs valamilyen távol van a másiktól, akkor nem biztos, hogy a másik is ugyanolyan távol van az egyiktől!

Dijkstra módszere

Feladat

A legrövidebb utak problémája (egy forrásból):
Adott egy $G = (V, E)$ irányított gráf, a $c : E \to \mathbb{R}^+$ nemnegatív értékű súlyfüggvény és egy $s \in V$ csúcs (a forrás). Határozzuk meg minden $v \in V$-re a $d(s, v)$ távolságot.

Dijkstra módszere

Az algoritmus futtatása során két dolgot tartunk nyilván:

- $D[1 : n]$:
  - Egy, a G csúcsaival indexelt tömb
  - az eljárás során addig megismert legrövidebb $s \rightsquigarrow v$ utak hossza
  - $D[v]$ végig felső közelítése a keresett $d(s, v)$ távolságnak, az addig megtalált legrövidebb út hossza. Lehet, hogy $D[v] > d(s, v)$, ha az eddig megtalált legrövidebb út nem a legrövidebb út.
  - A közelítést lépésről lépésre finomítjuk, végül elérjük $d(s, v)$-t
- KÉSZ:
  - G csúcsainak egy részhalmaza
  - Azok a csúcsok, amikre már biztosan tudjuk, hogy megtaláltuk a legrövidebb utat, ezekre $D[v] = d(s, v)$
  - Kezdetben KÉSZ $= \{s\}$, hiszen ekkor még csak azt tudjuk, hogy $d(s, s) = 0$
  - Amikor egy újabb csúcsba megtaláljuk a legrövidebb utat és már tudjuk, hogy ez a legrövidebb, akkor hozzávesszük a KÉSZ halmazhoz.
  - A KÉSZ halmaz folyamatosan növekszik.
  - Az algoritmus akkor fog véget érni, ha minden olyan csúcs belekerült a KÉSZ halmazba, ahova vezet irányított út s-ből.

Dijkstra módszere

Tegyük fel, hogy a G gráf az alábbi alakú C szomszédossági mátrixával adott:

$$C[v,w] = \begin{cases} 0 & \text{ha } v = w, \\ c(v,w) & \text{ha } v \neq w \text{ és } (v,w) \text{ éle } G\text{-nek}, \\ \infty & \text{különben.} \end{cases}$$

Kezdeti beállítások:

- $D[v] := C[s,v]$ minden $v \in V$ csúcsra
- KÉSZ $:= \{s\}$

Dijkstra módszere, az első lépések

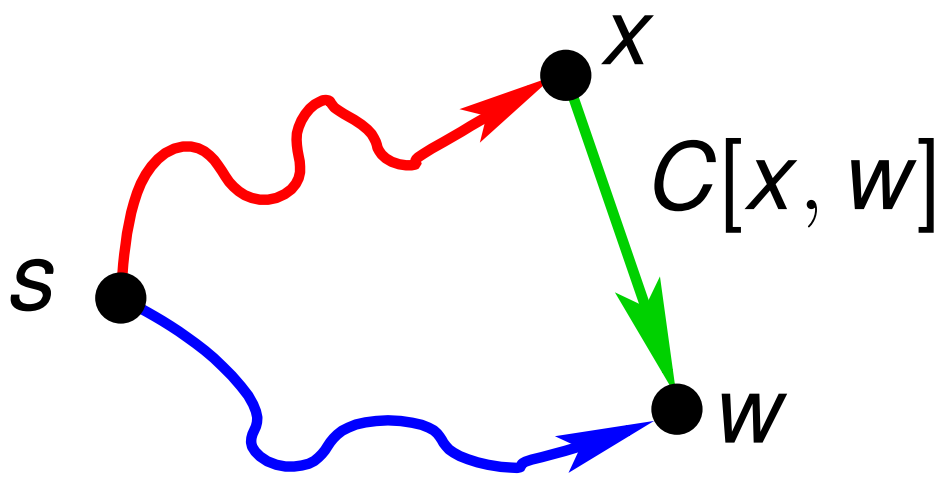
Kezdetben $D[x]$ csak akkor nem 0 és nem ∞, ha van él s-ből x-be. Ilyenkor $D[x]$ az oda mutató él súlya.

Válasszuk ki az s csúcs szomszédai közül a hozzá legközelebbit, vagyis egy olyan $x \in V \setminus \{s\}$ csúcsot, melyre $D[x]$ minimális.

Biztos, hogy így az egyetlen (s, x) élből álló út egy legrövidebb $s \rightsquigarrow x$ út, hiszen ha másik élen indulna az x-be vezető út, (az élsúlyok nemnegatívak!) akkor az nem lehet rövidebb.
$\Longrightarrow$ x-et betehetjük (s mellé) a KÉSZ halmazba.

Dijkstra módszere

Ezek után módosítsuk a többi csúcs $D[w]$ értékét, ha az eddig ismertnél rövidebb úton el lehet érni oda x-en keresztül, azaz ha $D[x] + C[x, w] < D[w]$.



Újra válasszunk ki a $v \in V \setminus$ KÉSZ csúcsok közül egy olyat, amelyre $D[v]$ minimális.
Ezen csúcs $D[\,]$-értéke már az s-től való távolságát tartalmazza.
Majd megint a most beválasztott v csúcs szomszédainál a $D[\,]$-értékeket módosítjuk, és így tovább, míg minden elérhető csúcs be nem kerül a KÉSZ halmazba.

Dijkstra algoritmusa szomszédossági mátrixszal

```
1:  procedure DIJKSTRA(G = (V, E, c), s)
2:      KÉSZ := {s}
3:      for all v ∈ V csúcsra do
4:          D[v] := C[s, v]          ▷ (a d(s, v) távolság első közelítése)
5:      end for
6:      while van olyan x ∈ V \ KÉSZ, amire D[x] ≠ ∞ do
7:          Válasszunk olyan x ∈ V \ KÉSZ csúcsot,
            melyre D[x] minimális.
8:          KÉSZ = KÉSZ ∪ {x}          ▷ Tegyük x-et a KÉSZ-be.
9:          for  x minden V \ KÉSZ-ben lévő w szomszédjára do
10:             D[w] := min{D[w], D[x] + C[x, w]}
                                        ▷ (d(s, w) új közelítése)
11:         end for
12:     end while
13: end procedure
```

Dijkstra algoritmusának lépésszáma

Lépésszám szomszédossági mátrixszal:

- (2-5): $O(n)$ (a mátrix egy sorának kiolvasása)
- (6-12:)
  - (6): a ciklus $O(n)$-szer fut le
  - A cikluson belül először egy minimum keresés (7:) $O(n)$, (8:) $O(1)$ majd a (9:) ciklus $O(n)$-szer
  - A (9:) cikluson belül (10:) konstans sok lépés
- (6-12:) $O(n^2)$
- Összesen: $O(n) + O(n^2) = O(n^2)$

Lépésszám szomszédossági listával (éllistával):

- (2-5): $O(n)$ (s listájának végigolvasása)
- (6-12:)
  - (6): a ciklus $O(n)$-szer fut le
  - A cikluson belül először egy minimum keresés (7:) $O(n)$, (8:) $O(1)$, majd a (9:) ciklus $O(d_{ki}(x))$-szer
  - A (9:) cikluson belül (10:) konstans sok lépés
- (6-12:) $\sum_{x \in V} [O(n) + O(1) + O(d_{ki}(x))] = O(n^2 + n + m) = O(n^2)$
- Összesen: $O(n) + O(n^2) = O(n^2)$

Dijkstra algoritmusának lépésszáma

Lehet-e gyorsabban?

Később megmutatjuk, hogy ha a kupac adatstruktúrát használjuk a $D[v]$ értékek tárolására, akkor

- adatstruktúra felépítése: $O(n)$
- minimum keresése és egyben törlése: $O(\log n)$
- adott érték módosítása a kupacban: $O(\log n)$

Lépésszám szomszédossági listával, kupaccal:

- (2-5): $O(n)$ (s listájának végigolvasása és kupac építés)
- (6-12:)
  - (6): a ciklus $O(n)$-szer fut le
  - A cikluson belül először egy minimum keresés (7:) $O(\log n)$, (8:) $O(1)$, majd a (9:) ciklus $O(d_{ki}(x))$-szer
  - A (9:) cikluson belül (10:) $O(\log n)$ lépés
- (6-12:)
  $\sum_{x \in V} [O(\log n) + O(1) + O(d_{ki}(x) \log n)] = O((n+m) \log n)$
- Összesen: $O(n) + O((n+m) \log n) = O((n+m) \log n)$

Dijkstra algoritmusának helyessége

Tétel

A KÉSZ halmaz pontjaira $D[v]$ a legrövidebb $s \rightsquigarrow v$ utak hossza, azaz $D[v] = d(s, v)$.

Ezt indukcióval bizonyítjuk. Ehhez egyrészt bevezetjük a következő definíciót, másrészt egy erősebb állítást bizonyítunk, hogy működjön az indukció.

Definíció

különleges út: egy $s \rightsquigarrow z$ irányított út különleges, ha a z végpontot kivéve minden pontja a KÉSZ halmazban van. A különleges úttal elérhető pontok éppen a KÉSZ-ből egyetlen éllel elérhető pontok.

Dijkstra algoritmusának helyessége

Tétel

A (6-12:) ciklus minden iterációs lépése után érvényesek a következők:

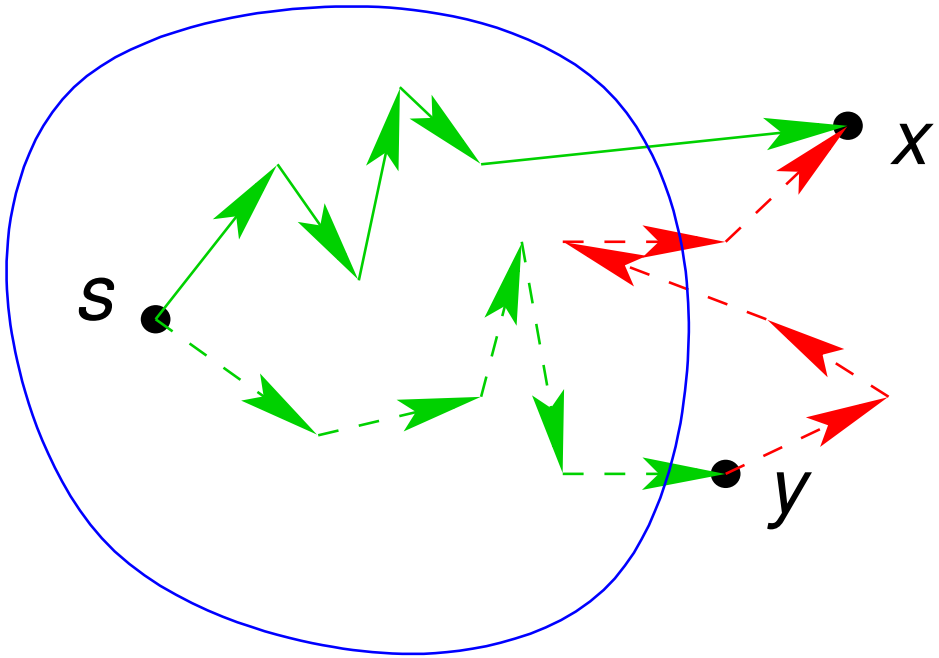
- *(a) KÉSZ pontjaira $D[v]$ a legrövidebb $s \rightsquigarrow v$ utak hossza.*
- *(b) Ha $v \in$ KÉSZ, akkor van olyan $d(s, v)$ hosszúságú (más szóval legrövidebb) $s \rightsquigarrow v$ út is, amelynek minden pontja a KÉSZ halmazban van.*
- *(c) Külső (vagyis $w \in V \setminus$ KÉSZ) pontokra $D[w]$ a legrövidebb különleges $s \rightsquigarrow w$ utak hossza.*

Bizonyítás:

Indukcióval (6:) előtt $\surd$

Tegyük fel, hogy igaz a j-edik iteráció után. Belátjuk, hogy igaz a $j+1$-edik iteráció után is.

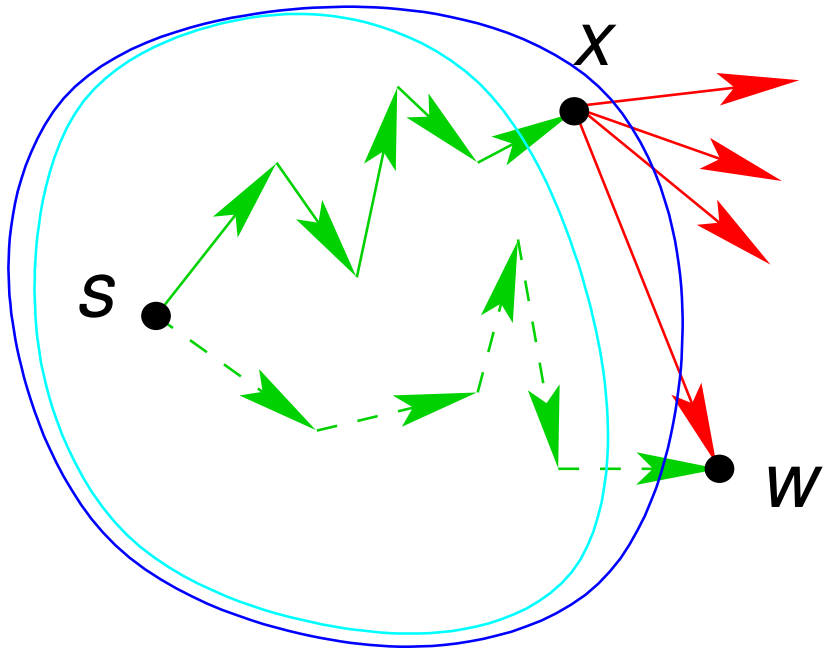
Tegyük fel, hogy az algoritmus a $(j+1)$. iterációs lépésben az x csúcsot választja a KÉSZ-be.



(a) Indirekt: mi van, ha $D[x]$ nem a $d(s,x)$ távolságot jelöli, azaz van ennél rövidebb $s \rightsquigarrow x$ út?
Ezen út „eleje" (az első olyan y-ig, ahol kilép a rövidebb út KÉSZ-ből) különleges, (c) miatt $\Longrightarrow$

Az út elejének hossza $D[y]$, mert az út eleje különleges. $D[y] \leq$ az egész út hossza, mert minden élsúly nemnegatív. Az indirekt feltevés szerint ez $< D[x] \implies D[y] < D[x]$ ↯

(b) Elég x-re $\Longleftarrow$ KÉSZ korábbi pontjaira az indukciós feltevésből (a KÉSZ halmaz mindig csak növekszik) $\surd$
Láttuk, hogy $d(s,x) = D[x]$, ez egy különleges $s \rightsquigarrow x$ út hossza volt a $(j+1)$. iteráció előtt (itt a (c)-re vonatkozó indukciós feltevést használtuk) annak végeztével az út minden pontja KÉSZ-beli lesz.



(c) A $(j+1)$. iteráció előtt

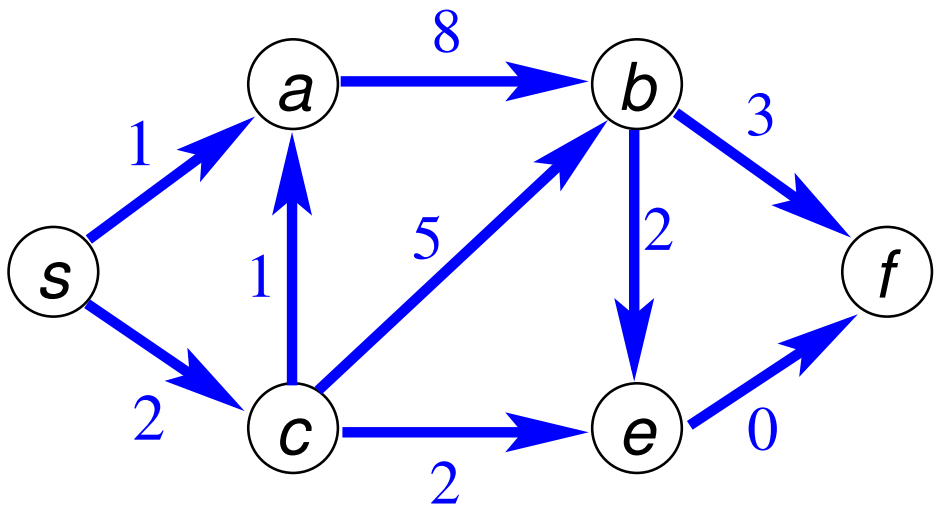
$$D[w] = \min_{v \in \text{KÉSZ} \setminus \{x\}} \{d(s,v) + C[v,w]\}.$$

Utána

$$D[w] = \min_{v \in \text{KÉSZ}} \{d(s,v) + C[v,w]\}.$$

$\Longrightarrow$ Elég megnézni, hogy $D[w]$ vagy $d(s,x) + C[x,w]$ nagyobb.

Példa



D	KÉSZ	a	b	c	e	f
0.	$\{s\}$	1	∞	2	∞	∞
1.	$\{s, a\}$	1	9	2	∞	∞
2.	$\{s, a, c\}$	1	7	2	4	∞
3.	$\{s, a, c, e\}$	1	7	2	4	4
4.	$\{s, a, c, e, f\}$	1	7	2	4	4
5.	$\{s, a, c, e, f, b\}$	1	7	2	4	4

A legrövidebb utak nyomon-követése

Minden pontra tárolunk és karbantartunk egy $P[x]$ csúcsot is, ami megadja egy, az eddig ismert hozzá vezető legrövidebb úton az utolsó előtti csúcsot.
Kezdetben $P[v] := s$ minden $v \in V$-re, ahova mutat él s-ből.
$\Longrightarrow$ (9-11:) ciklus belsejében, ha egy külső w csúcs $D[w]$ értékét megváltoztatjuk, akkor $P[w] := x$.
Lépésszám: $O(n^2)$

Dijkstra algoritmusa irányítatlan gráfokra: Kezelhetjük az irányítatlan gráfot úgy, mintha minden irányítatlan él helyett egy oda és egy visszafelé irányított élet veszünk ugyanolyan súllyal.