Algoritmusok és gráfok - Vizsga
2020. január 29.

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 pszeudokód inputja egy $n \geq 2$ hosszú A tömb. Igaz-e, hogy ez a kód $O(n)$-es, ha egy lépésnek az összehasonlítás, az összeadás és az értékadás számít? Ha igaz, akkor lássa be ezt, ha pedig nem igaz, akkor lássa be, hogy $O(n^2)$-es.

```
ciklus i = 0-tól (n-1)-ig:
    ha A[i] > 0:
        ciklus amíg A[i] < 5:
            A[i] := A[i] + 1
        ciklus vége
ciklus vége
```

Megoldás
Ez a kód $O(n)$-es. A külső ciklus n-szer fut le, ennek a magja pedig $O(1)$-es a következők miatt: Egy lépés után egy belső ciklus következik, ami legfeljebb 5-ször fut le, mert minden lefutáskor nő eggyel az $A[i]$ értéke és mivel legalább 0 volt, ezért legfeljebb 5 lefutás után nagyobb lesz, mint 5 és akkor már nem teljesül az amíg-feltétel. A belső ciklus magja konstans sok lépés, így a belső ciklus $O(1)$ lépésből áll, azaz a külső ciklus magja $1 + O(1)$, azaz $O(1)$, így az egész kód $O(n)$.

2. (a) Adja meg a bináris keresőfa definícióját (elég a bináris keresőfa tulajdonságot leírnia, nem kell definiálni azt, hogy mi a bináris fa).
(b) Írja le, hogy hogyan kell beszúrni egy új elemet bináris keresőfába. Mennyi ennek az eljárásnak a lépésszáma és miért?

Megoldás
(a) A bináris keresőfa tulajdonság azt jelenti, hogy minden csúcsra igaz, hogy a bal részfájában minden elem kisebb, a jobb részfájában pedig minden elem nagyobb nála.
(b) Egy új s érték beszúrása úgy történik, hogy először s-et összehasonlítjuk a gyökérrel és ha s kisebb, mint a gyökér, akkor balra, ha pedig nagyobb, akkor jobbra megyünk. Ezt az eljárást folytatjuk az aktuális csúccsal (ahova érkezünk) mindaddig, amíg üres gyerekhez (None mutatóhoz) érünk, ekkor ide beillesztjük az s értéket.
Ezen eljárás lépésszáma $O(h)$, ahol h a fa magassága, mert minden szinten konstans sok lépést teszünk.

3. Egy irányított gráf élei a következők: $AB, AC, BG, CB, CD, CG, DA, DE, DF, EF, GD, GF$. Ebben a gráfban BFS-t (szélességi bejárást) futtatunk az A kezdőcsúcsból úgy, hogy ha bármikor választási lehetőség adódik, akkor az ábécé szerint előbb levő csúcsot választjuk.
(a) Milyen sorrendben járjuk be a csúcsokat és mi lesz a kapott BFS fa? (Indokolni

ezt nem kell, elég a sorrend és a fa megadása.)
(b) Hogyan néz ki a *honnan* tömb a BFS futásának a végén? Indoklásképpen írja le röviden, hogy mit tárol a *honnan* tömb.
(c) Adja meg a Q sor állapotát a kód futásának kezdetén, majd minden egyes csúcs bejárása után. Indoklásképpen röviden írja le, hogy a Q sornak mi a szerepe a kódban.

Megoldás
(a) A, B, C, G, D, F, E, a fa élei pedig: AB, AC, BG, CD, GF, DE.
(b)

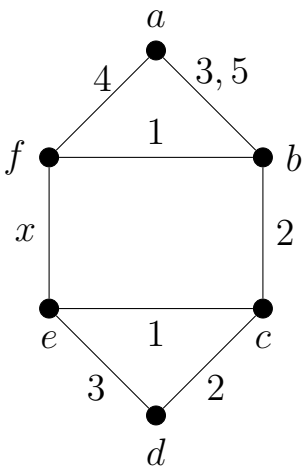
A	B	C	D	E	F	G
A	A	A	C	D	G	B

Ez azért van így, mert a $honnan[v]$ azt tárolja, hogy a BFS eljárás során a v csúcsot melyik csúcsból fedeztük fel.
(c) A Q sor azokat a csúcsokat tárolja, amiket már felfedeztünk, de amiknek a szomszédait még nem vizsgáltuk meg.
Az elején $Q = \{A\}$, később pedig
A bejárása után $Q = \{B, C\}$,
B bejárása után $Q = \{C, G\}$,
C bejárása után $Q = \{G, D\}$,
G bejárása után $Q = \{D, F\}$,
D bejárása után $Q = \{F, E\}$,
F bejárása után $Q = \{E\}$,
E bejárása után $Q = \{\}$.

4. A G irányított, élsúlyozott gráf élei a következők (zárójelben az élsúlyok):
$ac(8), af(7), bf(11), cd(10), ce(7), cf(6), eb(-5), ed(2), ef(3)$.
Ebben a gráfban az órán tanult, DAG-ban leghosszabb utat kereső algoritmust futtatjuk a-ból az a, c, e, b, d, f topologikus sorrendet használva. Az a, c, e, b, d csúcsokba vezető leghosszabb utak hosszát már meghatároztuk: *leghosszabb[a] = 0*, *leghosszabb[c] = 8*, *leghosszabb[e] = 15*, *leghosszabb[b] = 10*, *leghosszabb[d] = 18*. Hogyan kapjuk meg *leghosszabb[f]*-et a tanult eljárással ezekből az adatokból? A megoldásából látszódjon, hogy hogyan használja a tanult eljárást.

Megoldás
Az összes olyan v csúcsra, ahonnan vezet él f be, meg kell néznünk a *leghosszabb[v] + c(v,f)* értéket és ezek közül kell vennünk a maximumot. Ez azt jelenti, hogy a maximumát keressünk a
leghosszabb[a] + c(a,f), leghosszabb[c] + c(c,f), leghosszabb[e] + c(e,f),
leghosszabb[b] + c(b,f) értékeknek, ami b-nél található 21 értékben, azaz
leghosszabb[f] értéke 21.

5. Az alábbi gráfon, ahol az x élsúly nem ismert, futtatva Kruskal algorimusát az első három él, amit az eljárás beválaszt: bf, ec, bc. Adja meg x összes lehetséges értékét (röviden indokolva válaszát) és írja le, hogy mely éleket és milyen sorrendben választja be ezután az algoritmus és miért.



Megoldás

Az x értékére az igaz, hogy $x \geq 2$, mert a Kruskal algoritmus sorbarendezi az éleket súlyt szerint és az aktuális éle akkor veszi be a fába, ha az nem alkot kört az eddig választott élekkel (egyébként átlépi és megy a következő él vizsgálatára).
H x kisebb lenne, mint 2, akkor az fe élet vizsgálná bf és ec után és mivel nem alkotna velük kört, ezért ezt az fe élet venné be harmadiknak.
A bf, ec, bc élek után az fe élet biztos nem veheti be sose, mert kört alkotna, a többi élet pedig cd, ed, ab, af sorrendben nézi és ezekből először a cd, majd az ab élet veszi be.

6. Egy kezdetben üres, 11 méretű hash táblába nyílt címzéssel, lineáris próbával szúrtunk be nyolc egész számot, a használt hash függvény a $h(K) = \text{K}$ maradéka 11-gyel osztva függvény volt. A beszúrások után két számot kitöröltünk a táblából, a törlések helyét $*$ jelzi. Mutassa meg, hogy miért nem lehetséges, hogy az eljárás végén az alábbi táblát kaptuk:

0	1	2	3	4	5	6	7	8	9	10
16	*		14	6	*	17			9	21

Megoldás

A 16 nem kerülhetett oda, ahol van, mert ezt először a $h(16) = 5$-ös cellába akarjuk rakni és ha ide nem tudjuk, akkor balra indulva keresünk neki új helyet, amit legkésőbb a 2-es cellában megtalálnánk (mert ott sose volt semmi), vagyis ezen a cellán nem juthatott volna túl a 16.

7. Szomszédossági mátrixával adott egy n csúcsú, irányítatlan G gráf és adott egy, a csúcsokkal indexelt R tömbben a csúcsok egy részhalmaza oly módon, hogy ha a v csúcs nincsen benne a részhalmazban, akkor $R[v] = 0$, ha pedig v benne van a részhalmazban, akkor $R[v] = 1$.
Azt szeretnénk eldönteni, hogy igaz-e, hogy egyetlen él sem fut R-beli csúcsok közt a gráfban.

Adjon erre a feladatra $O(n^2)$ lépésszámú algoritmust.

Megoldás

Algo: végigmegyünk az R tömbön és ha valahol 1-t látunk egy i csúcsnál, akkor minden i után szereplő olyan j-re, ahol $R[j]$ is 1 megnézzük, hogy $A[i, j] = 0$ teljesüle. (Itt A jelöli a gráf szomszédossági mátrixát.) Ha ez minden R-beli 1-es párra teljesül ez, akkor egyetlen él sincsen a gráfban R-beli csúcspárok között, különben meg ez nem igaz

Pszeudokódban uez:

```
rendben: = True
ciklus i = 1-tól (n-1)-ig:
    ha R[i] == 1:
        ciklus j = (i + 1)-tól n-ig:
            ha (R[j] == 1 és A[i,j] ! = 0):
                rendben := False
        ciklus vége
ciklus vége
```

Jóság: Mivel minden R-beli 1-es párt megnézünk, ezért kiderül, hogy mindegyik él hiányzik-e a gráfból vagy sem.

Lépésszám: A belső ciklus legfeljebb n-szer fut le, a magja konstans sok lépés, vagyis a belső ciklus $O(n)$. A külső ciklus legfeljebb n-szer fut le, ennek magja 1 lépés plusz az $O(n)$-es belső ciklus, azaz a külső ciklus $O(n^2)$-es. Az egész kód csak egy lépéssel több, mint a külső ciklus, azaz ez is $O(n^2)$-es.

Vagy a szöveges leíráshoz: R-ben legfeljebb n darab 1-es lehet és mindegyiknél legfeljebb n másik értéket kell megnéznem R-ben is és A-ban is, azaz legfeljebb $O(n^2)$ lépés történik.

8. Egy A szomszédossági mátrix-szal adott egy város úthálózatának élsúlyozott, irányított gráfja: a csúcsok a csomópontok, az élek a csomópontok közötti közvetlen utak, az élek súlya pedig azt mutatja, hogy mennyi az átlagos idő, ami az út megtételéhez autóval szükséges. Adott továbbá egy másik B mátrix is, ami azt adja meg minden közvetlen útra, hogy mekkora a legnagyobb magasságú autó, amivel elakadás nélkül át lehet haladni az adott szakaszon: a B mátrix i-edik sorának, j-edik oszlopában álló $B[i, j]$ érték mutatja a legnagyobb magasságot, amivel még megtehető az i csomópontból a j csomópontba vezető közvetlen út.

   Adott a gráfban két kijelölt csúcs, X és Y. Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha $O(n^2)$ lépésben meg akarjuk találni a leggyorsabban megtehető olyan utat X-ből Y-ba, amit legfeljebb 350 centiméter magas autóval meg lehet tenni? (Itt n szokásos módon a gráf csúcsainak számát jelöli.)

Megoldás

Algoritmus:

1. Módosítsuk a gráfot úgy, hogy azokat az éleket, amiken nem lehet átmenni 350 cm magas autóval, azokat kitöröljük.
2. Ebben az új gráfban futtassuk a Dijkstra algoritmust az X csúcsból. Ha nem kapunk utat Y-ba, akkor nincs ilyen út, ha kapunk, akkor pedig ez lesz a legrövidebb kívánt út.

A gráf módosítása pontosabban: végigmegyek az A szomszédossági mátrixon, soronként, soron belül oszloponként és ha $A[i,j]$ nem végtelen, akkor megnézem, hogy $B[i,j] \geq 350$ teljesül-e. Ha igen, akkor nincs tennivalónk ezzel a bejegyzéssel, egyébként pedig $A[i,j]$-t végtelenre állítjuk.

```
ciklus 1 =1-tól n-ig:
    ciklus j = 1-tól n-ig:
        ha A[i,j] nem végtelen  és B[i,j] < 350:
            A[i,j] := végtelen
    ciklus vége
ciklus vége
```

Lépésszám:

1. Az új szomszédossági mátrix előállítása során minden cellánál konstans sok munkát végzek, azaz ez $O(n^2)$ lépés. (Vagy a pszeudokódra hivatkozva: a külső ciklus n-szer fut le, ennek a magja a belső ciklus, ami egy n-szer lefutó, konstans magú kód, azaz a belső ciklus $O(n)$, az egész kód pedig $O(n^2)$.)
2. A kapott mátrix is egy n csúcsú gráfhoz tartozik, ebben Dijkstra $O(n^2)$ lépésben fut, így a módosítás és a futtatás együtt $O(n^2) + O(n^2) = O(n^2)$.

Jóság: A módosítással éppen azok az élek válnak elérhetetlenné, amiket nem szabad használnunk, azaz az új gráf útjai megegyeznek az eredeti gráf használható útjaival. Dijkstra algoritmusa használható, mert minden élsúly pozitív.