A számítástudomány alapjai

Algoritmusok bonyolultsága

2020. december 1.

Bevezetés: algoritmusok lépésszámbecslése

A mai órán azt járjuk körül, hogy mitől hatékony egy algoritmus, és milyen szempontokra érdemes figyelni algoritmusok tervezésekor. Az algoritmust a továbbiakban olyan eljárásnak tekintjük, ami egy Π problémát old meg: az I input által megadott konkrét esethez kiszámít egy megoldást az O output képében. (Az outputnak mindig helyes megoldásnak kell lennie.)

Bevezetés: algoritmusok lépésszámbecslése

A mai órán azt járjuk körül, hogy mitől hatékony egy algoritmus, és milyen szempontokra érdemes figyelni algoritmusok tervezésekor. Az algoritmust a továbbiakban olyan eljárásnak tekintjük, ami egy Π problémát old meg: az I input által megadott konkrét esethez kiszámít egy megoldást az O output képében. (Az outputnak mindig helyes megoldásnak kell lennie.)

Π	mkffa	szél. keresés	mélységi keresés	legrövidebb út
I	G, k	$G, (r)$	$G, (r)$	G, ℓ, r
Alg	Kruskal	BFS	DFS	Dijkstra, Ford

Π	legh. út	gráfszínezés	maxfolyam	max. párosítás
I	G, c	G	G, s, t, c	$G = (A, B; E)$
Alg	PERT	mohó (!)	jav. utas	Alt. utas

Π	lnko	összedás	kanonikus alak	Prímteszt
I	a, b	a, b	n	n
O	(a, b)	$a + b$	$n = \Pi_{i=1}^{k} p_i^{\alpha_i}$	i/n
Alg	Euklideszi	írásbeli	??	??

Bevezetés: algoritmusok lépésszámbecslése

A mai órán azt járjuk körül, hogy mitől hatékony egy algoritmus, és milyen szempontokra érdemes figyelni algoritmusok tervezésekor. Az algoritmust a továbbiakban olyan eljárásnak tekintjük, ami egy Π problémát old meg: az I input által megadott konkrét esethez kiszámít egy megoldást az O output képében. (Az outputnak mindig helyes megoldásnak kell lennie.)

Bevezetés: algoritmusok lépésszámbecslése

A mai órán azt járjuk körül, hogy mitől hatékony egy algoritmus, és milyen szempontokra érdemes figyelni algoritmusok tervezésekor. Az algoritmust a továbbiakban olyan eljárásnak tekintjük, ami egy Π problémát old meg: az I input által megadott konkrét esethez kiszámít egy megoldást az O output képében. (Az outputnak mindig helyes megoldásnak kell lennie.)
Mikor gyors egy algoritmus? Akkor ha kevés lépést végez. Ám a lépésszám függ az inputtól: $f_A(I)$ jelöli az A algoritmus lépésszáma az I inputon. Cél: ,,bonyolult" input esetén is csak ,,kevés" lépésre legyen szükség. Mi a bonyolult input? Hát az, amit bonyolult leírni.
Def: Az I input $|I|$ **mérete** az I leírásához szükséges bitek száma.

Bevezetés: algoritmusok lépésszámbecslése

A mai órán azt járjuk körül, hogy mitől hatékony egy algoritmus, és milyen szempontokra érdemes figyelni algoritmusok tervezésekor. Az algoritmust a továbbiakban olyan eljárásnak tekintjük, ami egy Π problémát old meg: az I input által megadott konkrét esethez kiszámít egy megoldást az O output képében. (Az outputnak mindig helyes megoldásnak kell lennie.)
Mikor gyors egy algoritmus? Akkor ha kevés lépést végez. Ám a lépésszám függ az inputtól: $f_A(I)$ jelöli az A algoritmus lépésszáma az I inputon. Cél: ,,bonyolult" input esetén is csak ,,kevés" lépésre legyen szükség. Mi a bonyolult input? Hát az, amit bonyolult leírni.
Def: Az I input $|I|$ **mérete** az I leírásához szükséges bitek száma.
Példa: összeadás esetén a, b inputra a ill. b jegyei száma $\lfloor \log_2(a) \rfloor + 1$, ill. $\lfloor \log_2(b) \rfloor + 1$, az input mérete tehát $\lfloor \log_2(a) \rfloor + \lfloor \log_2(b) \rfloor + 2$.
Egy n-csúcsú, m-élű G gráf szomszédossági mátrixa egy $n \times n$ méretű 0/1 mátrix, így a G input mérete n^2. (Van szerencsésebb megadás is, ahol az inputméret $konst \cdot (n + m)$)

Bevezetés: algoritmusok lépésszámbecslése

Mikor gyors egy algoritmus? Akkor ha kevés lépést végez. Ám a lépésszám függ az inputtól: $f_A(I)$ jelöli az A algoritmus lépésszáma az I inputon. Cél: ,,bonyolult" input esetén is csak ,,kevés" lépésre legyen szükség. Mi a bonyolult input? Hát az, amit bonyolult leírni.
Def: Az I input $|I|$ **mérete** az I leírásához szükséges bitek száma.

Bevezetés: algoritmusok lépésszámbecslése

Mikor gyors egy algoritmus? Akkor ha kevés lépést végez. Ám a lépésszám függ az inputtól: $f_A(I)$ jelöli az A algoritmus lépésszáma az I inputon. Cél: ,,bonyolult" input esetén is csak ,,kevés" lépésre legyen szükség. Mi a bonyolult input? Hát az, amit bonyolult leírni.
Def: Az I input $|I|$ **mérete** az I leírásához szükséges bitek száma.
Az A algoritmus bonyolultságát azzal szeretnénk mérni, hogy mennyire gyorsan nő a lépésszám az input bonyolultabbá válásával, azaz az input méretének növekedtével. Jelölje az A algoritmus maximális lépésszámát a legfeljebb n-méretű inputokon $f_A(n) = \max\{f_A(I) : |I| \leq n\}$. Ezt nehéz pontosan kiszámítani, de ez általában szükségtelen. Elég egy (minél jobb) felső becslés: ha pl $f_A(n) \leq g(n)$, akkor az A algoritmus a legfeljebb n méretű inputokon sosem végez $g(n)$-nél több lépést.
Ebben a modellben az algoritmusnak garantáltan időben kell végeznie minden szóba jövő inputon, ezért innentől úgy tekintjük, hogy az n méretű inputon mindig $f_A(n)$ (ill. $g(n)$) a lépésszám.

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)
Az A algoritmus exponenciális futásidejű, ha van olyan $c > 0$ és $\alpha > 1$, amire $f_A(n) \geq c \cdot \alpha^n$ teljesül minden $n \in \mathbb{N}$ esetén.

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)
Az A algoritmus exponenciális futásidejű, ha van olyan $c > 0$ és $\alpha > 1$, amire $f_A(n) \geq c \cdot \alpha^n$ teljesül minden $n \in \mathbb{N}$ esetén.
Példa: Tfh ugyanarra a Π problémára A és A' egy-egy polinomiális ill. exponenciális futásidejű algoritmus: mondjuk $f_A(n) \leq 1000n^2$, és $f_{A'}(n) \geq 2^n$. Mekkora inputtal tudnak **garantáltan** végezni az egyes algoritmusok, ha összesen ℓ lépés kivárására van idő:

ℓ lépésszámkorlát	10^3	10^5	10^7	10^9
A esetén max $\lvert I \rvert$ legalább:	1	10	100	1000
A' esetén max $\lvert I \rvert$ legfeljebb:	10	16	23	30

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)
Az A algoritmus exponenciális futásidejű, ha van olyan $c > 0$ és $\alpha > 1$, amire $f_A(n) \geq c \cdot \alpha^n$ teljesül minden $n \in \mathbb{N}$ esetén.
Példa: Tfh ugyanarra a Π problémára A és A' egy-egy polinomiális ill. exponenciális futásidejű algoritmus: mondjuk $f_A(n) \leq 1000n^2$, és $f_{A'}(n) \geq 2^n$. Mekkora inputtal tudnak **garantáltan** végezni az egyes algoritmusok, ha összesen ℓ lépés kivárására van idő:

ℓ lépésszámkorlát	10^3	10^5	10^7	10^9
A esetén max $\lvert I \rvert$ legalább:	1	10	100	1000
A' esetén max $\lvert I \rvert$ legfeljebb:	10	16	23	30

Taulság: A Moore-törvény szerint a számítógépek sebessége kb 18 hónaponta megduplázódik. Polinomiális algoritmus esetén ez mérhetően (konstanszorosra) növeli a megoldható input méretét. Exponenciális futásidejű esetén csak elhanyagolható haszon (konstans növekedés) származik ebből.

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)
Az A algoritmus exponenciális futásidejű, ha van olyan $c > 0$ és $\alpha > 1$, amire $f_A(n) \geq c \cdot \alpha^n$ teljesül minden $n \in \mathbb{N}$ esetén.

Lépésszámbecslések nagyságrendje

Az A algoritmus polinomidejű, ha $f_A(n) \leq p(n)$ teljesül egy alkalmas $p(x)$ poliomra minden $n \in \mathbb{N}$ esetén. (Ez ekvivalens azzal, hogy van olyan $c > 0$ és $k \in \mathbb{N}$, amire $f_A(n) \leq c \cdot n^k$.)
Az A algoritmus exponenciális futásidejű, ha van olyan $c > 0$ és $\alpha > 1$, amire $f_A(n) \geq c \cdot \alpha^n$ teljesül minden $n \in \mathbb{N}$ esetén.
Megj: Nem igaz, hogy minden algoritmus vagy polinomidejű vagy exponenciális futásidejű. Ha pl. $f_A(n) = n^{ln(n)}$, akkor A-ra egyik tulajdonság sem áll (ugyanis A egy ún. szubexponenciális futásidejű algoritmus). A számunkra a lényeg, hogy (itt és most) a polinomidejű algoritmusokat szeretjük, és azokat tekintjük hatékonynak. Természetesen a gyakorlatban törekszünk minél jobb lépésszámra, így pl. a polinom minél alacsonyabb fokon tartására, de az elméleti hatékonyság tekintetében csak az érdekes, hogy becsülhető-e a lépésszám polinommal.

Konkrét lépésszámbecslések

Írásbeli összeadás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Helyiértékenként legfeljebb 2 összeadást kell végezni (2-t akkor, ha van maradék), így $f_A(n) \leq 3 \max(n, k) \leq$ $\leq 3|I|$, az algoritmus lineáris futásidejű, tehát polinomiális.

Konkrét lépésszámbecslések

Írásbeli összeadás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Helyiértékenként legfeljebb 2 összeadást kell végezni (2-t akkor, ha van maradék), így $f_A(n) \leq 3\max(n, k) \leq$ $\leq 3|I|$, az algoritmus lineáris futásidejű, tehát polinomiális.

Írásbeli szorzás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Kell nk jegyenkénti szorzás (binárisan ez nagyon egyszerű), ami legfeljebb $konst \cdot nk$ lépés. Ezután k jegyű számból kell legfeljebb n db-ot összeadni, egyetlen összeg sem lesz $n + k$-nál több jegyű. Tehát $f_A(n) \leq konst \cdot (nk + n(n + k)) \leq$ $\leq 2konst \cdot (n + k)^2 = 2konst \cdot |I|^2$, az algoritmus polinomiális.

Konkrét lépésszámbecslések

Írásbeli összeadás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Helyiértékenként legfeljebb 2 összeadást kell végezni (2-t akkor, ha van maradék), így $f_A(n) \leq 3\max(n, k) \leq$ $\leq 3|I|$, az algoritmus lineáris futásidejű, tehát polinomiális.

Írásbeli szorzás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Kell nk jegyenkénti szorzás (binárisan ez nagyon egyszerű), ami legfeljebb $konst \cdot nk$ lépés. Ezután k jegyű számból kell legfeljebb n db-ot összeadni, egyetlen összeg sem lesz $n + k$-nál több jegyű. Tehát $f_A(n) \leq konst \cdot (nk + n(n + k)) \leq$ $\leq 2konst \cdot (n + k)^2 = 2konst \cdot |I|^2$, az algoritmus polinomiális.

Hatványozás Egy n-jegyű szám n-jegyű kitevőre felemelésekor az inputméret $|I| = 2n$. A végeredmény jegyeinek száma legalább annyi, mint $(2^n)^{(2^n)}$ jegyeinek száma, azaz $n \cdot 2^n > 2^n = \sqrt{2}^{|I|}$. Ezért a hatványozásra minden algoritmus exponenciális futásidejű.

Konkrét lépésszámbecslések

Írásbeli összeadás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Helyiértékenként legfeljebb 2 összeadást kell végezni (2-t akkor, ha van maradék), így $f_A(n) \leq 3\max(n, k) \leq$ $\leq 3|I|$, az algoritmus lineáris futásidejű, tehát polinomiális.

Írásbeli szorzás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Kell nk jegyenkénti szorzás (binárisan ez nagyon egyszerű), ami legfeljebb $konst \cdot nk$ lépés. Ezután k jegyű számból kell legfeljebb n db-ot összeadni, egyetlen összeg sem lesz $n + k$-nál több jegyű. Tehát $f_A(n) \leq konst \cdot (nk + n(n + k)) \leq$ $\leq 2konst \cdot (n + k)^2 = 2konst \cdot |I|^2$, az algoritmus polinomiális.

Hatványozás Egy n-jegyű szám n-jegyű kitevőre felemelésekor az inputméret $|I| = 2n$. A végeredmény jegyeinek száma legalább annyi, mint $(2^n)^{(2^n)}$ jegyeinek száma, azaz $n \cdot 2^n > 2^n = \sqrt{2}^{|I|}$. Ezért a hatványozásra minden algoritmus exponenciális futásidejű.

Modulo m hatványozás a, b, m input esetén kell megmondani a^b értékét $mod\ m$. Nem triviális, de van erre polinomidejű algoritmus.

Konkrét lépésszámbecslések

Írásbeli összeadás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Helyiértékenként legfeljebb 2 összeadást kell végezni (2-t akkor, ha van maradék), így $f_A(n) \leq 3\max(n, k) \leq$ $\leq 3|I|$, az algoritmus lineáris futásidejű, tehát polinomiális.

Írásbeli szorzás Egy n ill. k jegyű (bináris) szám összeadásakor az inputméret $|I| = n + k$. Kell nk jegyenkénti szorzás (binárisan ez nagyon egyszerű), ami legfeljebb $konst \cdot nk$ lépés. Ezután k jegyű számból kell legfeljebb n db-ot összeadni, egyetlen összeg sem lesz $n + k$-nál több jegyű. Tehát $f_A(n) \leq konst \cdot (nk + n(n + k)) \leq$ $\leq 2konst \cdot (n + k)^2 = 2konst \cdot |I|^2$, az algoritmus polinomiális.

Hatványozás Egy n-jegyű szám n-jegyű kitevőre felemelésekor az inputméret $|I| = 2n$. A végeredmény jegyeinek száma legalább annyi, mint $(2^n)^{(2^n)}$ jegyeinek száma, azaz $n \cdot 2^n > 2^n = \sqrt{2}^{|I|}$. Ezért a hatványozásra minden algoritmus exponenciális futásidejű.

Modulo m hatványozás a, b, m input esetén kell megmondani a^b értékét $mod\ m$. Nem triviális, de van erre polinomidejű algoritmus.

Euklideszi algoritmus polinomidőben végezhető, nem bizonyítjuk.

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Az algoritmus minden mozzanata egy-egy esethez köthető.

Ia Van élért csúcs (u), és abból fut eléretlen csúcsba él ($\leq m$-szer)

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Az algoritmus minden mozzanata egy-egy esethez köthető.

Ia Van élért csúcs (u), és abból fut eléretlen csúcsba él ($\leq m$-szer)

Ib Van élért csúcs (u), de nem fut belőle él eléretlenbe ($\leq n$-szer)

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Az algoritmus minden mozzanata egy-egy esethez köthető.

Ia	Van élért csúcs (u), és abból fut eléretlen csúcsba él	($\leq m$-szer)
Ib	Van élért csúcs (u), de nem fut belőle él eléretlenbe	($\leq n$-szer)
IIa	Nincs élért csúcs, de van eléretlen	($\leq n$-szer)

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Az algoritmus minden mozzanata egy-egy esethez köthető.

Ia	Van élért csúcs (u), és abból fut eléretlen csúcsba él	($\leq m$-szer)
Ib	Van élért csúcs (u), de nem fut belőle él eléretlenbe	($\leq n$-szer)
IIa	Nincs élért csúcs, de van eléretlen	($\leq n$-szer)
IIb	Nincs se élért csúcs, se eléretlen	(≤ 1-szer)

Gráfalgoritmusok lépésszámbecslése

Bejárások Input egy n csúcsú, m élű gráf. Az inputméret n^2, vagy $konst \cdot (n+m)$, a konkrét adatstruktúrától függően. Mivel $m \leq n^2$, ezért $f_A(n)$ pontosan akkor becsülhető n^2 egy polinomjával, ha $konst \cdot (n+m)$ egy polinomjával felülről becsülhető. Ezért a polinomidejűség nem függ attól, hogy hogyan is adjuk meg az inputot.

Az algoritmus minden mozzanata egy-egy esethez köthető.

Ia	Van élért csúcs (u), és abból fut eléretlen csúcsba él	($\leq m$-szer)
Ib	Van élért csúcs (u), de nem fut belőle él eléretlenbe	($\leq n$-szer)
IIa	Nincs élért csúcs, de van eléretlen	($\leq n$-szer)
IIb	Nincs se élért csúcs, se eléretlen	($\leq$ 1-szer)

Az algoritmus megvalósításakor a fenti mozzanatok mindegyike valójában egynél több (de szerencsére konstans sok) lépést jelent, ezért a lépésszám legfeljebb $konst \cdot (n+m)$, ami a bemenet méretének elsőfokú polinomjával becsülhető, azaz a bejárási algoritmusok lineárisak, így polinomidejűek.

Gráfalgoritmusok lépésszámbecslése

Gráfalgoritmusok lépésszámbecslése

Kruskal Input egy n csúcsú, m élű gráf és egy k költségfv. Az inputméret G miatt $konst(n+m)$, a költségfüggvény tárolása annyiszor m bit, ahány jegyűek az egyes élköltségek. Tegyük fel, hogy az élköltségek kevés bites egész számok, így az inputméret továbbra is $konst \cdot (n+m)$-nek tekinthető.

Gráfalgoritmusok lépésszámbecslése

Kruskal Input egy n csúcsú, m élű gráf és egy k költségfv. Az inputméret G miatt $konst(n+m)$, a költségfüggvény tárolása annyiszor m bit, ahány jegyűek az egyes élköltségek. Tegyük fel, hogy az élköltségek kevés bites egész számok, így az inputméret továbbra is $konst \cdot (n+m)$-nek tekinthető.
Az algoritmus először az éleket növekvő költség szerint rendezi. Ez elvégezhető $konst \cdot m \log m$ lépésben (de a buborékrendezés is végez $konst \cdot m^2$ lépésben).
Ezután az élekről egyesével döntünk. Alkalmas (pl unió-holvan típusú) adatstruktúrával minden ilyen döntés elvégezhető $konst' \cdot \log n$ lépésben, de naív megközelítéssel sem kell $konst'' \cdot n$ lépésnél több. Ezért a Kruskal algoritmus lépésszáma legfeljebb $konst \cdot m^2 + konst'' \cdot mn \leq c \cdot |I|^2$, alkalmas c konstansra. Ezért a Kruskal algoritmus kvadratikus, vagyis polinomidejű.

Gráfalgoritmusok lépésszámbecslése

Gráfalgoritmusok lépésszámbecslése

Dijkstra Input egy n csúcsú, m élű $G = (V, E)$ gráf, $r \in V$ gyökércsúcs és egy $\ell : E \to \mathbb{R}_+$ hosszfv. Az élhosszokat legfeljebb konstans jegyű számoknak tekintve az inputméret ismét $konst \cdot (n + m)$.

Gráfalgoritmusok lépésszámbecslése

Dijkstra Input egy n csúcsú, m élű $G = (V, E)$ gráf, $r \in V$ gyökércsúcs és egy $\ell : E \to \mathbb{R}_+$ hosszfv. Az élhosszokat legfeljebb konstans jegyű számoknak tekintve az inputméret ismét $konst \cdot (n + m)$.
Az algoritmus élmenti javításokat (legfeljebb m-szer) ill. a KÉSZ halmaz növelését végzi (n-szer). Egyetlen elmenti javítás megoldható konstans sok lépésből, a KÉSZ halmaz növeléséhez kell még egy minimumképzés is, aholis legfeljebb n elemből választunk minimumot. Ez legfeljebb n összehasonlítással megtehető. Ezért a Kruskal algoritmus lépésszáma legfeljebb $konst \cdot m + konst' \cdot n^2 \leq c \cdot |I|^2$ alkalmas c konstanssal. Tehát a Dijkstra algoritmus is kvadratikus, vagyis polinomidejű.

Gráfalgoritmusok lépésszámbecslése

Dijkstra Input egy n csúcsú, m élű $G = (V, E)$ gráf, $r \in V$ gyökércsúcs és egy $\ell : E \to \mathbb{R}_+$ hosszfv. Az élhosszokat legfeljebb konstans jegyű számoknak tekintve az inputméret ismét $konst \cdot (n + m)$.
Az algoritmus élmenti javításokat (legfeljebb m-szer) ill. a KÉSZ halmaz növelését végzi (n-szer). Egyetlen elmenti javítás megoldható konstans sok lépésből, a KÉSZ halmaz növeléséhez kell még egy minimumképzés is, aholis legfeljebb n elemből választunk minimumot. Ez legfeljebb n összehasonlítással megtehető. Ezért a Kruskal algoritmus lépésszáma legfeljebb $konst \cdot m + konst' \cdot n^2 \leq c \cdot |I|^2$ alkalmas c konstanssal. Tehát a Dijkstra algoritmus is kvadratikus, vagyis polinomidejű.

És polinomidejű még a többi tanult gráfalgoritmus is: Ford, Floyd, PERT, Ford-Fulkerson (Edmonds-Karp kiegészítéssel), alternáló utas, de ezt még annyira sem ,,bizonyítjuk", mint az előbbieket.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Megj: A döntési problémák valamilyen értelmeben a legegyszerűbb problémák, ezért a továbbiakban ezeket vizsgáljuk. Az eddig vizsgált problémák ugyan nem ilyenek, de sokuk visszavezethető döntési problémára.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Megj: A döntési problémák valamilyen értelmeben a legegyszerűbb problémák, ezért a továbbiakban ezeket vizsgáljuk. Az eddig vizsgált problémák ugyan nem ilyenek, de sokuk visszavezethető döntési problémára.

Például, ha egy inputként megadott G gráfban maximális méretű párosítást kell keresni, akkor a rokon döntési probléma az, hogy G', k input esetén a G' gráfnak van-e az k méretű párosítása. Ezt a kérdést a G gráfra különböző k értékekkel feltéve kibarkochbázható a maximális méretű párosítás $\nu(G)$ mérete. Ezek után G-ből egymás után elhagyva az éleket ugyanilyen döntési problémák megoldásával megtudhatjuk, hogy az adott él elhagyása után is van-e ugyanekkora méretű párosítás. Ha minden olyan élt elhagyunk, amire nem csökken a maximális párosítás mérete, akkor G-ből végül egy maximális méretű párosítás marad.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Megj: A döntési problémák valamilyen értelmeben a legegyszerűbb problémák, ezért a továbbiakban ezeket vizsgáljuk. Az eddig vizsgált problémák ugyan nem ilyenek, de sokuk visszavezethető döntési problémára.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Megj: A döntési problémák valamilyen értelmeben a legegyszerűbb problémák, ezért a továbbiakban ezeket vizsgáljuk. Az eddig vizsgált problémák ugyan nem ilyenek, de sokuk visszavezethető döntési problémára.

Feladat: Határozzuk meg a Hamilton-kör keresés problémához tartozó döntési problémát, és találjunk az inputként megadott gráfban Hamilton-kört néhány alkalmasan választott döntési probléma megoldásával.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- Az ÖF probléma inputja egy G gráf, outputja IGEN, ha G összefüggő, NEM, ha nem az.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- Az ÖF probléma inputja egy G gráf, outputja IGEN, ha G összefüggő, NEM, ha nem az.
  A BFS polinomidejű algoritmus, és segítségével megválaszolható a kérdés $\Rightarrow$ ÖF$\in P$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- Az EULER probléma inputja egy G gráf, outputja IGEN, ha G-nek van Euler-körsétája, NEM, ha nincs.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- Az EULER probléma inputja egy G gráf, outputja IGEN, ha G-nek van Euler-körsétája, NEM, ha nincs.
  Az izolált pontoktól eltekintve összefüggőség egy BFS-sel megállapítható, a fokok párossága szintén polinomidőben ellenőrizhető $\Rightarrow$ EULER$\in P$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- A MAXFOLYAM probléma inputja egy (G, s, t, c) hálózat és egy k érték, outputja IGEN, ha G-ben van k nagyságú st-folyam, NEM, ha nincs.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- A MAXFOLYAM probléma inputja egy (G, s, t, c) hálózat és egy k érték, outputja IGEN, ha G-ben van k nagyságú st-folyam, NEM, ha nincs.
  Az Edmonds-Karp módszer szerint végzett jav utas algoritmus polinomidejű, ezért MAXFOLYAM$\in P$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Példák:

- A MAXFOLYAM probléma inputja egy (G, s, t, c) hálózat és egy k érték, outputja IGEN, ha G-ben van k nagyságú st-folyam, NEM, ha nincs.
  Az Edmonds-Karp módszer szerint végzett jav utas algoritmus polinomidejű, ezért MAXFOLYAM$\in P$.
- További P-beli problémák: létezik-e legfeljebb k súlyú feszítőfa G-ben, van-e k méretű párosítás G-ben, PERT feladat elvégezhető-e k időegység alatt, G síkbarajzolható-e, ...

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

A következő cél olyan módszer kidolgozása, aminek a segítségével megtudhatjuk, ha egy problémára nincs polinomidejű algoritmus (helyesebben reménytelen ilyet keresni).

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Példa: HAM input: G gráf, output IGEN, ha G-nek van H-köre.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Példa: HAM input: G gráf, output IGEN, ha G-nek van H-köre.

HAM$\in NP$: a H-kör polinomidőben ellenőrizhető tanú az IGENre.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Def: A *P* problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az *NP* problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden *I* input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Példa: HAM input: *G* gráf, output IGEN, ha *G*-nek van H-köre.

HAM$\in NP$: a H-kör polinomidőben ellenőrizhető tanú az IGENre.

3-SZÍN input: *G* gráf, output IGEN, ha $\chi(G) \leq 3$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) *I* inputra az output egyetlen bit: IGEN vagy NEM.

Def: A *P* problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az *NP* problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden *I* input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Példa: HAM input: *G* gráf, output IGEN, ha *G*-nek van H-köre.

HAM$\in NP$: a H-kör polinomidőben ellenőrizhető tanú az IGENre.

3-SZÍN input: *G* gráf, output IGEN, ha $\chi(G) \leq 3$.

3-SZÍN$\in NP$: 3-színezés pol.időben ellenőrizhető tanú az IGENre.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Biz: A pol.idejű algoritmus futása tanú az IGEN/NEM-re.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

Példa: MAXKLIKK input: G gráf, $k \in \mathbb{N}$, output IGEN, ha $\omega(G) \geq k$, különben NEM.

MAXFTN input: G gráf, $k \in \mathbb{N}$, output IGEN, ha $\alpha(G) \geq k$, különben NEM. Ekkor MAXKLIKK$\prec$MAXFTN: $(G,k) \overset{pol}{\rightarrow} (\overline{G},k)$, u.i. $\omega(G) = \alpha(\overline{G})$.

$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

Tétel: (1) $\Pi \prec \Pi' \in P \Rightarrow \Pi \in P$ és
(2) $\Pi \prec \Pi' \prec \Pi'' \Rightarrow \Pi \prec \Pi''$

$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

Tétel: (1) $\Pi \prec \Pi' \in P \Rightarrow \Pi \in P$ és
(2) $\Pi \prec \Pi' \prec \Pi'' \Rightarrow \Pi \prec \Pi''$

„Biz”: Polinomok egymásba helyettesítése polinom.

$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

Döntési problémák

Def: A Π probléma döntési probléma, ha minden (értelmes) I inputra az output egyetlen bit: IGEN vagy NEM.

Def: A P problémaosztályt mindazon Π döntési problémák alkotják, amelyekre van polinomidejű algoritmus.

Def: Az NP problémaosztály mindazon Π döntési problémákból áll, amelyekre az IGEN válaszhoz tartozó minden I input esetén a válasz helyessége polinomidőben bizonyítható.

A $co - NP$ problémaosztály ugyanez, a NEM válasz esetén.

Tétel: $P \subseteq NP \cap co - NP$.

Def: A Π döntési probléma polinomiálisan visszavezethető a Π' döntési problémára (jel: $\Pi \prec \Pi'$), ha van olyan polinomidejű algoritmus, ami Π minden I inputjához kiszámítja a Π' egy olyan I' inputját, amire ugyanaz a válasz Π'-ben, mint I-re Π-ben:

Tétel: (1) $\Pi \prec \Pi' \in P \Rightarrow \Pi \in P$ és
(2) $\Pi \prec \Pi' \prec \Pi'' \Rightarrow \Pi \prec \Pi''$

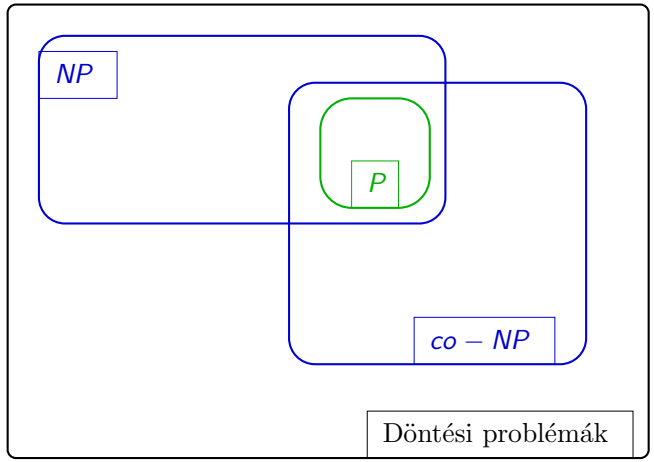
$$\begin{array}{ccc} \Pi & \prec & \Pi' \\ | & & | \\ I & \overset{pol}{\rightarrow} & I' \end{array}$$

„Biz”: Polinomok egymásba helyettesítése polinom.

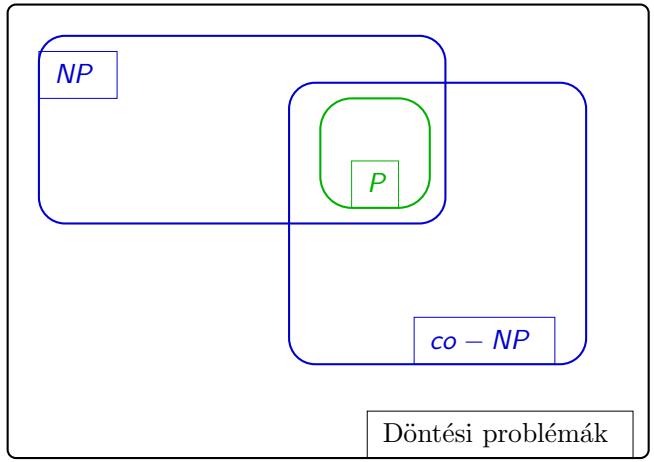
Megj: Ha $\Pi \prec \Pi'$, akkor Π'-t nehezebbnek tekintjük Π-nél u.i. Π' segítségével Π is megoldható.

NP-teljesség

Tétel: $P \subseteq NP \cap co - NP$.

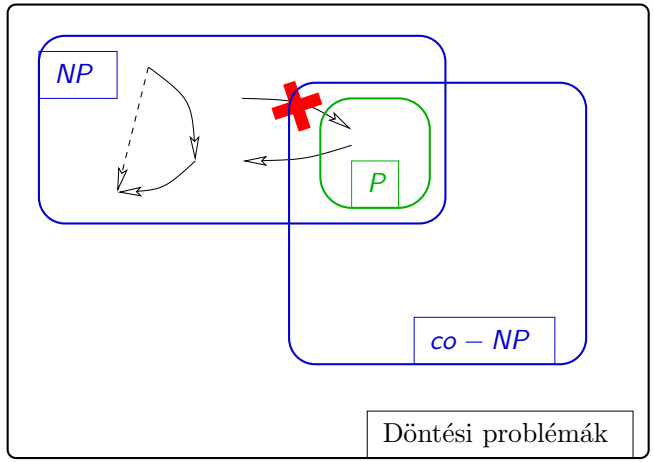


NP-teljesség



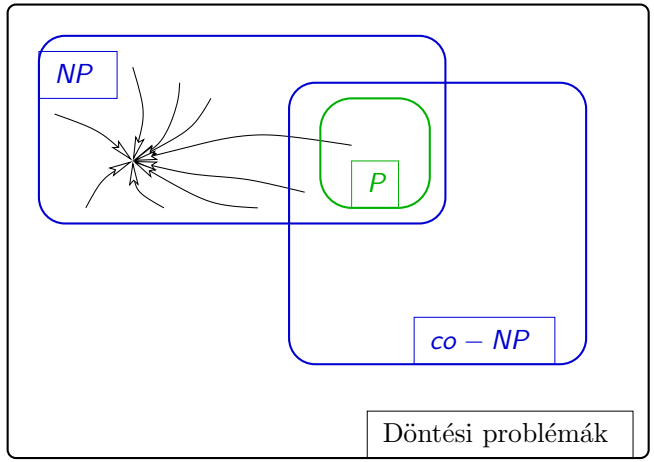
NP-teljesség

Tétel: (1) $\Pi \prec \Pi' \in P \Rightarrow \Pi \in P$ és
(2) $\Pi \prec \Pi' \prec \Pi'' \Rightarrow \Pi \prec \Pi''$

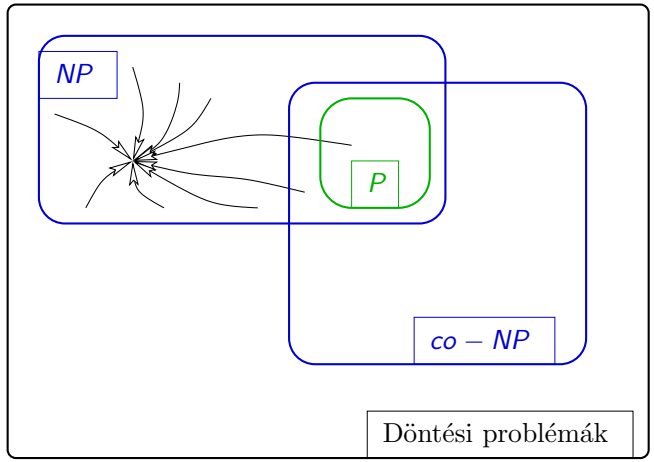


NP-teljesség

Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?



NP-teljesség



Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?

Cook-Levin-tétel: A SAT ilyen.

Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.

CNF: $\text{klóz}_1 \wedge \text{klóz}_2 \wedge \ldots \wedge \text{klóz}_k$.

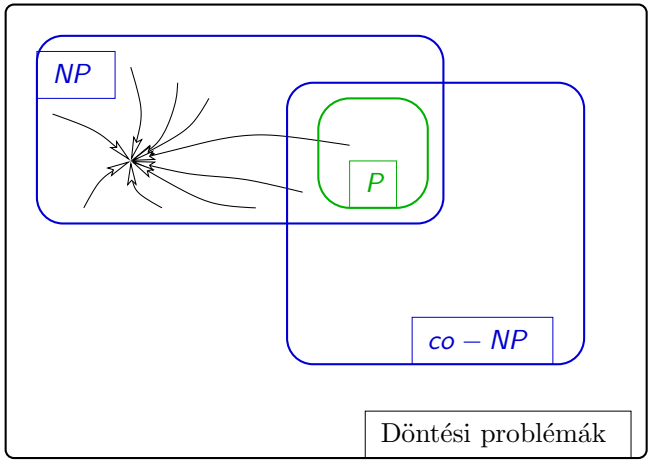
klóz: $\text{literál}_1 \vee \text{literál}_2 \vee \ldots \vee \text{literál}_\ell$. literál: változó vagy a negáltja.

Φ CNF kielégíthető, ha a változók logikai értéke választható úgy, hogy Φ kiértékelése IGAZ legyen.

Példa: A $\Phi = (x \vee \overline{y} \vee w) \wedge (y \vee \overline{t} \vee \overline{z}) \wedge (t \vee x)$ CNF kielégíthető pl $x = y = 1$, $t = w = z = 0$ választással.

NP-teljesség

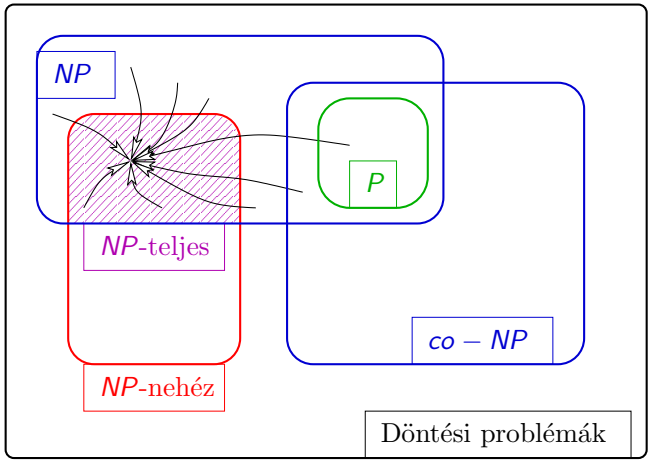
Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?
Cook-Levin-tétel: A SAT ilyen.
Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.



NP-teljesség

Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?

Cook-Levin-tétel: A SAT ilyen.



Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.

Def: A Π probléma *NP*-nehéz, ha $\Pi' \prec \Pi$ áll minden $\Pi' \in NP$-re.

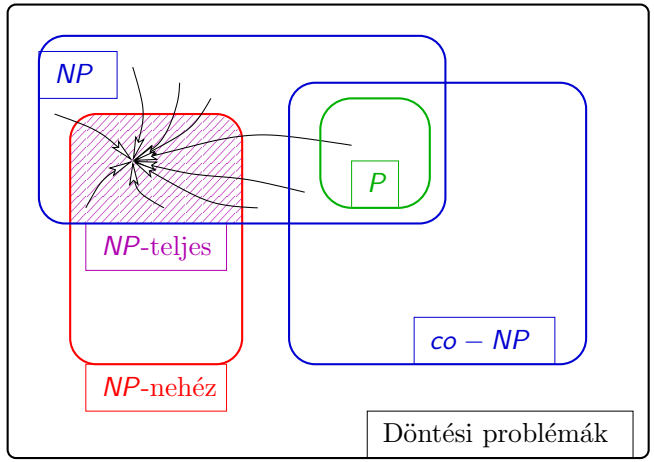
A Π probléma *NP*-teljes, ha Π *NP*-nehéz és $\Pi \in NP$.

Az *NP*-teljes problémák az *NP* problémaosztály legnehezebb problémái. Ilyen pl. a SAT.

NP-teljesség

Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?

Cook-Levin-tétel: A SAT ilyen.

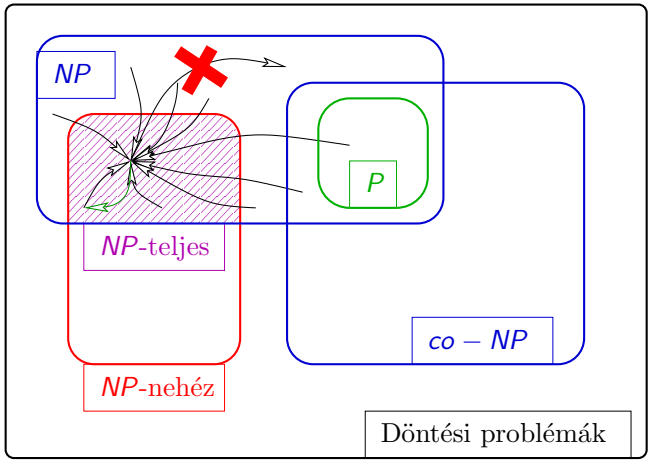


Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.

Def: A Π probléma *NP*-nehéz, ha $\Pi' \prec \Pi$ áll minden $\Pi' \in NP$-re.
A Π probléma *NP*-teljes, ha Π *NP*-nehéz és $\Pi \in NP$.
Az *NP*-teljes problémák az *NP* problémaosztály legnehezebb problémái. Ilyen pl. a SAT.

Megf: Ha *P* tartalmaz *NP*-teljes problémát, akkor $P = NP = co - NP$. Így pl az *NP*-beliségből következik a *P*-beliség. Abban hiszünk, hogy ez nem igaz.
Úgy képzeljük, hogy *P*-ben nincs *NP*-teljes probléma, ezért egyetlen *NP*-teljes problémára sincs hatékony algoritmus.

NP-teljesség



Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?

Cook-Levin-tétel: A SAT ilyen.

Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.

Def: A Π probléma *NP*-nehéz, ha $\Pi' \prec \Pi$ áll minden $\Pi' \in NP$-re.

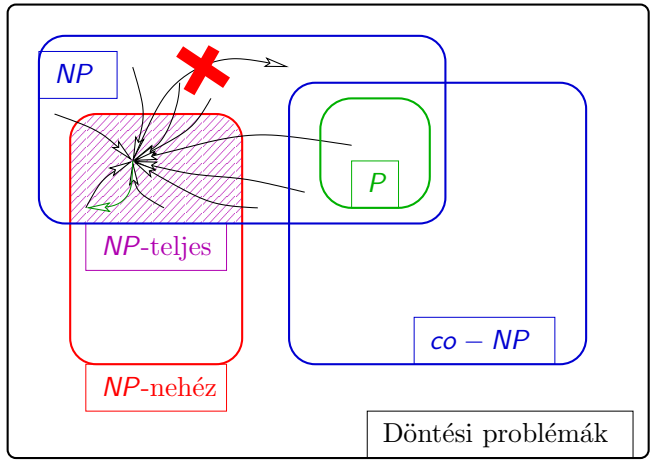
A Π probléma *NP*-teljes, ha Π *NP*-nehéz és $\Pi \in NP$.

Az *NP*-teljes problémák az *NP* problémaosztály legnehezebb problémái. Ilyen pl. a SAT.

Tétel: Ha $\Pi \in NP$ és $\Pi' \prec \Pi$ valamely ismert *NP*-nehéz Π' problémára, akkor Π *NP*-teljes.

Számos problémára lehet így *NP*-teljességet igazolni. Ezek tehát -azt hisszük- nem *P*-beliek, így hatékony algoritmus sincs rájuk.

NP-teljesség



Van-e vajon az *NP*-beli problémák között legnehezebb? Olyan t.i., amire minden más *NP*-beli visszavezethető?

Cook-Levin-tétel: A SAT ilyen.

Def: A SAT inputja egy CNF, outputja IGEN ha a CNF kielégíthető.

Def: A Π probléma *NP*-nehéz, ha $\Pi' \prec \Pi$ áll minden $\Pi' \in NP$-re.
A Π probléma *NP*-teljes, ha Π *NP*-nehéz és $\Pi \in NP$.
Az *NP*-teljes problémák az *NP* problémaosztály legnehezebb problémái. Ilyen pl. a SAT.

Tétel: Ha $\Pi \in NP$ és $\Pi' \prec \Pi$ valamely ismert *NP*-nehéz Π' problémára, akkor Π *NP*-teljes.
Számos problémára lehet így *NP*-teljességet igazolni. Ezek tehát -azt hisszük- nem *P*-beliek, így hatékony algoritmus sincs rájuk.
Vigyázat! A Π *NP*-teljességéhez nem Π-t kell visszavezetni egy nehéz problémára (ebből u.i. nem következik semmi), hanem egy nehezet kell Π-re visszavezetni.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.
Tétel: k-SAT, MAXFTN, MAXKLIKK, HAM, HAMÚT, RÉSZGR$\in NP$, P RÍM$\in co - NP$.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.
Tétel: k-SAT, MAXFTN, MAXKLIKK, HAM, HAMÚT, RÉSZGR$\in NP$, P RÍM$\in co - NP$.
Biz: Polinomidőben ellenőrizhető tanúk: helyes értékadás, k méretű ftn/klikk, Hamilton-kör/út, $V(H) \to V(G)$ injekció, valódi osztó.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.
Tétel: k-SAT, MAXFTN, MAXKLIKK, HAM, HAMÚT, RÉSZGR$\in NP$, P RÍM$\in co - NP$.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.
Tétel: k-SAT, MAXFTN, MAXKLIKK, HAM, HAMÚT, RÉSZGR$\in NP$, P RÍM$\in co - NP$.
Tétel: 2-SAT$\in P$.

NP-teljes problémák

k-SAT: a SAT problémának az a speciális elete, ahol a CNF minden klózában legfeljebb k literál van összeéselve.
HAMÚT: Input: G. Output: IGEN, ha G-nek van Hamilton-útja.
k-SZÍN: Input: G gráf és $k \in \mathbb{N}$. Output: IGEN, ha $\chi(G) \leq k$.
RÉSZGR: Input: G, H gráfok. Output: IGEN, ha G-nek van H-val izomorf részgráfja.
PRÍM: Input: $n \in \mathbb{N}$. Output: IGEN, ha n prím.
Tétel: k-SAT, MAXFTN, MAXKLIKK, HAM, HAMÚT, RÉSZGR$\in NP$, P RÍM$\in co - NP$.
Tétel: 2-SAT$\in P$.
Tétel: SAT$\prec$ 3-SAT$\prec$ 3-SZÍN$\prec$MAXFTN$\prec$MAXKLIKK, valamint 3-SZÍN$\prec$HAM$\prec$HAMÚT$\prec$RÉSZGR és ha $k \geq 3$ akkor 3-SZÍN$\prec$ k-SZÍN. Ezen problémák mindegyike NP-teljes.

Köszönöm a figyelmet!