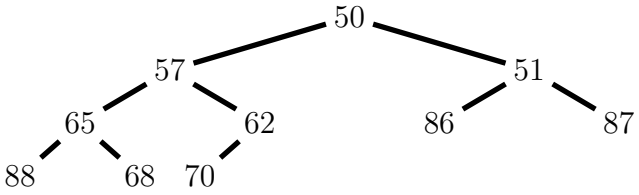
Algoritmuselmélet - 2. ZH
2024 tavasza

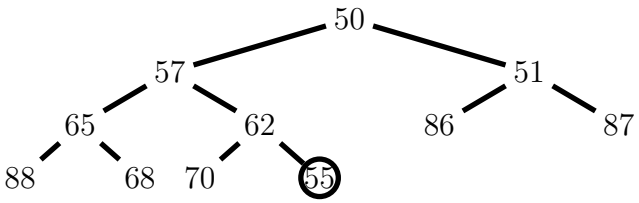
A munkaidő 90 perc. A VÁLASZOKAT INDOKOLNI KELL.
Hivatkozni csak az előadáson tanultakra lehet.

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) Szúrja be ebbe a kupacba (a fa alakban) az 55-t, majd hajtson végre egy MINTÖR-t, a megoldása során látszódjanak az egyes lépései a műveleteknek.

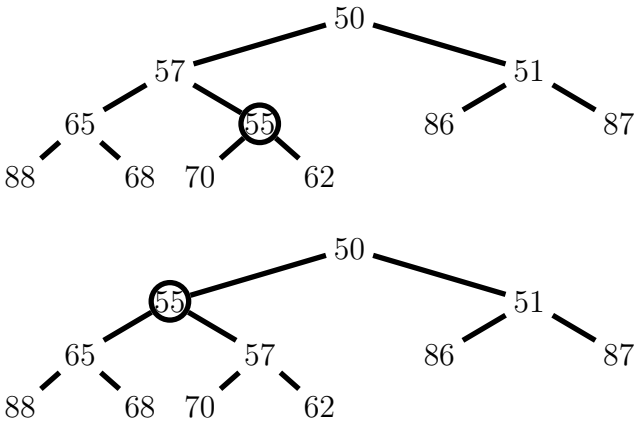
Megoldás: (a) Jól felrajzolt fa: **2 pont**



(b) Jó helyre illeszti be az új elemet: **1 pont**

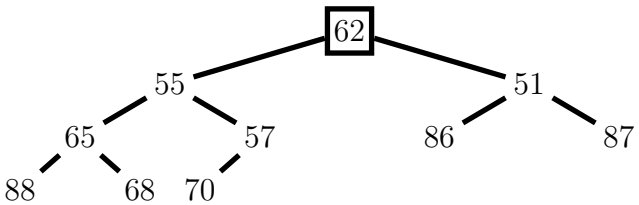


Kiderül, hogy felfele mozgatja az elemet (akár azért, mert lerajzolja lépésenként vagy jelöli a cseréket vagy leírja, hogy mi történik): **1 pont**

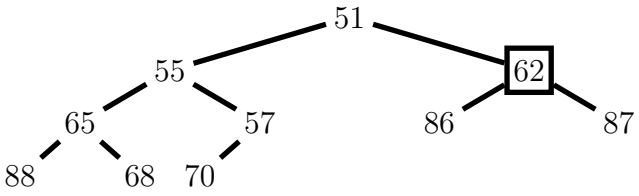


Jó kupac a végén: **2 pont**

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**



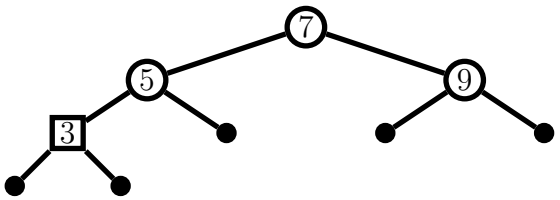
Jó kupac a végén: **1 pont**

2. Adott két bináris keresőfa. Az egyik a csupa különböző $a_1, \ldots, a_n$ elemeket tárolja és a magassága $2024 \log n$. A másik a csupa különböző $b_1, \ldots, b_n$ 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.

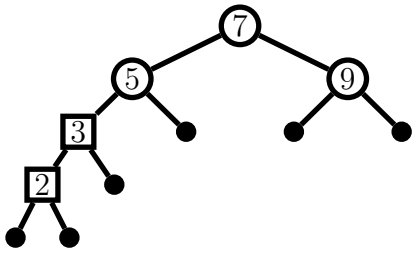
Megoldás: Megkeressük az első fa maximális elemét és töröljük az első fából. Tegyük fel, hogy ez épp a_n. **2 pont**
Ez $O(\log n)$ lépés, mert ennek a fának a magassága $2024 \log n \in O(\log n)$. **1 pont**
Az új fa gyökere legyen a_n. **2 pont**
A bal részfája legyen az első fa maradéka (a_n törlése 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. Szúrja be a következő piros-fekete fába a 2-t.
(A ○ csúcs fekete, a □ pedig piros.)
A megoldásban látszódjanak a beszúrás egyes részlépései, de indokolni nem kell.

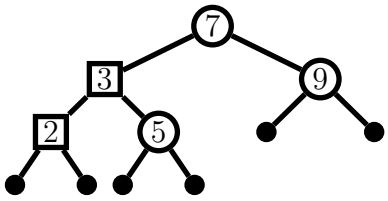


Megoldás: Először beszúrjuk naiv módon a 2-t és az új elemet pirosra színezi:



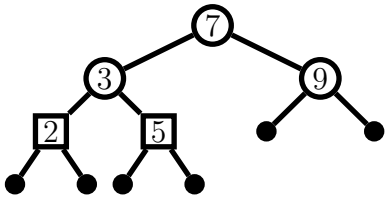
2+1 pont

Az 3-as csúcs piros, testvére fekete, ezért átszínezni nem lehet. Mivel az 3 és 2 azonos oldali gyerek, 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** szúrja be a 25, majd a 14 számot a $h(x) = x \pmod{11}$ hash függvénnyel. (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: $0, 1, -1, 4, -4, 9, -9, \ldots$ **3 pont**
$25 \equiv 3 \pmod{11}$, sorban a $3, 4, 2, 7, 10$ cellákat próbáljuk. **2 pont**
A törölt jel helyére be lehet szúrni új elemet. **1 pont**
A 10. cellába tesszük. **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. **2 pont**
Az 1. cellába tesszük. **1 pont**

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

Ha a próba sorozatként: $0, -1, 1, -4, 4, -9, 9, \ldots$*-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 megfelelőt, akkor az első 3 pont nem jár, de a többit 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 $O(p^2 n^2) \subseteq O(n^3)$. 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.)*

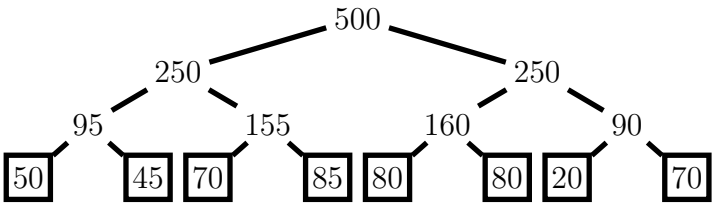
Input: G irányítatlan gráf szomszédossági mátrixával, k, ℓ egész számok (tízes számrendszerbeli alakjukkal).

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 $\prec$ 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') előállítása 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 kapcsolat, 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 $50, 45, 70, 85, 80, 80, 20, 70$, 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 részfában folytatjuk a k keresését rekurzívan. **1 pont**
Ha $k > elem(bal(x))$, akkor a $jobb(x)$ gyökerű részfában folytatjuk a keresését rekurzívan, de a $k - elem(bal(x))$ értékkel, **1 pont**
mert ebben a részfában 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 feljegyzünk 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**