Leghosszabb közös részsorozat

Egy sorozat, akkor részsorozata egy másiknak, ha abból elemeinek elhagyásával megkapható. A feladat két sorozat $X = (x_1, \ldots, x_m)$ és $Y = (y_1, \ldots, y_n)$ leghosszabb közös részsorozatának meghatározása.

A továbbiakban X_i az X sorozat i hosszú prefixét jelöli $X_i = (x_1, \ldots, x_i)$ és hasonlóan jelöljük a prefixeket az Y és Z sorozatokra is.

Lemma: Legyen $X = (x_1, \ldots, x_m)$ és $Y = (y_1, \ldots, y_n)$ két sorozat és $Z = (z_1, \ldots, z_k)$ ezek LKR-je. Ekkor:
- Ha $x_m = y_n$, akkor $z_k = x_m = y_n$ és Z_{k-1} az X_{m-1} és Y_{n-1} sorozatok egy LKR-je.
- Ha $x_m \neq y_n$, akkor Z az X_{m-1} és Y vagy az X és Y_{n-1} sorozatok egy LKR-je.

Megoldás dinamikus programozással:

Részprobléma: X_i és Y_j LKR-je. Az LKR hossza legyen c[i,j]. Nyilvánvalóan c[0,j]=c[i, 0]= 0.

Rekurzív összefüggés: A lemma alapján

$$c[i,j] = \begin{cases} 0, & \text{ha } i = 0 \text{ vagy } j = 0, \\ c[i-1,j-1]+1, & \text{ha } x_i = y_j, \\ \max\{c[i-1,j], c[i,j-1]\} & \text{egyébként,} \end{cases}$$

Táblázatkitöltés: c[i,j]-hez használjuk az c[i,j-1] és c[i-1,j] értékeket, ezeknek kell meglenni a c[i,j] érték számításánál. Így a helyes kitöltési sorrend soronként minden sorban a nagyobb j érték felé.

A megoldás meghatározását feljegyzéses módszerrel oldjuk meg, S[i,j]-ben feljegyezzük, hogy mi volt az optimális döntés c[i,j] számításakor.

```
LKR
for i:=0 to m c[i, 0]:=0
for j:=1 to n  c[0,j]:= 0
for i=:1 to m
    {for j:=1 to n
        {if x[i]=y[j]
           then {c[i,j]:=c[i-1,j-1]+1
                 S[i,j]:=2}
        else if c[i-1,j]>= c[i,j-1]
            then {c[i,j]:=c[i-1,j]
                  S[i,j]:=1}
        else   {c[i,j]:=c[i,j-1]
               S[i,j]:= 0}}}
```

Megoldás meghatározása

Ez a szakasz kitölti a c és S táblázatokat, a kiíratás S alapján egy rekurzív algoritmussal megtehető.

```
KIIR(i,j)
if i=0 or j=0 then return
if S[i,j]=2
    then {KIIR(i-1,j-1)
          Print "x[i]"}
else if  S[i,j]=1 then KIIR(i-1,j)
else KIIR(i,j-1)
```

1. táblázat. Az c[i,j] értékek táblázata

1	2	3	4	5	5	6	6	6
1	2	3	4	4	5	5	5	6
1	2	3	4	4	4	5	5	5
1	2	3	3	3	4	4	4	5
1	2	2	3	3	3	4	4	4
1	1	2	2	2	3	3	3	3
1	1	2	2	2	2	2	2	2
0	1	1	1	1	1	1	1	1

Példa Határozzuk meg az (a,b,b,a,b,a,b,a) és (b,a,b,a,a,b,a,a,b) sorozatok leghosszabb közös részsorozatát!

Tehát az LKR hossza 6. Az LKR-t megkapjuk, ha felírjuk az S táblázatot, vagy visszafejtéssel, ahol az átlós érték növekszik, ott van közös betű. Az i-edik sor j-edik elemének, az X i-edik és az Y j-edik betűje felel meg. Következésképp egy LKR (b,b,a,a,b,a).

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 $i = 1,\ldots,n$, $W = 0,\ldots,S$ 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 $s_1 \le W$ 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 $s[i+1] \le W$.

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.

2. táblázat. A hátizsák megoldó 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 értékhalmaz, M a műveletek halmaza.
Fő tulajdonságok

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

Verem

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

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

Megvalósítás: tömb, lánc, kombinált.

Sor

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

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

Megvalósítás: tömb, lánc kombinált.

Prioritási Sor

Értékhalmaz: 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) $\{x = \min(Pre(S)), Pre(S) = S \cup \{x\}\}$
- $\{S = \{a_1,\ldots,a_n\}\}$ 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: $H = \{a_1,\ldots,a_n\}$, 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

Értékhalmaz: 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) $\{H = \emptyset\}$
- $\{H = \{a_1, ..., a_n\}\}$ Elemszam(H) $\{Elemszam = n\}$
- $\{H = H\}$ 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.
- $R = \{r_1, \ldots, r_k\}$ a cellák közötti szerkezeti kapcsolatok.
- Adat a cellák adattartalmát megadó (parciális) függvény, $c \in M$ esetén Adat(c) a c cellában lévő adat.

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

Lánc

A = (M,R,Adat) lánc, ha $R = \{\text{kovet}\}$, ahol kovet $M \to M$ parciális függvény, amelyre teljesül:
$(\exists \text{fej} \in M)(\forall x \in M)(\exists! k \geq 0)(x = kovet^k(fej))$

Nyilvánvalóan pontosan egy olyan $c \in M$ cella van, amelyre $kovet(c) = \perp$, ezt láncvégnek nevezzük. A lánc hosszán a cellák n számát értjük.
Ha $n > 0$, akkor $kovet^{(n-1)}(fej) =$láncvég.

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

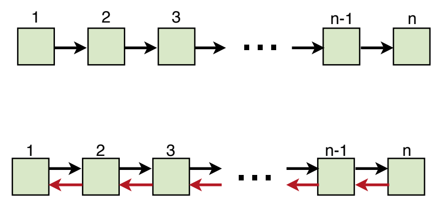
Az A = (M,R,Adat) körlánc, ha $R = \{kovet\}$, ahol kovet $M \to M$ (totális) függvény, amelyre teljesül: $(\forall x, y \in M)(\exists k \geq 0)(y = kovet^k(x))$

Kétirányú lánc

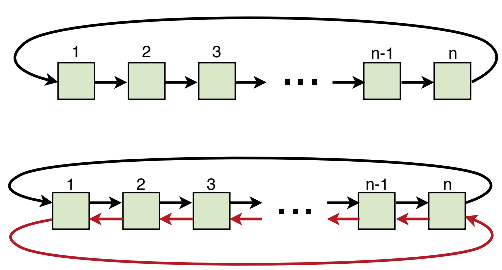
Az A = (M,R,Adat) kétirányú lánc, ha $R = \{kovet, eloz\}$, ahol kovet, eloz $M \to M$ parciális függvények, hogy: $(M, \{kovet,\}, Adat)$ és $(M, \{eloz\}, Adat)$ mindegyike lánc, és
$(\forall x \in M)(x = eloz(kovet(x)) = kovet(eloz(x)))$

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

Fa



1. ábra.



2. ábra.

A = (M,R,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 $f : M \to M$ parciális függvény gráfját kapjuk, amely csak g-re nincs értelmezve, és f körmentes.

Tehát minden nem-rendezett fa egyértelműen megadható egy olyan $Apa : M \to M$ parciális függvénnyel, amelyre teljesül a következő feltétel.
$(\exists g \in M)((Apa(g) = \perp) \wedge (\forall x \in M)(\exists! k \geq 0)(Apa^k(x) = g))$

Rendezett fa

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

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

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
- $(\forall x \neq g \in M)(\exists i_1, \ldots, i_k)(x = f_{i_k} \ldots f_{i_1}(g))$ minden pontba vezet út a gyökérből.

Fa megadása elsőfiú-testvér kapcsolattal

Legyen A = (M,R,Adat) olyan absztrakt adatszerkezet, hogy $R = \{e, t\}$, $e, t : M \to M \cup \{\perp\})$. 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:

- $\perp \in Fa(E)$
- $(\forall a \in E) a \in Fa(E)$
- $(\forall a \in E)(\forall t_1, \ldots, t_k \in Fa(E))(a(t_1, ..., t_k) \in Fa(E))$

Kiskérdések

- Leghosszabb Közös Részsorozat (KIIR is)
- Hátizsák feladat (KIIR is)
- Hátizsák feladat táblázatkitöltés végrehajtása adott inputra
- Lánc, körlánc, kétirányú lánc, körlánc definíciója
- fa algebrai definíciója