

Algoritmuselmélet 1. ZH

A rendelkezésre álló munkaidő 90 perc. Minden megoldást indokoljon!

2024. április 18.

Minden feladat egységesen 10 pontot ér. Az aláírás megszerzéséhez minimum 24 pontot kell elérni.

Kérjük, minden résztvevő írja fel a **nevét**, **NEPTUN kódját** és a **gyakorlatvezető nevét** az összetűzött lapok jobb felső sarkába.

Általános alapelvek. A pontozási útmutató célja, hogy a javítók a dolgozatokat egységesen értékeljék. Ezért az útmutató minden feladat (legalább egy lehetséges) megoldásának főbb gondolatait és az ezekhez rendelt részpontszámokat közli. Az útmutatónak nem célja a feladatok teljes értékű megoldásának részletes leírása; a leírt lépések egy maximális pontszámot érő megoldás vázlatának tekinthetők. Az útmutatóban feltüntetett részpontszámok csak akkor járnak a megoldónak, ha a kapcsolódó gondolat egy áttekinthető, világosan leírt és megindokolt megoldás egy lépéseként szerepel a dolgozatban. Így például az anyagban szereplő ismeretek, definíciók, tételek puszta leírása azok alkalmazása nélkül nem ér pontot (még akkor sem, ha egyébként valamelyik leírt tény a megoldásban valóban szerephez jut). Annak mérlegelése, hogy az útmutatóban feltüntetett pontszám a fentiek figyelembevételével a megoldónak (részben vagy egészében) jár-e, teljes mértékben a javító hatásköre. Részpontszám jár minden olyan ötletért, rész megoldásért, amelyből a dolgozatban leírt gondolatmenet alkalmas kiegészítésével a feladat hibátlan megoldása volna kapható. Ha egy megoldó egy feladatra több, egymástól lényegesen különböző megoldást is elkezd, akkor legföljebb az egyikre adható pontszám. Ha mindegyik leírt megoldás vagy megoldásrészlet helyes vagy helyessé kiegészíthető, akkor a legtöbb részpontot érő megoldáskezdeményt értékeljük. Ha azonban több megoldási kísérlet között van helyes és (lényeges) hibát tartalmazó is, továbbá a dolgozathoz nem derül ki, hogy a megoldó melyiket tartotta helyesnek, akkor a kevesebb pontot érő megoldáskezdeményt értékeljük (akkor is, ha ez a pontszám 0). Az útmutatóban szereplő részpontszámok szükség esetén tovább is oszthatók. Az útmutatóban leírtól eltérő jó megoldás természetesen maximális pontot ér, de bizonyítás nélkül csak az előadáson szereplő tételekre és állításokra lehet hivatkozni.

1. Legyen $f(n) = 2024$ ha $n < 15$ és $f(n) = n \cdot 2^{n+10}$ különben. Valamint legyen $g(n) = 2023$ ha $n < 42$ és $g(n) = n^2 \cdot 2^n$ különben. Igaz-e, hogy
- (a) $f(n) \in O(g(n))$? (b) $g(n) \in O(f(n))$?

Megoldás: $f(n) = 2^{10} \cdot n \cdot 2^n$ ha $n \geq 15$ **2 pont**

(a) Ha $n \geq 42$, akkor $n < n^2$, ezért $f(n) < 2^{10}g(n)$. **2 pont**

Tehát az $n_0 = 42$ és $c = 2^{10}$ választás mutatja, hogy $f(n) \in O(g(n))$. **2 pont**

(b) Tegyük fel, hogy valamilyen $n \geq n_0$ és c esetén $n^2 \cdot 2^n \leq c \cdot 2^{10} \cdot n \cdot 2^n$. **2 pont**

Ekkor $n \leq c \cdot 2^{10}$ -nek teljesülni kell minden elég nagy n -re, ami nem igaz. **1 pont**

Tehát nem igaz, hogy $g(n) \in O(f(n))$. **1 pont**

2. Adottak c_1, c_2, \dots, c_n különböző egész számok. Ezeket szeretnénk nagyság szerint rendezni növekvő, vagy csökkenő sorrendbe úgy, hogy a szokásos összehasonlítás helyett most a következő kérdéseket lehet feltenni 1 lépésben: Három kiválasztott

elem közül melyik esik a rendezés szerint a másik kettő közé? (Például, ha a c_i, c_j, c_k elemeket kérdezzük ahol $c_i = 3, c_j = 1, c_k = 7$, akkor a c_i elemet kapjuk válaszként.) Adjon $O(n \log n)$ kérdést használó algoritmust, aminek kimenete a csökkenő vagy növekvő sorrendben lévő számok. (Azt nem kell az algoritmusnak meghatároznia, hogy csökkenő vagy növekvő-e a sorozat.)

Megoldás: Először vegyünk az első 3 elemet, c_1, c_2, c_3 -at és kérdezzük meg, hogy melyik a középső. Ha pl. c_2 , akkor a sorrend vagy $c_1 < c_2 < c_3$, vagy $c_1 > c_2 > c_3$, írjuk le őket balról jobbra c_1, c_2, c_3 sorrendben **2 pont**

Ezután a beszűrős rendezéshez hasonlóan járunk. **2 pont**

Ha c_1, c_2, \dots, c_k sorba van rakva (növeően vagy csökkenően), akkor a bináris kereséshez hasonlóan beszűrjük c_{k+1} -et. **2 pont**

Először a $c_{k+1}, c_{\lceil k/2 \rceil - 1}, c_{\lceil k/2 \rceil}$ hármásra kérdezzük rá. A választ meghatározza, hogy c_{k+1} a sorrend bal vagy jobb felében van-e, vagy pedig épp megtaláltuk a helyét. **2 pont**

A szóbejövő helyek száma minden kérdés után felére csökken, így egy elem beszűrása $O(\log n)$ lépés. Az összes lépésszám $O(n \log n)$. **2 pont**

3. Adott egy n pontú m élű egyszerű, összefüggő, irányítatlan gráf. Adjon $O(n + m)$ futásidejű algoritmust, ami megtalál egy olyan pontot, amelyet a gráfból elhagyva a gráf összefüggő marad.

Megoldás: Futtassunk egy DFS-t a gráf tetszőleges pontjából. **2 pont**

Ennek vegyünk az 1 befejezési számú csúcsát (itt le is állíthatjuk a DFS-t). **2 pont**

Ez a DFS fának egy levele lesz. **2 pont**

Ha ezt a levelet elhagyjuk a gráfból, akkor a DFS fa maradék élei mutatják, hogy a maradék gráf összefüggő marad. **2 pont**

A DFS lépésszáma a befejezési szám meghatározásával együtt is $O(n + m)$ **2 pont**

Ha DFS helyett pl. BFS fát veszünk, akkor pl. az utolsó meglátogatott pont lesz biztosan levél.

4. Egy irányított kört nem tartalmazó irányított gráfban (DAG) lefuttattuk a DFS algoritmust valamelyik pontjából indulva. Azt kaptuk, hogy az (u, v) élre teljesül, hogy $\text{mszám}(v) < \text{mszám}(u)$. Bizonyítsa be, hogy $\text{bszám}(v) < \text{bszám}(u)$ teljesül.

Megoldás: Mivel $\text{mszám}(v) < \text{mszám}(u)$, ezért (u, v) csak visszaél vagy keresztél lehet. **3 pont**

Ha visszaél lenne, akkor lenne a gráfban irányított kör, vagyis nem DAG. **4 pont**

Tehát (u, v) keresztél. **1 pont**

Keresztélre $\text{bszám}(v) < \text{bszám}(u)$ **2 pont**

5. Egy n hosszú 0-1 sorozatot olyan, legalább 4 hosszú darabokra kell szétvágni, hogy minden keletkezett részben az első két bit megegyezzen az utolsó két bittel. (Például

a 0100111110110 sorozat felvágható 3 részre: 01001, 1111, 10110.) Adjon $O(n^2)$ lépésszámú dinamikus programozást használó algoritmust, amely eldönti, hogy van-e ilyen szétvágás!

Megoldás: Legyenek a sorozat elemei s_1, \dots, s_n .

Legyen $M[i] = \text{IGAZ}$, ha a sorozat a első i bitjének van megfelelő szétvágása, különben $M[i] = \text{HAMIS}$. **2 pont**

$M[0] = \text{IGAZ}$, $i = 1, 2, 3$ -ra legyen $M[i] = \text{HAMIS}$. **1 pont**

$i \geq 4$ -re $M[i]$ akkor IGAZ, ha valamely $0 \leq j \leq i - 4$ esetén $M[j] = \text{IGAZ}$, valamint $s_{j+1} = s_{i-1}$ és $s_{j+2} = s_i$ teljesülnek. **3 pont**

Ha van ilyen j , akkor az s_1, \dots, s_j rész felvágása az s_{j+1}, \dots, s_i darabbal az s_1, \dots, s_i egy felvágását adja. **1 pont**

Akkor és csak akkor van az egész sorozatnak megfelelő felvágása, ha $M[n] = \text{IGAZ}$. **1 pont**

Adott i -re a megfelelő j keresése $O(n)$ lépés. Ez minden i -re elvégezve, az összes lépésszám $O(n^2)$. **2 pont**

Más jó megoldás, ami nem dinamikus programozást használ: **5 pont**

Az nem derül ki a feladat szövegéből, hogy ha a sorozat első és utolsó két bitje megegyezik, akkor az 1 részre vágás megfelelő-e. Mindkét féle értelmezést elfogadjuk.

6. Egy irányított gráf éllistája az élek súlyaival:

$a : (b, 6), (c, 5), (e, 8)$
 $b : (a, 6), (e, 1), (f, 2)$
 $c : (b, 2), (f, 4),$
 $e : (b, 6), (g, 3)$
 $f : (e, 1), (g, 1)$
 $g :$

Dijkstra algoritmusával határozza meg a -ból az összes többi csúcsba vezető legrövidebb út hosszát. (Indokolni nem kell, de látszódjon, lépésenként hogyan változik a távolságokat tároló $D[]$ tömb és a KÉSZ halmaz.)

Megoldás:

	a	b	c	e	f	g	KÉSZ
1.	0	6	5	8	∞	∞	$\{a\}$
2.	0	6	5	8	9	∞	$\{a, c\}$
3.	0	6	5	7	8	∞	$\{a, c, b\}$
4.	0	6	5	7	8	10	$\{a, c, b, e\}$
5.	0	6	5	7	8	9	$\{a, c, b, e, f\}$
6.	0	6	5	7	8	9	$\{a, c, b, e, f, g\}$

Az a oszlopa csupa 0, (vagy nincs is felírva) **1 pont**

Az esetek nagy részében jól választja ki, hogy melyik csúcs kerüljön át a KÉSZ

halmazba.

2 pont

Az esetek nagy részében jól végzi el a javítást.

2 pont

Ha sok számolási hiba van, akkor csak az eddigi 5 pont jár.

Ha viszont minden számolás jó, beleértve, hogy jól választotta ki a minimálisat, akkor további

5 pont

5-nél több elszámolásnál ezért a részért nem jár pont.

7. Adott egy $n \times n$ -es mátrix. Adjon $O(n^2 \log n)$ összehasonlítást használó algoritmust, amely eldönti, van-e két olyan sor, amelyek az első elem kivételével megegyeznek, viszont az első elemük meg különböző!

Megoldás: Rendezzük a sorokat először a 2. oszlop szerint.

1 pont

Bináris keresést használó beszűrásos rendezéssel vagy összefésüléses vagy kupacos rendezéssel ez $O(n \log n)$ lépés

1 pont

Így egymás utáni blokkokba rendeztük a sorokat, hogy egy blokkban a 2. elem egyenlő.

1 pont

Most rendezzük külön-külön a blokkokat a harmadik elem szerint, így olyan blokkokat kapunk, ahol a 2. és 3. oszlop elemei is egyformák. (Ehelyett rendezhetünk az egész 3. oszlop szerint a beszűrásos rendezés olyan változatával, ami az azonos elemek sorrendjét megtartja.) Ezután hasonlóan folytatjuk a többi oszlopra.

2 pont

Egy oszlop szerint a rendezés lépésszáma

$$O(n_1 \log n_1) + \dots + O(n_k \log n_k) \subseteq O\left(\left(\sum_{i=1}^k n_i\right) \cdot \log n\right) \subseteq O(n \log n),$$

ahol n_i a blokkok mérete.

2 pont

Végül minden blokkban végigmegyünk az első oszlopon. Ha nem mind egyforma, akkor találtunk két megfelelő sort. (Ehelyett lassabb algoritmus is megfelelő lehet.)

1 pont

Ennek lépésszáma összesen $O(n)$

1 pont

Az össz lépésszám $n \cdot O(n \log n) + O(n) \subseteq O(n^2 \log n)$.

1 pont

Algoritmuselmélet 1. pótZH

A rendelkezésre álló munkaidő 90 perc. Minden megoldást indokoljon!

2024. április 30.

Minden feladat egységesen 10 pontot ér. Az aláírás megszerzéséhez minimum 24 pontot kell elérni.

Kérjük, minden résztvevő írja fel a **nevét**, **NEPTUN kódját** és a **gyakorlatvezető nevét** az összetűzött lapok jobb felső sarkába.

Általános alapelvek. A pontozási útmutató célja, hogy a javítók a dolgozatokat egységesen értékeljék. Ezért az útmutató minden feladat (legalább egy lehetséges) megoldásának főbb gondolatait és az ezekhez rendelt részpontszámokat közli. Az útmutatónak nem célja a feladatok teljes értékű megoldásának részletes leírása; a leírt lépések egy maximális pontszámot érő megoldás vázlatának tekinthetők. Az útmutatóban feltüntetett részpontszámok csak akkor járnak a megoldónak, ha a kapcsolódó gondolat egy áttekinthető, világosan leírt és megindokolt megoldás egy lépéseként szerepel a dolgozatban. Így például az anyagban szereplő ismeretek, definíciók, tételek pusztán leírása azok alkalmazása nélkül nem ér pontot (még akkor sem, ha egyébként valamelyik leírt tény a megoldásban valóban szerephez jut). Annak mérlegelése, hogy az útmutatóban feltüntetett pontszám a fentiek figyelembevételével a megoldónak (részben vagy egészében) jár-e, teljes mértékben a javító hatásköre. Részpontszám jár minden olyan ötletért, rész megoldásért, amelyből a dolgozatban leírt gondolatmenet alkalmas kiegészítésével a feladat hibátlan megoldása volna kapható. Ha egy megoldó egy feladatra több, egymástól lényegesen különböző megoldást is elkezd, akkor legfőbb az egyikre adható pontszám. Ha mindegyik leírt megoldás vagy megoldásrészlet helyes vagy helyessé kiegészíthető, akkor a legtöbb részpontot érő megoldáskezdeményt értékeljük. Ha azonban több megoldási kísérlet között van helyes és (lényeges) hibát tartalmazó is, továbbá a dolgozathoz nem derül ki, hogy a megoldó melyiket tartotta helyesnek, akkor a kevesebb pontot érő megoldáskezdeményt értékeljük (akkor is, ha ez a pontszám 0). Az útmutatóban szereplő részpontszámok szükség esetén tovább is oszthatók. Az útmutatóban leírttól eltérő jó megoldás természetesen maximális pontot ér, de bizonyítás nélkül csak az előadáson szereplő tételekre és állításokra lehet hivatkozni.

1. Bizonyítsa be, hogy $(3n^3 + 4 \log n)(2n^2 - 5\sqrt{n}) \in \Omega(n^5)$.

Megoldás: Be kell látni, hogy van olyan c és n_0 , amire $(3n^3 + 4 \log n)(2n^2 - 5\sqrt{n}) \geq cn^5$
teljesül minden $n \geq n_0$ -ra. **2 pont**

$(3n^3 + 4 \log n) \geq 3n^3$ minden $n > 0$ esetén **1 pont**

$n^2 = n \cdot n \geq 5\sqrt{n}$ minden $n \geq 5$ esetén (már $n > 1,8$ -ra is igaz) **2 pont**

$2n^2 - 5\sqrt{n} \geq n^2$ minden $n \geq 5$ esetén **2 pont**

$(3n^3 + 4 \log n)(2n^2 - 5\sqrt{n}) \geq 3n^3 \cdot n^2 = 3n^5$ minden $n \geq 5$ esetén **1 pont**

Tehát például a $c = 3, n_0 = 5$ megfelelő konstansok **2 pont**

Ha valaki (tévedésből) $O(n^5)$ -t bizonyít max. **5 pont**

2. Adjon $O(n^2 \log n)$ összehasonlító algoritmust, ami eldönti, hogy az adott x_1, x_2, \dots, x_n és S racionális számok esetén van-e olyan x_i, x_j, x_k számhármasság ($1 \leq i, j, k \leq n$), hogy $x_i + x_j + x_k = S$. (A három indexnek nem kell feltétlenül különbözőnek lennie, akár mindegyik is lehet ugyanaz.)

Megoldás: Rendezzük a számokat nagyság szerint például összefésüléssel **2 pont**

- Minden $1 \leq i, j \leq n$ párra számítsuk ki az $S - x_i - x_j$ értéket **2 pont**
 Bináris kereséssel döntsük el, hogy a kapott szám szerepel-e a rendezett sorozatban. Ha igen, akkor találtunk megfelelő számokat. **2 pont**
 A rendezés lépésszáma $O(n \log n)$ **1 pont**
 Egy párra a keresés $O(\log n)$ **1 pont**
 Mivel $O(n^2)$ pár van, az összes lépésszám $O(n^2 \log n) + O(n \log n) \subseteq O(n^2 \log n)$ **2 pont**

3. Egy $2k \geq 4$ csúcsú egyszerű, irányítatlan gráf mélységi bejárása során azt tapasztaltuk, hogy minden csúcsra a befejezési és a mélységi szám különbsége kisebb, mint k . Bizonyítsa be, hogy a gráf nem összefüggő és minden komponensben legfeljebb k csúcs van.

Megoldás: Irányítatlan gráfban az 1-es mélységi számú csúcsból indított bejárás bejárja az összes vele egy komponensben lévő csúcsot **2 pont**

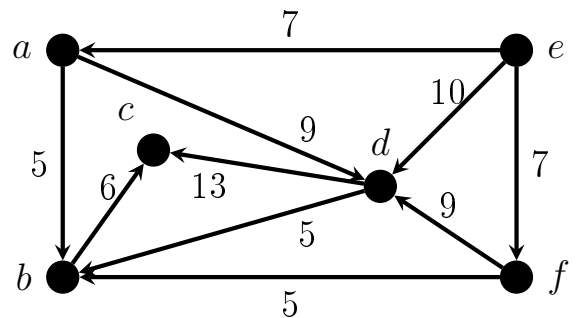
Ha az 1-es mélységi számú csúcs befejezési száma b , akkor ebben a komponensben pontosan b csúcs van **2 pont**

Mivel most $\text{bszám} - \text{mszám} = b - 1 \leq k - 1 < k$, ebben a komponensben legfeljebb k csúcs van **2 pont**

Ilyenkor a DFS egy még be nem járt csúcsból újra indítja a keresést, (ennek mélységi száma $b + 1$ lesz) **2 pont**

Az előzőekhez hasonlóan látszik, hogy ebben a komponensben is legfeljebb k csúcs van **2 pont**

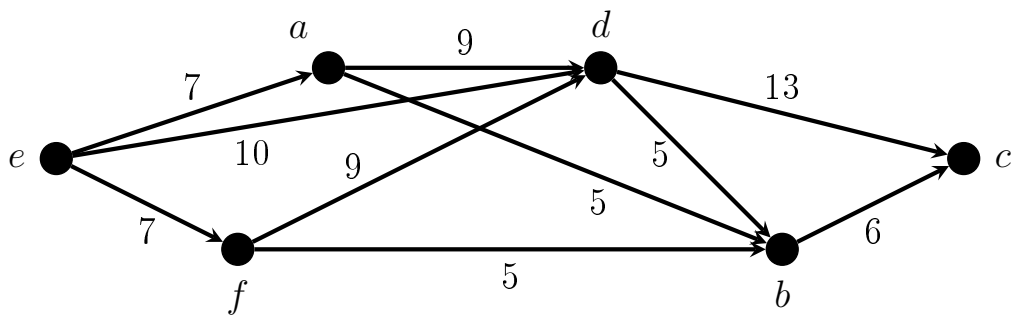
4. a) Az előadáson tanult, DFS-t használó módszerrel adja meg a jobboldali gráf egy topologikus rendezését.
 b) Határozza meg a leghosszabb út hosszát a topologikus rendezés első és utolsó pontja között az órán tanult módszerrel.



Megoldás: a) A DFS futtatásánál meg kell határozni a mélységi és befejezési számokat **1 pont**

Ha pl. e -ből indítjuk: $e(1,6), a(2,4), b(3,2), c(4,1), d(5,3), f(6,5)$ (az első a mélységi szám, a másodig a befejezési szám), a jó számok **3 pont**

Egy topologikus sorrend a befejezési számok szerinti csökkenő sorrend: e, f, a, d, b, c
 (Ha más pontból indítjuk a DFS-t, pl. f -ből, vagy más sorrendben választjuk a következő csúcsot akkor kijöhet másik sorrend is: e, a, f, d, b, c) **1 pont**



b) A leghosszabb utak e -ből indulva ($t(v)$ jelöli az e -ből v -be vezető leghosszabb út hosszát).
 Annak felírása/indoklása, hogyan kell ezt csinálni. **2 pont**

$$t(e) = 0; t(a) = 7; t(f) = 7, t(d) = \max(t(e) + 10, t(a) + 9, t(f) + 9) = 16;$$

$$t(b) = \max(t(a) + 5, t(d) + 5, t(f) + 5) = 21; t(c) = \max(t(d) + 13, t(b) + 6) = 29$$

a jó számolás

3 pont

Ha a számolásnál ki vannak írva a megfelelő maximumok, akkor azért is jár az indoklásért adandó pont.

5. A párizsi olimpiára kerékpárral szeretnénk eljutni. Az utat már megterveztük. Kiderül, hogy éppen minden km-nél van egy szállás. Az út hossza n km, egy nap legfeljebb k km-t tudunk haladni. Ha még nem értünk Párizsba, akkor az előző szállástól számítva legfeljebb k km-en belül meg kell szállnunk. Minden szállás költsége adott, az i -edikben $A[i]$ euro. Adjon algoritmust, ami dinamikus programozást használva meghatározza az út során a szállások minimális összköltségét. Az algoritmus lépésszáma legyen $O(kn)$.

Megoldás: Legyen $C[i]$ a minimális költsége annak, hogy eljutottunk az i -edik km-ig. (Ha megszállunk az i -edik km-nél, azt még nem számoljuk bele. Az indulás városában való megszállást sem számoljuk, azaz $A[0] = 0$.) $C[0] = 0$ **2 pont**

$$C[i] = \min\{C[j] + A[j] \mid i - k \leq j < i \text{ és } j \geq 0\}$$
 3 pont

Helyesség indoklása: Az előző k km-en belül meg kellett szállni, ez legyen az j -edik km. Eddig már tudjuk, hogy mennyi a minimális költség ($C[j]$) és ki kell fizetni még $A[j]$ -t.

2 pont

A keresett érték $C[n]$

1 pont

Lépésszám: minden $C[i]$ kiszámolásához k szám minimumát kell meghatározni, ez $O(k)$ lépés, n értéket kell kiszámolni, a lépésszám összesen $O(kn)$. **2 pont**

6. Jobboldalon látható egy irányított, súlyozott gráf szomszédossági mátrixa. Dijkstra algoritmusával határozza meg a -ból az összes többi csúcsba vezető legrövidebb út hosszát. (Indokolni nem kell, de látszódjon, lépésenként hogyan változik a távolságokat tároló $D[]$ tömb és a KÉSZ halmaz.)

	a	b	c	e	f	g
a	∞	6	5	8	∞	∞
b	6	∞	∞	1	4	∞
c	∞	2	∞	∞	6	∞
e	∞	6	∞	∞	∞	2
f	∞	∞	∞	1	∞	1
g	∞	∞	∞	∞	∞	∞

Megoldás:

	a	b	c	e	f	g	KÉSZ
1.	0	6	5	8	∞	∞	$\{a\}$
2.	0	6	5	8	11	∞	$\{a,c\}$
3.	0	6	5	7	10	∞	$\{a,c,b\}$
4.	0	6	5	7	10	9	$\{a,c,b,e\}$
5.	0	6	5	7	10	9	$\{a,c,b,e,g\}$
6.	0	6	5	7	10	9	$\{a,c,b,e,g,f\}$

Az a oszlopa csupa 0, (vagy nincs is felírva)

1 pont

Az esetek nagy részében jól választja ki, hogy melyik csúcs kerüljön át a KÉSZ halmazba.

2 pont

Az esetek nagy részében jól végzi el a javítást.

2 pont

Ha sok számolási hiba van, akkor csak az eddigi 5 pont jár.

Ha viszont minden számolás jó, beleértve, hogy jól választotta ki a minimálisat, akkor további

5 pont

5-nél több elszámolásnál ezért a részért nem jár pont.

7. Adott egy városok közötti úthálózatot leíró irányított gráf éllistája, ahol csúcsok a városok, élek a közvetlen utak és az élek súlya a városok közötti közvetlen útszakaszok hosszát adja meg. Egyes városokban nevezetességek is vannak, ezek száma minden városra meg van adva egy tömbben (a nevezetességek száma mindegyik városnál egy nemnegatív egész szám). Szeretnénk eljutni az A városból a B városba, az elsődleges szempont, hogy az út a lehető legrövidebb legyen. De ha esetleg több egyforma hosszú legrövidebb út is van, akkor ezek közül azt akarjuk kiválasztani, ami során az érintett városokban a lehető legtöbb nevezetességet tudjuk megnézni. Adjon algoritmust egy ilyen út megkeresésére, az algoritmus lépésszáma legyen $O(n^2)$.

Megoldás: Módosítsuk a Dijkstra algoritmust, minden csúcsra az eddigi legrövidebb utat tároló $D[]$ tömbön kívül egy plusz információt is tároljunk. Mindegyik legrövidebb úton megszámloljuk hány nevezetesség látható. $N[]$ tömbben ezek maximumát.

2 pont

Az algoritmust úgy módosítjuk, hogy minden körben azt a csúcsot tesszük át a KÉSZ halmazba, amire a $D[]$ érték minimális, de ha több minimális van, akkor azok közül azt választjuk, amire $N[]$ maximális.

2 pont

Amikor áttettük a v csúcsot, akkor a $D[]$ és $N[]$ értékeket értelemszerűen módosítjuk a v -ből kimenő éleken.

2 pont

Az algoritmus helyességének bizonyítása nagyon hasonló a Dijkstra algoritmus helyességének bizonyításához. Csak a következőt kell meggondolni: A kezdőpontból induló bármely út elejére teljesül, hogy az út hossza nem lehet nagyobb, mint az egész út hossza (mivel az élsúlyok nemnegatívak), valamint az első részen nem lehet több látnivaló, mint az egész úton.

2 pont

A lépésszám ugyanannyi, mint a Dijkstra lépésszáma, ami $O(n^2)$

2 pont

2. megoldás: Bár a feladat szövegében nem szerepel, feltételezhetjük, hogy minden közvetlen útszakasz hossza pozitív egész. A súlyok átskálázásával ez elérhető. Ezt indoklás nélkül is elfogadjuk. Legyen a összes város látnivalóinak számának összege C . Szorozzunk meg minden élsúlyt $2nC$ -vel. Az így kapott G' gráfban ugyanaz lesz a legrövidebb út bármely két pont között, mint az eredeti G gráfban, csak a hossza lesz $2nC$ -szerese. **2 pont**

Most minden v csúcsot kettőzünk meg, legyen egy v' csúcs is. Minden v -ből kimenő él menjen ki most v' -ből és legyen egy él v -ből v' -be. Ennek súlya legyen C -ből kivonva a v város látnivalóinak száma. **2 pont**

Keressük meg G' -ben a legrövidebb utat A -ból B -be, ez adja meg a keresett utat. **2 pont**

Helyesség: A új élek súlyai minden csúcson legfeljebb C -vel növelik az út hosszát, így bármely úton legfeljebb $2nC$ -vel nő a hossz. Mivel ez kisebb, mint $2nC$, ezek nem befolyásolják azt, hogy az algoritmus egy eredeti legrövidebb útnak megfelelőt fog megtalálni.

Viszont ezek között a legtöbb látnivalót adót adja meg. **2 pont**

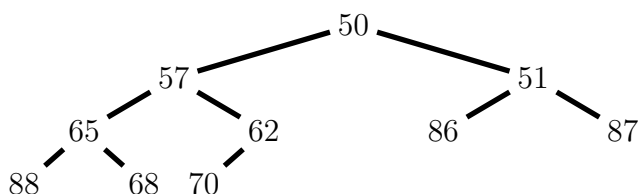
A lépésszám ugyanannyi, mint a Dijkstra lépésszáma, $O(n^2)$, ebbe belefér az G' előállítás is. **2 pont**

A munkaidő 90 perc. A VÁLASZOKAT INDOKOLNI KELL.
Hivatkozni csak az előadáson tanultakra lehet.

1. (a) Rajzolja fel az 50, 57, 51, 65, 62, 86, 87, 88, 68, 70 tömbbel adott kupacot bináris fa alakban. *(Itt indoklás nem szükséges.)*
(b) Szűrje be ebbe a kupacba (a fa alakban) az 55-t, majd hajtson végre egy MINTÖR-t, a megoldása során látszódnak az egyes lépései a műveleteknek.

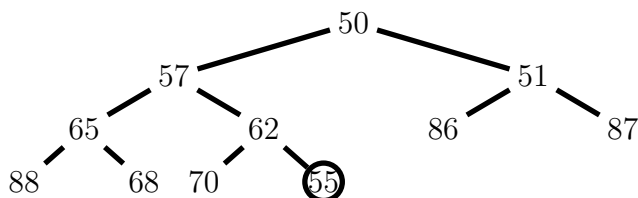
Megoldás: (a) Jól felrajzolt fa:

2 pont



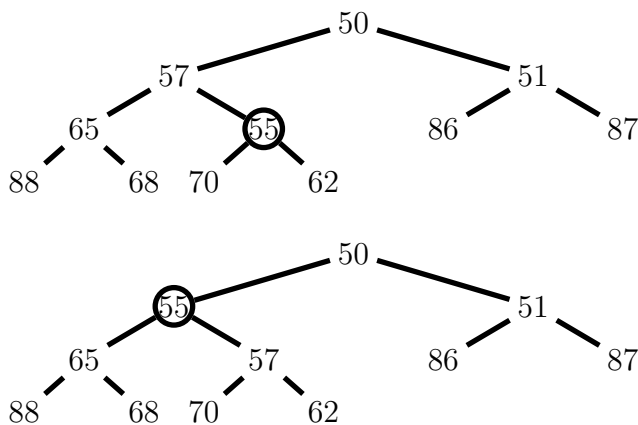
(b) Jó helyre illeszti be az új elemet:

1 pont



Kiderül, hogy felfele mozgatja az elemet (akár azért, mert lerajzolja lépésenként vagy jelöli a cseréket vagy leírja, hogy mi történik):

1 pont

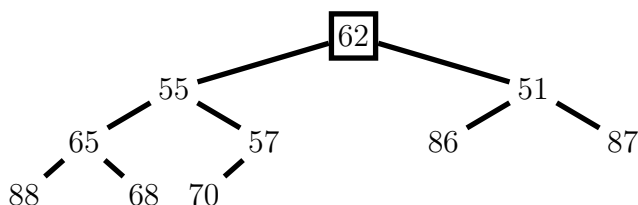


Jó kupac a végén:

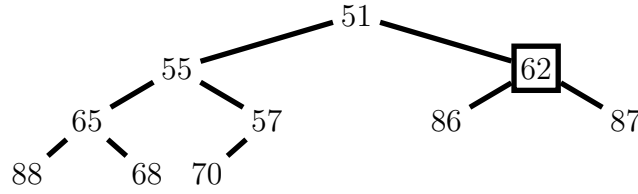
2 pont

MINTÖR-nél a 62-t felviszi a gyökérbe:

1 pont



Kiderül, hogy lefelé, a kisebb gyerek helyére mozgatja az elemet (akár azért, mert lerajzolja lépésenként vagy jelöli a cseréket vagy leírja, hogy mi történik): 1+1 pont



Jó kupac a végén: 1 pont

2. Adott két bináris keresőfa. Az egyik a csupa különböző a_1, \dots, a_n elemeket tárolja és a magassága $2024 \log n$. A másik a csupa különböző b_1, \dots, b_n elemeket tárolja és magassága $n/2024$. Tudjuk, hogy minden a_i kisebb minden b_j -nél. Adjon olyan $O(\log n)$ lépésszámú algoritmust, ami előállít egy olyan bináris keresőfát, ami éppen az összes a_i -t és b_j -t tárolja.

Megoldás: Megkeressük az első fa maximális elemét és töröljük az első fából. Tegyük fel, hogy ez épp a_n . 2 pont

Ez $O(\log n)$ lépés, mert ennek a fának a magassága $2024 \log n \in O(\log n)$. 1 pont

Az új fa gyökere legyen a_n . 2 pont

A bal részfája legyen az első fa maradéka (a_n törlése után). A jobb részfája legyen a második fa. 2 pont

Így a gyökérre teljesül a keresőfa tulajdonság a_n választása miatt. A többi pontra eddig is teljesült. 2 pont

Az új mutatók beállítása konstans sok lépés 1 pont

2. Megoldás: Megkeressük az első fa maximális elemét. Tegyük fel, hogy ez épp a_n . 2 pont

Ez $O(\log n)$ lépés, mert ennek a fának a magassága $2024 \log n \in O(\log n)$. 1 pont

a_n -nek nem lehet jobb gyereke, mert maximális elem (ez volt előadáson). 2 pont

A második fa gyökere legyen az a_n jobb gyereke. 2 pont

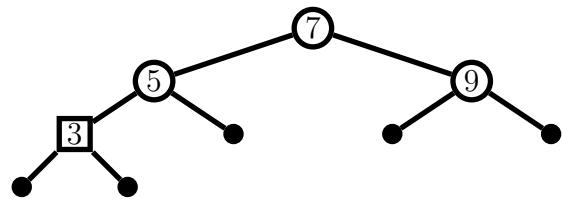
Így a_n -re is teljesül a keresőfa tulajdonság a_n választása miatt és ezért a_n őseire is. A többi pontra eddig is teljesült. 2 pont

Az új mutatók beállítása konstans sok lépés 1 pont

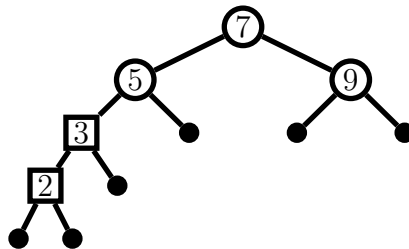
3. Szűrje be a következő piros-fekete fába a 2-t.

(A \bigcirc csúcs fekete, a \square pedig piros.)

A megoldásban látszódnak a beszúrás egyes részlépései, de indokolni nem kell.

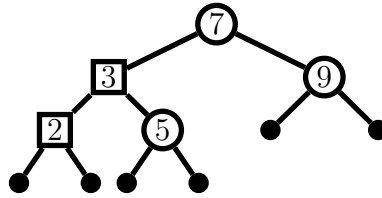


Megoldás: Először beszúrjuk naiv módon a 2-t és az új elemet pirosra színezi:



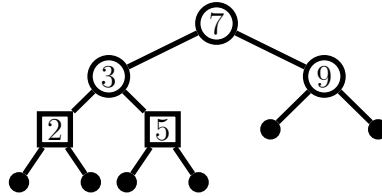
2+1 pont

Az 3-as csúcs piros, testvére fekete, ezért átszínezni nem lehet. Mivel az 3 és 2 azonos oldali gyerekek, forgatni kell, hogy az 3 feljebb kerüljön:



4 pont

Átszínezés:



3 pont

Ha nincs magyarázat, hogy mikor melyik lépést használjuk, de jól vannak végrehajtva a műveletek, akkor is jár a teljes pontszám.

A fenti pontok csak akkor járnak, ha az algoritmus használatával készültek el a fák. De ha az utolsó fa jól fel van rajzolva, akkor azért 1 pont jár.

4. Az alábbi, 11 méretű hash táblába nyílt címzéssel, **kvadratikus próbával** szűrje be a 25, majd a 14 számot a $h(x) = x \pmod{11}$ hash függvényvel. (A * jel a törölt elem helyét jelzi.) A megoldásban látszódjon milyen sorrendben vizsgáljuk meg a különböző cellákat.

0	1	2	3	4	5	6	7	8	9	10
*		57	47	37			40		20	*

Megoldás: A próba sorozat: 0, 1, -1, 4, -4, 9, -9, ...

$25 \equiv 3 \pmod{11}$, sorban a 3, 4, 2, 7, 10 cellákat próbáljuk.

A törölt jel helyére be lehet szűrni új elemet.

A 10. cellába tesszük.

3 pont

2 pont

1 pont

1 pont

0	1	2	3	4	5	6	7	8	9	10
*		57	47	37			40		20	25

$14 \equiv 3 \pmod{11}$, sorban a 3, 4, 2, 7, 10, 1 cellákat próbáljuk.

Az 1. cellába tesszük.

2 pont

1 pont

0	1	2	3	4	5	6	7	8	9	10
*	14	57	47	37			40		20	25

Ha a próba sorozatként: 0, -1, 1, -4, 4, -9, 9, ...-et használja, akkor ezért de helyes számolás után a többi pont jár.

-1 pont

Ha valamilyen kvadratikus próbára emlékeztető próbasorozat használ a megoldás, de nem a megfelelő, akkor az első 3 pont nem jár, de a többi meg lehet adni.

Lineáris próbával való beszúrásért, ha minden stimmel.

2 pont

5. Szomszédossági mátrixával adott egy irányítatlan gráf. Bizonyítsa be, hogy a következő probléma NP-ben van: Van-e a gráf minden u és v pontja között Hamilton-út? (A Hamilton-út két végpontja között lehet él, attól még Hamilton-útnak számít.)

Megoldás: Azt kell belátni, hogy a probléma NP-beli, azaz van rövid, gyorsan ellenőrizhető tanú a probléma „igen” inputjaira. **1 pont**

Jó tanú lesz ha minden u, v pontpárra megadunk egy Hamilton-utat. **2 pont**

Egy Hamilton-utat a csúcsok sorszámainak sorozatával lehet megadni, ami $p \log p \in O(n)$, ahol p a pontok száma. (Elég azt írni, hogy $O(n)$.) **1 pont**

A pontpárok száma $O(p^2) \subseteq O(n)$, tehát a tanú mérete $O(n^2)$. **2 pont**

Az ellenőrzés során meg kell nézni:

• Minden út Hamilton-út? **1 pont**

• Minden pontpár szerepel, mint valamelyik Hamilton-út két végpontja? **1 pont**

Egy Hamilton-út ellenőrzése elvégezhető $O(n)$ lépésben (ez volt előadáson), minden út ellenőrzése $O(n^2)$ lépésben. **1 pont**

Egy adott pontpárra végignézve a teljes tanút, megnézhetjük, hogy a pontpár szerepel-e végpontként, ez $O(n^2)$. Az összes párra pedig $O(p^2n^2) \subseteq O(n^3)$. Az ellenőrzés összideje $O(n^2) + O(n^3) \subseteq O(n^3)$, ami polinomiális. **1 pont**

A lépésszámokra bármilyen helyes korlát elfogadható, nem csak a fentiek. Van ezeknél jobb is, de gyengébb is megfelelő, ha polinomiális.

6. P-ben van vagy NP-teljes az alábbi NP-beli eldöntési feladat? (Azt a tényt fel szabad használni, hogy ez a feladat NP-ben van.)

Input: G irányítatlan gráf szomszédossági mátrixával, k, ℓ egész számok (tízes számrendszerbeli alakjukkal).

Kérdés: Igaz-e, hogy G -ben van egy k és egy ℓ méretű klikk (teljes részgráf), melyeknek nincs közös csúcsuk?

Megoldás: Nevezzük a fenti problémát KÉTKLIKK problémának.

Ha a KÉTKLIKK probléma NP-beli és visszavezethető rá egy NP-teljes probléma, akkor KÉTKLIKK is NP-teljes. Az Iszonyú Hasznos Tétel miatt ez már bizonyítja az NP-teljességet. Az NP-beliséget nem kell most bizonyítani a feladat szerint. (Ha ezek nincsenek így leírva, de aztán ennek megfelelően folytatódik a bizonyítás, akkor is jár ez a pont.) **1 pont**

Megadunk egy MAXKLIKK \rightarrow KÉTKLIKK Karp-redukciót. **2 pont**

A redukció során egy (G, k) párhoz azt a (G', k', ℓ') hármast rendeljük, amiben $k' = \ell' = k$ és G' -t úgy kapjuk G -ből, hogy felveszünk G mellé egy még egy példányban G -t. **2 pont**

(G', k', ℓ') előállítás gyors, mert a másik G hozzávétele $O(n)$ -ben megvan. **1 pont**

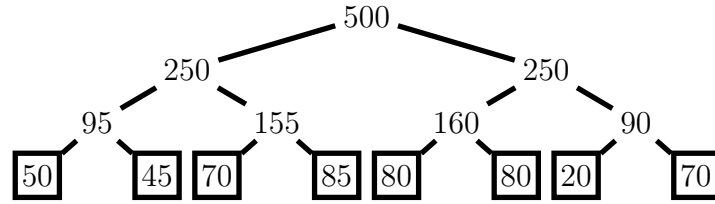
Ha G -ben van k -as klikk ponthalmaz, akkor G mindkét példányában van egy k -as klikk és ezeknek nincs közös pontja. Azaz G' -ben van diszjunkt k' -s és ℓ' -s klikk. **2 pont**

Ha G' -ben van diszjunkt k' -s és ℓ' -s klikk, akkor a k -as klikk G egyik példányában kell, hogy legyen (mindegy, hogy a másik klikk hol van), tehát G -ben van k -as klikk. **2 pont**

Ha nem áll össze a bizonyítás, de megjelenik a MAXKLIKK, ennek NP-teljessége és valamiféle kapcsolatot, akkor lehet adni max **3 pont**

7. A World Of SZIT számítógépes játéknak $n = 2^q$ (q egész) szintje van. (A szintek számozását 0-val kezdjük.) A játék során az i -edik szinten $p[i]$ pontot lehet szerezni. A pontokat a következő adatstruktúrában tároljuk: Egy teljes bináris fa leveleit balról jobbra számozzuk, az i . levél tárolja

a $p[i]$ pontszámot (a balszélső a nulladik). Az i sorszámot **nem** tároljuk a levélben. Egy nem levél csúcsban levő szám a csúcs gyerekeiben levő számok összege. Például, ha a $p[i]$ pontok rendre 50, 45, 70, 85, 80, 20, 70, akkor így néz ki a fa:



A $SZINT(k)$ művelet eredménye, hogy a játék hányadik szintjén lehet először elérni összesítésben k pontot. A szint sorszámának bináris alakját szeretnénk megkapni. Adjon olyan algoritmust a $SZINT(k)$ művelet megvalósítására, ami $O(\log n)$ lépést használ. Például, $SZINT(100) = 2$. (Az adatstruktúra adott, csak a $SZINT(k)$ műveletet kell megvalósítani.)

Megoldás: Legyen a gyökér x .

Ha $k > elem(x)$, akkor nem lehet k pontot szerezni a játékban.

1 pont

Hasonlítsuk össze k -t $elem(bal(x))$ -el. Ha $k \leq elem(bal(x))$, akkor balra kell továbblépni, mert k pontot a játék első $n/2$ szintjének valamelyikén lehet megszerezni, hiszen $elem(bal(x))$ értéke az első $n/2$ szinten megszerezhető összes pontszám.

1 pont

Ha $k > elem(bal(x))$, akkor jobbra kell lépni, mert k pontot a játék második $n/2$ szintjének valamelyikén lehet megszerezni.

1 pont

Ha $k \leq elem(bal(x))$, akkor a $bal(x)$ gyökerű részében folytatjuk a k keresését rekurzívan.

1 pont

Ha $k > elem(bal(x))$, akkor a $jobb(x)$ gyökerű részében folytatjuk a keresését rekurzívan, de a $k - elem(bal(x))$ értékkel,

1 pont

mert ebben a részében már csak az $elem(bal(x))$ értéken túli pontok szerepelnek.

1 pont

Amikor egy levélhez érünk, akkor megtaláltuk a megfelelő levelet. Azt kell meghatározni, hogy az itt lévő pont hanyadik szinthez tartozik, azaz hanyadik levél balról számítva.

1 pont

A lépések során minden balra lépésnél feljegyezzük egy 0-t, jobbra lépéskor pedig 1-et írunk fel. Az így kapott szám a keresett szintszám bináris alakja lesz.

2 pont

Mivel teljes bináris fáról van szó, minden út $q = \log_2 n$ hosszú (ez volt előadáson), tehát a lépésszám $O(\log n)$.

1 pont

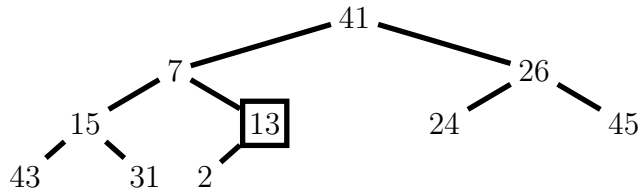
A munkaidő 90 perc. A VÁLASZOKAT INDOKOLNI KELL.
Hivatkozni csak az előadáson tanultakra lehet.

1. Egy (majdnem) teljes bináris fa tömb reprezentációja 41, 7, 26, 15, 13, 24, 45, 43, 31, 2. Ábrázolja ennek fa alakját és hajtsa végre az órán tanult kupacépítés algoritmust rajta. (Minden lépés után adja meg az aktuális állapotot, de indokolni nem kell.)

Megoldás:

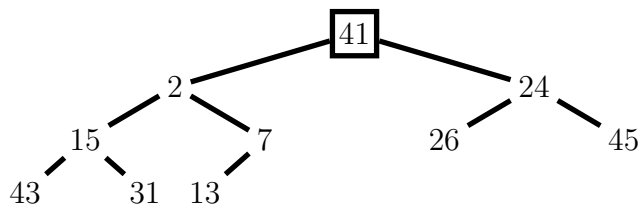
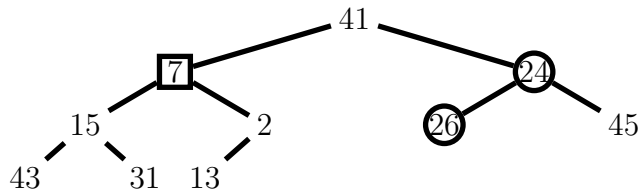
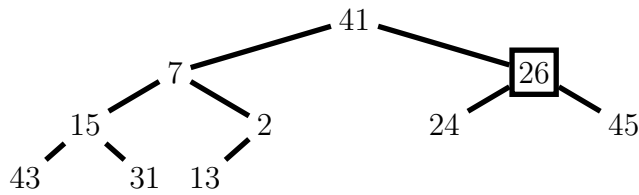
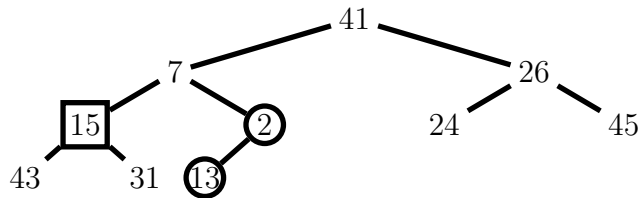
Jó fa.

2 pont



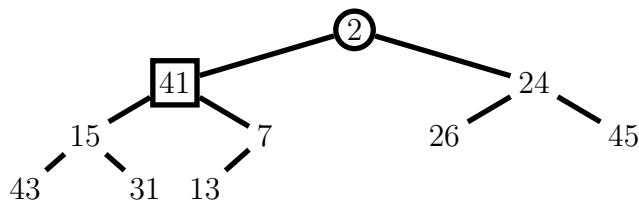
Jó helyről indul a kupacol eljárás.

1 pont



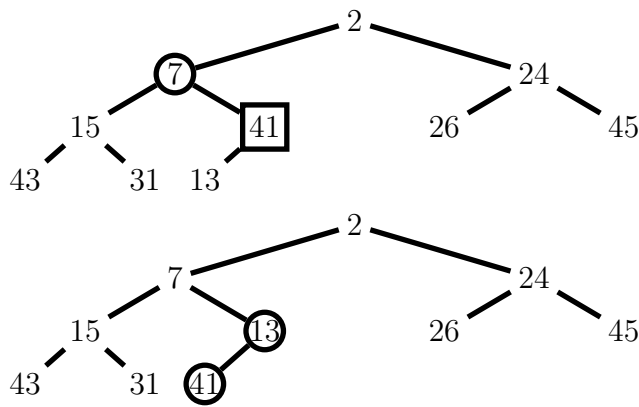
Jó lépések eddig.

3 pont



A 41-nek le kell szivárognia.

2 pont



Jó cserék, mindig a kisebb kerül feljebb.

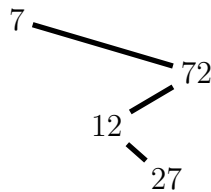
2 pont

2. Az alábbi bináris keresőfába

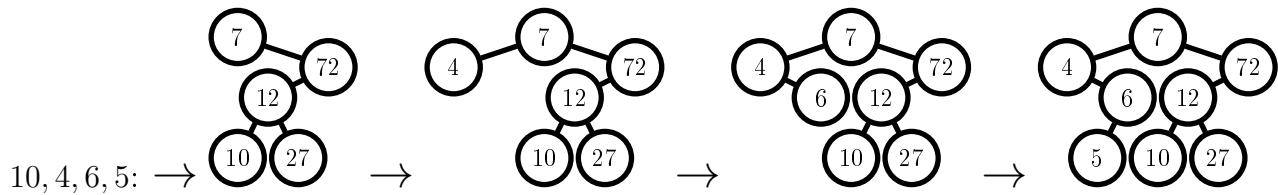
(a) Szúrja be sorban a 10, 4, 6, 5 számokat az órán tanult módszerrel.

(b) Ezután törölje ki a 7-et, majd pedig a 72-t az órán tanult módszerrel..

Minden lépés után adja meg az aktuális állapotot, de indokolni nem kell.

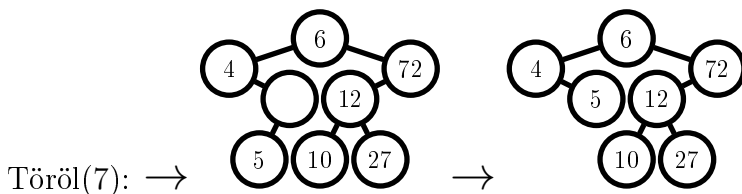


Megoldás:



Jó faépítés.

5 pont

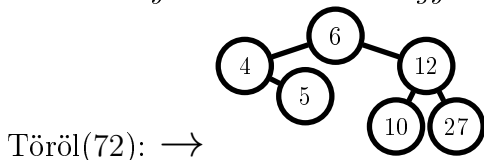


A 7 jó törlése.

Ha a 6 helyett a 10-et teszi a gyökérbe, akkor

3 pont

-1 pont



Töröl(72): →

A 72 jó törlése.

2 pont

3. Mekkora lehet egy olyan piros-fekete fa magassága, amiben 7 elemet tárolunk? Adja meg az összes lehetséges értéket.

Megoldás: A magasság a gyökértől a levélig vezető úton az élek száma.

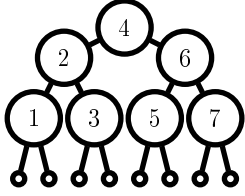
Nem kell levonni, ha más magasság definícióval számol.

A magasság legalább 3, hiszen 2 magasságú fában nem fér el, csak 3 elem.

2 pont

3 lehet is.

1 pont

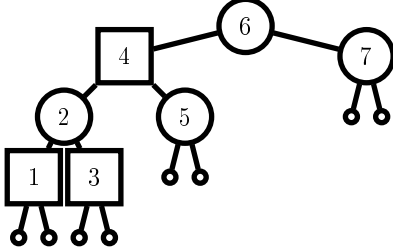


4 is lehet.

1 pont

Indoklás vagy példa a 4 magasságúra.

2 pont



4-nél több nem lehet, mert akkor a leghosszabb ágon legalább 5 él kellene legyen, emiatt pedig (mivel a levél és a gyökér is fekete, legfeljebb 2 piros csúcs lehet rajta (2. és 4. vagy 2. és 5. vagy 3. és 5.)), tehát kell legyen legalább 3+1 (3 belső csúcs, plusz 1 levél) fekete csúcs. Azaz, mivel a fekete-magasság minden irányba ugyanannyi, bármely ágon kell legyen 3+1 fekete csúcs, ez pedig azt jelenti, hogy legalább 7 fekete elem van a fa belsejében. Ez pedig ellentmondás, hiszen akkor nem lehet piros csúcs már, ha 7 elemet tárolunk, hisz mind fekete. Tehát 4-nél több nem lehet a magasság. **4 pont**

4. A kezdetben üres $M = 8$ méretű hashtáblába a $h(x) = 5x \pmod{M}$ hash-függvény és a $h'(x) = (x \pmod{3}) + 1$ másodlagos hash-függvény segítségével az adott sorrendben szűrja be a 3, 2, 4, 11, 5, 1 elemeket. Ábrázolja az egyes beszúrások menetét is!

Megoldás: Kettős hashelésnél a próbasorozat: $h_i = -i \cdot h'(K)$.

2 pont

Ha nem ezzel a próbasorozattal számol, de valami hasonlóval, akkor

-1 pont

Ha nem a $h(x) = 5x \pmod{M}$ hash függvényt használja.

-1 pont

A 3, 2, 4, 1 esetén a hash-függvény által adott cella üres, ezek jó beszúrása.

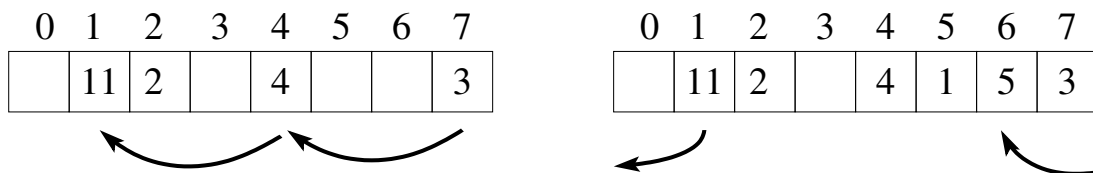
3 pont

11 esetén a 7, 4, 1 cellákat próbáljuk,

3 pont

az 5 esetén pedig az 1, 6 cellákat.

2 pont



5. Bizonyítsa be, hogy ha egy X eldöntési problémáról be tudnánk látni, hogy $X \in \text{NP}$, de $X \notin \text{P}$, akkor $\text{MAXFÜGGETLEN} \notin \text{P}$ is teljesülne.

Megoldás: Tegyük fel indirekt, hogy $\text{MAXFÜGGETLEN} \in \text{P}$

1 pont

Tudjuk, hogy $\text{MAXFÜGGETLEN} \in \text{NP}$ -teljes.

1 pont

Ha egy NP-teljes P-ben van, akkor $\text{P} = \text{NP}$, ez volt előadáson (vagy indoklás).

2 pont

Ezért, ha $\text{MAXFÜGGETLEN} \in \text{P}$ -ben van, akkor ebből következik, hogy $\text{P} = \text{NP}$.

4 pont

Innen ellentmondás, mert így $X \in P$ is teljesülne.

2 pont

6. Egy céges vacsorán a résztvevők között vannak olyanok, akik nem kedvelnek néhány másik résztvevőt. A „nem kedvelés” kölcsönös. Adottak az egymást nem kedvelő párok.

P-ben van vagy NP-teljes az alábbi NP-beli eldöntési feladat? *(Az a tényt fel szabad használni, hogy ez a feladat NP-ben van.)*

Input: Az egymást nem kedvelő párok listája.

Kérdés: Le lehet-e ültetni a résztvevőket 2 egyforma méretű kör alakú asztal köré úgy, hogy ne üljön egymás mellett két olyan résztvevő, aki nem kedveli egymást?

Megoldás: Nevezzük a fenti problémát ASZTAL problémának.

Ha az ASZTAL probléma NP-beli és visszavezethető rá egy NP-teljes probléma, akkor ASZTAL is NP-teljes. Az Iszonyú Hasznos Tétel miatt ez már bizonyítja az NP-teljességet. Az NP-beliséget nem kell most bizonyítani a feladat szerint. *(Ha ezek nincsenek így leírva, de aztán ennek megfelelően folytatódik a bizonyítás, akkor is jár ez a pont.)*

1 pont

Megadunk egy $H \prec$ ASZTAL Karp-redukciót.

1 pont

A redukció során egy G gráfhoz először azt a G' gráfot rendeljük, amiben G' -t úgy kapjuk G -ből, hogy a G gráfból, veszünk két diszjunkt példányt, majd az így kapott gráfnak vesszük a komplementerét. Ezután G' élei legyenek az egymást nem kedvelő párok listája.

2 pont

Ennek előállítása gyors, mert a másik G hozzávétele $O(n)$ -ben megvan és az éllista is gyorsan előállítható (akár mátrixszal, akár éllistával van adva G).

1 pont

Ha G -ben van Hamilton-kör, akkor G' pontjainak van egy felsorolása, amiben a szomszédosak között sehol nincs él. Ebből kapunk megfelelő ültetést mindkét asztalhoz.

2 pont

Ha G' -ben van megfelelő ültetés, akkor ez G -ben két olyan kört ad, amik G két példányában egy-egy Hamilton-kört ad, tehát G -ben van Hamilton-kör.

3 pont

Ha nem áll össze a bizonyítás, de lefordítja a feladatot gráfelméleti nyelvre, megjelenik a H probléma, ennek NP-teljessége és valamiféle kapcsolat, akkor lehet adni max

4 pont

7. Írja fel a HÁTIZSÁK feladat optimalizációs változatát egészértékű programozási feladatként.

Megoldás: Legyen a HÁTIZSÁK inputja: $s_1, \dots, s_m; v_1, \dots, v_m; b$

1 pont

Maximalizálni akarjuk a bepakolt tárgyak összértékét.

1 pont

Az x_i változó legyen 1, ha az i -edik tárgyat bepakoltuk, és 0 különben.

2 pont

Az egyenlőtlenségek: $\sum_{i=1}^m s_i x_i \leq b$

3 pont

Minden $0 \leq x_i \leq 1$ és x_i egész.

1 pont

Célfüggvény: $\max \sum_{i=1}^m v_i x_i$

2 pont