Algoritmusok és gráfok - Pótzárthelyi
2019. december 5.

A válaszokat indokolni kell. Hivatkozni csak az előadáson tanultakra lehet.

1. Igaz-e, hogy ha egy algoritmus lépésszáma $42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 - 7 \cdot \sqrt{n} + 30$, akkor az algoritmus lépésszáma $O(n^3)$? Ha úgy véli, hogy ez igaz, akkor megfelelő c konstans és n_0 küszöbérték megadásával lássa ezt be, ha pedig úgy véli, hogy hamis, akkor bizonyítsa be ezt.

   Megoldás
   Igaz az állítás, az $42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 - 7 \cdot \sqrt{n} + 30$ függvény $O(n^3)$-ös. Ehhez egy olyan c pozitív konstanst és egy olyan n_0 küszöbindexet kell találnunk, melyre igaz, hogy

   $$42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 - 7 \cdot \sqrt{n} + 30 \leq c \cdot n^3, \text{ha } n \geq n_0$$

   Ilyen c és n_0 értéket például az alábbi egyenlőtlenség-sorozattal tudunk találni:
   $42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 - 7 \cdot \sqrt{n} + 30 \leq 42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 + 30$
   mert a negatív tagot el lehet hagyni, tetszőleges $n \geq 1$ esetén
   $42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 + 30 \leq 42n^3 + 2020 \cdot n^3 + 30n^3$
   mert $\log n \leq n$, $n - 1 \leq n$ és $30 \leq 30n^3$ tetszőleges $n \geq 1$ esetén.
   Azt kaptuk tehát, hogy
   $42 \cdot n^2 \log n + 2020 \cdot (n-1)^3 - 7 \cdot \sqrt{n} + 30 \leq 42n^3 + 2020 \cdot n^3 + 30n^3 = (42 + 2020 + 30)n^3$
   tetszőleges $n \geq 1$ esetén, vagyis $c = 42 + 2020 + 30 = 2092$ és $n_0 = 1$ jó választás.

2. A $2, 10, 1, 3, 7, 8, 4$ tömb rendezése során előállhat-e az $1, 2, 3, 10, 7, 8, 4$ helyzet, ha
   (a) buborékrendezést használunk?
   (b) beszúrásos rendezést használunk?

   Megoldás
   (a) A buborékrendezésnél ez nem állhat elő. A buborékrendezés során először végigmegyünk az egész tömbön és a szomszédos elemeket hasonlítjuk: ha egy nagyobb elem egy kisebb elem előtt áll közvetlenül, akkor megcseréljük őket. Így az első körben a 10-et megcseréljük az 1, majd a 3 értékkel, de eközben a tömb elején a többi elem sorrendje nem változik, vagyis mikorra a 10 elér a 4. pozícióba a tömb állapota $2, 1, 3, 10, 7, 8, 4$ lesz, azaz eddig nem állhatott elő a kívánt átmeneti helyzet, mert a 2 nem léphette még át az 1-et. Ha viszont tovább folytatjuk a rendezést, akkor a 10 továbbmegy jobbra és sose lesz már a 4. pozícióban vagyis a későbbiekben sem állhat elő ez a helyzet.
   (b) A beszúrásos rendezést futtatva az első két elem rendezetten áll, az első érdemi történés az, amikor a 3. pozícióban álló 1-et a helyére mozgatjuk (balra haladva, mindaddig cserélve, amíg nála nagyobb elem áll előtte), így az $1, 2, 10, 3, 7, 8, 4$ tömböt kapjuk. Ekkor a 3-at mozgatjuk balra, cserélve az előtte levővel, ha az nagyobb nála és az $1, 2, 3, 10, 7, 8, 4$ tömböt kapjuk, a kívánt átmeneti helyzetet, vagyis ennél az algoritmusnál lehetséges ez az átmeneti állapot.

3. Egy kezdetben üres, 11 méretű hash táblába nyílt címzéssel, lineáris próbával szúrtunk be nyolc egész számot, a beszúrások sorrendje nem ismert. A használt hash függvény értéke a K kulcs esetén K maradéka 11-gyel osztva. Csak beszúrások történtek, törlés nem volt és az alábbi táblát kaptuk.

0	1	2	3	4	5	6	7	8	9	10
	13	2	3	5	17	7	18			21

Milyen sorrendben történhetett a $3, 5, 17, 7, 18$ elemek beszúrása, ha ezt a táblát kaptuk? Az összes lehetséges sorrendjét adja meg ennek az öt elemnek.

Megoldás

Az 5-öt biztosan később szúrtuk be, mint a 17-et, mert különben az 5 az 5-ös indexű cellába került volna, mivel nem ott van, ezért annak akkor már foglaltnak kellett lennie. Hasonló érveléssel látható, hogy a 17-et biztos később szúrtuk be, mint a 7-et, mert különben a 17 a 6-os indexű cellában lenne. Ugyanígy látható, hogy a 7–et a 18 után szúrtuk be, mert különben a 7 a 7-es indexű cellában lenne. Az derült tehát ki, hogy az 5 később jött, mint a 17, az később jött, mint a 7, az pedig később jött, mint a 18, vagyis ezen négy elem sorendje biztosan $18, 7, 17, 5$ volt.

A 3 beszúrása ezekhez képest bármikor történhetett, mert a 3 a hash függvény szerinti helyére került és ez a hely teljesen független a többi elem által bejárt celláktól, a 3 beszúrása nem befolyásolja a másik négy elem sorsát.

Vagyis a lehetséges sorrendek: $3, 18, 7, 17, 5$ vagy $18, 3, 7, 17, 5$ vagy $18, 7, 3, 17, 5$ vagy $18, 7, 17, 3, 5$ vagy $18, 7, 17, 5, 3$.

4. Egy hét csúcsú bináris keresőfát posztorder bejárással bejárva a csúcsokat $1, 10, 11, 8, 20, 13, 3$ sorrendben látogatjuk meg. Adja meg, hogy hogyan néz ki ez a fa.

Megoldás

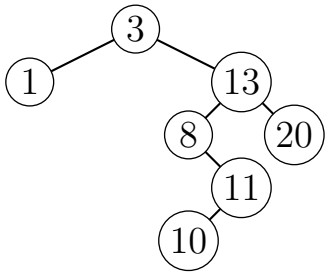
A posztorder bejárásban a gyökeret látogatjuk meg utoljára, ezért a fa gyökere a 3 és a bináris keresőfa tulajdonság miatt tőle balra csak az 1 van, a $8, 10, 11, 13, 20$ értékek a jobb fában vannak.

A jobb fa elemei közül a posztorder bejárás a 13-at járja be utoljára, vagyis ez a gyökér a jobb részfában és a bináris keresőfa tulajdonság miatt a jobb részfájában csak a 20 van, a balban pedig $8, 10, 11$.

Az most a kérdés, hogy milyen elrendezésben vannak a $8, 10, 11$ elemek a 13 bal részfájában. Ezt onnan tudjuk kideríteni, hogy a posztorder bejárás ezen értékek közül a 8-at látja utoljára, vagyis ez a gyökér és a bináris keresőfa tulajdonság miatt ennek bal gyereke nincs, a jobb fájában pedig 10 és 11 van.

Az most már csak a kérdés, hogy a 10 és 11 hogyan állnak ebben a jobb fában, de ezt megint csak a posztorder bejárással tudjuk kitalálni, mivel a posztorder bejárás a 11-et látja később, ezért ez a gyökér, ennek pedig bal gyereke lesz a 10.

A kapott fa tehát ez:



5. Adott egy n hosszú tömb, mely csupa különböző egész számot tartalmaz. Adjon $O(n \log n)$ lépésszámú algoritmust, ami vagy talál a tömbben egy olyan k egész számot, melyre sem $k-1$, sem $k+1$ nincs a tömbben vagy jelzi, ha nincs ilyen k szám.

   Egy lehetséges megoldás

   Algo:

   (a) Rendezzük a tömböt összefésüléses rendezéssel.

   (b) Menjünk végig a rendezett tömbön és a tömb minden $A[i]$ elemére nézzük meg, hogy előtte $A[i]-1$, utána pedig $A[i]+1$ van-e. Ha találunk olyan i indexet, ahol $A[i]$ előtt nem $A[i]-1$, utána pedig nem $A[i]+1$ van, akkor $A[i]=k$ olyan szám, amit keresünk, egyébként pedig nincsen k szám.

   Jóság: A rendezés után az egymást követő számok egymás után lesznek a tömbben, vagyis ha a rendezett sorban egy szám előtt közvetlenül nem a nála eggyel kisebb szám áll, akkor az eggyel kisebb szám nincs a tömbben, hasonlóan ha egy szám után közvetlenül nem a nála eggyel nagyobb szám áll, akkor az eggyel nagyobb szám nincs a tömbben.

   Lépésszám: A rendezés $O(n \log n)$, utána pedig minden i értékre két ellenőrzést kell tennünk, ami $O(n)$, így az algoritmus $O(n \log n)$ lépésszámú.

6. Egy szomszédossági mátrixával adott n csúcsú, irányított G gráfban néhány csúcs kékre van színezve, a többi csúcs színtelen. A kék csúcsok egy, a csúcsokkal indexelt K tömbben adottak úgy, hogy $K[v]$ értéke 1, ha a v csúcs kék, egyébként pedig $K[v]=0$. Adott továbbá két kijelölt színtelen csúcs, s és t is. Adjon $O(n^2)$ lépésszámú algoritmust, ami eldönti, hogy el lehet-e jutni s-ből t-be kék csúcs érintése nélkül.

   Megoldás

   Ötlet: Pontosan akkor lehet eljutni s-ből t-be kék csúcs érintése nélkül, ha van s-ből t-be olyan út, melyen minden csúcs színtelen.

   Algo:

   1. Módosítsuk a gráf szomszédossági mátrixát úgy, hogy minden olyan élet kitörlünk belőle, aminek legalább egy végpontja kék. Ezt úgy lehet megtenni, hogy végigmegyünk a mátrixon soronként, soron belül oszloponként és ha $A[i,j]=1$ (azaz van él az i

csúcsból a j csúcsba) és vagy $K[i] = 1$ vagy $K[j] = 1$, akkor $A[i, j] = 0$-t állítunk be (azaz töröljük az élet a mátrixból).
Pszeudokódban ugyanez:

```
ciklus i =1-tól n-ig:
    ciklus j= 1-tól n-ig:
        ha A[i,j] == 1:
            ha (K[i]  == 1 vagy  K[j]  == 1):
                A[i,j] := 0
    ciklus vége
ciklus vége
```

2. BFS-t vagy DFS-t futtatunk a kapott mátrixon s-ből és ha t bejárt lesz a végén, akkor van jó út, különben meg nincsen.

Jóság: Az új gráfban pontosan azok az utak maradnak meg, amik nem használnak kék csúcsokat, mert a kék csúcsokba se bemenni, se kijönni belőlük nem lehet és a BFS/DFS el tudja dönteni, hogy az új gráfban van-e út s-ből t-be.

Lépésszám: A gráf módosítása $O(n^2)$, mert a belső ciklus magja konstans lépés és n-szer fut le, azaz a belső ciklus $O(n)$-es, a külső ciklus magja ez az $O(n)$-es belső ciklus és n-szer fut le, vagyis $O(n^2)$ összesen.
A BFS/DFS futtatása $O(n^2)$, mert az új gráfnak is n csúcsa van (csak éleket töröltünk.)