

## Hátizsák feladat

Egy adott hátizsákba tárgyakat akarunk pakolni. Adott  $n$  tárgy minden tárgynak van egy fontossági értéke ( $f[i]$ ), és egy súlya ( $s[i]$ ), a hátizsákba maximum összesen  $S$  súlyt pakolhatunk. Az  $s[i]$  és  $S$  értékek egészek. Szeretnénk úgy választani tárgyakat, hogy az összfontosság maximális legyen. Tehát feladatunk, hogy kiválasszuk a tárgyaknak olyan halmazai közül, amelyekre az összsúly nem haladja meg  $S$ -t azt, amelyre maximális az összfontosság.

Definiáljuk az  $F(i, W)$  függvényt, minden  $i = 1, \dots, n$ ,  $W = 0, \dots, S$  értékre. Ez a függvény azon hátizsák probléma optimális függvényértékét adja meg, amelyben a tárgyak listája az első  $i$  tárgyat tartalmazza, és a hátizsák mérete  $W$ .

Ekkor a kezdeti értékekre  $F(1, W) = f[1]$ , ha  $s_1 \leq W$  és 0 különben. Másrészt a következő rekurzió teljesül:

$$F(i+1, W) = \max\{F(i, W), f[i+1] + F(i, W - s[i+1])\},$$

ha  $s[i+1] \leq W$ .

Továbbá  $F(i+1, W) = F(i, W)$ , ha  $s[i+1] > W$ ,

A rekurzió valóban fennáll. A részprobléma optimális megoldásában vagy szerepel az  $i+1$ -edik tárgy vagy nem, és ezen két eset maximuma adja az optimális célfüggvényértéket.

Hatizsak

```
for x:=0 to s[1]-1
    F[x,1]:=0
for x:=s[1] to S
    F[x,1]:=f[1]
for i:=2 to n
    for x:=0 to S
        F[x][i]:= F[x][i-1]
        if (s[i]<=x and F[x][i]<F[x-s[i]][i-1]+f[i])
            Then F[x][i]:=F[x-s[i]][i-1]+f[i]
```

KIIR

```
while (F[x][i]>0)
    while (i>=1 and F[x][i]==F[x][i-1])
        i=i-1
    print "i"
    x:=x-s[i]
    i:=i-1
```

### Példa:

A tárgyak (súly, fontosság) párokban (4,6) (3,5) (2,3) (2,3) a hátizsák kapacitása 8.

1. táblázat. A partíció algoritmus teljes táblázata

0	0	3	5	6	8	9	11	12
0	0	3	5	6	8	9	11	11
0	0	0	5	6	6	6	11	11
0	0	0	0	6	6	6	6	6

Megoldás: 4,3,1.

**Az adatkezelés szintjei**

- Probléma szintje.
- Modell szintje.
- Absztrakt adattípus szintje.
- Absztrakt adatszerkezet szintje.
- Adatszerkezet szintje.
- Gépi szint.

### Absztrakt adattípus

Az absztrakt adattípus egy  $(E, M)$  párral adható meg, ahol  $E$  az értékhalmoz,  $M$  a műveletek halmaza.  
Fő tulajdonságok

- Nem ismert az adatokat tároló adatszerkezet.
- Nem ismertek a műveleteket megvalósító algoritmusok, a műveletek specifikációjukkal definiáltak.

### Verem

Értékhalmoz:  $E$   $Verem = [a_1, \dots, a_n : a_i \in E, i = 1, \dots, n, ]$  Műveletek:  $V : Verem, x : E$

- $\{Igaz\}$  Letesit(V)  $\{V = []\}$
- $\{V = V\}$  Megszuntet(V)  $\{Igaz\}$
- $\{V = V\}$  Uresit(V)  $\{V = []\}$
- $\{V = [a_1, \dots, a_n]\}$  VeremBe(V,x)  $\{V = [a_1, \dots, a_n, x]\}$
- $\{V = [a_1, \dots, a_n], n > 0\}$  VeremBol(V,x)  $\{V = [a_1, \dots, a_{n-1}], x = a_n\}$
- $\{V = V\}$  NemUres(V)  $\{NemUres = (Pre(V) \neq [])\}$
- $\{V = [a_1, \dots, a_n], n > 0\}$  Teteje(V,x)  $\{x = a_n, V = Pre(V)\}$
- $\{V = [a_1, \dots, a_n], n > 0\}$  Torol(V)  $\{V = [a_1, \dots, a_{n-1}]\}$

Megvalósítás: tömb, lánc, kombinált.

### Sor

Értékhalmoz:  $E$   $Sor = [a_1, \dots, a_n : a_i \in E, i = 1, \dots, n, ]$  Műveletek:  $S : Sor, x : E$

- $\{Igaz\}$  Letesit(S)  $\{S = []\}$
- $\{S = S\}$  Megszuntet(S)  $\{Igaz\}$
- $\{S = S\}$  Uresit(S)  $\{S = []\}$
- $\{S = [a_1, \dots, a_n]\}$  SorBa(S,x)  $\{S = [a_1, \dots, a_n, x]\}$
- $\{S = [a_1, \dots, a_n], n > 0\}$  SorBol(S,x)  $\{S = [a_2, \dots, a_n], x = a_1\}$

- $\{S = [a_1, \dots, a_n]\}$  Elemszam(S)  $\{Elemszam = n\}$
- $\{S = [a_1, \dots, a_n], n > 0\}$  Elso(S,x)  $\{x = a_1, S = Pre(S)\}$
- $\{S = [a_1, \dots, a_n], n > 0\}$  Torol(S)  $\{S = [a_2, \dots, a_n]\}$

Megvalósítás: tömb, lánc kombinált.

### Prioritási Sor

Értékhalmoz: E, E-n értelmezett a  $\leq$  lineáris rendezési reláció  $PriSor = S \subseteq E$ . Műveletek: S : PriSor, x : E.

- $\{Igaz\}$  Letesit(S, $\leq$ )  $\{S = \emptyset\}$
- $\{S = S\}$  Megszuntet(S)  $\{Igaz\}$
- $\{S = S\}$  Uresit(S)  $\{S = \emptyset\}$
- $\{S = S\}$  SorBa(S,x)  $\{S = Pre(S) \cup \{x\}\}$
- $\{S \neq \emptyset\}$  SorBol(S,x)  $\{x = \min(Pre(S)), Pre(S) = S \cup \{x\}\}$
- $\{S = \{a_1, \dots, a_n\}\}$  Elemszam(S)  $\{Elemszam = n\}$
- $\{S \neq \emptyset\}$  Elso(S,x)  $\{x = \min(Pre(S)), Pre(S) = S\}$
- $\{S \neq \emptyset\}$  Torol(S)  $\{S = Pre(S) \setminus \{\min(Pre(S))\}\}$

Prioritási sor megvalósítások:

PriSorT: kupac-tömb adatszerkezettel,

PriSorR: kupac-tömb adatszerkezettel, a rendezési reláció konstruktor paraméter.

### Példa

**Probléma:** Halmaz k-adik legkisebb elemének kiválasztása.

**Bemenet:**  $H = \{a_1, \dots, a_n\}$ , különböző egész számok,  $1 \leq k \leq n$ .

**Kimenet:**  $a_i \in H$ , amelyre  $|\{x : x \in H, x \leq a_i\}| = k$ ,

Feltételezzük, hogy az input egy T tömbben van eltárolva. Ekkor a megoldás.

```

Valaszt
Letesit S:Prisor
for i:=1 to n
  Sorba(S,T[i])
for i=1 to k
  Sorbol(S,x)
Return x

```

### Halmaz

Értékhalmoz: E, Halmaz=  $H \subseteq E$  Műveletek: H : Halmaz, x : E, I : Iterator

- $\{Igaz\}$  Letesit(H)  $\{H = \emptyset\}$
- $\{H = H\}$  Megszuntet(H)  $\{Igaz\}$

- $\{H = H\}$  Uresit(H)  $\{H = \emptyset\}$
- $\{H = \{a_1, \dots, a_n\}\}$  Elemszam(H)  $\{Elemeszam = n\}$
- $\{H = H\}$  Eleme(H,x)  $\{Eleme = (x \in Pre(H))\}$
- $\{H = H\}$  Bovit(H,x)  $\{H = Pre(H) \cup \{x\}\}$
- $\{H = H\}$  Torol(H,x)  $\{H = Pre(H) \setminus \{x\}\}$
- $\{H = H\}$  IterKezd(H, I)  $\{\}$
- $\{I = I\}$  IterAd(I,x)  $\{\}$
- $\{I = I\}$  IterVege(I)  $\{\}$

Megvalósítás: bitvektor, tömb, lánc, hasítótábla.

### Iterátor használata

*Pascal megvalósítás:*

```
IterKezd(H, I);
While Not IterVege(I) Do Begin
    IterAd(I, x);
    M(x);
End;
```

*java megvalósítás:*

```
Iterator<E> I=H.iterator(); //IterKezd(H, I)
while (I.hasNext()){ //!IterVege(I)
    x=I.next(); //IterAd(I, x)
    M(x);
}
```

vagy kiterjesztett for-ciklussal:

```
for (E x:H)
    M(x);
```

### Példák további absztrakt adattípusokra

- Lista
- Kétirányú Lista
- Tömb
- Sorozat
- Függvény
- Reláció

## Absztrakt adatszerkezetek

Absztrakt adatszerkezet egy  $A = (M, R, Adat)$  rendezett hármas, ahol

- $M$  az absztrakt memóriahelyek, cellák halmaza.
- $R = \{r_1, \dots, r_k\}$  a cellák közötti szerkezeti kapcsolatok.
- $Adat$  a cellák adattartalmát megadó (parciális) függvény,  $c \in M$  esetén  $Adat(c)$  a  $c$  cellában lévő adat.

$f : A \rightarrow B$  parciális függvény esetén, ha  $x \in A$  elemre  $f$  nem értelmezett, akkor az  $f(x) = \perp$  jelölést alkalmazzuk, tehát mint ha  $f : A \rightarrow B \cup \{\perp\}$  mindenütt értelmezett (totális) függvény lenne.

### Lánc

$A = (M, R, Adat)$  lánc, ha  $R = \{kovet\}$ , ahol  $kovet : M \rightarrow M$  parciális függvény, amelyre teljesül:

$$(\exists fej \in M)(\forall x \in M)(\exists ! k \geq 0)(x = kovet^k(fej))$$

Nyilvánvalóan pontosan egy olyan  $c \in M$  cella van, amelyre  $kovet(c) = \perp$ , ezt láncvégnak nevezzük. A lánc hosszán a cellák  $n$  számát értjük.

Ha  $n > 0$ , akkor  $kovet^{(n-1)}(fej) = \text{láncvég}$ .

Az  $A = (M, R, Adat)$  lánc rendezett lánc a  $\leq$  relációra nézve, ha

$$(\forall x \in M)(Adat(x) \leq Adat(kovet(x)))$$

Az  $A = (M, R, Adat)$  körlánc, ha  $R = \{kovet\}$ , ahol  $kovet : M \rightarrow M$  (totális) függvény, amelyre teljesül:  $(\forall x, y \in M)(\exists k \geq 0)(y = kovet^k(x))$

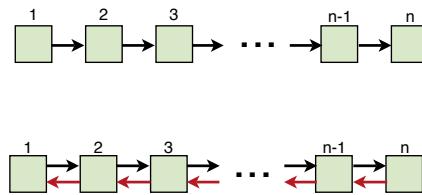
### Kétirányú lánc

Az  $A = (M, R, Adat)$  kétirányú lánc, ha  $R = \{kovet, eloz\}$ , ahol  $kovet, eloz : M \rightarrow M$  parciális függvények, hogy:  $(M, \{kovet, \}, Adat)$  és  $(M, \{eloz\}, Adat)$  mindegyike lánc, és

$$(\forall x \in M)(x = eloz(kovet(x)) = kovet(eloz(x)))$$

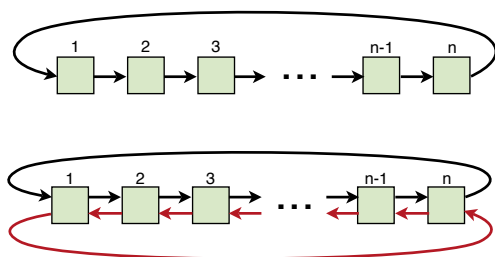
Az  $A = (M, R, Adat)$  kétirányú körlánc, ha  $R = \{kovet, eloz\}$ , ahol  $kovet, eloz : M \rightarrow M$  (teljesen értelmezett) függvények, hogy:  $(M, \{kovet, \}, Adat)$  és  $(M, \{eloz\}, Adat)$  mindegyike körlánc, és

$$(\forall x \in M)(x = eloz(kovet(x)) = kovet(eloz(x)))$$



1. ábra.

**Fa**



2. ábra.

$A = (M, R, \text{Adat})$  nem-rendezett fa, ha  $R = \{r\}$ ,  $r \in M \times M$  bináris reláció, és van olyan  $g \in M$ , hogy a  $G = (M, r)$  irányított gráfban bármely  $x \in M$ -re pontosan egy  $(g, x)$  út vezet.  $g$ -t a fa gyökerének nevezzük.

Vegyük észre, hogy ha minden él irányát megfordítjuk, akkor egy olyan  $f : M \rightarrow M$  parciális függvény gráfját kapjuk, amely csak  $g$ -re nincs értelmezve, és  $f$  körmentes.

Tehát minden nem-rendezett fa egyértelműen megadható egy olyan  $\text{Apa} : M \rightarrow M$  parciális függvénnyel, amelyre teljesül a következő feltétel.

$$(\exists g \in M)((\text{Apa}(g) = \perp) \wedge (\forall x \in M)(\exists! k \geq 0)(\text{Apa}^k(x) = g))$$

### Rendezett fa

Legyen  $A = (M, R, \text{Adat})$  olyan absztrakt adatszerkezet, hogy  $R = \{f\}$ ,  $f : M \rightarrow (M \cup \{\perp\})^*$ .

Tehát  $x \in M$ ,  $f(x) = (y_1, \dots, y_k)$ , ahol  $y_i \in (M \cup \{\perp\})$ ,  $i = 1, \dots, k$ .

Minden  $i > 0$  természetes számra és  $x \in M$ -re legyen  $f_i(x)$  az  $f(x)$   $i$ -edik komponense. Tehát  $f_i(x)$  az  $x$   $i$ -edik fiát adja. Ha  $f_i(x) = \perp$ , akkor hiányzik az  $i$ -edik fiú. Az  $A$  adatszerkezetet fának nevezzük, ha van olyan  $g \in M$ , hogy teljesül az alábbi négy feltétel:

- $(\forall x \in M)(\forall i)(g \neq f_i(x))$  a gyökér nem fia senkinek,
- $(\forall y \neq g \in M)(\exists x \in M)(\exists i)(f_i(x) = y)$  minden pont, ami nem gyökér fia valakinek
- $(\forall x, y \in M)(\forall i, j)(f_i(x) = f_j(y)) \Rightarrow (x = y \wedge i = j)$  minden pontnak legfeljebb egy apja van
- $(\forall x \neq g \in M)(\exists i_1, \dots, i_k)(x = f_{i_k} \dots f_{i_1}(g))$  minden pontba vezet út a gyökérből.

### Fa megadása elsőfiú-testvér kapcsolattal

Legyen  $A = (M, R, \text{Adat})$  olyan absztrakt adatszerkezet, hogy  $R = \{e, t\}$ ,  $e, t : M \rightarrow M \cup \{\perp\}$ . Legyen  $f_i(x) = t^{i-1}(e(x))$   $i > 0$ . Az  $A$  adatszerkezet fa, ha az  $f_i$  függvények teljesítik a megfelelő feltételeket.

### Fák algebrai definíciója

Legyen  $E$  tetszőleges adathalmaz. Az  $E$ -feletti fák  $Fa(E)$  halmaza az a legszűkebb halmaz, amelyre teljesül az alábbi három feltétel:

- $\perp \in Fa(E)$
- $(\forall a \in E)a \in Fa(E)$

- $(\forall a \in E)(\forall t_1, \dots, t_k \in Fa(E))(a(t_1, \dots, t_k) \in Fa(E))$

## Adatszerkezetek

Egy  $A = (M, R, Adat)$  absztrakt adatszerkezet megvalósítása:

- Konkrét memória allokálás az M-beli absztrakt memória cellák számára.
- Az R szerkezeti kapcsolatok ábrázolása.
- Alapműveletek algoritmusainak megadása.

### Belső adatszerkezet

A cellákat a főtárban lefoglalt memóriamezők tárolják. Minden cellát a számára lefoglalt memóriamező kezdőcíme azonosít.

### Külső adatszerkezet

A cellákat külső tárolón (lemez) fájl tárolja. Minden cellát egy  $(F, p)$  pár azonosít, ahol F a tároló fájl azonosítója és p az F fájlban belüli rekordsorszám.

### Elosztott adatszerkezet

Az egyes cellákat különböző számítógépeken tárolhatjuk. Egy cella azonosításához egy  $(G, F, p)$  hármast kell megadni, ahol G a hálózatba kapcsolt számítógép (adott hálózati protokoll szerinti) azonosítója, F a fájl azonosítója és p a fájlban belüli rekordsorszám.

### Endogén ábrázolás

Az adatot és a szerkezeti kapcsolatot ugyanaz a cella tartalmazza.

### Exogén ábrázolás

Külön cella tartalmazza az adatot és a szerkezeti kapcsolatokat. A szerkezeti kapcsolatokat tartalmazó cellában a megfelelő adatra mutató hivatkozást tároljuk. Objektum orientált programozási nyelv esetén az adatszerkezetek ábrázolása alapvetően exogén. Például a java esetén csak akkor alkalmazható endogén ábrázolás, ha az adatok típusa elemi típus (int, long, float, double, char, boolean).

### Heterogén ábrázolás

Egyes cellák csak szerkezeti kapcsolatot tartalmaznak, mások tartalmazhatnak adatot és a szerkezeti kapcsolatokat.

### Statikus ábrázolás

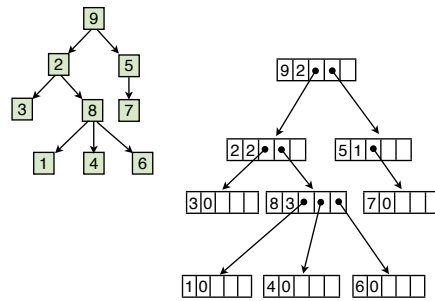
A memória lefoglalás statikus tömbbel történik. Minden cellát tömbbeli indexe azonosít. Hiányzó szomszédot a 0 (vagy -1) index azonosítja.

### Dinamikus ábrázolás

A cellák számára dinamikusan foglalunk memóriát, minden cellát pointer érték - memóriacím - azonosít. A hiányzó kapcsolatot a null (nil) pointer érték ábrázolja. A szerkezeti kapcsolatot, mint pointer értéket a cellában tároljuk.

### Szerkezeti kapcsolat számítása

A szerkezeti kapcsolat esetenként megadható számítási eljárással is, nincs szükség a szerkezeti kapcsolat tárolására. Minden x cellára és r szerkezeti kapcsolatra és adott i-re kiszámítható x-nek r-szerinti i-edik szomszédja. Például, teljes bináris fák esetén tömbben tárolva a bal és jobb fiú indexe számolható.



3. ábra.

### Kapcsolati tömb

Kapcsolati tömb esetén egy pont fiaira mutatókat tárolunk statikusan, és tároljuk a fiúk számát is.

A kapcsolati tömb előnye:

Minden pont  $i$ -edik fia közvetlenül (konstans időben) elérhető.

A kapcsolati tömb hátránya:

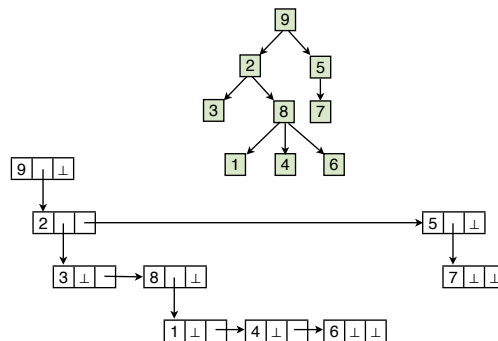
Statikus, ezért nem lehet konstans időben bővíteni és törölni. Nagy memóriaigény.

**Kapcsolati lánc** esetén egy pont fiaira mutatókat dinamikusan tároljuk láncban.

### Elsőfiú Apa testvér

Elsőfiú tesvér ábrázolás esetén a pont az adaton kívül két mutatót tartalmaz, egyet a közvetlen testvérére egyet az elsőfiára.

Elsőfiú Apa tesvér ábrázolás esetén a pont az adaton kívül három mutatót tartalmaz, egyet a közvetlen testvérére egyet az elsőfiára, egyet az apjára.



4. ábra.

### Kiskérdések

- Hátizsák feladat megoldó algoritmus, KIIR is
- Hátizsák megoldó algoritmus végrehajtása
- Lánc, körlánc, kétirányú lánc, körlánc definíciója

- rendezett fák  $f_i$  függvényes definíciója