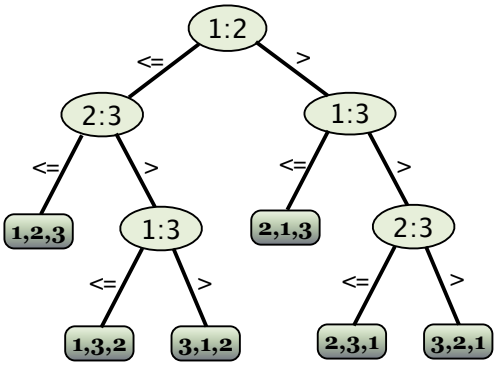


1. ábra.

Alsó korlát rendezési algoritmusokra

Minden olyan rendezési algoritmusnak a futását, amely elempárok egymással való összehasonlítása alapján működik leírja egy bináris döntési fa.

Az algoritmus által a legrosszabb esetben szükséges összahasonlítások számát a neki megfelelő döntési fa magassága adja meg.

Tétel Egy n darab elemet rendező döntési fa magassága $\Omega(nlogn)$.

Bizonyítás A fa leveleinek száma a lehetséges sorrendek száma, azaz $n!$. Ha a fa magassága h, akkor $2^{h-1} \geq n!$ kell teljesüljön, következésképpen $h-1 \geq \lg(n!) \geq lg((n/e)^n) = n\lg n - n\lg e = \Omega(n\lg n)$.

Következmény Minden olyan rendezési algoritmusra, amely csak az elempárok összehasonlítása alapján működik $T_{lr}(n) = \Omega(n\lg n)$.

Számláló rendezés

Amennyiben a rendezendő elemek által felvehető értékek halmazának számossága kicsi, akkor megadható lineáris időigényű algoritmus. A bemenet a rendezendő elemek egy n méretű A tömbben eltárolva, a kimenet ezen számok egy B tömbben rendezetten. Továbbá feltesszük, hogy a rendezendő elemek mindegyike az $\{1,\ldots,k\}$ halmazba esik. $C[i]$ tartalmazza azt az információt, hogy a legfeljebb i nagyságú elemből hány darab van.

```
Szamlalorend(A,B,k)
  for i=1 to k
    C[i]:=0
  for j=1 to n
    C[A[j]]:=C[A[j]]+1
```

```
  for i=2 to k
    C[i]:=C[i]+C[i-1]
  for j=n downto 1
     {B[C[A[j]]]:=A[j]
      C[A[j]]:=C[A[j]]-1}
```

Megjegyzés: Az algoritmus ötlete nem csak akkor használható, ha az elemek az $\{1,\ldots,k\}$ halmazba esnek. Általánosabb esetben meg kell keresni a minimális elemet *min*-t, és $C[i]$ a legfeljebb $i-min+1$ nagyságú elemeket tartalmazza.

Definíció Egy rendezést stabilnak nevezünk, ha az azonos értékű elemek ugyanabban a sorrendben jelennek meg a kimeneti tömbben, mint ahogy a bemeneti tömbben szerepeltek.

A leszámláló rendezés stabil.

Futási idő: $\Theta(n+k)$

Példa

```
A=[3,1,3,1,2,1,1,2,3]
C=[4,2,3]
C=[4,6,9]
B=[0,0,0,0,0,0,0,0,3]
C=[4,6,8]
B=[0,0,0,0,0,2,0,0,3]
C=[4,5,8]
B=[0,0,0,1,0,2,0,0,3]
C=[3,5,8]
```

Számjegyes vagy radix rendezés

Példa

```
329 720 720 329
457 436 329 436
657 457 436 457
839 657 839 657
436 329 457 720
720 839 657 839
```

Bemenet: H k dimenziós vektorok halmaza, ahol az *i*-edik komponens minden *i*-re egy olyan elemtípusból kerül ki, amelyen adott egy rendezési reláció.

Kimenet: H elemei a lexikografikus rendezés alapján rendezve.

Definíció Egy $X=(x_1,\ldots,x_k)$ vektor megelőzi az $Y=(y_i,\ldots,y_k)$ vektort a lexikografikus rendezésben, ha $\exists i$, amelyre teljesül, hogy $\forall j<i\ x_j=y_j$ és $x_i<y_i$.

Elvi algoritmus

```
for i:=d downto 1
   H Stabil rendezése a kulcs i-edik jegye szerint
```

Helyesség: A ciklusmag végrehajtása után a H halmaz rendezett lesz arra a kulcsra nézve, amely az eredeti kulcs [i..d] jegyeit (karaktereit) tartalmazza. Bizonyítás i-szinti indukcióval.

Futási idő Ha a jegyek szerinti rendezés lineáris idejű (pl számláló), akkor a futási idő $O(dn)$.

Edényrendezés

Tegyük fel, hogy a rendezendő $H = \{a_1, \ldots, a_n\}$ halmaz elemei a $[0,1)$ intervallumba eső valós számok. Vegyünk m db vödröt, $V[0], \ldots, V[m-1]$ és osszuk szét a rendezendő halmaz elemeit a vödrökbe úgy, hogy az a_i elem az $\lfloor m \cdot a_i \rfloor$ sorszámú vödörbe kerüljön. Majd rendezzük az egy vödörbe került elemeket, és a vödrök sorszáma szerinti növekvő sorrendben fűzzük össze a rendezett részsorozatokat (vödröket).

Edényrendezés

For i=1 to n
    Beszúrjuk a_i-t a $B[\lfloor na_i \rfloor]$ listába
For i=0 to m-1
    Rendezzük a $B[i]$ listát beszúró rendezéssel
Összefűzzük a $B[0], B[1], \ldots, B[m-1]$ listákat.

Helyesség: Ha két elem ugyanabba a vödörbe kerül, akkor a beszúró rendezés miatt lesz megfelelő a sorrendjük, egyébként pedig a vödrök összefűzésének sorrendje miatt.

Példa

Legyen $A = \{0.12, 0.21, 0.68, 0.26, 0.72, 0.94, 0.78, 0.17, 0.23, 0.39\}$, és legyen $m = 10$.

Ekkor a vödrök tartalma $B[1] = \{0.12, 0.17\}$, $B[2] = \{0.21, 0.26, 0.23\}$, $B[3] = \{0.39\}$, $B[6] = \{0.68\}$, $B[7] = \{0.72, 0.78\}$, $B[9] = \{0.94\}$.

Futási idő: Tegyük fel, hogy $m = n$, ekkor:

- legjobb eset $\Theta(n)$,
- legrosszabb eset $\Theta(n^2)$,
- átlagos eset $\Theta(n)$.

Kiválasztási probléma

Bemenet: Azonos típusú (különböző) elemek $H = \{a_1, \ldots, a_n\}$ halmaza, amelyeken értelmezve van egy $\leq$ lineáris rendezési reláció és egy i $(1 \leq i \leq n)$ index.

Kimenet: Olyan $x \in H$, hogy $|\{y : y \in H \wedge y \leq x\}| = i$.

A kimenetben szereplő x-et a H rendezett minta i-edik (legkisebb) elemének nevezzük. A középső elemet a rendezett minta mediánjának hívjuk. Pontosabban, az $i = \lfloor (n+1)/2 \rfloor$ -edik elem az alsó, az $i = \lceil (n+1)/2 \rceil$ -edik elem a felső medián.

Feloszt eljárás (Hoare féle)

```
Feloszt(A,p,r)
  x:=A[p]
  i:=p-1
  j:=r+1
  while(Igaz)
       {repeat j:=j-1
              until A[j]<=x
        repeat i:=i+1
              until A[i]>=x
         if i<j
             then csere(A[i],A[j])
             else return j}
```

Példa:
$i[5,3,2,6,1,4,3,7]\,j$ x=5
$[5^i,3,2,6,1,4,3^j,7]$
$[3^i,3,2,6,1,4,5^j,7]$
$[3,3,2,6^i,1,4^j,5,7]$
$[3,3,2,4^i,1,6^j,5,7]$
$[3,3,2,4,1^j,6^i,5,7]$
Return 5

Kiválasztás ismételt felosztással

Az algoritmus használja a gyorsrendezésnél definiált Feloszt(A,p,r) eljárást, amely átrendezi az A tömb A[p] és A[r] közé eső szakaszát úgy, hogy a visszaadott q értékre $A[p,\ldots,q]$ minden eleme kisebb vagy egyenlő $A[q+1,\ldots,r]$ minden eleménél. Az alábbi algoritmus a Hoare felosztást használja, ahol nem garantált, hogy $A[p,\ldots,q-1]$ minden eleme kisebb vagy egyenlő, mint $A[q]$.

Elvi algoritmus

```
Kiválaszt(A,p,r,i)
  If p=r
      Then return A[p]
  q:=Feloszt(A,p,r)
  k:=q-p+1
  if i ≤ k
      Return Kiválaszt(A,p,q,i)
  else
      Return Kiválaszt(A,q+1,r,i-k)
```

1. Példa: A=[6,4,3,7,5,1,8,9], i=2
A feloszt eljárás után $A=[1,4,3,5,7,6,8,9]$ és $q=4$
A rekurzív hívás az A,1,4,2 értékekre történik azaz a $[1,4,3,5]$ résztömbben keressük a második legkisebb elemet.

2. Példa: A=[6,4,3,7,5,1,8,9], i=6
A feloszt eljárás után $A=[1,4,3,5,7,6,8,9]$ és $q=4$
A rekurzív hívás az A,5,8,2 értékekre történik azaz a $[7,6,8,9]$ tömb minimális elemét keressük.

Megjegyzés Ha olyan felosztást használunk, ahol a felosztó elemtől balra eső elemek a felosztó elemnél mind kisebbek vagy egyenlők (pl Lomuto) akkor a baloldali rekurzív hívás az $A[p,q-1]$ tömbre vonatkozik, és $q=i$ esetén a felosztó elem az i-edik legkisebb így az eljárást nem kell folytatnunk.

Futási idő elemzés

Legjobb eset: $O(n)$
Legrosszabb eset: $O(n^2)$
Átlagos eset: $O(n)$

Bizonyítás: Feltesszük, hogy minden bemenet egyformán valószínű. Legyen $T_a(n)$ az n méretű input átlagos költsége. Az q felosztó index lehetséges értékei: $q=1,2,\ldots,n-1$. A keresés vagy a bal oldali részben folytatódik, amelynek hossza ekkor q, vagy a jobboldaliban, amelynek hossza $n-q$.

Feltéve, hogy $T_a(n)$ monoton növekvő függvény, használva, hogy a felosztás lineáris idejű kapjuk, hogy

$$T_a(n) \leq \frac{1}{n-1}\sum_{q=1}^{n-1}\max\{T_a(q),T_a(n-q)\}+O(n).$$

Az állítást indukcióval igazoljuk, ehhez tegyük fel, hogy van olyan c, amelyre $T_a(k) \leq c \cdot k$ teljesül $k < n$ esetén. Továbbá legyen α a feloszt eljárás futási idejében szereplő multiplikatív konstans. Ekkor azt kapjuk

$$T_a(n) \leq \frac{2}{n-1} \sum_{q=\lfloor n/2 \rfloor}^{n-1} c \cdot q + \alpha n$$

Végrehajtva az összegzéseket adódik, hogy

$$T_a(n) \leq 3cn/4 + c/2 + an = cn - (cn/4 - c/2 - an).$$

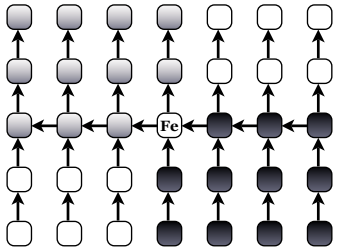
Megfelelően nagy c-t választva a fenti kifejezés jobboldala kisebb lesz, mint cn.

Kiválasztás lineáris időben

Linkiválaszt algoritmus

- 1. Osszuk a bemeneti tömb n elemét $\lfloor n/5 \rfloor$ 5 elemű csoportra, a maradék elemekből alkossunk egy további csoportot.
- 2. Minden csoportnak keressük meg a mediánját úgy, hogy az 5 elemű csoportokat (és az esetleges egyetlen kisebb elemszámú csoportot) rendezzük.
- 3. A Linkiválaszt algoritmus rekurzív hívásával határozzuk meg a kapott mediánokból álló halmaz mediánját.
- 4. A Feloszt eljárás alapján osszuk fel az így kiválasztott elem körül a bemeneti tömböt, legyen k az elemnél nem nagyobb tőle különböző elemek száma, $n-k-1$ az elemnél nagyobb elemek száma.
- 5. A Linkiválaszt algoritmus rekurzív hívásával határozzuk meg a baloldali részben az i-edik elemet, ha $i \leq k$, illetve a jobboldali részben az $i-k-1$-edik elemet, ha $i > k+1$. Az $i = k+1$ esetben a felosztó elem a keresett elem.

Elemzés



2. ábra.

Az ábrán a sötét elemek kisebbek a felosztó elemnél, a világosak nagyobbak, így a rekurzív hívás során legfeljebb $7n/10+6$ elemre kell végrehajtani az algoritmust. Tehát elegendően nagy n esetén a legrosszabb eset korlátra fennáll a következő rekurzív összefüggés:

$$T_{lr}(n) \leq T_{lr}(\lceil n/5 \rceil) + T_{lr}(7n/10+6) + O(n).$$

Teljes indukcióval igazolható, hogy $T_{lr}(n) = O(n)$.

Megvalósítás: Az 5-ös csoportokat úgy választjuk ki, hogy a tömb első, második, harmadik, negyedik és ötödik ötödéből is veszünk egy-egy elemet, így az ötösöket tudjuk úgy rendezni, hogy a mediánok a tömb közepső részére kerüljenek. Ezáltal a Linkiválaszt algoritmus helyben megvalósítható csak a kiindulási tömb felhasználásával.

Megjegyzés: A lineáris idejű mediánválasztással egyszerűen módosítható a gyorsrendezés algoritmusa úgy, hogy legrosszabb esetben is $O(n \log n)$ legyen a futási ideje. Annyit kell csak tenni, hogy a felosztó elemnek mindig a mediánt választjuk.

Kiválasztás kupaccal

A bináris kupac egy majdnem teljes bináris fa, amely minden szintjén teljesen kitöltött kivéve a legalacsonyabb szintet, ahol balról jobbra haladva egy adott csúcsig vannak elemek. A fát egy tömbben reprezentáljuk, minden elem a szint szerinti bejárás szerinti sorszámának megfelelő eleme a tömbnek. A kupacot reprezentáló A tömbhöz két értéket rendelünk, hossz(A) a tömb mérete, kupacmeret(A) a kupac elemeinek a száma.

A kupac gyökere A[1], a szerkezeti kapcsolatok egyszerűen számolhatóak:

- $A[i]$ bal fia $A[2i]$
- $A[i]$ jobb fia $A[2i+1]$
- $A[i]$ apja $A[\lfloor i/2 \rfloor]$

A kupac minden gyökértől különböző elemére teljesül, hogy az értéke nem lehet nagyobb, mint az apjáé. Ennek következménye, hogy a kupac minden részfájára teljesül, hogy a gyökéreleme maximális.

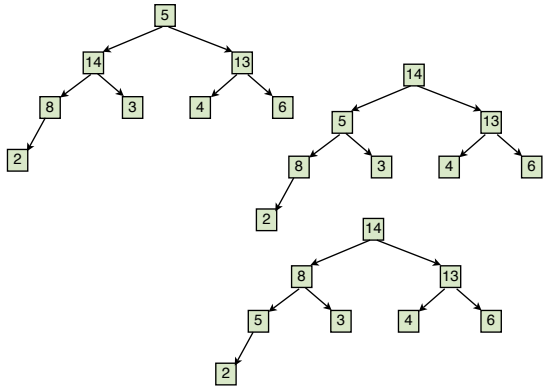
MAXIMUM-KUPACOL Eljárás

A MAXIMUM-KUPACOL eljárás helyreállítja az A[i] elemre a kupactulajdonságot. Az elemet süllyeszti cserékkel mindaddig, amíg a tulajdonság sérül.

```
MAXIMUM-KUPACOL(A,i)
l:=2i              //az A[i] elem bal fiának indexe
r:=2i+1            //az A[i] elem jobb fiának indexe
if l<=kupacmeret(A) and A[l]>A[i]
     then max:=l
     else max:=i
if r<=kupacmeret(A) and A[r]>A[max]
     then max:=r      // A[max] a három elem közül a legnagyobb
if max!=i
   then {Csere(A[i],A[max])
         MAXIMUM-KUPACOL(A,max)}
```

Példa

```
A=[5,14,13,8,3,4,6,2]
MAXIMUM-KUPACOL(A,1)
A=[14,5,13,8,3,4,6,2]
A=[14,8,13,5,3,4,6,2]
```



3. ábra.

Kiválasztás kupaccal

Az algoritmus során a for ciklus j értékkel való végrehajtása után a kupac az első j elemből az i legkisebbet tartalmazza.

```
Kupacvalaszt(A,i)
 Kupacmeret(A):=i
 for j= [i/2] to 1
    MAXIMUM KUPACOL(A,j)
 // az elso i elembol egy kupac
 for j=i+1 to n
    {if A[1]>A[j]
      then {Csere(A[1],A[j])
            MAXIMUM KUPACOL(A,1)}}
return A[1]
```

Futási idő: $O(i)+(n-i)O(\log i)$

Példa

```
A=[2,5,7,3,6,4,8], k=3
A=[2,5,7|3,6,4,8]
A=[7,5,2|3,6,4,8]
A=[3,5,2|7,6,4,8]
A=[5,3,2|7,6,4,8]
A=[4,3,2|7,6,5,8]
```

Minimum és maximum egyidejű választása

A feladat adott n méretű tömb elemeiből kiválasztani a maximális és minimális elemet.

```
MaxMin(A)
if (n%2=0) then
    {k:=2
    if (A[1]<A[2]) then
```

```
       {mini:=1
        maxi:=2}
    else
       {mini:=2
        maxi:=1}}
else
    {k:=1
     mini:=1
     maxi:=1}

while(k<n)
    {if A[k+1]<A[k+2] then
        {if A[k+1]<A[mini] then mini:=k+1
         if A[k+2]>A[maxi] then maxi:=k+2}
    else
        {if A[k+2]<A[mini] then mini:=k+2
        if A[k+1]>A[maxi] then maxi:=k+1}
    k:=k+2}
```

Futási idő Az algoritmus legfeljebb $\lfloor 3n/2 \rfloor$ összehasonlítást végez.

Kiskérdések ZH utáni hétre

- Kruskal algoritmus
- Prím algoritmus
- Ford Fulkerson algoritmus
- Stabil párosítási probléma és lánykérő algoritmus

Szorgalmi feladat

Adjunk meg egy eljárást, amely kiválasztja a legkisebb és a második legkisebb elemet n elemből $n-1+\lceil \log n \rceil$ elempár összehasonlításával.

Beküldés: cimreh@inf.u-szeged.hu,

Leírás +magyarázat + futási idő elemzés

- első négy megoldó 8-8 pont
- a második négy megoldó 5-5 pont

A szerzett plusszpontok a vizsga minimumkövetelményébe nem számítanak bele.