

Algoritmuselmélet 2. előadás

Katona Gyula Y.

Budapesti Műszaki és Gazdaságtudományi Egyetem

Számítástudományi Tsz.

I. B. 137/b

kiskat@cs.bme.hu

2002 Február 12.

Buborék-rendezés

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 a cserélgetve eljutunk a tömb végéig.

Buborék-rendezés

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 a cserélgetve eljutunk a tömb végéig. Ekkor a legnagyobb elem $A[n]$ -ben van.

Buborék-rendezés

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 a 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.

Buborék-rendezés

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 a 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 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: $n - 1 + n - 2 + \dots + 1 = \frac{n(n-1)}{2}$

cserék száma: $\frac{n(n-1)}{2}$

Buborék-rendezés

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 a 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 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: $n - 1 + n - 2 + \dots + 1 = \frac{n(n-1)}{2}$

cserék száma: $\frac{n(n-1)}{2}$

Java animáció: Buborék rendezés

Beszűrásos rendezés

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

Beszűrásos rendezés

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

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

Beszűrásos rendezés

Ha az $A[1 : k]$ résztömb már rendezett, akkor szűrjük 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	$\frac{(n + 2)(n - 1)}{2}$	$\frac{(n + 2)(n - 1)}{2}$

Beszűrásos rendezés

Ha az $A[1 : k]$ résztömb már rendezett, akkor szűrjük 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	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$
átlagos összehasonlítás	$\frac{n(n-1)}{4}$	$\sum_{k=1}^{n-1} \lceil \log_2(n+1) \rceil$

Beszűrásos rendezés

Ha az $A[1 : k]$ résztömb már rendezett, akkor szűrjük 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	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$
átlagos összehasonlítás	$\frac{n(n-1)}{4}$	$\sum_{k=1}^{n-1} \lceil \log_2(n+1) \rceil$
átlagos mozgatás	$\frac{n^2}{4}$	$\frac{n^2}{4}$

Bináris beszúrásos rendezés lépésszáma

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

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

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

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

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

Ezért $K \leq n(\log_2 n - 0,442)$ elég nagy n -re.

Bináris beszúrásos rendezés lépésszáma

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

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

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

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

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

Ezért $K \leq n(\log_2 n - 0,442)$ elég nagy n -re.

Java animáció: Beszúrásos rendezés

Alsó becslés összehasonlítás alapú rendezésre

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

Alsó becslés összehasonlítás alapú rendezésre

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

Kezdetben $n!$ lehetséges sorrend jön szóba.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha Bar Kochba-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, amikben x hátrébb van y -nál, már nem jönnek szóba.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha Bar Kochba-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, amikben 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.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha Bar Kochba-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, amikben 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. \implies

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

Összefésüléssel rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

Összefésüléssel rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

$A[1 : k]$ és $B[1 : l]$ rendezett tömbök $\implies C[1 : k + l]$ rendezett tömb

Összefésüléssel rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

$A[1 : k]$ és $B[1 : l]$ rendezett tömbök $\implies C[1 : k + l]$ rendezett tömb
Nyilván $C[1] = \min\{A[1], B[1]\}$, pl. $A[1]$,

Összefésüléssel rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

$A[1 : k]$ és $B[1 : l]$ rendezett tömbök $\implies C[1 : k + l]$ rendezett tömb

Nyilván $C[1] = \min\{A[1], B[1]\}$, pl. $A[1]$,
ezt rakjuk át C -be és töröljük A -ból.

Összefésüléssel rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

$A[1 : k]$ és $B[1 : l]$ rendezett tömbök $\implies C[1 : k + l]$ 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

<i>A</i>	<i>B</i>	<i>C</i>
12 , 15, 20, 31	13, 16, 18	

Példa

<i>A</i>	<i>B</i>	<i>C</i>
12 , 15, 20, 31	13, 16, 18	
15, 20, 31	13 , 16, 18	12,

Példa

<i>A</i>	<i>B</i>	<i>C</i>
12 , 15, 20, 31	13, 16, 18	
15, 20, 31	13 , 16, 18	12,
15 , 20, 31	16, 18	12, 13

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Példa

<i>A</i>	<i>B</i>	<i>C</i>
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

Összefésüléssel rendezés

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

Összefésüléssel 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 $\text{MSORT}(A[i, i])$ az üres utasítás.

Összehasonlítások száma

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

Összehasonlítások száma

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

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

Összehasonlítások száma

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

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

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

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Az összefésüléssel rendezés konstans szorzó erejéig optimális.

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Az összefésüléssel rendezés konstans szorzó erejéig optimális.

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

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Az összefésüléssel rendezés konstans szorzó erejéig optimális.

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

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

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Az összefésüléssel rendezés konstans szorzó erejéig optimális.

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

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

Java animáció: Összefésüléssel rendezés

Kupac adatszerkezet

Egy $(U, <)$ rendezett halmaz egy S véges részhalmazát szeretnénk tárolni, hogy a *beszúrás* és a *minimális elem törlése (mintör)* hatékony legyen.

Kupac adatszerkezet

Egy $(U, <)$ rendezett halmaz egy S véges részhalmazát szeretnénk tárolni, hogy a *beszúrás* és a *minimális elem törlése (mintör)* hatékony legyen.

Alkalmazások:

- Jobok indítása

Kupac adatszerkezet

Egy $(U, <)$ rendezett halmaz egy S véges részalmazát szeretnénk tárolni, hogy a *beszúrás* és a *minimális elem törlése (mintör)* hatékony legyen.

Alkalmazások:

- Jobok indítása
- Több rendezett halmaz összefésülése

Kupac adatszerkezet

Egy $(U, <)$ rendezett halmaz egy S véges részhalmazát szeretnénk tárolni, hogy a *beszűrés* és a *minimális elem törlése (mintör)* hatékony legyen.

Alkalmazások:

- Jobok indítása
- Több rendezett halmaz összefésülése
- Gyors rendezési algoritmus

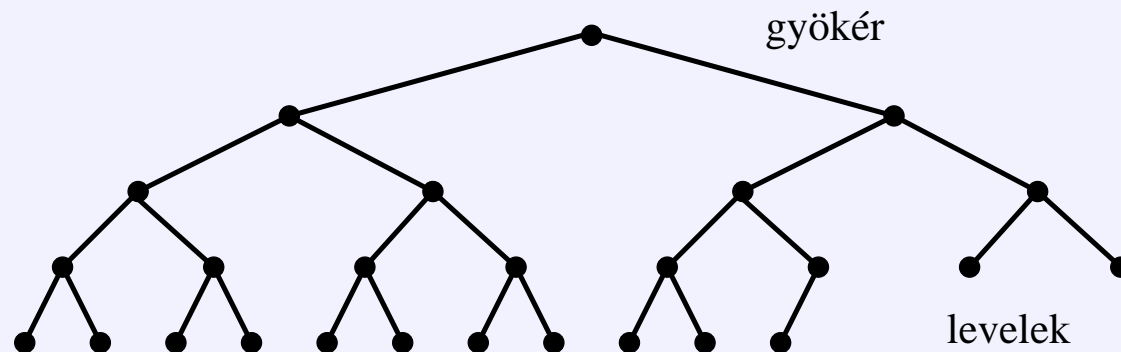
Kupac adatszerkezet

Egy $(U, <)$ rendezett halmaz egy S véges részhalmazát szeretnénk tárolni, hogy a *beszúrás* és a *minimális elem törlése (mintör)* hatékony legyen.

Alkalmazások:

- Jobok indítása
- Több rendezett halmaz összefésülése
- Gyors rendezési algoritmus

Teljes bináris fa:



Bináris fa ábrázolása tömbbel

A fa csúcsai az $A[1 : n]$ tömb elemei.

Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i + 1]$.

Bináris fa ábrázolása tömbbel

A fa csúcsai az $A[1 : n]$ tömb elemei.

Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i + 1]$.

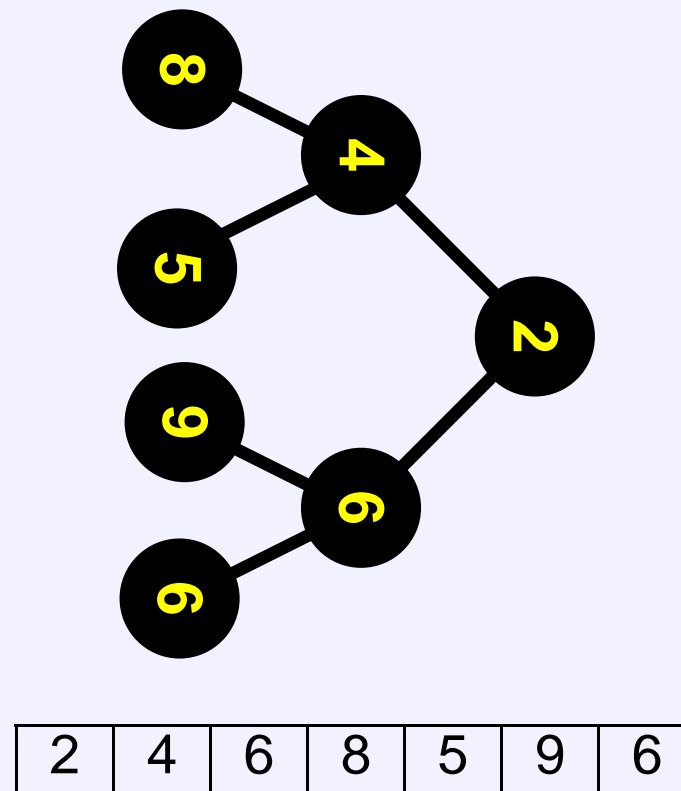
$\implies A[j]$ csúcs apja $A[\lfloor j/2 \rfloor]$

Bináris fa ábrázolása tömbbel

A fa csúcsai az $A[1 : n]$ tömb elemei.

Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i + 1]$.

$\implies A[j]$ csúcs apja $A[\lfloor j/2 \rfloor]$

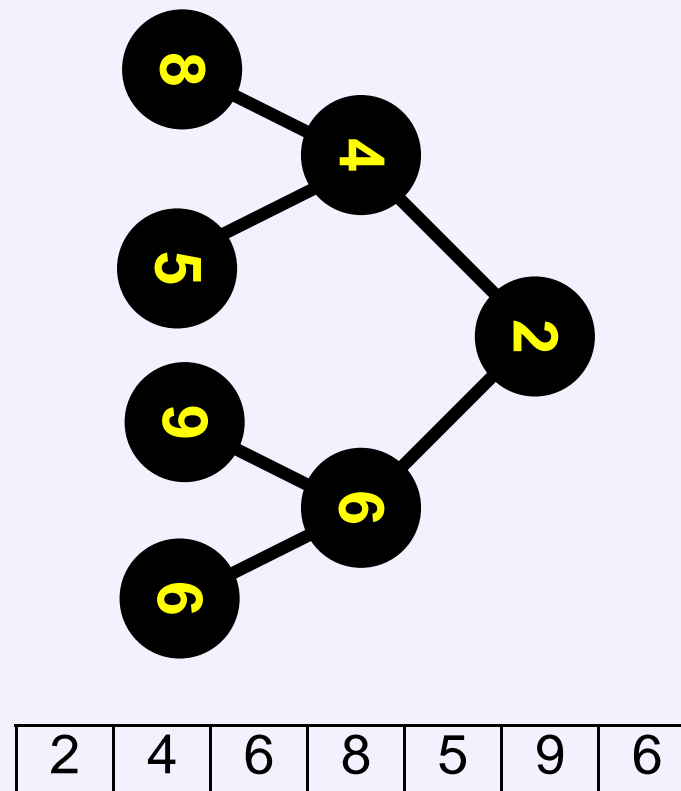


Bináris fa ábrázolása tömbbel

A fa csúcsai az $A[1 : n]$ tömb elemei.

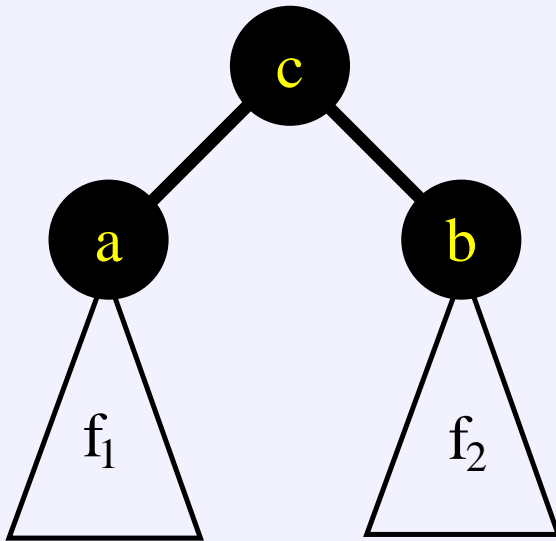
Az $A[i]$ csúcs bal fia $A[2i]$, a jobb fia pedig $A[2i + 1]$.

$\implies A[j]$ csúcs apja $A[\lfloor j/2 \rfloor]$



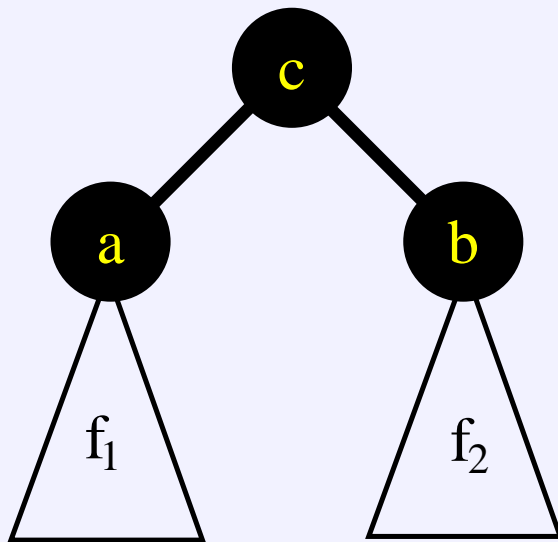
Kupac tulajdonság: $apa < fia$

Kupacépítés



f_1 és f_2 kupacok

Kupacépítés



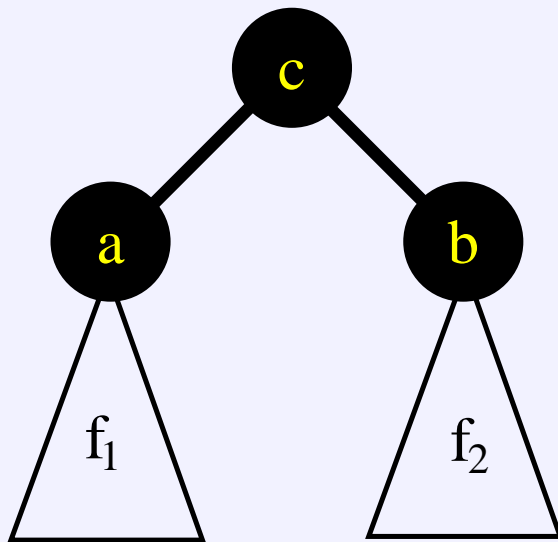
f_1 és f_2 kupacok

feszítésgörge(f)

{ Ha $\min\{a, b\} < c$, akkor $\min\{a, b\}$ és c helyet cserél

Ha a c elem a -val cserélt helyet, akkor *feszítésgörge*(f_1), ha b -vel, akkor *feszítésgörge*(f_2) }

Kupacépítés



f_1 és f_2 kupacok

felszivárog(f)

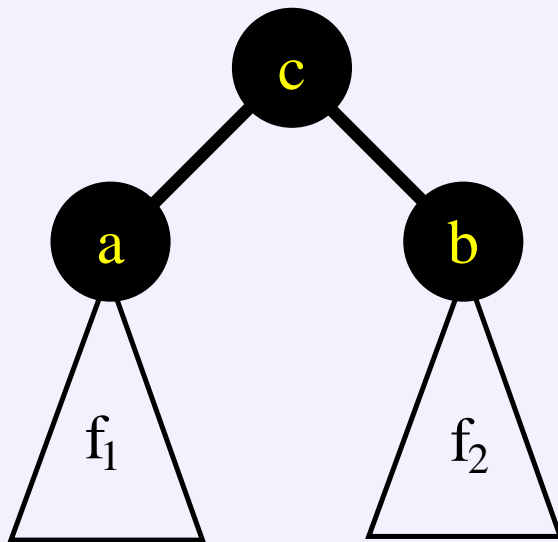
{ Ha $\min\{a, b\} < c$, akkor $\min\{a, b\}$ és c helyet cserél

Ha a c elem a -val cserélt helyet, akkor *felszivárog*(f_1), ha b -vel, akkor *felszivárog*(f_2) }

c addig megy lefelé, amíg sérti a kupac tulajdonságot.

Lépésszám: Ha l a fa szintjeinek száma, akkor $\leq l - 1$ csere és $\leq 2(l - 1)$ összehasonlítás

Kupacépítés



f_1 és f_2 kupacok

felszivárog(f)

{ Ha $\min\{a, b\} < c$, akkor $\min\{a, b\}$ és c helyet cserél

Ha a c elem a -val cserélt helyet, akkor *felszivárog*(f_1), ha b -vel, akkor *felszivárog*(f_2) }

c addig megy lefelé, amíg sérti a kupac tulajdonságot.

Lépésszám: Ha l a fa szintjeinek száma, akkor $\leq l - 1$ csere és $\leq 2(l - 1)$ összehasonlítás

kupacépítés(f)

{ Az f fa v csúcsaira letről felfelé, jobbról balra *felszivárog*(v). }

Kupacépítés költsége

Bináris fában:

1. szint: 1 pont

2. szint: 2 pont

3. szint: 2^2 pont

⋮

l -edik szint: > 1 és $\leq 2^{l-1}$ pont

Kupacépítés költsége

Bináris fában:

1. szint: 1 pont

2. szint: 2 pont

3. szint: 2^2 pont

⋮

l -edik szint: > 1 és $\leq 2^{l-1}$ pont

$$\implies n \geq 1 + \sum_{i=0}^{l-2} 2^i = 2^{l-1} \implies l \geq 1 + \log_2 n$$

Kupacépítés költsége

Bináris fában:

1. szint: 1 pont

2. szint: 2 pont

3. szint: 2^2 pont

⋮

l -edik szint: > 1 és $\leq 2^{l-1}$ pont

$$\implies n \geq 1 + \sum_{i=0}^{l-2} 2^i = 2^{l-1} \implies l \geq 1 + \log_2 n$$

Az i -edik szinten levő v csúcsra $felszivárog(v)$ költsége legfeljebb $l - i$ csere és legfeljebb $2(l - i)$ összehasonlítás.

Kupacépítés költsége

Bináris fában:

1. szint: 1 pont

2. szint: 2 pont

3. szint: 2^2 pont

⋮

l -edik szint: > 1 és $\leq 2^{l-1}$ pont

$$\implies n \geq 1 + \sum_{i=0}^{l-2} 2^i = 2^{l-1} \implies l \geq 1 + \log_2 n$$

Az i -edik szinten levő v csúcsra $felszivárog(v)$ költsége legfeljebb $l - i$ csere és legfeljebb $2(l - i)$ összehasonlítás.

A cserék száma ezért összesen legfeljebb $\sum_{i=1}^l (l - i)2^{i-1}$.
 $j = l - i$ (azaz $i = l - j$) helyettesítéssel

A cserék száma ezért összesen legfeljebb $\sum_{i=1}^l (l - i)2^{i-1}$.
 $j = l - i$ (azaz $i = l - j$) helyettesítéssel

$$\sum_{j=0}^{l-1} j2^{l-j-1} = 2^{l-1} \sum_{j=0}^{l-1} j/2^j < 2^l \leq 2n.$$

A cserék száma ezért összesen legfeljebb $\sum_{i=1}^l (l-i)2^{i-1}$.
 $j = l - i$ (azaz $i = l - j$) helyettesítéssel

$$\sum_{j=0}^{l-1} j2^{l-j-1} = 2^{l-1} \sum_{j=0}^{l-1} j/2^j < 2^l \leq 2n.$$

$$\begin{array}{ccccccc}
 1/2 & & & & & & \\
 1/4 & 1/4 & & & & & \\
 1/8 & 1/8 & 1/8 & & & & \\
 \vdots & & & & & & \\
 \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \cdots & \frac{1}{2^{l-1}} & & \\
 \hline
 < 1 & < 1/2 & < 1/4 & \cdots & < \frac{1}{2^{l-1}} & &
 \end{array}$$

A cserék száma ezért összesen legfeljebb $\sum_{i=1}^l (l - i)2^{i-1}$.
 $j = l - i$ (azaz $i = l - j$) helyettesítéssel

$$\sum_{j=0}^{l-1} j2^{l-j-1} = 2^{l-1} \sum_{j=0}^{l-1} j/2^j < 2^l \leq 2n.$$

$$\begin{array}{ccccccc} 1/2 & & & & & & \\ 1/4 & 1/4 & & & & & \\ 1/8 & 1/8 & 1/8 & & & & \\ \vdots & & & & & & \\ \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \cdots & \frac{1}{2^{l-1}} & & \\ \hline < 1 & < 1/2 & < 1/4 & \cdots & < \frac{1}{2^{l-1}} & & \end{array}$$

Tétel. Kupacépítés költsége: $O(n)$

A cserék száma ezért összesen legfeljebb $\sum_{i=1}^l (l - i)2^{i-1}$.
 $j = l - i$ (azaz $i = l - j$) helyettesítéssel

$$\sum_{j=0}^{l-1} j2^{l-j-1} = 2^{l-1} \sum_{j=0}^{l-1} j/2^j < 2^l \leq 2n.$$

$$\begin{array}{ccccccc} 1/2 & & & & & & \\ 1/4 & 1/4 & & & & & \\ 1/8 & 1/8 & 1/8 & & & & \\ \vdots & & & & & & \\ \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \frac{1}{2^{l-1}} & \cdots & \frac{1}{2^{l-1}} & & \\ \hline < 1 & < 1/2 & < 1/4 & \cdots & < \frac{1}{2^{l-1}} & & \end{array}$$

Tétel. Kupacépítés költsége: $O(n)$

MINTÖR

A minimális elem az f gyökérben van, ezt töröljük.

MINTÖR

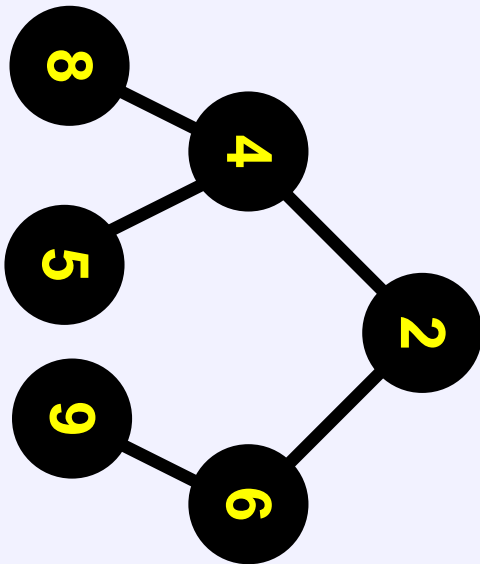
A minimális elem az f gyökérben van, ezt töröljük.

A f -be tesszük a fa utolsó szintjének jobb szélső elemét, majd *felszivárogo*(f).

MINTÖR

A minimális elem az f gyökérben van, ezt töröljük.

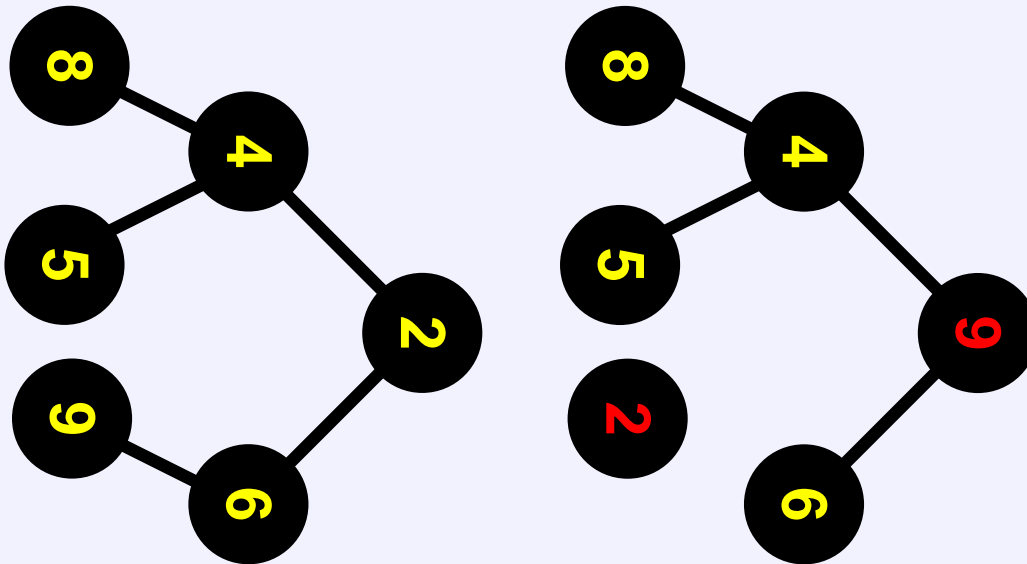
A f -be tesszük a fa utolsó szintjének jobb szélső elemét, majd $felszivárogo(f)$.



MINTÖR

A minimális elem az f gyökérben van, ezt töröljük.

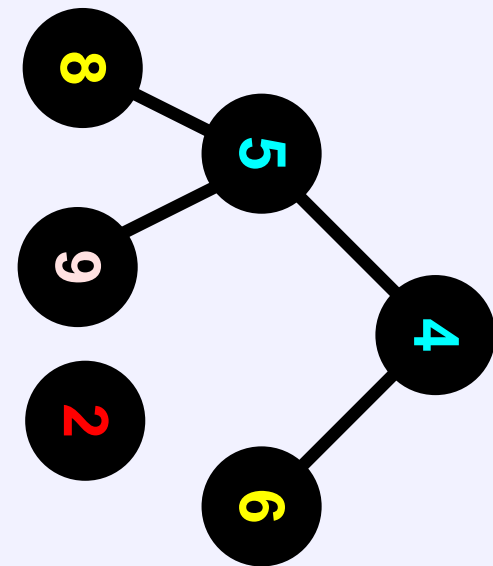
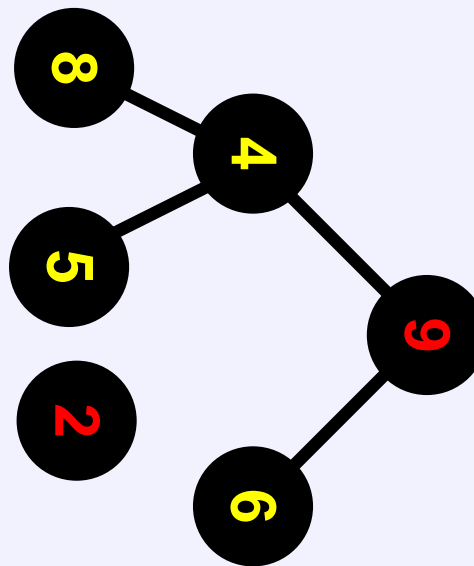
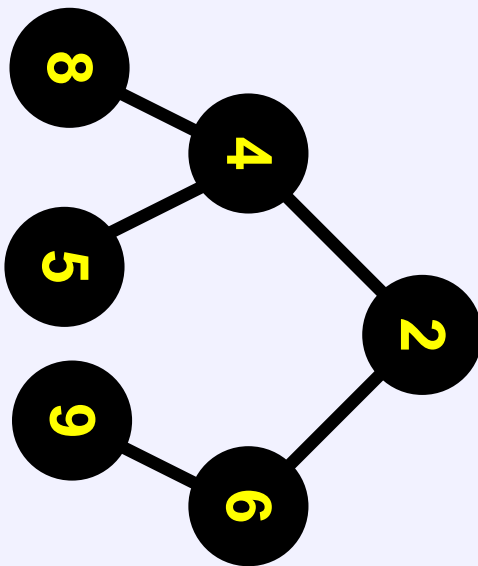
A f -be tesszük a fa utolsó szintjének jobb szélső elemét, majd $felszivárogo(f)$.



MINTÖR

A minimális elem az f gyökérben van, ezt töröljük.

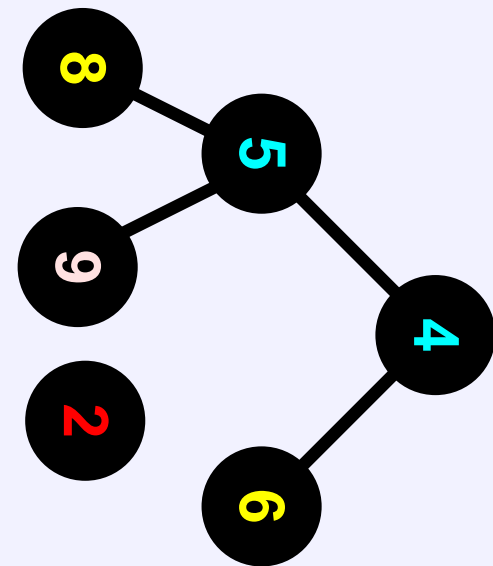
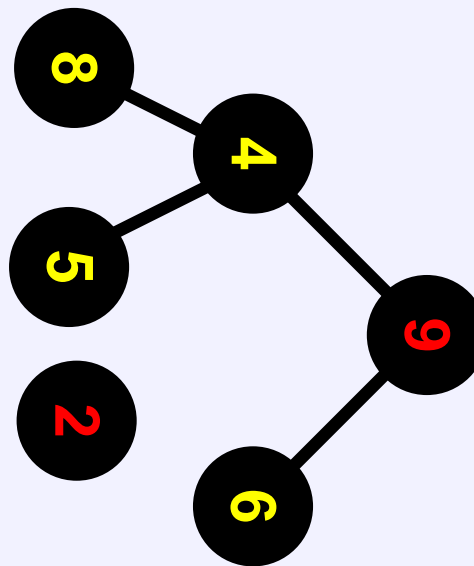
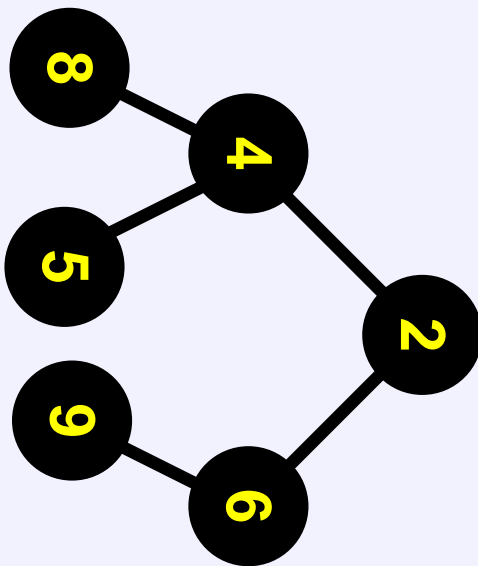
A f -be tesszük a fa utolsó szintjének jobb szélső elemét, majd $felszivárogoz(f)$.



MINTÖR

A minimális elem az f gyökérben van, ezt töröljük.

A f -be tesszük a fa utolsó szintjének jobb szélső elemét, majd *felszivároog*(f).



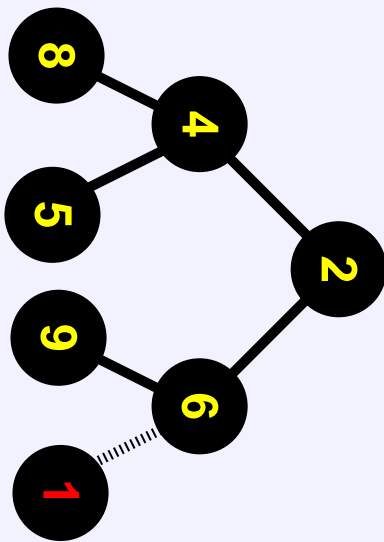
Költség: $O(l) = O(\log_2 n)$

BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s -et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.

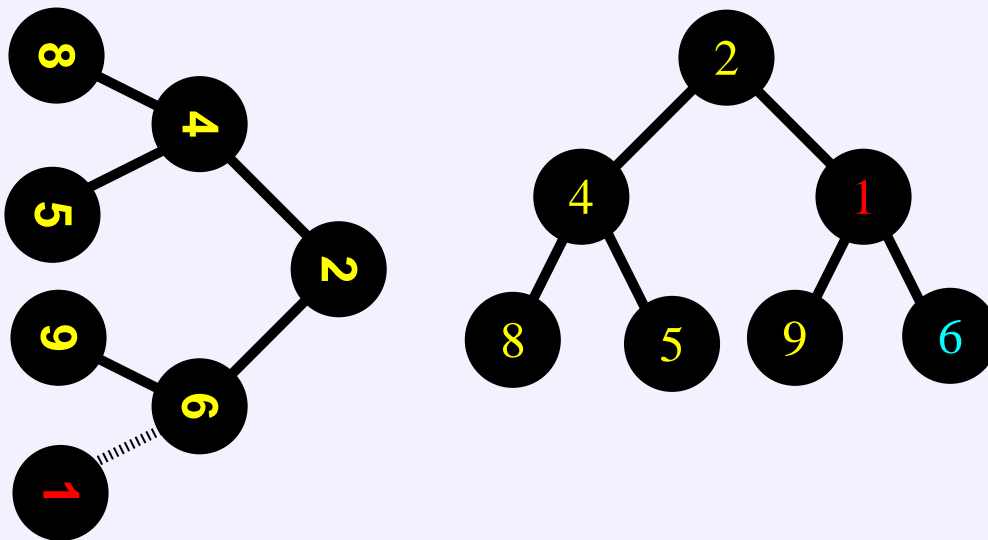
BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s -et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.



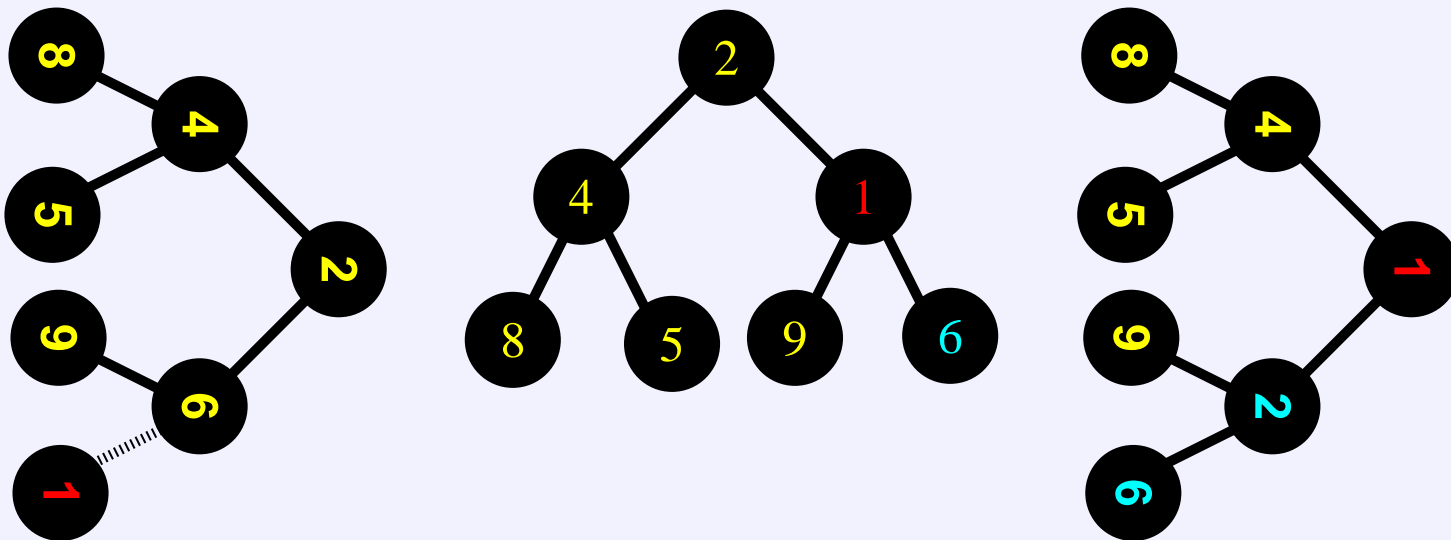
BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s -et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.



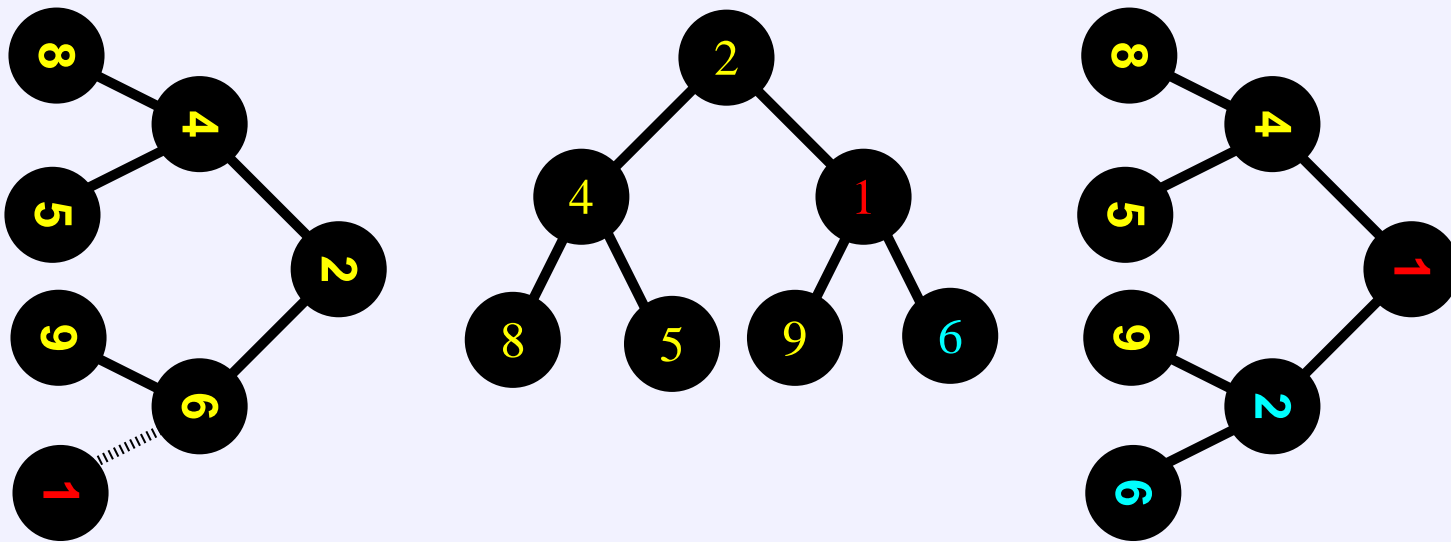
BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s -et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.



BESZÚR

Új levelet adunk a fához (ügyelve a teljességre), ide tesszük az s elemet. Ezután s -et mozgatjuk felfelé, mindig összehasonlítjuk az apjával.



Költség: $O(l) = O(\log_2 n)$

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.

[J. W. J. Williams és R. W. Floyd, 1964]

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.

[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.

[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

Legjobb ismert rendező algoritmus.

Pontos implementációval:

$2n \lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n \lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.

[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

Legjobb ismert rendező algoritmus.

Pontos implementációval:

$2n \lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n \lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).

Java animáció: Kupacos rendezés