Algel V. gyakorlat

Rendezettek vagyunk

2013. március 11.

1. Legalább hány összehasonlítás kell ahhoz, hogy egy n elemű tömbből egy olyan tagot találjunk meg, ami a tömb 10 legkisebb eleme közé tartozik?

2. **[ZH: 2004. április 8.]** Az $A[1:n]$ tömbben levő elemekről tudjuk, hogy $A[1] \neq A[n]$. Adjon $O(\log n)$ összehasonlítást használó algoritmust, amely talál egy olyan i indexet, hogy $A[i] \neq A[i+1]$!

3. Rendezzük a következő listát buborék- beszúrásos- és összefésüléses rendezéssel!
4,11,9,10,5,6,8,1,2,16

4. **[ZH: 2004. március 29.]** Az $A[1\dots n]$ tömbben egész számokat tárolunk, ugyanaz a szám többször is szerepelhet. Határozzuk meg $O(n\log n)$ lépésben az összes olyan számot, amelyik egynél többször fordul elő a tömbben.

5. **[ZH: 2009. április 24.]** Adottak a $p_0 = (0,0), p_1 = (x_1, y_1), \dots p_n = (x_n, y_n), p_{n+1} = (100, 0)$ pontok a síkban ($n \geq 1$) úgy, hogy $1 \leq i \leq n$ esetén x_i és y_i racionális számok, $0 < x_i < 100$, és semelyik három pont nem esik egy egyenesbe. Egyenes szakaszokkal akarjuk ezeket a pontokat valamilyen sorrendben összekötni úgy, hogy egy $n+2$ csúcsú zárt töröttvonalat kapjunk, amiben a behúzott szakaszok nem metszik egymást. Adjon egy $O(n\log n)$ lépésszámú algoritmust annak meghatározására, melyik pontot melyikkel kössük össze!

---

6. **[ZH: 2007. április 27.]** A valós számokból álló $a_1^2 \dots a_n^2$ sorozat egy darabig nő, utána csökken. Adjon $O(n)$ összehasonlítást használó algoritmust, ami rendezi az $a_1, \dots a_n$ sorozat!

7. Az $A[1:n]$ tömbben egy rendezett univerzum n különböző eleme volt, nagyság szerint növekvő sorrendben. Valaki időközben megkeverte a tömb elemeit, de csak annyira, hogy minden egyes elem új helye az eredetitől legfeljebb 5 távolságra esik. Adjunk $O(n)$ futásidejű algoritmust az eredeti állapot helyreállítására!

8. **[ZH: 2010. április 19.]** Az A tömbben n különböző számot tárolunk. Tudjuk, hogy $A[1] > A[2]$ és $A[n-1] < A[n]$. Adjon algoritmust, mely $O(\log n)$ összehasonlítással megtalál a tömbben egy lokális minimumot (ha van), azaz egy olyan $1 \leq i \leq n$ indexet, hogy $A[i]$ tömbbeli szomszédai nagyobbak, mint $A[i]$.

9. **[Vizsga: 2007. június 19.]** Adott a síkon n pont, melyek koordinátái $(a_1, b_1) \dots (a_n, b_n)$. Olyan $P = (x, y)$ pontot keresünk a síkon, amire az alábbi összeg minimális.
$$\sum_{i=1}^{n} (|a_i - x| + |b_i - y|)$$
Adjon algoritmust, ami $O(n\log n)$ lépésben meghatároz egy ilyen P pontot.

10. Legyen adott egy egészekből álló $A[1:n]$ tömb valamint egy b egész szám. Szeretnénk hatékonyan eldönteni, hogy van-e két olyan $i, j \in \{1, \dots, n\}$ index, melyekre $A[i] + A[j] = b$. Oldjuk meg ezt a feladatot $O(n\log n)$ időben!

11. **[Vizsga: 2009. június 11.]** Adott a számegyenesen n intervallum, $[a_1, b_1], \dots, [a_n, b_n]$. Azt akarjuk tudni, hogy együtt milyen hosszú részt fednek le a számegyenesből (azaz, hogy mennyi az $\cup_{i=1}^{n} [a_i, b_i]$ összhossza). Adjon $O(n\log n)$ lépéses algoritmust ennek a hossznak a meghatározására!

12. Adottak a sík egész koordinátájú $P_1 = (x_1, y_1), \ldots, P_n = (x_n, y_n)$ koordinátájú pontjai. Javasoljunk $O(n)$ költségű módszert olyan $P_i \neq P_j$ pontok kiválasztására, amelyeken átmenő egyenes által meghatározott félsíkok közül az egyik tartalmazza az összes pontot!

13. A (növekvően) rendezett $A[1:n]$ tömb k elemét valaki megváltoztatta. A változtatások helyeit nem ismerjük. Javasoljunk $O(n + k \log k)$ költségű algoritmust az így módosított tömb rendezésére!

14. **[Vizsga: 2004. június 10.]** Az n méretű (nem feltétlenül rendezett) A tömb elemei különböző pozitív egész számok. Adjon algoritmust, amely meghatároz egy $1 \leq k \leq n$ számot és kiválaszt k különböző elemet az A tömbből úgy, hogy a kiválasztott elemek összege nem több, mint k^3. Ha nincs ilyen k, akkor az algoritmus jelezze ezt a tényt! Az algoritmus lépésszáma legyen $O(n \log n)$! (Két szám összehasonlítása, összadása vagy szorzása egy lépésnek számít.)

15. **[Vizsga: 2004. június 3.]** A $2^k - 1$ elemű A tömb elemei mind különbözőek és növekvő sorrendben vannak. Minden elemet egy k hosszú bitsorozat ír le, tehát tekinthetjük úgy, hogy a $0, 1, 2, \ldots, 2^k - 1$ számokat tároljuk egy kivételével. A feladat ennek a hiányzó számnak a megkeresése. Ehhez egy lépésben valamelyik elem egy bitjére kérdezhetünk rá: a $BIT(i, j)$ eljárás az $A[i]$ elem j-edik bitjét mondja meg. Adjon olyan algoritmust, amely a BIT eljárás $O(k)$-szori hívásával megtalálja a hiányzó számot (bitsorozatot).

16. Igazoljuk, hogy egy n elemből álló kupac felépítése $\Omega(n)$ összehasonlítást igényel!

17. **[pZH: 2012. április 25.]** Egy n-szer n-es táblázatban az $1, 2, \ldots, n^2$ egész számok vannak valamilyen sorrendben (az n^2 szám mindegyike pontosan egyszer szerepel). Szeretnénk ezeket a számokat rendberakni, úgy, hogy az i. sorban ($(i-1)n+1$)-től in-ig legyenek növekvően a számok, minden $1 \leq i \leq n$ esetén. A számokat csak egyféleképpen tudjuk mozgatni: két (nem feltétlenül szomszédos) elemet felcserélhetünk.

    (a) Adjon $O(n^2)$ cserét használó algoritmust a feladat megoldására! (A cserén kívül bármi mást korlátlanul szabad csinálnunk.)

    (b) Létezik-e olyan algoritmus, ami $O(n \log n)$ cserével megoldja a feladatot?

18. **[Vizsga: 2003. május 30.]** Adott összesen $2n$ különböző szám két n elemű halmazban, $A = \{a_1, \ldots, a_n\}$ és $B = \{b_1, \ldots, b_n\}$. Azt szeretnénk eldönteni minimális számú összehasonlításssal, hogy a $2n$ szám közül a legnagyobb az A vagy a B halmazban van-e. (Azaz nem kell feltétlenül meghatározni, melyik elem a legnagyobb, csak azt, hogy melyik halmazba tartozik.) Mutassa meg, hogy ehhez a feladathoz is legalább annyi összehasonlítás kell, amennyi $2n$ elem közül a maximális meghatározásához szükséges.

19. **[ZH: 2002. április 8.]** Adottak a $c_1, c_2, \ldots, c_n$ különböző egész számok. Ezeket szeretnénk nagyság szerint rendezni növekvő, vagy csökkenő sorrendbe úgy, hogy a szokásos összehasonlítás helyett, most a következő kérdéseket lehet feltenni: *Három kiválasztott elem közül melyik a középső?* Bizonyítsuk be, hogy a leghatékonyabb algoritmus $\Theta(n \log_2 n)$ összehasonlítást használ!