

# Algoritmuselmélet

## Bonyolultságelmélet

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék  
Budapesti Műszaki és Gazdaságtudományi Egyetem

# Mi az algoritmus?

A továbbiakhoz szükség volna egy pontos definícióra, hogy mit értünk algoritmus alatt.

Ilyen definíciót majd csak Msc-n a Nyelvek és automaták tárgyban adunk.

Helyette a továbbiakban úgy tekintjük, hogy azt tekintjük algoritmusnak, amit egy mai normál számítógéppel végre lehet hajtani. (pl. nem normál számítógép ebből a szempontból a quantum számítógép)

Az eddigi algoritmusok mind ilyenek voltak. De mikor arról beszélünk, hogy valamilyen problémára nincs algoritmus, akkor igazából arra gondolunk, hogy normál számítógépen nincs ilyen algoritmus.

# Algoritmusok hatékonysága

Algoritmus lépésszáma, ha  $n$  az input mérete:

- $\log n \implies$  remek
- $n \implies$  nagyon jó
- $n \log n \implies$  egész jó
- $n^2 \implies$  nem túl nagy  $n$ -re jó
- $n^3 \implies$  nem túl nagy  $n$ -re lehet jó
- $n^4 \implies$  nem túl nagy  $n$ -re néha jó
- $n^{10} \implies$  50 év múlva, nem túl nagy  $n$ -re lehet jó
- $n^{100} \implies$  1 millió év múlva lehet jó
- $2^n \implies$  nagyon kis  $n$ -re lehet jó, de nagy  $n$ -re sohasem

# Algoritmusok hatékonysága

Az eddig tanult algoritmusok általában hatékonyak voltak:

- $a + b$  kiszámítása: Input mérete:  $n = \log a + \log b$ ,  
Lépésszám:  $O(\log a + \log b) = O(n)$
- $ab$  kiszámítása: Input mérete:  $n = \log a + \log b$ ,  
Lépésszám:  $O((\log a + \log b) \log b) = O(n^2)$
- maradékos osztás, legnagyobb közös osztó:  
Input mérete:  $n = \log a + \log b$ , Lépésszám:  $O(n^2)$
- Gráfalgoritmusok (összefüggőség, legrövidebb út, párosítás, ...)  
Input mérete:  $n =$  éllista vagy szomszédossági mátrix mérete  
Lépésszám:  $O(n), O(n^2), O(n^3)$

# Algoritmusok hatékonysága

Van olyan algoritmikus feladat, amiről nem tudjuk, hogy ki lehet-e számolni hatékonyan:

- Később lesz szó ilyenekről.

Van olyan, amit nem lehet hatékonyan kiszámolni:

- $2^a$  kiszámítása: Input mérete:  $n = \log a$ ,  
Lépésszám  $\geq$  output mérete:  $\log(2^a) = a = \Omega(2^n)$
- általánosított sakk:  $n \times n$ -es táblán  $2n$  figura esetén adott állásból van-e nyerő stratégiája a kezdőnek?

Olyan is van, amire bizonyítható, hogy algoritmussal nem lehet kiszámolni!

- Msc-n a Nyelvek és automaták tárgyban lesz ilyen.  
(Rónyai-Ivanyos-Szabó könyv 7.8.3: Dominó probléma)

# Eldöntési problémák

Mostantól csak olyan típusú feladatokkal foglalkozunk, melyekre a válasz 1 bit: **IGEN** vagy **NEM**, ezek az úgynevezett *eldöntési problémák*.

- **PRÍM**

**Bemenet:**  $m > 0$  egész.

**Kérdés:**  $m$  prímszám?

- **ÚT**

**Bemenet:**  $G = (V, E)$  irányítatlan gráf és két kitüntetett csúcs,  $s, t \in V$ .

**Kérdés:** Van-e  $s$  és  $t$  között út a gráfban?

- **MINÚT**

**Bemenet:**  $G = (V, E)$  irányítatlan gráf és két kitüntetett csúcs,  $s, t \in V$ ,  $k$  egész.

**Kérdés:** Van-e  $s$  és  $t$  között legfeljebb  $k$  élű út a gráfban?

# Eldöntési problémák

Legtöbbször igaz, hogy egy optimalizálási feladat megoldható egy megfelelő eldöntési feladat algoritmusának néhány futtatásával.

Például:

Ha MINÚT-ra van egy gyors algoritmusunk, akkor bináris kereséssel  $O(\log m)$  db futtatással meghatározhatjuk a legrövidebb út hosszát.

Elég az eldöntési problémákat vizsgálni.

## Definíció

*Egy eldöntési problémához tartozó  $L$  nyelv azoknak a bemeneteknek a halmaza, amelyekre a válasz IGEN. A lehetséges bemeneteket (amelyek tehát vagy beletartoznak  $L$ -be vagy nem), szavaknak hívjuk. Egy  $X$  eldöntési probléma és  $x$  bemenet esetén  $x \in X$  jelöli, hogy az  $x$  bemenetre a válasz IGEN.*

# Polinom időben eldönthető problémák

## Definíció

Jelölje  $P$  azoknak az  $X$  eldöntési problémáknak a halmazát, amelyekhez van olyan  $A$  algoritmus, amely minden  $x$  bemenetre helyesen megválaszolja a kérdést, és az algoritmus lépésszáma **polinomiális**, azaz  $O(|x|^k)$  valamely  $k$  pozitív konstansra.

(Itt  $|x|$  az  $x$  bemenet hosszát jelöli,  $k$  független  $x$ -től.)

Jelölés:  $X \in P$ .

Az eddig tanult algoritmusok eldöntési változata  $P$ -ben van.

- Összefüggő-e egy adott  $G$  gráf?
- Van-e teljes párosítás egy adott  $G$  páros gráfban?
- Mekkora a legrövidebb út  $u$ -ból  $v$ -be?
- Mekkora legkisebb súlyú feszítőfa  $G$ -ben?



# Polinom időben eldönthető problémák

- Ha az input egy élsúlyozatlan gráf, akkor a korábban adott lépésszámok megfelelnek ennek a definíciónak.
- Ha az input több jegyű bináris vagy decimális számokat is tartalmaz, akkor jobban meg kell vizsgálni a lépésszámot.
- Ha az input súlyozott gráf, a súlyok binárisan vagy decimálisan adottak, a súlyokkal csak alapműveleteket végzünk, akkor az eddigi algoritmusok mind polinomiálisak.
- Ha az inputban a  $k$  szám bináris alakban van megadva (azaz mérete  $x = \log_2 k$ ) és az algoritmus  $k$  lépést végez, akkor ez nem polinomiális, hiszen  $k = 2^x$ .

A hátizsák feladatra adott dinamikus programozási algoritmus sem polinomiális, mert a lépésszám a hátizsák méretének függvénye, az inputban viszont ennek logaritmusa szerepel.

# Polinom időben eldönthető problémák

## PRÍM

**Bemenet:**  $m > 0$  egész szám bináris formában

**Kérdés:**  $m$  prímszám?

Az input mérete  $|x| = \log_2 m \implies m = 2^{|x|}$

- 1. algortimus:** Nézzük meg, hogy  $m$  osztható-e a  $2, 3, \dots, \sqrt{m}$  számok bármelyikével. **Lépésszám:**  $> \sqrt{m} = \sqrt{2^{|x|}} = 2^{|x|/2}$   
Ez nem polinomiális  $\not\Rightarrow$  PRÍM  $\in$  P
- 2. algortimus:** Fermat-teszt. **Lépésszám:**  $O(|x|^2 \log |x|)$ , polinomiális  
Nem mindig ad jó választ.  $\not\Rightarrow$  PRÍM  $\in$  P
- 3. algortimus:** AKS-teszt (Agrawal–Kayal–Saxena, 2002)  
**Lépésszám:**  $O(|x|^{12})$ , polinomiális  
Mindig jó választ ad.  $\implies$  PRÍM  $\in$  P

# Polinom időben eldönthető problémák

Ha belátjuk, hogy egy eldöntési probléma **P**-ben van, az nem feltétlen jelenti azt, hogy a gyakorlatban is tényleg hatékonyan megoldható. Ha a lépésszám  $|x|^{1000}$ , azzal nem megyünk sokra.

De ha belátnánk, hogy egy eldöntési probléma nincsen **P**-ben, akkor tudnánk, hogy arra nincs hatékony algoritmus normál számítógéppel.

## Probléma

*Tudunk-e mondani valami enyhébb követelményt, ami a **P**-beli problémákon kívül esetleg teljesül több nem **P**-beli problémára is?*

# Hatékony tanúsítvány

## Definíció

Azt mondjuk, hogy az  $X$  eldöntési problémához van **hatékony tanúsítvány**, ha van olyan  $\mathcal{T}$  algoritmus, melynek a bemenete  $(x, t)$  párokból áll, ahol  $x$  az  $X$  probléma egy lehetséges bemenete és a következő feltételek teljesülnek: léteznek olyan  $c$  és  $k$  pozitív konstansok, hogy

- ha  $x \in X$ , akkor van olyan  $t$ , aminek hossza  $|t| = O(|x|^c)$  és  $\mathcal{T}(x, t) = \text{IGEN}$ ,
- ha  $x \notin X$ , akkor nincs olyan  $t$ , aminek hossza  $|t| = O(|x|^c)$  és  $\mathcal{T}(x, t) = \text{IGEN}$ .
- A  $\mathcal{T}$  algoritmus polinomiális, azaz a lépésszáma  $O((|x| + |t|)^k)$ .

**Röviden:** Ha  $x$  bemenet esetén az eldöntési problémára a válasz **IGEN**, akkor erre van olyan polinom hosszú tanú ( $t$ ), amiről  $\mathcal{T}$ -vel polinom időben ellenőrizhető, hogy valóban **IGEN**. Ha a válasz **NEM**, akkor viszont nincs olyan rövid érvelés, amivel be lehet minket csapni.

# NP-beli problémák

## Definíció

Jelölje **NP** azoknak az eldöntési problémáknak a halmazát, amelyekre van hatékony tanúsítvány.

## Megjegyzés

Az **NP** a **nemdeterminisztikus polinom idő** rövidítése, ami arra utal, hogy  $t$  „megtalálása” nem feltétlenül determinisztikusan történik, egy  $x \in X$  esetén elég, ha megsejtjük, mi lesz jó. Viszont utána az ellenőrzés, hogy valóban jó a  $t$ , amit választottunk (kaptunk valakitől), már polinom időben megy.

# coNP-beli problémák

## Definíció

Egy  $X$  eldöntési probléma **komplementere** az az  $\bar{X}$ -sal jelölt probléma, melynek bemenete ugyanolyan mint az  $X$  esetén, de a válasz ellentétes. Azaz minden lehetséges  $x$  bemenetre

$$x \in \bar{X} \Leftrightarrow x \notin X.$$

Például: ÖSSZETETT =  $\overline{\text{PRÍM}}$

## Definíció

Jelölje **coNP** az **NP**-beli problémák **komplementereiből** álló halmazt, azaz  $X \in \text{coNP} \Leftrightarrow \bar{X} \in \text{NP}$ .

**Vagyis:** Az **NP**-beli problémák esetében a definíció szerint az **IGEN** válaszra van polinom hosszú, polinom időben ellenőrizhető bizonyíték, a **coNP**-beli problémák azok, ahol a **NEM** válaszra van polinom hosszú, polinom időben ellenőrizhető bizonyíték.

# Példák

- H

**Bemenet:**  $G$  gráf.

**Kérdés:** Van-e  $G$ -ben Hamilton-kör?

$\in \text{NP}$ :  $t \rightarrow$  egy  $C$  Hamilton-kör  $G$ -ben;  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $C$  Hamilton-kör-e.

$\in \text{coNP}$ : **Nem tudjuk, hogy van-e hatékony tanúsítvány.**

- 3SZÍN

**Bemenet:**  $G$  gráf.

**Kérdés:** Igaz-e, hogy  $G$  színezhető 3 színnel?

$\in$ NP:  $t \rightarrow$  egy színezés megadása  $G$ -ben; (azaz egy  $f: V(G) \rightarrow \{p, k, s\}$  függvény)  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $f$  tényleg egy 3-színezése  $G$ -nek.

$\in$ coNP: **Nem tudjuk, hogy van-e hatékony tanúsítvány.**



# Példák

- 3SZÍN

Bemenet:  $G$  gráf.

Kérdés: Igaz-e, hogy  $G$  nem színezhető 3 színnel?

∈NP: Nem tudjuk, hogy van-e hatékony tanúsítvány.

∈coNP:  $t \rightarrow$  egy színezés megadása  $G$ -ben; (azaz egy  $f: V(G) \rightarrow \{p, k, s\}$  függvény)  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $f$  tényleg egy 3-színezése  $G$ -nek.

# Példák

- ÖSSZEFÜGGŐSÉG

**Bemenet:**  $G = (V, E)$  irányítatlan gráf.

**Kérdés:**  $G$  összefüggő?

$\in \text{NP}$ :  $t \rightarrow G$  egy feszítőfája:  $F$ ;  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $F$  tényleg feszítőfa-e  $G$ -ben.

$\in \text{coNP}$ :  $t \rightarrow U \subset V$   
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy nincs él  $U$  és  $V - U$ -beli pontok között

A tanúróól mindig be kell látni, hogy polinomiális méretű, és az ellenőrző algoritusról is mindig be kell látni, hogy polinom időben fut.

Ezek általában könnyen beláthatóak.

- PÁROS-GRÁF-PÁROSÍTÁS

**Bemenet:**  $G = (A, B; E)$  páros gráf

**Kérdés:** Van-e  $G$ -ben teljes párosítás?

$t \rightarrow$  élek  $E'$  részhalmaza;  
 $\in \text{NP}$ :  $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $E'$  tényleg teljes párosítás-e  $G$ -ben.

$\in \text{coNP}$ :  $t \rightarrow X \subseteq A$   
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy teljesül-e  $|X| > |N(X)|$ .

# Példák

- PRÍM

**Bemenet:**  $m > 0$  egész.

**Kérdés:**  $m$  prímszám?

$\in \text{NP}$ :  $t \rightarrow$  Bonyolult, de van ilyen (pl. AKS-teszt);  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $t$  tényleg tanúsítvány.

$\in \text{coNP}$ :  $t \rightarrow 1 < k < m$  szám, ami  $m$  osztója;  
 $\mathcal{T} \rightarrow$  ellenőrzi, hogy  $k$  osztja-e  $m$ -et.

# Hatékony tanúsítvány P-beli problémákra

A **ÖSSZEFÜGGŐSÉG**, **PÁROS-GRÁF-PÁROSÍTÁS**, **PRÍM** problémákra ismert polinomiális algoritmus is.  $\implies$

Az algoritmus lefutásának leírása egy hatékony tanúsítvány ( $t$ ) arra is, hogy a probléma **NP**-ben van, és arra is, hogy **coNP**-ben van, hiszen a végén a válasz vagy **IGEN** vagy **NEM**, a leírás pedig polinom hosszú. Az ellenőrző algoritmus ( $\mathcal{T}$ ) pedig polinom időben tudja ellenőrizni, hogy a leírás helyes-e.  $\implies$

## Tétel

$$P \subseteq NP \text{ és } P \subseteq \text{coNP}$$

Azaz:  $P \subseteq NP \cap \text{coNP}$

# A legfontosabb nyitott kérdések

## Sejtés

$$P \neq NP$$

Ha  $P = NP$  teljesülne, akkor minden olyan problémára, amelyre van hatékony tanúsítvány (azaz  $NP$ -beli), lenne polinomiális algoritmus is. Fogunk mutatni olyan problémákat, amik  $NP$ -beliek, de senki nem tud rájuk polinomiális algoritmust.

## Kérdés

$$P \stackrel{?}{=} NP \cap \text{coNP}$$

Eddig majdnem minden fontos  $NP \cap \text{coNP}$ -beli problémáról kiderült, hogy van rá polinomiális algoritmus, azaz  $P$ -beli.