Algoritmusok és gráfok
NEGYEDIK GYAKORLAT, 2019. október 4.
Megoldások néhány feladathoz

3. **(Vizsga 2018)** Az órán tanult összefésülés eljárást futtatjuk az $1, 3, 5, 9$ és $2, 10, y, 15, 16$ rendezett tömbökön, ahol y értéke nem ismert.
   (a) Hány összehasonlításban vesz részt a 9-es érték és miért?
   (b) Hány összehasonlításban vesz részt az y elem és miért?

   Megoldás Az összefésül eljárás során először az 1-et hasonlítjuk a 2-vel és mivel $1 < 2$, ezért 1 kerül először az outputra és az első tömbben lépünk egyet.

   Ezután $3 > 2$ miatt a 2 kerül az outputra és a 2. tömbben lépünk, majd $3 < 10$ és $5 < 10$ miatt a 3 és az 5 kerülnek az outputra. Ekkor az első tömbben már csak a 9 maradt, a második tömbben pedig $10, y, 15, 16$.

   Ekkor 9-et hasonlítjuk 10-zel és mivel $9 < 10$, ezért 9 kerül az outputra, vagyis a 9-es számot többet nem hasonlítjuk, így a 9-es egy összehasonlításban szerepel. Ez a válasz az (a) kérdésre.

   (b) Ekkor az első tömb elfogyott, vagyis innentől kezdve a második tömb minden elemét további összehasonlítás nélkül az outputra mozgatjuk, vagyis y egy összehasonlításban sem vett részt.

4. Adott egy tömbben $n \geq 2$ darab különböző szám és szeretnénk megtalálni azt a párt, melynek különbsége minimális (vagyis keressük a legközelebbi számpárt). Adjon erre a feladatra $O(n \log n)$ összehasonlítást használó algoritmust.

   Megoldás Az algoritmus elve a következő:

   (a) Rendezzük az A tömböt összefésüléses rendezéssel

   (b) A rendezett A tömbön végigmenve hasonlítsuk össze a szomszédos elemeket, közben jegyezzük meg, hogy mi volt az eddig látott legkisebb különbség és mely két szám között vevődött fel, ezek lesznek a keresett értékek.

   Az algoritmus pontosabban: a rendezést nem kell részletezni, mert ha egy órán tanult eljárást használunk módosítás nélkül, akkor arra lehet hivatkozni, csak a 2. lépést kell kifejteni, példáil így:

```
elso := A[0]
masodik := A[1]
minimum := A[1] - A[0]

ciklus i = 1-tól (n-1)-ig:
    ha A[i] - A[i-1] < minimum:
        minimum := A[i] - A[i-1]
        elso := A[i-1]
        masodik := A[i]
ciklus vége

return elso és masodik
```

   Ez az algoritmus helyes mert a rendezett tömbben minden elemre igaz, hogy a hozzá legközelebb álló szám a szomszédja vagyis elég csak a szomszédos párokat nézni. Az algoritmus ezeket mind megviszgálja, így biztosan megtalálja a legkisebb különbséget.

Az algoritmus lépésszáma $O(n \log n)$, mert az összefésüléses rendezés $O(n \log n)$, a második lépésben pedig egy legfeljebb n-szer lefutó ciklus van, melynek magja konstans sok lépésből áll, vagyis ez a 2. rész $O(n)$, az egész pedig $O(n \log n) + O(n)$, ami $O(n \log n)$.

5. **(ZH 2018)** Egy $n \geq 3$ elemű rendezett tömbben pontosan három különböző érték szerepel. Adjon $O(\log n)$ lépésszámú algoritmust ennek a három értéknek a megkeresésére. Például ha az input $0, 0, 1, 1, 1, 8$, akkor az elvárt kimenet $0, 1, 8$.

   Megoldás
   A tömb első és utolsó eleme két keresett értéket megad, vagyis csak a harmadik értéket kell megkeresnünk. A harmadik értéket a bináris kereséshez hasonlóan keressük, így:
   Egyre kisebb résztömbökön dolgozunk (az elején az egész tömbbel kezdünk) úgy, hogy az aktuális tömb középső elemét összehasonlítjuk a már ismert két értékkel és ha a középső elem mindkettőtől különböző, akkor megvan a harmadik elem, ha a kisebbel egyezik meg, akkor a tömb hátsó felével dolgozunk tovább (a középső után jövő cellákkal), ha pedig a nagyobbik ismert elemmel egyezik meg a középső elem, akkor a középső cella előtti részzel dolgozunk tovább.
   Lehet persze pszeudokóddal is, ahol *kicsi* és *nagy* a már ismert két érték:

```
eleje:= 0
vége:= n-1
harmadik:= ‘‘Nincs meg még!’’
ciklus amíg (megvan == ‘‘Nincs meg még!’’):
    közép := alsó egész része (eleje + vége)/2-nek
    ha A[közép] > kicsi és A[közép] < nagy:
        harmadik := A[közép]
    egyébként ha A[közép] == nagy:
        vége :=   közép -1
    egyébként:
        eleje := közép + 1
    elágazás vége
ciklus vége
return harmadik
```

   Ez jó, mert amikor szűkítem az aktuális tömböt, akkor abban mindig szerepel a keresett harmadik érték, hiszen a tömb rendezett volt.

   Lépésszám indolása: minden körben felezzük a tömböt, ezt csak $\log n$-szer lehet megtenni, egy körben konstans sok munka van, tehát $O(\log n)$ vagy az is jó, hogy pont olyan a szerkezete az algoritmusnak, mint a bináris keresésnek csak itt 2 összehasonlítás van minden felezés előtt, míg a bináris keresésben csak 1, azaz legfeljebb kétszer annyi lépés, mint a bináris keresés, de az meg $O(\log n)$, vagyis akkor ez is $O(\log n)$.