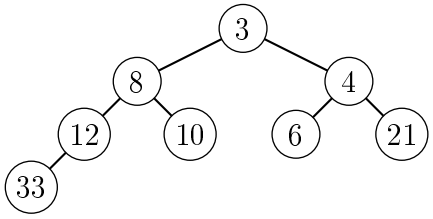
Algoritmuselmélet - Minta 2. ZH, megoldásokkal
2024 tavasza

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

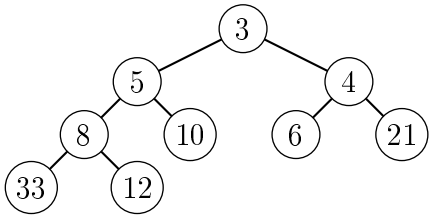
1. (a) Rajzolja fel a $3, 8, 4, 12, 10, 6, 21, 33$ tömbbel adott kupacot bináris fa alakban. *(Itt indoklás nem szükséges.)*
(b) Szúrja be ebbe a kupacba (a fa alakban) az 5-t, majd hajtson végre egy MINTÖR-t, a megoldása során látszódjanak az egyes lépései a műveleteknek.

Megoldás:

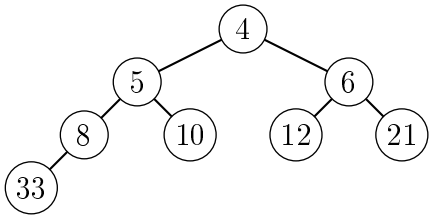
(a)



(b) Az 5-t a 12 jobb gyerekeként illesztjük be a fába, majd felfele mozgatjuk, felcserélve a szülővel addig, amíg már a szülője kisebb lesz nála, azaz 12-vel és 8-cal kell cseréni, de 3-mal már nem. Ezt kapjuk a végén:



A MINTÖR a 3 helyére teszi a 12-t, majd a 12-t addig cseréli fel a kisebb gyerekével, amíg már nem lesz magánál kisebb gyereke, ebben az esetben ez akkor következik be, amikor a 12 levélbe kerül, azaz a 4-gyel és a 6-tal kellett cserélni. Ezt kapjuk:



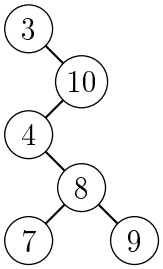
A pontozás vázlata

(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 12-t felviszi a gyökérbe: **1 pont**
Kiderül, hogy lefele 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**

2. Egy bináris keresőfa preorder bejárása során a fa csúcsait $3, 10, 4, 8, 7, 9$ sorrendben látogatjuk meg. Rajzolja fel ezt a hat csúcsú bináris keresőfát, ahol ez megtörténhetett, majd lássa be, hogy a fa csak így nézhet ki.

Megoldás:

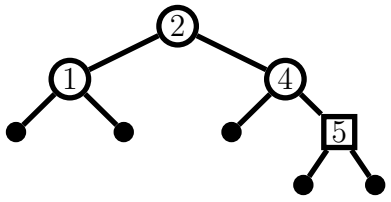
Jó fa: **3 pont**



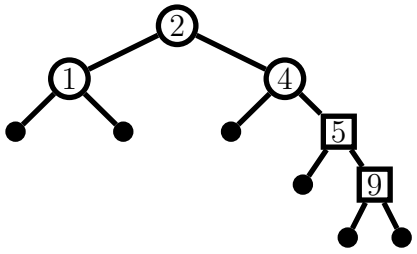
A gyökérben a 3-nak kell állnia, mert a preorder bejárás a gyökeret látja elsőnek.
A 3-tól balra nincsen senki, mert a bináris keresőfa tulajdonság miatt ott csak 3-nál kisebb számok lehetnek, de olyan nem szerepel. Tehát az összes többi elem a 3-as szám jobb fájában van és csak az a kérdés, hogy milyen elrendezésben.
A $10, 4, 8, 7, 9$ számok közül a 10-et látjuk először, ezért 10 van a 3-as szám jobb részfájának a gyökerében és mindenki más a 10-től balra van, mert mindegyik szám kisebb nála.
A $4, 8, 7, 9$ számok közül a 4-et látjuk először, ezért ez lesz a gyökérben és tőle jobbra van a másik három, mert nagyobbak nála.
A $8, 7, 9$ számok közül pedig a 8-t látjuk először, ezért ez lesz a gyökérben és tőle balra van a 7 és jobbra a 9, megint csak a bináris keresőfa tulajdonság miatt.

Az indoklásban jól hivatkozza a preorder bejárást (a számok sorrendjében elöl álló elem lesz a gyökérben): **3 pont**
Az indoklásban jól hivatkozza a bináris keresőfa tulajdonságot (a számok melyik oldalára esnek a gyökérnek): **4 pont**
Önmagában a preorder bejárás illetve a bináris keresőfa tulajdonság felírására nem jár pont, csak arra, ha ezekből a feladat szempontjából releváns következtetésekre jut a hallgató.

3. Szúrja be a következő piros-fekete fába a 9-et. (A ○ csúcs fekete, a □ pedig piros.) *A megoldásban látszódjanak melyik műveleteket végzi, de indokolni nem kell.*

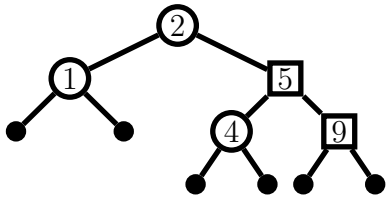


Megoldás: Először beszúrjuk naiv módon a 9-et:



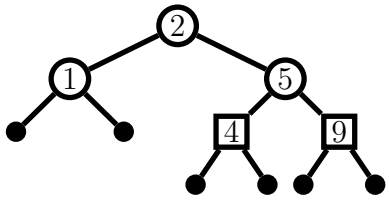
3 pont

Az 5-ös csúcs piros, testvére fekete, ezért átszínezni nem lehet. Mivel az 5 és 9 azon oldali gyerek, forgatni kell, hogy az 5 feljebb kerüljön:



4 pont

Átszínezés:



3 pont

4. Egy kezdetben üres 2-3-fába az $1, 2, \ldots, n$ számokat szúrtuk be ebben a sorrendben. Bizonyítsa be, hogy a keletkezett fában a háromgyerekes csúcsok száma $O(\log n)$.

Megoldás: Háromgyerekes csúcs bármikor is csak úgy tud keletkezni, hogy egy kétgyerekes csúcsnak keletkezik harmadik gyereke (akár a levél szinten az új elem beszúrásával, akár feljebb). **2 pont**
A növekvő sorrend miatt a beszúrt elem mindig az aktuális fa legjobboldalibb eleme lesz, mert a levelek rendezettek, ezért a legalsó nem-levél szinten csak a jobboldali elem lehet harmadfokú. **2 pont**
A fentebbi szinteken is csak a legjobboldalibb csúcs lehet háromgyerekes, mert a csúcsvágások is mindig a legjobboldalibb csúcsnál történnek. **3 pont**
Vagyis szintenként egy háromgyerekes lehet a keletkezett fában, de szintből $O(\log n)$ van a legvégén. **3 pont**
A fa magasságának felírása csak akkor ér pontot, ha ez az érték valamilyen módon szerepet kap a megoldás során.

5. Lássa be, hogy az alábbi eldöntési probléma coNP-ben van:

Input: G egyszerű, összefüggő, irányítatlan gráf

Kérdés: Igaz-e, hogy G minden feszítőfájában van olyan csúcs, aminek a feszítőfában négy szomszédja van (azaz a csúcs a fában negyedfokú)?

Megoldás: Azt kell belátni, hogy a probléma komplementere NP-beli, azaz van rövid, gyorsan ellenőrizhető tanú a komplementer probléma „igen” inputjaira. (Vagy úgy is lehet mondani, hogy azt kell belátni, hogy van rövid, gyorsan ellenőrizhető tanú a „nem” inputokra.) **1 pont**
A probléma komplementerében az a kérdés, hogy igaz-e, hogy G-nek van olyan feszítőfája, amiben nincsen negyedfokú csúcs. (Vagy úgy is lehet mondani, hogy arra kell tanú, hogy egy G-nek van olyan feszítőfája, amiben minden csúcs foka 4-től különböző.) **2 pont**
Jó tanú lesz egy F feszítőfa. **2 pont**
A tanú mérete nem nagyobb, mint az input mérete, azaz $O(n)$ méretű. **1 pont**
Az ellenőrzés során ezeket kell megnézni: **2 pont**

- F minden éle szerepel G-ben?

- F összefüggő: BFS-sel eldönthető
- F-nek $v-1$ éle van (ahol v a gráf pontszáma)?
- F-ben egyik fokszám sem 4?

A BFS $O(v+e) \subseteq O(n)$ lépés, a másik három ellenőrzés F éleinek egyszeri végigjárása során végrehajtható. Vagyis az ellenőrzés $O(n)$ lépés. **2 pont**

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 és k pozitív egész szám

Kérdés: Igaz-e, hogy G-ben van k méretű klikk és k elemű független ponthalmaz is?

Megoldás: Adunk egy visszavezetést egy NP-teljes problémáról erre a feladatra, ez az NP-beliséggel együtt az Iszonyú Hasznos Tétel miatt már bizonyítja az NP-teljességet. **1 pont**
Jó választás lesz a MaxFtln probléma. **1 pont**
Ha $k > v$, akkor $G' = G$ és $k' = k$ (azaz az input nem változik semmit), egyébként pedig a redukció során egy (G, k) párhoz azt a (G', k') párt rendeljük, amiben $k' = k$ és G'-t úgy kapjuk G-ből, hogy felveszünk G mellé egy k-as klikket, aminek minden csúcsát összekötjük G minden csúcsával.**3 pont**
(G', k') előállítása gyors, mert a k-as klikk hozzávétele $O(v^2)$-ben megvan, ami biztosan $O(n)$. **1 pont**
Ha $k > v$ (ahol v a csúcsok száma G-ben), akkor nincsen se k-as független, se k-as klikk G-ben, azaz ilyenkor a redukció 2. pontjában kívánt ekvivalencia fennáll.

Az érdekes esetben ($k \leq v$) pedig:

- ha G-ben van k-as független ponthalmaz, akkor G'-ben van k-as független ponthalmaz és k-as klikk is, mert a k-as független ponthalmaz megmarad G-ben is, az újonnan felvett k-as klikk pedig biztosítja a k-as klikket. **2 pont**
- ha G'-ben van k-as klikk és k-as független ponthalmaz is, akkor a k-as független ponthalmaz G-n belül kell, hogy legyen, mert az új csúcsok minden G-belivel és egymással is össze vannak kötve. **2 pont**

Ha valaki nem kezeli a $k > v$ esetet külön, de ad egy jó redukciót $k \leq v$ esetre, ami minden részletében helyes, akkor az 9 pontot kap.

Ha MaxKlikk-ről csinál valaki jó redukciót az is tökéletes persze (ekkor k izolált csúcsot kell felvenni G mellé).

7. Ha adott n szám, akkor hívjuk közülük középső elemnek a rendezés szerinti $\lceil n/2 \rceil$-ediket. Kezdetben adottak az $a_1, a_2, \ldots, a_n$ egész számok, amikről tudjuk, hogy az a_1 a középső elem, egyébként a számok rendezetlenek. Ezekből építsen fel egy adatszerkezetet $O(n)$ összehasonlítást felhasználva úgy, hogy a felépített adatszerkezetben az alábbi két műveletet k tárolt elem esetén $O(\log k)$ lépésben meg lehessen valósítani:
BESZÚR: egy új elemet illeszt az adatszerkezetbe,
KÖZÉPTÖR: az aktuális középső elemet törli.

Megoldás: Két kupacot használunk, az egyik egy min-kupac (ahogy órán tanultuk), ebben lesznek a középső elemnél nagyobb számok, a másik pedig egy max-kupac, ebben lesz a középső elem és a nála kisebb számok. A max-kupacban az a tulajdonság áll fenn a számokra, hogy minden elem nagyobb, mint a gyerekei, a műveletek pedig hasonlóan megvalósíthatók, mint a min-kupacban. **2 pont**

Az a_1-nél nagyobb elemekből a tanult módszerrel $O(n)$-ben lehet megépíteni a min-kupacot, hasonlóan lehet megépíteni a max-kupacot is $O(n)$-ben az a_1-nél kisebb elemekből és a_1-ből. **1 pont**

Mindig igaz, hogy a max-kupacban vagy ugyanannyi elem van, mint a min-kupacban vagy eggyel több és az is mindig igaz, hogy a max-kupac gyökerében a középső elem található. **1 pont**
BESZÚR esetén az új elemet összehasonlítom a max-kupac gyökerében levő középső elemmel és ha kisebb nála, akkor a max-kupacba, ha meg nagyobb nála, akkor a min-kupacba szúrom be. **1 pont**
Ha ezután a két kupac mérete több, mint eggyel eltér (azaz a különbség 2), akkor a nagyobb méretű kupacban (ez csak a max-kupac lehet) MAXTÖR-t csinálok (annak mintájára, ahogy a MINTÖR volt a min-kupacban), majd a kivett elemet beszúrom a min-kupacba (a tanult beszúrás eljárással). Hasonlóan járok el akkor is, amikor a beszúás után a min-kupac mérete nagyobb lesz a max-kupac méreténél: ekkor MINTÖR-t csinálok a min-kupacban, majd a kivett elemet beszúrom a max-kupacba. **1 pont**
Mindegyik lépés $O(\log k)$-ban megvalósítható, mert a kupacok mérete $O(k)$. **1 pont**
KÖZÉPTÖR során a max-kupac gyökerét kell törölni, azaz egy MAXTÖR-t kell csinálni, majd ha a min-kupac mérete ettől nagyobbá válik, mint a max-kupacé, akkor egy hagyományos MINTÖR-t kell csinálni a min-kupacban, majd a kivett elemet be kell szúrni a max-kupacba. **2 pont**
Mindegyik lépés $O(\log k)$-ban megvalósítható, mert a kupacok mérete $O(k)$. **1 pont**