Algoritmusok és gráfok pótzárthelyi
2025. december 4.

Név:

Gyakorlatvezető:

1.	2.	3.	4.	5.	6.	∑:

1. Egy algoritmus lépésszáma az n-méretű tömbökön $2n^2 + 3\sqrt{n} - 14$.

   (a) Alkalmas c konstans és n_0 küszöbérték megadásával bizonyítsuk be, hogy ez egy $O(n^2)$-es algoritmus.

   Tetszőleges $n \geq 1$ esetén *(2 pont)*

   $$2n^2 + 3\sqrt{n} - 14 \leq 2n^2 + 3n^2 + 0 = 5n^2. \quad \textit{(4 pont)}$$

   Vagyis a definícióban a $c = 5$ és az $n_0 = 1$ értékeket választva látjuk, hogy ez valóban egy $O(n^2)$-es algoritmus. *(2 pont)*

   (b) Mutassuk meg, hogy ez egy $O(2^n)$-es algoritmus.

   Az előadáson tanultak szerint minden $O(n^2)$-es lépésszámú algoritmus egyben $O(2^n)$-es is. *(2 pont)*

2. Adott egy n darab különböző számot tartalmazó T tömb, és egy m darab különböző számot tartalmazó S tömb. (A tömbök nem feltétlenül rendezettek.) Adjunk $O(n+m)$ lépésszámú algoritmust annak eldöntésére, hogy a T tömb második legkisebb eleme egyenlő-e az S tömb legnagyobb elemével.

   (a) Algoritmus leírása (pszeudokóddal vagy szöveggel):

   Az előadáson tanult algoritmussal keressük meg T legkisebb elemét: legyen ez $T[i]$. Cseréljük fel a $T[0]$ és a $T[i]$ elemeket, majd az így kapott T tömbnek a $T[1:n-1]$ résztömbjében keressük meg a legkisebb elemet.
   Az algoritmusnak a gyakorlaton látott módosításával keressük meg az S tömb legnagyobb elemét. Legyen ez s.
   Ha $s = t$, akkor a válaszunk `igaz`, különben `hamis`. *(5 pont)*

   (b) Algoritmus helyességének bizonyítása:

   A T tömb második legkisebb elemét úgy tudjuk megtalálni, hogy a T tömb legkisebb elemét nem tekintjük, és a többi elem között keressük meg a legkisebbet. *(2 pont)*

   (c) Lépésszám meghatározása (indoklással):

   A T tömb legkisebb elemének a megkeresése $O(n)$ lépés. A két elem cseréje $O(1)$ lépés. A csere után kapott T tömb $T[1:n-1]$ résztömbjében a legkisebb elem megkeresése $O(n)$ lépés. Az S tömb legnagyobb elemének megkeresése $O(m)$ lépés. A végén az összehasonlítás és a végeredmény visszaadása $O(1)$ lépés.
   Ez összesen $O(n)+O(1)+O(n)+O(m)+O(1) = O(n+m)$ lépés. *(3 pont)*

Név:

3. Binárisan kerestünk egy x egész számot az alábbi rendezett tömbben, és a keresés 2 darab összehasonlítás után véget ért. Határozzuk meg az x egész szám összes lehetséges értékét (és indokoljuk meg, hogy x értéke nem lehet más).

1	4	5	7	10	11	12

A bináris keresés során először a tömb középső elemével, azaz a 7-essel hasonlítjuk össze az x-et. *(1 pont)*

1. eset: $x = 7$.

Ekkor a keresés itt véget is ér (mindössze egyetlen összehasonlítás után), ami ellentmond annak, hogy két összehasonlítás történt. Tehát ez az eset nem lehetséges. *(1 pont)*

2. eset: $x < 7$.

Ekkor a tömb első felében, azaz a

1	4	5

résztömbben folytatjuk az eljárást. Most ennek a résztömbnek a középső elemével, azaz a 4-gyel hasonlítjuk össze az x-et. *(1 pont)*

Ha $x = 4$, akkor a keresés véget ér, méghozzá összesen két összehasonlítás után. *(1 pont)*

Ha $x \neq 4$, akkor még van további elem, amivel össze kell hasonlítanunk az x-et ($x < 4$ esetén az 1, illetve $x > 4$ esetén az 5), ami már összesen három összehasonlítás lenne, ami nem lehetséges. *(2 pont)*

3. eset: $x > 7$.

Ekkor a tömb második felében, azaz a

10	11	12

résztömbben folytatjuk az eljárást. Most ennek a résztömbnek a középső elemével, azaz a 11-gyel hasonlítjuk össze az x-et. *(1 pont)*

Ha $x = 11$, akkor a keresés véget ér, méghozzá összesen két összehasonlítás után, ami lehetséges. *(1 pont)*

Ha $x \neq 11$, akkor még van további elem, amivel össze kell hasonlítanunk az x-et ($x < 11$ esetén a 10, illetve $x > 11$ esetén a 12), ami már összesen három összehasonlítás lenne, ami nem lehetséges. *(2 pont)*

Tehát $x = 4$ vagy $x = 11$ volt a keresett szám. *(0 pont)*

4. Rendezzük az alábbi tömböt beszúrásos rendezéssel és lépésenként dokumentáljuk a tömb állapotát. Hány összehasonlítás történik a rendezés teljes futása során?

5	1	7	4

Az első lépésben összehasonlítjuk az 1-et és az 5-öt, és mivel $1 < 5$, ezért az 5-öt és az 1-et felcseréljük.

1	5	7	4

(2 pont)

A következő lépésben összehasonlítjuk a 7-et és az 5-öt, és mivel $7 > 5$, ezért nem cserélünk. *(2 pont)*

A következő lépésben összehasonlítjuk a 4-et és a 7-et, és mivel $4 < 7$, ezért a 4-et és a 7-et felcseréljük.

1	5	4	7

A következő lépésben összehasonlítjuk a 4-et és az 5-öt, és mivel $4 < 5$, ezért a 4-et és az 5-öt felcseréljük.

1	4	5	7

A következő lépésben összehasonlítjuk a 4-et és az 1-et, és mivel $4 > 1$, ezért nem csinálunk semmit. *(5 pont)*

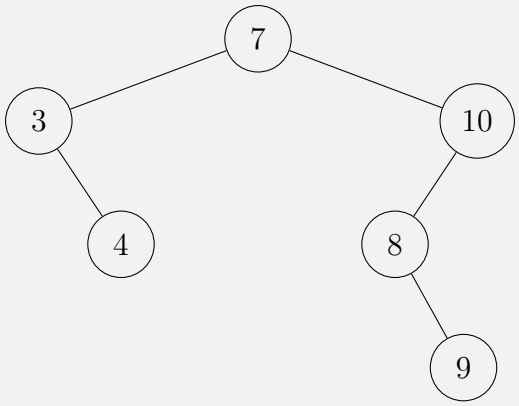
Ezzel az algoritmus véget ért.

Összesen 5 összehasonlítás történt a rendezés során. *(1 pont)*

Név:

5. Egy bináris keresőfa preorder bejárása során a fa csúcsait 7, 3, 4, 10, 8, 9 sorrendben látogatjuk meg. Rajzoljuk fel ezt a bináris keresőfát, majd indokoljuk meg, hogy miért csak így nézhet ki a fa.

Mivel a preorder bejárás először a fa gyökerét írja ki, ezért a gyökérben a 7-es van. A preorder ezután a 7-es bal részfájában lévő elemeket fogja kiírni, majd a jobb részfában lévőket. *(3 pont)*
A keresőfa-tulajdonság miatt a 7-es bal részfájában a 7-nél kisebb elemek, azaz a 3-as és a 4-es vannak, a jobb részfában pedig a 7-nél nagyobb elemek, azaz a 10-es, a 8-as és a 9-es. *(2 pont)*
Hasonlóan folytatható az érvelés a bal és a jobb részfára. *(2 pont)*



(3 pont)

6. Egy kezdetben üres, M-méretű T hash-táblába nyílt címzéssel, a $h(K) = K \pmod M$ hash-függvényt és lineáris próbát használva szúrtunk be különböző egész számokat, miközben történhettek törlések is. Adott egy olyan x egész szám, ami jelenleg T-ben van. Adjunk $O(M)$ lépésszámú algoritmust annak meghatározására, hogy hány ütközés történt az x beszúrásakor.

   (a) Algoritmus leírása (pszeudokóddal vagy szöveggel):

   Számoljuk ki a $h(x)$ értéket. Sorra nézzük meg a $h(x) - 1 \pmod M$, a $h(x) - 2 \pmod M$, ... indexű cellák tartalmát. Ha az x elemet a $h(x) - i \pmod M$ indexű cellában találjuk meg, akkor a válaszunk i. *(4 pont)*

   (b) Algoritmus helyességének bizonyítása:

   Az x elem beszúrása úgy történt, hogy először megvizsgáltuk a $h(x)$ indexű cellát, és ha az üres volt vagy csak $*$ van benne (azaz töröltünk onnan korábban egy elemet), akkor oda beszúrtuk x-et. Ha nem volt üres, akkor addig léptünk balra (ha közben elérjük a tábla elejét, akkor onnan a tábla utolsó mezőjére lépünk tovább), amíg nem találtunk egy üres vagy egy $*$-ot tartalmazó mezőt, és oda szúrtuk be x-et.
   A fenti algoritmus a $h(x)$ indexű cellából indulva számolja meg, hogy hányat kell balra lépni, amíg eljutunk az x elemhez, vagyis ez valóban az ütközések számát adja meg. *(3 pont)*

   (c) Lépésszám meghatározása (indoklással):

   A tábla minden elemét legfeljebb egyszer vizsgáljuk meg, ami $O(M)$ lépés. *(3 pont)*
   (A $h(x)$ értékének kiszámolása, illetve az eredmény visszaadása $O(1)$ lépés. Így összesen $O(M) + O(1) = O(M)$ lépést végzünk.)