Algoritmuselmélet

Keresés, minimumkeresés, rendezés

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Keresés, ha nincs összehasonlítás

Feladat

Adott az U halmaz véges $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$ részhalmaza és $s \in U$. El akarjuk eldönteni, hogy igaz-e $s \in S$, és ha igen, akkor melyik i-re teljesül $s_i = s$.
Hány lépés kell?

Itt egy lépés: Igaz-e, hogy $s_i = s$?
Válasz: Igen vagy nem.

Legrosszabb esetben minden elemet végig kell nézni $\Longrightarrow$
n lépés kell legrosszabb esetben.

Ha bármely két elem összehasonlítható

Példák:

- $\mathbb{Z}$ az egész számok halmaza. A $<$ rendezés a nagyság szerinti rendezés.
- Az *abc* betűinek Σ halmaza; a $<$ rendezést az *abc-sorrend* adja. Az x betű kisebb, mint az y betű, ha x előbb szerepel az *abc*-sorrendben, mint y.
- A Σ betűiből alkotott szavak Σ^* halmaza a szótárszerű vagy lexikografikus rendezéssel. $\Rightarrow$ legyen $X = x_1 x_2 \cdots x_k$ és $Y = y_1 y_2 \cdots y_l$ két szó.
  Az X kisebb mint Y, ha vagy $l > k$ és $x_i = y_i$ ha minden $i = 1, 2, \ldots, k$ esetén;
  vagy pedig $x_j < y_j$ teljesül a legkisebb olyan j indexre, melyre $x_j \neq y_j$. Tehát például *kar* $<$ *karika* és *bor* $<$ *bot*.

Keresés, ha van összehasonlítás

Barkochba játék: gondolok egy számot 1 és 100 között, hány eldöntendő kérdésből lehet kitalálni?

Feladat

Adott az U halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ részhalmaza és $s \in U$.
Összehasonlításokkal akarjuk eldönteni, hogy igaz-e $s \in S$, és ha igen, akkor melyik i-re teljesül $s_i = s$.
Hány összehasonlítás kell?

Itt összehasonlítás: Mi a viszonya s-nek és s_i-nek?
Válasz: $s_i = s$ vagy $s_i < s$ vagy $s_i > s$.

Lineáris keresés

Sorban mindegyik elemmel összehasonlítjuk.
Költség a legrosszabb esetben: n, mert lehet, hogy pont az utolsó volt.

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk: $i = \lceil \frac{n+1}{2} \rceil$
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

$$|S_2| \leq \frac{|S|}{4}, |S_3| \leq \frac{|S|}{2^3}, \ldots |S_k| \leq \frac{|S|}{2^k}$$

Pl. keressük meg, benne van-e 21 az alábbi sorozatban!

$$15, 22, 25, 37, \boxed{48}, 56, 70, 82 \quad (1)$$

$$15, 22, \boxed{25}, 37, 48, 56, 70, 82 \quad (2)$$

$$15, \boxed{22}, 25, 37, 48, 56, 70, 82 \quad (3)$$

$$\boxed{15}, 22, 25, 37, 48, 56, 70, 82 \quad (4)$$

Bináris keresés

Legfeljebb hány összehasonlítás kell?

Megválaszthatjuk k-t úgy, hogy $2^k \leq n < 2^{k+1}$ teljesüljön.

Ekkor $|S_k| \leq \frac{|S|}{2^k} = \frac{n}{2^k} < 2$, mivel $|S_k|$ egész szám, ezért $|S_k| \leq 1$.
Tehát k összehasonlítás után legfeljebb egy elem jöhet még szóba.
Ezért már csak legfeljebb egy további összehasonlítás kell.

$$2^k \leq n \implies k \leq \lfloor \log_2 n \rfloor$$

$$k + 1 \leq \lfloor \log_2 n \rfloor + 1 = \lceil \log_2 (n+1) \rceil$$

Tehát az összehasonlítások száma legfeljebb: $\lceil \log_2 (n+1) \rceil$

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség (E) tud mutatni egy olyan rendezett tömböt, aminél az algoritmusnak használnia kell $\log(n+1)$ felső egészrész kérdést. Az ellenség az algoritmus kérdéseinek hatására határozza meg a tömb elemeit a következő módon:

Ha az algoritmus s_i-vel hasonlít, akkor E megnézi, hogy hány s_i-nél kisebb és hány nagyobb olyan cella van, ami még tartalmazhatja a keresett s elemet (amit még nem zárt ki az algoritmus). Ha több ilyen cella van s_i után, akkor E azt válaszolja, hogy s nagyobb, mint s_i (ekkor E kitölti az s_i előtti és az s_i celláját tetszőleges s-nél kisebb számokkal), különben meg azt válaszolja, hogy s kisebb, mint s_i (akkor kitölti az s_i utáni cellákat és s_i celláját tetszőleges s-nél nagyobb értékekkel).

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Ha E így válaszol és egy kérdés előtt volt x cella, ahol s lehetett (ennyit nem zárt még ki az algoritmus), akkor a kérdés után marad még legalább $(x-1)/2$ cella, ahol s lehet. (A „-1” az s_i miatt van, az már biztos nem jöhet szóba).

Vagyis az első kérdés után marad legalább $(n-1)/2$ lehetőség.

$\Longrightarrow$ 2 kérdés után legalább $\frac{\frac{n-1}{2}-1}{2} = \frac{n}{2^2} - \frac{1}{2} - \frac{1}{2^2}$ marad.

$\Longrightarrow$ k kérdés után is marad még $\frac{n}{2^k} - \frac{1}{2} - \cdots - \frac{1}{2^k}$ lehetőség.

Tehát teljesülnie kell $\frac{n}{2^k} - \frac{1}{2} - \cdots - \frac{1}{2^k} \leq 1$-nek.

Vagyis $n \leq 2^k + 2^{k-1} + \ldots + 1 = 2^{k+1} - 1$. $\Longrightarrow \lceil \log_2(n+1) \rceil - 1 \leq k$.

Ha még van egy lehetséges elem, akkor még +1 egy kérdés. □

Rendezés

Feladat

Adott az U rendezett halmaz véges $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$ *részhalmaza. Összehasonlításokkal rendezzük az S elemeit a rendezés szerint növekvő sorrendbe, azaz keressünk olyan* σ *permutációt, hogy* $s_{\sigma(1)} < s_{\sigma(2)} < \cdots < s_{\sigma(n)}$.

Input: tömb, láncolt lista, (vagy bármi)

Output: általában, mint az input

Lépések: elemek mozgatása, cseréje, összehasonlítása

A rendezés önmagában is előforduló feladat, de előjön, mint hasznos adatstruktúra is. Rendezett tömbben keresni (pl. telefonkönyv).

- Hány összehasonlítás kell a legrosszabb esetben?
- Hány összehasonlítás kell átlagos esetben?
- Hány csere kell a legrosszabb esetben?
- Mennyi plusz tárhely szükséges?

Buborékrendezés

Volt Prog1-ből

Input: $A[1:n]$ (rendezetlen) tömb
Ha valamely i-re $A[i] > A[i+1]$, akkor a két cella tartalmát kicseréljük.
A tömb elejéről indulva, közben cserélgetve eljutunk a tömb végéig.
Ekkor a legnagyobb elem $A[n]$-ben van. Ismételjük ezt az $A[1:n-1]$ tömbre, majd az $A[1:n-2]$ tömbre, stb.

procedure buborék
(az $A[1:n]$ tömböt növekvően (nem csökkenően) rendezi *)*
for $(j = n-1, j > 0, j := j-1)$ **do**
  for $(i = 1, i \leq j, i := i+1)$ **do**
    { ha $A[i+1] < A[i]$, akkor cseréljük ki őket.}

összehasonlítások száma: $n-1+n-2+\ldots+1 = \frac{n(n-1)}{2}$

cserék száma: $\leq \frac{n(n-1)}{2}$

Animáció
Video tánc: Buborék rendezés

Beszúrásos rendezés

Ha az $A[1:k]$ résztömb már rendezett, akkor szúrjuk be a következő elemet, $A[k+1]$-et, lineáris vagy bináris kereséssel, majd a következőt ebbe, stb.

	lineáris	bináris
összehasonlítás	$\frac{n(n-1)}{2}$	$\sum_{k=1}^{n-1} \lceil \log_2(k+1) \rceil$
mozgatás	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$

Bináris beszúrásos rendezés lépésszáma

$$K := \lceil \log_2 2 \rceil + \lceil \log_2 3 \rceil + \cdots + \lceil \log_2 n \rceil \leq n \lceil \log_2 n \rceil$$

Jobb becslés: használjuk fel, hogy $\lceil \log_2 k \rceil \leq 1 + \log_2 k$

$$K < n - 1 + \log_2 2 + \cdots + \log_2 n = n - 1 + \log_2(n!)$$

Felhasználva a Stirling formulát: $n! \sim (n/e)^n \sqrt{2\pi n}$ kapjuk, hogy

$$\log_2 n! \sim n(\log_2 n - \log_2 e) + \frac{1}{2} \log_2 n + \log_2 \sqrt{2\pi} \sim n(\log_2 n - 1,442)$$

Ezért $K \leq n(\log_2 n - 0,442)$ elég nagy n-re.
Animáció
Video tánc: Beszúrásos rendezés

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha barochba-ban kellene kitalálni, hogy az elemek melyik sorrendje (permutációja) az igazi sorrend.

Kezdetben $n!$ lehetséges sorrend jön szóba.
Két elemet összehasonlítva a válasz két részre osztja a sorrendeket.
Ha pl. azt kapjuk, hogy $x < y$, akkor az olyan sorrendek, amelyekben x hátrébb van y-nál, már nem jönnek szóba.
Ha az ellenség megint úgy válaszol, hogy minél több sorrend maradjon meg, akkor k kérdés után még szóba jön $\frac{n!}{2^k}$ sorrend.
Ha $\frac{n!}{2^k} > 1$, nem tudjuk megadni a rendezést. $\Longrightarrow$

Tétel

Minden összehasonlítás alapú rendező módszer n elem rendezésekor legalább $\log_2(n!)$ *összehasonlítást használ.*

Összefésüléses rendezés

Volt Prog1-ből

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:
$A[1:k]$ és $B[1:l]$ rendezett tömbök $\longrightarrow$ $C[1:k+l]$ rendezett tömb
Nyilván $C[1] = \min\{A[1], B[1]\}$, pl. $A[1]$,
ezt rakjuk át C-be és töröljük A-ból.
$C[2] = \min\{A[2], B[1]\}$, stb.

Példa

A	B	C
12, 15, 20, 31	13, 16, 18	
15, 20, 31	13, 16, 18	12,
15, 20, 31	16, 18	12, 13
20, 31	16, 18	12, 13, 15
20, 31	18	12, 13, 15, 16
20, 31		12, 13, 15, 16, 18
31		12, 13, 15, 16, 18, 20
		12, 13, 15, 16, 18, 20, 31

összehasonlítások száma: $k + l - 1$, ahol k, l a két tömb hossza

Összefésüléses rendezés

Alapötlet: Rendezzük külön a tömb első felét, majd a második felét, végül fésüljük össze.
Ezt csináljuk rekurzívan.

$$\text{MSORT}(A[1:n]) := \text{MERGE}(\text{MSORT}(A[1:\lceil n/2 \rceil]), \text{MSORT}(A[\lceil n/2 \rceil + 1 : n])).$$

Hogy elvarrjuk a rekurzió alját, legyen MSORT$(A[i, i])$ az üres utasítás.

Összehasonlítások száma

Jelöljük $T(n)$-el a lépésszámot n hosszú tömb rendezésekor. Az egyszerűség kedvéért tegyük fel, hogy $n = 2^k$.

$$T(n) \leq n - 1 + 2T(n/2),$$

$$T(n) \leq n - 1 + 2(n/2 - 1 + 2T(n/4)) = n - 1 + 2(n/2 - 1) + 4T(n/4).$$

$$T(n) \leq n-1+2(n/2-1)+4(n/4-1)+\cdots+2^{k-1}(n/2^{k-1}-1) \leq n\lceil \log_2 n \rceil.$$

Felhasználva, hogy $T(1) = 0$.

Az összefésüléses rendezés konstans szorzó erejéig optimális.

Mozgatások száma: $2n\lceil \log_2 n \rceil$

Tárigény: $2n$ cella (bonyolultabban megcsinálva elég $n + konst.$)

Animáció

Video animáció: Összefésüléses rendezés

Video animáció: Összefésüléses rendezés táncban

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3		4		5		6		7		8

Gyorsrendezés

[C. A. R. Hoare, 1960]
oszd meg és uralkodj: véletlen s elem a tömbből $\longrightarrow$ PARTÍCIÓ(s) $\longrightarrow$

s-nél kisebb elemek	s	*s-nél nagyobb elemek*

GYORSREND($A[1:n]$)
1. Válasszunk egy véletlen s elemet az A tömbből.
2. PARTÍCIÓ(s); az eredmény legyen az $A[1:k-1]$, $A[k:k]$, $A[k+1:n]$ felbontás.
3. GYORSREND($A[1:k-1]$); GYORSREND($A[k+1:n]$).

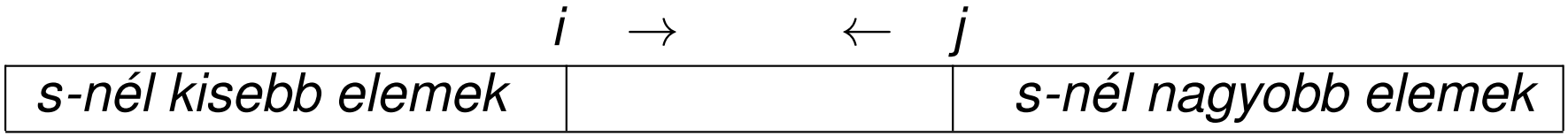
Véletlen elemnek választhatjuk mindig a tömb első helyén állót.

A PARTÍCIÓ(s) működése

Legyen $i := 2, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül, utána
- j-t csökkentjük, amíg $A[j] > s$

$\Longrightarrow$



Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] > s$ és $A[j] < s \Longrightarrow$
Kicseréljük $A[i]$ és $A[j]$ tartalmát, majd az i-t tovább növeljük, amíg $A[i] < s$ teljesül, stb. . . . . Ha a két mutató összeér (már nem teljesül $i < j$), akkor kicseréljük $A[1] = s$-et és $A[i-1]$-et
PARTÍCIÓ lépésszáma: $O(n)$
GYORSREND lépésszáma *legrosszabb esetben*: $O(n^2)$
GYORSREND lépésszáma *átlagos esetben*:
$1,39 n \log_2 n + O(n) = O(n \log n)$
Animáció

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.
Pl. ismerjük az elemek számát, belső szerkezetét.

Ládarendezés (binsort)

Legyen $U = \{u_1 < u_2 < \ldots < u_m\}$ egy m elemű halmaz, aminek elemei összehasonlíthatóak és tudjuk a nagyság szerinti sorrendjüket. A legegyszerűbb példa az $\in \{1, \ldots, m\}$ halmaz. Tudjuk, hogy $A[1 : n]$ elemei az U halmazból kerülnek ki.

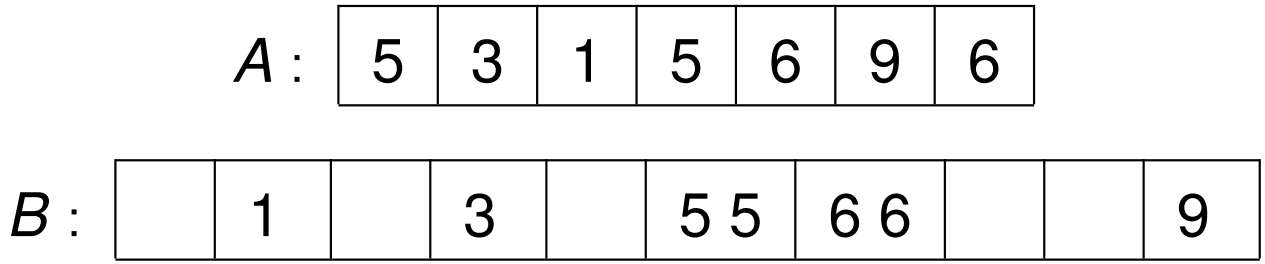
$\Longrightarrow$ Lefoglalunk egy U elemeivel indexelt B tömböt (m db ládát), először mind üres.

Első fázis: végigolvassuk az A-t, és az $s = A[i]$ elemet a $B[s]$ lista végére fűzzük.

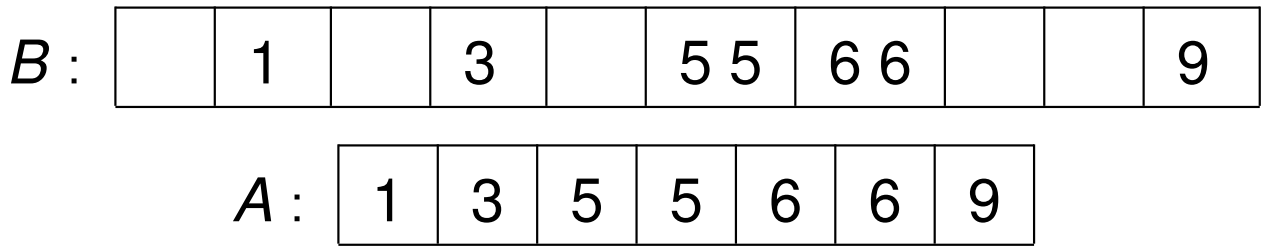
$\Longrightarrow$ konzervatív rendezés, azaz az egyenlő elemek sorrendjét megtartja.

Ládarendezés

Példa: Tegyük fel, hogy a rendezendő $A[1:7]$ tömb elemei 0 és 9 közötti egészek:



Második fázis: elejétől a végéig U elemeinek növekvő sorrendjében végigmegyünk B-n, és a $B[i]$ listák tartalmát visszaírjuk A-ba.



Lépésszám: B létrehozása $O(m)$, első fázis $O(n)$, második fázis $O(n+m)$, összesen $O(n+m)$.
Ez gyorsabb, mint az általános alsó korlát, ha pl. $m \leq cn$.
Animáció

Radix rendezés

A kulcsok összetettek, több komponensből állnak, $t_1 \dots t_k$ alakú szavak, ahol a t_i komponens az L_i rendezett típusból való, legyen $|L_i| = s_i$, a rendezés lexikografikus.
Példa: Legyen $(U, <)$ a *huszadik századi dátumok* összessége az időrendnek megfelelő rendezéssel.

$$L_1 = \{1900, 1901, \dots, 1999\}, \quad s_1 = 100.$$

$$L_2 = \{\textit{január, február,}\dots, \textit{december}\}, \quad s_2 = 12.$$

$$L_3 = \{1, 2, \dots, 31\}, \quad s_3 = 31.$$

A dátumok rendezése éppen az L_i típusokból származó lexikografikus rendezés lesz.

Radix rendezés

- Rendezzük a sorozatot az utolsó, a k-adik komponens szerint ládarendezéssel.
- A kapottat rendezzük a $k-1$-edik komponens szerint ládarendezéssel.
- stb.

Fontos, hogy a ládarendezésnél, az elemeket a ládában mindig a lista végére tettük. Így ha két azonos kulcsú elem közül az egyik megelőzi a másikat, akkor a rendezés után sem változik a sorrendjük.
$\longrightarrow$ Az ilyen rendezést konzervatív rendezésnek nevezzük.

Miért működik a radix jól?

Ha $X < Y$, az első $i-1$ tag megegyezik, de $x_i < y_i$, akkor az i-edik komponens rendezésekor X előre kerül.

A láderendezés konzervatív $\Longrightarrow$ később már nem változik a sorrendjük.

Példa:

1969.01.18.	1969.01.01.	1955.12.18.	1955.01.18.	1918.12.18.

Napok szerint rendezve:

1969.01.01.	1969.01.18.	1955.12.18.	1955.01.18.	1918.12.18.

Hónapok szerint rendezve:

1969.01.01.	1969.01.18.	1955.01.18.	1955.12.18.	1918.12.18.

Évek szerint rendezve:

1918.12.18.	1955.01.18.	1955.12.18.	1969.01.01.	1969.01.18.

Radix rendezés

Lépésszám: k ládarendezés összköltsége: $O(kn + \sum_{i=1}^{k} s_i)$

Ez lehet gyorsabb az általános korlátnál

- c, k állandók és $s_i \leq cn$
  $\Longrightarrow O(kn + \sum_{i=1}^{k} cn) = O(k(c+1)n) = O(n)$.
  pl. az $[1, n^{10} - 1]$ intervallumból való egészek rendezése
- $k = \log n$, $s_i = 2 \implies O(n \log n + 2 \log n) = O(n \log n)$.

Animáció