Algoritmusok és gráfok - Vizsga
2020. január 22.

A válaszokat indokolni kell, de a feladatokban szereplő tanult algoritmusokat nem kell részletesen leírni, elég csak azokat a részeket kifejteni, amelyek az indokláshoz szükségesek.

1. Az alábbi függvények közül pontosan egyre igaz, hogy $O(n^2)$-es. Válassza ki ezt a függvényt és lássa be megfelelő c konstans és n_0 küszöbérték megadásával, hogy $O(n^2)$-es.

   $$\log(n^2) + \tfrac{n(n-1)(n-2)}{2010} \qquad 2020n\log n + \frac{2^n \cdot n(n-1)}{3^n} \qquad 20n^2(\log n)^2 - 8$$

   Megoldás

   A három függvény közül az $2020n\log n + \dfrac{2^n \cdot n(n-1)}{3^n}$-re igaz, hogy $O(n^2)$-es.
   Ezt úgy látjuk be, hogy megadunk olyan c konstans és n_0 küszöbértéket, melyre $2020n\log n + \dfrac{2^n \cdot n(n-1)}{3^n} \leq cn^2$ fennáll, ha $n \geq n_0$.
   Ezen c és n_0 párt egy egyenlőtlenségrendszerrel találjuk meg:
   $2020n\log n + \dfrac{2^n \cdot n(n-1)}{3^n} \leq 2020n^2 + \dfrac{2^n \cdot n(n-1)}{3^n}$, ha $n \geq 1$, mivel $\log n \leq n$, ha $n \geq 1$.
   $2020n^2 + \dfrac{2^n \cdot n(n-1)}{3^n} \leq 2020n^2 + n(n-1)$, ha $n \geq 1$, mivel $\dfrac{2^n}{3^n} \leq 1$, ha $n \geq 1$.
   $2020n^2 + n(n-1) \leq 2020n^2 + n^2 = 2021n^2$, ha $n \geq 1$, mivel $n - 1 \leq n$, ha $n \geq 1$.
   Azt kaptuk tehát, hogy $2020n\log n + \dfrac{2^n \cdot n(n-1)}{3^n} \leq 2021n^2$, ha $n \geq 1$, azaz $n_0 = 1$ és $c = 2021$ jó választás.

2. A Prim algoritmus lényegi része egy amíg-ciklus, mely két részlet (a két üresen hagyott téglalap) kivételével így néz ki:

```
ciklus amíg van olyan csúcs, ahol legolcsóbb[v] nem *:
    v* := az a csúcs, ahol legolcsóbb[v] minimális
    v* LEFEDVE-be megy
    legolcsóbb[v*] := *
    (közeli[v*], v*) F-be kerül
    ciklus w = 1-tól n-ig:
        ha A[v*,w] nem végtelen és legolcsóbb[w] nem *:
            ha c(v*, w) < legolcsóbb[w]:
                legolcsóbb[w] := [          ]
                közeli[w] := [          ]
    ciklus vége
ciklus vége
```

(a) Mit tárol `legolcsóbb[w]` egy w csúcs esetén?
(b) Mit tárol a `közeli[w]` egy w csúcs esetén?
(c) Egészítse ki a kódot a két üres téglalap kitöltésével és magyarázza el röviden (2-3 mondatban), hogy miért így kell a két hiányzó értéket meghatározni.

Megoldás
(a) legolcsóbb[w] a LEFEDVE halmazból w-be menő legrövidebb él súlyát tárolja
(b) közeli[w] azt a LEFEDVE halmazban levő csúcsot adja meg, ahonnan ez a legrövidebb él jön.
(c) legolcsóbb[w] := c(v*, w) és közeli[w] := v*, mert ha $c(v^*, w) <$ legolcsóbb[w], akkor találtunk egy, az eddigi legjobbnál rövidebb élet w-be a LEFEDVE halmazból, nevezetesen a v^* csúcsból érkező $c(v^*, w)$ súlyú élet.

3. Egy kezdetben üres, 11 méretű hash táblába nyílt címzéssel, lineáris próbával szúrtunk be hét egész számot, a használt hash függvény a $h(K) =$ K maradéka 11-gyel osztva függvény volt. A beszúrások után egy számot kitöröltünk a táblából, a törölt elem helyét * jelzi, az eljárás végén az alábbi táblát kaptuk:

0	1	2	3	4	5	6	7	8	9	10
		14	3		5	17	*	8	20	

(a) Szúrja be ebbe a táblába a 31-et, majd törölje ki a 17-et. Adja meg a tábla állapotát mindkét művelet után és mindkét esetben jelezze, hogy melyik cellákat érintettük.
(b) Mely cellákat érintjük a 19 keresése során az (a) pontban kapott táblában?

Megoldás
(a) A 31-et a $h(31) = 9$-es cellába akarjuk először berakni, de ez foglalt, ezért megyünk balra. A 8-as is foglalt, innen is balra lépünk, a 7-es törölt, ide beszúrhatjuk.
A tábla ez lesz:

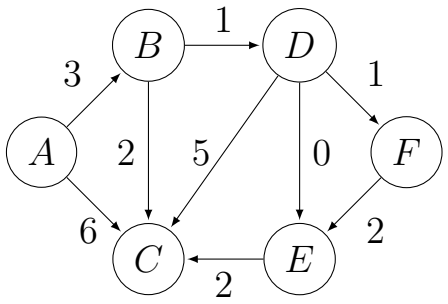
0	1	2	3	4	5	6	7	8	9	10
		14	3		5	17	31	8	20	

A 17-es törlése a 17-es keresésével kezdődik, a $h(17) = 6$-os cellában, itt meg is találjuk, kitöröljük (azaz $*$-ot írunk a helyére.)
A tábla ez lesz:

0	1	2	3	4	5	6	7	8	9	10
		14	3		5	*	31	8	20	

(b) A 19-es érték keresése a $h(19) = 8$-as cellában kezdődik, ez foglalt, megyünk balra, amíg üres cellát nem találunk (azaz átmegyünk a 8, 7, 6, 5 indexű cellákon és végül a 4-es cellában állunk meg, itt látjuk meg, hogy a 19 nincs a táblában.

4. Az alábbi gráfon futtatjuk Dijkstra algorimusát az A csúcsból, az A és B csúcsok KÉSZ-be kerülése után a d tömb így néz ki:



A	B	C	D	E	F
$\star$	$\star$	5	4	∞	∞

(a) Magyarázza el röviden, hogy $d(F)$ miért végtelen és $d(D)$ miért 4.
(b) Melyik csúcs kerül a következő lépésben a KÉSZ halmazba és miért?
(c) Hogyan néz ki a d tömb az új csúcs KÉSZ-be kerülése miatti módosítások után és miért?

Megoldás
(a) A d tömbben $d[v]$ a legrövidebb olyan út hosszát jelöli, ami a kezdőcsúcsból indul, v-be érkezik és az utolsó él kivételével végig a KÉSZ halmazban halad. Mivel F-re nincs ilyen út (mert a KÉSZ halmazból nem vezet él F-be), ezért $d(F) = \infty$.
A D csúcsba csak B-ből érkezhetünk a KÉSZ-ből, a legrövidebb ilyen út az ABD, ennek hossza 4, ezért $d(D) = 4$.
(b) A KÉSZ-be a D kerül be, mert ennek d értéke a legkisebb.
(c) Ha D KÉSZ-be kerül, akkor $d(D) = \star$ lesz és változások lehetnek még D KÉSZ-en kívüli szomszédainál, máshol biztosan nem. D szomszédai C, E, F, ezek mind KÉSZ-en kívül vannak, azt kell megnézni, hogy a D-n át egy éllel ezekbe a csúcsokba vezető legrövidebb út hossza kisebb-e az aktuális d értéknél:
C-re a legjobb D-ből érkező út 9, ez rosszabb, mint az eddigi 5, marad az 5,
E-re és F-re eddig nem volt út, most lett 4 és 5 hosszú, ezért $d(E) = 4, d(F) = 5$.

5. Az $5, 2, x, 10$ tömb összefésüléses rendezése során 4 összehasonlítás történik. Adja meg x összes lehetséges értékét, ha tudjuk, hogy x olyan egész szám, ami máshol nem szerepel a tömbben. Válaszát indokolja is.

Megoldás
Az algoritmus először egy-egy összehasonlítással rendezi az $5, 2$ és $x, 10$ kételemű részeket. Ez eddig biztosan két összehasonlítás, az a kérdés, hogy mikor, mely x értékek esetén lehet két további összehasonlítással befejezni a rendezést.

Megvizsgáljuk az összes esetet:
ha $x < 2$: ekkor a $2, 5$ tömböt fésüljük össze az $x, 10$ tömbbel, de ekkor 2 és x, 2 és 10 , majd 5 és 10 is össze lesz hasonlítva, ez már túl sok összehasonlítás.

Ha $2 < x < 5$, akkor megint csak a $2, 5$ tömböt fésüljük össze az $x, 10$ tömbbel, ekkor 2 és x, 5 és x , majd 5 és 10 is össze lesz hasonlítva, ez ismét túl sok összehasonlítás.
Ha $5 < x < 10$, akkor megint a $2, 5$ és $x, 10$ tömböket fésüljük össze, ekkor 2 és x, majd 5 és x lesz hasonlítva, majd az első tömb elfogy, nem kell több összehasonlítás. Ekkor 4 összehasonlítás volt összesen, ez az eset lehetséges.
Ha $10 < x$, akkor a $2, 5$ és $10, x$ tömböket fésüljük össze, ekkor 2 és 10, majd 5 és 10 lesz hasonlítva, majd az első tömb elfogy, nem kell több összehasonlítás. Ekkor is 4 összehasonlítás volt összesen, ez az eset is lehetséges.

6. Egy G irányított, élsúlyozott gráfban az a, b, c, d, e sorrend topologikus sorrend. Ezen topologikus sorrendet használva a tanult algoritmust futtatjuk az a-tól vett legrövidebb utak hosszának meghatározására. Az a, b, c, d csúcsokra ezeket a távolságokat kaptuk: *távolság*[a] = 0, *távolság[b]* = 2, *távolság[c]* = -7, *távolság[d]* = 5.
(a) Miért biztos, hogy van a gráfban ab él? Mi ennek a súlya és miért?
(b) Ha van a gráfban cd él, akkor mi lehet ennek a súlya és miért?

Megoldás
(a) A b csúcs előtt csak az a van a topologikus sorrendben, ezért egy a-ból b-be menő út csak egy közvetlen a-ból b-be menő él lehet. Mivel ennek az útnak a hossza 2, de az út maga az él, ezért az él hossza 2.
(b) A d-re kapott távolság az algoritmus szerint a legkisebb érték a távolság[a] + c(a,d), távolság[b] + c(b,d), távolság[c] + c(c,d) értékek közül (amennyiben mindhárom él létezik). Ha van él c-ből d-be, akkor távolság[c] $+c(c, d) = -7 + c(c, d) \geq$ távolság[d] = 5, azaz $c(c, d) - 7 \geq 5$, vagyis $c(c, d) \geq 12$.

7. Szomszédossági mátrixával adott egy n csúcsú, irányítatlan G gráf. Adjon $O(n^2)$ lépésszámú algoritmust, ami eldönti, hogy van-e a gráfban olyan másodfokú csúcs, aminek a két szomszédja között van él a gráfban.

Megoldás
Algo szövegesen:
Az A szomszédossági mátrix minden sorára megcsináljuk ezt: végigmegyünk a soron és megszámoljuk, hogy hány 1 van, közben a szomszédos csúcsokat egy listába rakjuk. Ha nem két 1 van, akkor megyünk a következő sorra, ha pedig két darab 1 van, akkor a listában két csúcs van, i és j és megnézzük, hogy $A[i, j]$ 1 vagy 0. Ha 1, akkor találtunk jó másodfokú csúcsot, ha pedig 0, akkor megyünk tovább a következő sorra. Ha végigmentünk az összes soron és így nem találtunk jó másodfokú csúcsot, akkor nincs ilyen.

Algo kóddal:

```
van_jo_csucs := False
ciklus k = 1-tól n-ig:   // végig a sorokon
    szomszedok_szama: = 0
    szomszedok_listaja: =  üres lista
    ciklus s = 1-tol n-ig:            // végig az oszlopokon a k. sorban
        ha A[k,s] == 1:
            szomszedok_szama +=1
            s bele szomszedok_listaja-ba
    ciklus vége
    ha szomszedok_szama == 2:
        ha A[szomszedok_listaja[0], szomszedok_listaja[1]] == 1:
            van_jo_csucs: = True
ciklus vége
return van_jo_csucs
```

Jóság: Az 1-ek száma a sorban a csúcs fokszámát adja meg, pont ezt nézzük meg minden sorban. Ha a szomszédok száma 2, akkor $A[i, j]$ mutatja, hogy ők össze vannak-e kötve vagy sem.

Lépésszám:
A szöveges leíráshoz: Összesen n darab sor van és minden sorban megnézek n értéket, amikor a fokszámokat számolom, értékenként konstans lépés történik (számláló és lista karbantartása), ez eddig $n \cdot O(n) = O(n^2)$. Minden sorban van még egy ellenőrzés (hogy 2 szomszéd volt-e) és legfeljebb egy érték kiolvasása a szomszédossági mátrixból, ez soronként konstans lépés, azaz $O(n)$ összesen, vagyis a lépésszám $O(n^2) + O(n) = O(n^2)$.

Pszeudokód: A külső ciklus n-szer fut le, ennek magja konstans sok lépés, aztán egy belső ciklus, majd konstans sok lépés. A belső ciklus n-szer fut le, magja konstans, vagyis a belső ciklus $O(n)$, a külső ciklus pedig $n(O(1) + O(n) + O(1)) = nO(n) = O(n^2)$. A külső cikluson kívül kívül csak konstans sok lépés van, így az egész kód $O(n^2)$.

8. Szomszédossági mátrixával adott egy n csúcsú, irányított G gráf és benne egy kijelölt (u, v) irányított él. Szeretnénk megtalálni a legkevesebb élből álló, az (u, v) élen átmenő irányított kört a gráfban.

   Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha $O(n^2)$ lépésben meg akarjuk oldani ezt a feladatot?

   Megoldás
   Algo:

Futtassuk a BFS-t a v csúcsból és nézzük meg a BFS fában a v-ből u-ba vezető utat. Ha nincs ilyen út, akkor nincs ezen az élen át kör, ha van ilyen út, akkor ez az út és az él maga alkotja a legkevesebb élű keresett kört.
Jóság:
Minden uv-n átmenő kör az uv élből és egy vu útból áll, ha megtaláljuk a legrövidebb vu utat, akkor megvan a a legrövidebb keresett kör is.
Lépésszám: BFS lépésszáma (útmegtalálással együtt is) $O(n^2)$, ezután a vu út megkeresése a fában $O(n)$, azaz együtt $O(n^2)$.