1. Rendezze a $3, 12, 1, 34, 4, 6, 0$ számsorozatot beszúrásos rendezéssel! Hány összehasonlításra volt szükség?

   Megoldás: A beszúrásos rendezésnek két fajtája van: amikor lineárisan és amikor binárisan keresünk. A beszúrásonként kapott sorozatok ugyanazok, de az összehasonlítások száma eltérhet.

   Az eljárás az, hogy az i-edik elemet szúrjuk be a már rendezett első $i-1$ elem közé, tehát a sorozat az egyes lépésekben így alakul: $(3, \mathbf{12}, 1, 34, 4, 6, 0) \longrightarrow (3, 12, \mathbf{1}, 34, 4, 6, 0) \longrightarrow (1, 3, 12, \mathbf{34}, 4, 6, 0) \longrightarrow (1, 3, 12, 34, \mathbf{4}, 6, 0) \longrightarrow (1, 3, 4, 12, 34, \mathbf{6}, 0) \longrightarrow (1, 3, 4, 6, 12, 34, \mathbf{0}) \longrightarrow (0, 1, 3, 4, 6, 12, 34)$.

   Az összehasonlítások száma lineáris keresésnél (itt feltesszük, hogy hátulról előre haladva keresünk): $1 + 2 + 1 + 3 + 3 + 6 = 16$, bináris keresésnél: $1 + 2 + 2 + 2 + 3 + 3 = 13$.

2. Rendezze a $16, 17, 2, 6, 11, 33, 28, 22$ sorozatot gyorsrendezéssel úgy, hogy mindig a tömb első elemét választja particionáló elemnek!

   Megoldás: $(\mathbf{16}, 17, 2, 6, 11, 33, 28, 22) \longrightarrow$
   $(\mathbf{6}, 11, 2 \mid 16 \mid \mathbf{17}, 33, 28, 22) \longrightarrow$
   $(2 \mid 6 \mid 11 \mid 16 \mid 17 \mid \mathbf{33}, 28, 22) \longrightarrow$
   $(2 \mid 6 \mid 11 \mid 16 \mid 17 \mid \mathbf{22}, 28 \mid 33) \longrightarrow$
   $(2 \mid 6 \mid 11 \mid 16 \mid 17 \mid 22 \mid 28 \mid 33)$

3. Rendezze a következő láncokat a radix rendezés segítségével: *abc*, *acb*, *bca*, *bbc*, *acc*, *bac*, *baa*.

   Megoldás:
   bca, *baa*, *acb*, *abc*, *bbc*, *acc*, *bac*
   baa, *bac*, *abc*, *bbc*, *bca*, *acb*, *acc*
   abc, *acb*, *acc*, *baa*, *bac*, *bbc*, *bca*

4. Dr. Watson azzal állít be Sherlock Holmes-hoz, hogy olyan összehasonlítás-alapú rendezési algoritmust talált, ami úgy rendez akármekkora tömböt, hogy minden egyes tömbbeli szám legfeljebb 2024 összehasonlításban szerepel. Mivel indokolhatja Sherlock Holmes, hogy Watson téved?

   Megoldás: Ha minden elem legfeljebb 2024 összehasonlításban szerepel, akkor az algoritmus az n elemmel összesen $2024n/2$ összehasonlítást végez. De tudjuk, hogy minden összehasonlítás-alapú algoritmusnak kell legalább $\log n! = \Omega(n \log n)$ összehasonlítás, és $2024n/2 \notin \Omega(n \log n)$, tehát valóban nincs ilyen algoritmus.

---

5. Adjon egy $O(n)$ időigényű algoritmust n olyan egész számból álló sorozat rendezésére, melynek elemei az

   (a) $\{1, \ldots, 3n\}$ halmazból vannak!

   (b) $\{1, \ldots, n^3 - 1\}$ halmazból vannak!

   Megoldás: (a) Használjunk ládarendezést $3n$ ládával! Ekkor a lépésszám $O(n + 3n) = O(n)$.

   (b) Az egyszerű ládarendezés itt nem elég, használjuk a radix rendezést! Képzeljük el a számokat az n alapú számrendszerben felírva. Ekkor mindegyik (legfeljebb) 3 jegyű. Ha ebben az alakban használjuk a radix rendezést, akkor 3 ládarendezés kell, mindegyiknél n láda, tehát az egész valóban $O(n)$ lépést használ.

   Megjegyzés: Az n alapú számrendszerbe való átírás is megoldható $O(n)$ aritmetikai művelettel, hiszen minden számot maradékosan el kell osztani n-nel, a maradék adja az utolsó jegyet, a hányadost újra elosztjuk n-nel, az itteni maradék adja a következő számjegyet, a hányados meg az elsőt.

6. Az $A[1 : n]$ tömbben számokat tárolunk. Határozza meg $O(n \log n)$ lépésben

   (a) azokat az értékeket, amelyek egynél többször fordulnak elő;

   (b) a leggyakoribb értékeket (vagyis azokat, amelyeknél többször semelyik másik szám sem fordul elő a tömbben)!

Megoldás: (a) Ha a tömb rendezve van, akkor egyszerűbb a feladat, hiszen akkor az egyforma értékek egymás mellett vannak. *Algoritmus:* Először rendezzük a tömböt, például az összefésüléses rendezéssel, aminek lépésszáma $O(n \log n)$ (a többi eddig tanult rendező algoritmus nem lenne jó, mert azok nem $O(n \log n)$ lépésszámúak, ha nem csak az összehasonlításokat számoljuk). Majd végigmegyünk a tömbön, kiírjuk minden olyan elemet, ami határozottan nagyobb az előzőnél (vagy nincs előző) és egyenlő a következővel. *Jóság:* A rendezés miatt az egyforma elemek egy blokkban lesznek, minden blokk első elemét írjuk ki. Ha egy szám csak egyszer fordul elő, akkor nem fogjuk kiírni. *Lépésszám:* A rendezés $O(n \log n)$, utána a kiírás lépés, tehát összesen $O(n \log n)$ lépést végzünk.

(b) Rendezzük a tömböt, mint előbb. Most azt kell meghatározni, hogy a rendezett tömbben melyik elem alkotja a leghosszabb blokkot. *Algoritmus:* Először rendezzük a tömböt. Utána végigmegyünk a tömbön és egy-egy számlálóval nyilvántartjuk, hogy mennyi az eddigi legnagyobb blokk hossza, ott milyen érték vagy értékek voltak, és számoljuk az aktuális blokk hosszát. Ha az aktuális blokk hossza nagyobb vagy egyenlő a korábbi legnagyobbnál, akkor frissítjük az első két számlálót. *Jóság:* Ha hosszabb blokk következik, akkor csak az ott levő értéket tartjuk már nyilván. Ha az eddigi leghosszabb blokkal egyenlőt találunk, akkor pedig az új értéket is megjegyezzük. *Lépésszám:* A rendezésen kívül itt is $O(n)$ sok lépés történik, ezért a lépésszám $O(n \log n)$.

7. Egy tömbön gyorsrendezést futtatva az első particionálás után az eredmény: $4, 2, 3, 1, 6, 8, 11$. Mi lehetett a particionáló elem?

   Megoldás: Olyan elemet kell keresni, hogy az előtte levők a nála kisebbek, az utána levők a nála nagyobbak. Tehát a 6, a 8 és a 11 is jó, de más nem.

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 $A[i] \neq A[i + 1]$.

   Megoldás: Bináris kereséshez hasonló algoritmust használunk: legyen $i = \lceil n/2 \rceil$. Ha $A[i] \neq A[1]$, akkor az $A[1 : i]$ tömb is ugyanolyan tulajdonságú, mint az eredeti, elég ebben keresni. Ha viszont $A[i] = A[1]$, akkor nyilván $A[i] \neq A[n]$, és akkor a továbbiakban az $A[i : n]$ tömbben kereshetünk. Így ennél a lépésnél biztosan egy kisebb tömböt kapunk, ha $n \geq 3$.

   Az eljárás végén a tömb leszűkül egy 2 eleműre, azaz két szomszédos nem egyenlő elemre, ekkor készen vagyunk.

   A lépésszám, ahogy a bináris kereséséhez hasonlóan $O(\log n)$ lesz.

9. Legyen adott egy egészekből álló $A[1 : n]$ tömb valamint egy b egész szám. Egy olyan $i, j \in \{1, \ldots, n\}$ indexpárt keresünk, melyre $A[i] + A[j] = b$. Hogyan lehet ezt $O(n \log n)$ időben megoldani?

   Megoldás: Vegyük észre, hogy ha minden lehetséges indexpárt ki akarunk próbálni, akkor $\binom{n}{2} \in \Theta(n^2)$ lehetőség van, ami több, mint a kívánt lépésszám.

   Itt is sokat segít a rendezettség, ezért rendezzük a tömböt $O(n \log n)$ lépésben, összefésüléses rendezéssel.

   Ezután az $i = 1, 2, \ldots, n$ indexekre keressük a tömbben a $b - A[i]$ értéket. Tegyük fel, hogy $A[j] = b - A[i]$, ekkor egy jó indexpár a rendezett tömbben az (i,j). Meg kell még keresni ezeknek az $A[i]$ és $A[j]$ elemeknek az eredeti helyét az eredeti tömbben, ez további két bináris kereséssel megkapható.

   Ez az eljárás azért jó, mert $A[i] + A[j] = b$ pontosan akkor teljesül valami i és j indexpárra, ha $A[j] = b - A[i]$, azaz $b - A[i]$ benne van a tömbben.

   Ha bináris keresést használunk $b - A[i]$ megkeresésére is, akkor mindegyik (összesen n darab) keresés $\Theta(\log n)$ lépésszámú, tehát a lépésszám (a végén levő két extra kereséssel együtt) $O((n + 2) \log n) = O(n \log n)$.

   Megjegyzés: a rendezés utáni rész lineárisan is megoldható. Kiindulunk az $i = 1$, $j = n$ indexekből és általában, ha $A[i] + A[j] < b$, akkor az i értékét növeljük eggyel, ha pedig $A[i] + A[j] > b$ akkor a j értékét csökkentjük eggyel. Akkor hagyjuk abba, ha $A[i] + A[j] = b$ vagy ha $i > j$ lenne – ez utóbbi esetben nincs megoldás. (Miért?)

10. Az eredetileg növekvő $a_1, \ldots, a_n$ sorozatban egy elem értéke megváltozott, de nem tudjuk melyik. Hogyan lehet $O(n)$ lépésben újra növekvő sorrendbe rendezni az elemeket?

    Megoldás: Vegyük észre, hogy ha a változtatás után is $a_i < a_{i+1}$ minden i-re, akkor bár nem tudjuk megállapítani, melyik változott, de erre nincs is szükség, a sorozat továbbra is rendezett.

    Ha viszont van olyan i, melyre $a_i > a_{i+1}$, akkor ez úgy keletkezett, hogy a_i és a_{i+1} valamelyike változott, a_i nőtt vagy a_{i+1} csökkent. A legegyszerűb (nem feltételnül leggyorsabb) módszer, ha ezt a két problémás elemet beszúrjuk a többi elem által alkotott növekvő sorozatba. Ezt végezhetjük lineáris kereséssel, amikor $O(n)$ lépést végzünk. (Bináris keresésnél az összehasonlítások száma logaritmikus, de a mozgatásokra akkor is elmehet lineárisan sok lépés.)

    Megjegyzés: Egyetlen beszúrással is megoldható, hogyan?

11. Adjon minél kevesebb összehasonlítást használó algoritmust, ami n elem közül megtalálja a két legkisebbet!

    Megoldás: Előbb a legkisebb, majd a többiből megint a legkisebb meghatározása $n-1+n-2 = 2n-3$ összehasonlítás. Ennél jobbat is kaphatunk. Gondoljunk egy kieséses bajnokságra. A győztes legyen a legkisebb. A második nyilván csak olyan lehet, akit a későbbi győztes vert ki (a többieknél van legalább kettő nagyobb). Tehát a másodikra körönként egyetlen jelölt van. A bajnokság lebonyolítható $\lceil \log n \rceil$ körben, ennyi közül kell kiválasztani a másodikat, tehát a feladat összesen $n + \lceil \log n \rceil - 2$ lépésben megoldható.

12. Tudjuk, hogy az $a_1, \ldots, a_n$ sorozat olyan, hogy egy darabig növekszik, utána csökken. Adjon $O(n)$ összehasonlítást használó algoritmust, ami növekvő sorrendbe rendezi az elemeit!

    Megoldás: Ötlet: a legnagyobb elemnél kettévágjuk a sorozatot. A kapott két rész mindegyike rendezett, ezeket össze tudjuk fésülni egyetlen rendezett sorozattá.

    Algoritmus: Sorban az a_i – a_{i+1} szomszédokat összehasonlítva megtalálhatjuk a legnagyobb elemet: Ha $a_{x-1} < a_x$ és $a_x > a_{x+1}$ akkor a_x a legnagyobb elem. Ha nincs ilyen, vagyis $a_{i-1} < a_i$ minden i-re teljesül akkor nincs további tennivalónk, hiszen a sorozat növekszik. Eddig $n-1$ összehasonlítást használunk. Az $a_1, \ldots, a_x$ és az $a_n, a_{n-1}, \ldots, a_{x+1}$ növekvő sorozatokat a tanult módon legfeljebb $n-1$ összehasonlítással összefésüljük, és így legfeljebb $2n-2 \in O(n)$ összehasonlítással készen vagyunk. (Az algoritmus mozgatja is az elemeket, de összesen ez is $O(n)$ lépés.)

    Megoldhatjuk az a_x előzetes megkeresése nélkül is. Elég azt képzelni, hogy két sorozatunk van, az egyik első eleme a_1, a másiké a_n (és kezdetben nem tudjuk, hol érnek véget). Legyen $i = 1$ és a $j = n$, és hasonlítsuk össze az a_i és az a_j elemet. A kisebb a készülő rendezett lista következő eleme, és ha $a_i < a_j$, akkor az i értékét növeljük, különben a j értékét csökkentjük. A továbbiakban is így, az összefésüléshez hasonlóan folytatjuk. Akkor lesz vége, ha $i = j$ (és akkor ez a legnagyobb elem). Ez az eljárás legfeljebb $n-1 \in O(n)$ összehasonlítást használ.

    Megjegyzés: mindkét eljárás helyesen működik abban az esetben is, amikor a legnagyobb elem az a_1 vagy az a_n, akkor eredetileg is egy rendezett sorozatunk van (növekvő vagy csökkenő).

13. Az n méretű (nem feltétlenül rendezett) A tömb elemei különböző pozitív 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 $O(n \log n)$.

    Megoldás: Vegyük észre, hogy egy rögzített k esetén, ha van megoldás, akkor a tömb legkisebb k eleme is megoldás. Ha ezek nem jók, akkor pedig nincs megoldás. Tehát azt kell vizsgálni, hogy van-e olyan k, hogy a legkisebb k elem összege legfeljebb k^3.

    Ehhez rendezzük a tömböt $O(n \log n)$ lépésben, például összefésüléses rendezéssel. Legyen $k = 1$ és $T = A[1]$. Ha $T \leq k^3$, akkor készen vagyunk, különben növeljük k-t és adjuk T-hez az $A[k]$ számot. Ha egyik $k \leq n$ sem jó, akkor nincs megoldás.

    A rendezés utáni rész n összehasonlítást és $O(n)$ összeadást használ, ezért összesen a lépésszám $O(n \log n)$.

14. Adott a síkon n pont, melyek koordinátái $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$. Olyan $P = (x, y)$ pontot keresünk a síkon, amire a $\sum_{i=1}^{n}(|a_i - x| + |b_i - y|)$ összeg minimális. Adjon algoritmust, amely $O(n \log n)$ lépésben meghatároz egy ilyen P pontot!

Megoldás: A felírt összeget minimalizálhatjuk úgy, hogy külön meghatározzuk az x koordinátában és külön az y koordinátában a minimumot. Tehát például olyan x kell, amire a $\sum_{i=1}^{n} |a_i - x|$ összeg minimális. Ehhez képzeljük el az a_i pontokat a számegyenesen. Ha x kisebb, mint a legkisebb a_i, akkor x-et növelve az összeg csökken. Amíg kevesebb a_i van ami kisebb x-nél, mint ami nagyobb nála, addig x-et növelve az összeg tovább csökken. Akkortól nő, amikor már több a_i van ami x-nél kisebb, mint ahány nagyobb. Tehát az optimális x a rendezés szerinti középső a_i, ha n páratlan. Páros esetben mindegy, hogy hol van a középső intervallumon belül, például az $x = a_{n/2}$ választás jó.

Ezek után az algoritmus: rendezzük az a_i értékeket, és legyen x a rendezés szerinti $\lceil n/2 \rceil$ szám. Ugyanezt megcsináljuk y-ra: rendezzük a b_i értékeket, és legyen y a rendezés szerinti $\lceil n/2 \rceil$ szám.

Az algoritmus lépésszáma, összefésüléses rendezés esetén $O(n \log n)$.

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

Megoldás: Egy intervallum hossza $b_i - a_i$, de az unió lehet ezek összegénél kevesebb, ha az intervallumok metszik egymást. Ezeket az átfedéseket kell valahogy kezelni, pontosabban megtalálni azokat a diszjunkt intervallumokat, amik az uniót alkotják.

Rendezzük a végpontokat (mind a $2n$ számot) úgy, hogy tároljuk, melyik volt egy intervallum kezdőpontja és melyik egy intervallum végpontja. Az unió első része a legkisebb értéknél kezdődik (és ez szükségszerűen egy intervallum a_j kezdete). Menjünk a rendezett sorozatban addig, amíg először megegyezik a kezdő és végpontok száma. Ez egy végpontnál kövezkezik be (b_k), és vegyük észre, hogy itt ér véget az unió első része, ennek a résznek a hossza $b_k - a_j$. Ha még maradtak a sorozatban elemek, ugyanígy folytathatjuk.

A lépésszámhoz azt kell látni, hogy a rendezés után már csak végig kell menni a sorozaton, számlálva a kezdő- és végpontokat. A hossz meghatározásához legfeljebb n kivonás kell, majd ezek eredményét kell összeadni. A rendezés lépésszáma $O(2n \log(2n)) = O(n \log n)$, az utána következőké $O(n)$, tehát ez összesen $O(n \log n)$.

16. Adjon minél kevesebb összehasonlítást használó algoritmust, ami n elem közül megtalálja a legkisebbet és a legnagyobbat is!

Megoldás: Külön a legkisebbet és a legnagyobbat is meg lehet találni $n - 1$ összehasonlítással, azaz megoldható $2n - 2$ összehasonlítással. (Sőt $(2n - 3)$-mal is. Miért?)

Ennél jobb a következő: előbb hasonlítsuk össze az elsőt a másodikkal, a harmadikat a negyedikkel, stb. Ez után vegyük minden párból a kisebb elemet. A legkisebb közülük kerül ki. Használjuk ezekre a minimumot kereső algoritmust. Hasonlóan a páronkénti nagyobb elemek között találjuk meg a maximumot. Ha páratlan sok elem van, akkor az utolsó pár helyett 3 elemből határozzuk meg a kicsit és nagyot.

A lépésszám: $\lfloor n/2 \rfloor + 2 + 2(\lfloor n/2 \rfloor - 1)$, ami kb. $1{,}5n$. Pontosabban, ha n páros, akkor $1{,}5n - 2$, páratlan esetben pedig $(n-1)/2 + 2 + 2(n-1)/2 - 2 = 1{,}5(n-1)$.

Megjegyzés: meggondolható, hogy $1{,}5n - 2$ kell is. Legyen B azoknak az elemeknek a száma, amelyek még lehetnek legkisebbek és legnagyobbak is, I, ami már csak legkisebb lehet (volt már nála nagyobb), A, ami már csak legnagyobb lehet (volt nála kisebb, E pedig a többi, aminél már volt kisebb és nagyobb is. Kezdetben $|B| = n$, a többi üres, a végén B üres, $|I| = |A| = 1$.

Két B-beli összehasonlításakor B kettővel csökken, I és A eggyel-eggyel nő. Ha egy B-belit nem B-belivel hasonlítunk, akkor B csökken és mindig lehet olyan eredmény, amikor I és A egyike nő, a másik nem csökken (azaz I és A uniója nem csökken). Ha két I-belit vagy két A-belit hasonlítunk, akkor egyikük átkerül az E-be. A többi esetben mindig lehet olyan válasz, hogy sem I sem A nem változik.

Ahhoz, hogy a B kiürüljön ezek szerint kell legalább $n/2$ összehasonlítás és eközben I és A uniója nem változik. A B-ből minden elem előbb az I-be vagy A-ba kerül, ahonnan viszont egyszerre csak egy tud kikerülni, tehát ezek (majdnem) kiürítéséhez kell összesen legalább $n-2$ összehasonlítás. Tehát együtt van legalább $1{,}5n-2$.