Algoritmizálás

Horváth Gyula
Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
horvath@inf.u-szeged.hu

5. Mohó algoritmusok

A mohó stratégia elemi

1. Fogalmazzuk meg az optimalizációs feladatot úgy, hogy választások sorozatával építjük fel a megoldást.
2. Mutassuk meg, hogy mindig van olyan megoldása az eredeti feladatnak, amely a mohó választással kezdődik. Ezt mohó választási tulajdonságnak nevezzük.
3. Bizonyítsuk be, hogy a mohó választással olyan redukált problémát kapunk, amelynek optimális megoldásához hozzávéve a mohó választást, az eredeti probléma megoldását kapjuk. Ezt optimális részprobléma tulajdonságnak nevezzük.

Általában sokféle mohó választás kínálkozik, de nem mindegyik mohó választás eredményez optimális megoldást, ezért fontos, hogy bebizonyítsuk, hogy az adott mohó választás tényleg optimumot eredményez.

5.1. Feladat: Fénykép probléma

Egy rendezvényre n vendéget hívtak meg. Minden vendég előre jelezte, hogy mettől meddig lesz jelen. A szervezők fényképeken akarják megörökíteni a rendezvényen résztvevőket. Azt tervezik, hogy kiválasztanak k időpontot és minden kiválasztott időpontban az akkor éppen jelenlevőkről csoportképet készítenek. Az a céljuk, hogy a lehető legkevesebb képet kelljen készíteni, de mindenki rajta legyen legalább egy képen.
Írjunk olyan programot, amely kiszámítja, hogy legkevesebb hány fényképet kell készíteni, és megadja azokat az időpontokat is amikor csoportképet kell készíteni!

Bemenet

A `fenykep.be` szöveges állomány első sorában a vendégek n száma van $(1 \leq n \leq 3000)$. A következő n sor mindegyike két egész számot tartalmaz egy szóközzel elválasztva, egy vendég e érkezési és t távozási időpontját $(1 \leq e < t \leq 1000)$. Ha egy fényképet az x időpontban készítik és $e \leq x < t$, akkor azon a fényképen rajta lesz az e időben érkező és t időben távozó vendég.

Kimenet

A `fenykep.ki` szöveges állomány első sorába a készítendő fényképek k számát kell írni! A második sor pontosan k egész számot tartalmazzon egy-egy szóközzel elválasztva, azon időpontokat (tetszőleges sorrendben), amikor a csoportképeket készíteni kell.

Példa bemenet és kimenet

bemenet

```
6
2 4
1 4
2 7
7 13
5 10
3 9
```

kimenet

```
2
3 9
```

Megoldás

Tehát a bemenet intervallumok egy

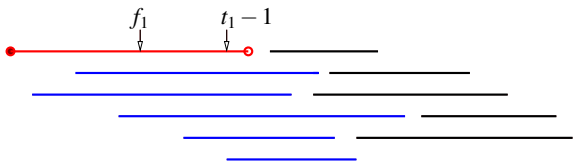
$$I = \{[e_1,t_1),\ldots,[e_n,t_n)\}$$

halmaza, a kimenet pedig olyan minimális elemszámú

$$M = \{f_1,\ldots,f_k\}$$

számhalmaz, hogy minden i-re $i = 1,\ldots,n$ van olyan $f \in M$, hogy $e_i \le f < t_i$.
Vegyük észre, hogy ha két intervallum jobb-végpontja megegyezik, $t_i = t_j$ akkor amelyik bal-végpontja kisebb,$e_i < e_j$ az elhagyható, hiszen ha $f \in [e_j,t_j)$, akkor $f \in [e_i,t_i)$.
A megoldás elemzése.
Tegyük fel, hogy az intervallumok jobb-végpontjuk szerint növekvően rendezettek, tehát $t_i < t_{i+1}$, $i = 1,\ldots,n-1$ és az M megoldáshalmazra $f_1 < \ldots < f_k$.

Mohó választás.
Válasszuk a megoldáshalmaz első elemének $t_1 - 1$-et.
Megmutatjuk, hogy az optimális megoldásban f_1 helyett állhat a mohó választás, tehát $t_1 - 1$. Először is $f_1 < t_1$, mert különben az 1. intervallumba nem esne egy pontja sem az optimális megoldásnak. Továbbá, minden olyan intervallum, amelyben benne van f_1, benne van $t_1 - 1$ is, hiszen ha $e_i \le f_1 < t_i$.
Redukált részprobléma.



1. ábra. Mohó választás és probléma redukálás.

Töröljünk I-ből mindan olyan intervallumot, amelyben benne van a $t_1 - 1$ mohó választás: $I' = I - \{[e_i,t_i) : e_i < t_1\}$. Az $M' = \{f_2,\ldots,f_k\}$ ponthalmaz megoldása lesz az I' problémának. I' optimális is, mert ha lenne kevesebb pontot tartalmazó megoldása I'-nek, akkor hozzávéve $t_1 - 1$-et, vagy f_1-et, a kiindulási I probléma kisebb elemszámú megoldását kapnánk, mint $|M|$.
Megvalósítás

```
1  Program Fenykep;
2  (Λ Bemenet : Intervallumok {[e1,t1],...,[en,tn]} halmaza.
3     Kimenet: Legkevesebb elemszámú olyan M halmaz, hogy minden
4              intervallumba esik M-nek legalább egy eleme. Λ)
5  const
6    MaxN=3000;                    { az intervallumok max. száma  }
7    MinE=1;
8    MaxT=1000;
9  var
10   N     :Word;                  { az intervallumok száma }
11   Int   :array[1..MaxT] of Word;{ az intervallumok: [Int[t],t), ha Int[t]>0 }
12   k:Word;                       { a megoldás elemszáma}
13   M:array[1..MaxN] of Word;     { a megoldás halmaz}
14   i,x:Integer;
15   Utolso:Integer;

16 Procedure Beolvas;
17 {Global:N, Int}
18   var
19     Bef:Text;
20     i,e,t:Word;
```

```
21    begin
22      for i:=1 to MaxT do Int[i]:=0;
23      Assign(Bef,'fenykep.be'); Reset(Bef);
24      ReadLn(Bef,N);
25      for i:=1 to N do begin
26        ReadLn(Bef,e,t);
27        if e>Int[t] then Int[t]:=e;
28      end;
29      Close(Bef);
30    end{Beolvas};

31  Procedure KiIr;
32  {Global: k, M }
33    var
34      Kif:Text;
35      i:Word;
36    begin
37      Assign(KiF,'fenykep.ki'); Rewrite(KiF);
38      WriteLn(Kif,K);
39
40      for i:=1 to k do
41        Write(Kif,M[i],'␣');
42
43      WriteLn(Kif);
44
45      Close(KiF);
46    end{KiIr};

47  begin{Program}
48    Beolvas;
49
50    Utolso:=0;{az utolsó beválasztott pont}
51    k:=0;      {a bevalásztott pontok száma}
52    for x:=1 to MaxT do
53      if (Int[x]>0)and(Utolso<Int[x]) then begin
54        Utolso:=x-1;
55        Inc(k);
56        M[k]:=Utolso;
57      end;
58    {for i};
59
60    KiIr;
61  end.
```

5.2. Feladat: Egységnyi végrehajtású munkák ütemezése

Mekk Elek ezermester népszerű vállalkozó, sokan keresik fel megrendelésekkel. Minden munkája pontosan egy napig tart és egyszerre csak egy munkán tud dolgozni. Minden megrendelés határidős, és amit elvállal, azt határidőre el kell végeznie. Minden elvégzett munka után meghatározott haszon jár. A mester a következő évre beérkezett megrendelések közül ki akar választani egy olyan részhalmazt, amely a lehető legtöbb hasznot eredményezi.
Készítsünk programot a következő évi megrendelések egy lehető legnagyobb elemszámú részhalmazának a kiválasztására és ütemezésére annak érdekében, hogy a mester a kiválasztott munkákat határidőre el tudja végezni, és az összhaszon a lehető legnagyobb legyen. A programnak egy ilyen ütemezést kell eredményül adnia.

Bemenet

Az `utemez.be` állomány első sora a megrendelések n számát ($1 \leq n \leq 10000$) tartalmazza. A következő n sor mindegyikében két pozitív egész szám van egy-egy szóközzel elválasztva, az adott megrendelés h határideje ($1 \leq h \leq 365$), és a munka után járó

p haszon. Tehát az i-edik munkát az állomány $i+1$-edik sora írja le.

Kimenet

Az `utemez.ki` állomány első sorában a kiválasztott munkák k száma legyen. A következő k sor mindegyikébe két számot kell írni egy-egy szóközzel elválasztva. Az első szám a kiválasztott munka száma legyen, a második pedig annak a napnak a sorszáma, amelyiken az adott munkát el kell végezni. Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni az állományba!

Példa bemenet és kimenet

bemenet

```
6
3 7
2 4
7 2
4 6
2 4
1 3
```

kimenet

```
5
5 1
1 3
2 2
4 4
3 7
```

Megoldás

Próbáljuk meg a megoldást olyan formában megfogalmazni, hogy lépésenként választást végzünk, minden lépésben egy munkát ütemezünk be egy olyan napra, amely a választott munka határidejénél nem nagyobb. A választást befolyásolja, hogy melyek a még szabad napok és milyen munkákat nem választottunk még. Legyen S a még szabad napok halmaza és M a még beosztásra váró munkák halmaza. Tehát egy ütemezési (rész)probléma az (S,M) párral adható meg általánosan.
Megoldás elemzése.

5.1. lemma. *Legyen $M \subseteq \{1,\ldots,n\}$ a munkáknak egy részhalmaza. Az M-beli munkáknak akkor és csak akkor van határidőt nem sértő beosztása, ha M elemeinek határidő szerint nemcsökkenő felsorolása határidőt nem sértő.*

Bizonyítás. Ha az M-beli munkáknak van határidőt nem sértő beosztása, akkor van olyan is, amikor a beosztás folyamatos, tehát a beosztás megadható M elemeinek egy felsorolásával. M elemeinek egy $\langle m_1,\ldots,m_k\rangle$ felsorolása akkor határidőt nem sértő, ha minden i-re, $i=1,\ldots,k$ teljesül az $i \le h_{m_i}$ egyenlőtlenség. Ha a felsorolásra nem teljesül, hogy határidő szerint nemcsökkenő, akkor van olyan i index, hogy $h_{m_i} > h_{m_{i+1}}$. A i-edik és $i+1$-edik munkát megcserélve továbbra is határidőt nem sértő beosztást kapunk, mert $i < i+1 \le h_{m_{i+1}} < h_{m_i}$.
Fordítva nyilvánvaló. ■

Mohó választási tulajdonság.
Válasszuk a legnagyobb hasznú munkát. Ha van olyan szabad nap, amely nem nagyobb, mint a munka határideje, akkor üte-

2. ábra.

mezzük be a munkát a legnagyobb ilyen szabad napra. Ha nincs ilyen nap, akkor a munkát csak töröljük a beosztandó munkák halmazából. Bebizonyítjuk, hogy bármely (S,M) részprobléma esetén van olyan optimális megoldás, amely tartalmazza a mohó választást. Legyen

$$\{(m_1,d_1),\ldots,(m_k,d_k)\}$$

egy optimális megoldása az (S,M) részproblémának. Tehát az m_i munka a d_i napra van ütemezve, így $d_i \le H[m_i]$, és a d_i értékek páronként különbözőek. Feltehetjük, hogy a munkák hasznuk szerint nemnövekvően vannak felsorolva, azaz

$$P[m_1] \ge P[m_2] \ge \cdots \ge P[m_k].$$

Legyen m^* a mohó választás, azaz m^* a legnagyobb hasznú munka, amelyre d^* a legnagyobb olyan nap, amely még szabad és $d^* \le H[m^*]$. Tehát $P[m^*] \ge P[m_i], (i=1,\ldots,k)$. Ha az optimális megoldásban valamelyik munka a d^* napra van beosztva, azaz $d^* = d_i$, akkor m_i helyettesíthető m_*-al, úgy, hogy az összhaszon nem csökken, mivel $P[m^*] \ge P[m_i]$. Ha nem lenne egyetlen munka sem beosztva a d_* napra, akkor m_* hozzá-vételével jobb megoldást kapnánk, mint az optimális, ami ellentmondás. Tehát feltehető, hogy az optimális megoldásban m_1 a mohó választás és $d_1 = d^*$.

Optimális részproblémák tulajdonság.
Legyen (m^*, d^*) mohó választása az (S, M) részproblémának. Ekkor a mohó választás eredményeként az (S', M') részprobléma keletkezik, ahol $S' = S - \{d^*\}$, és $M' = M - \{m^*\}$. Azt kell megmutatni, hogy (S', M') egy optimális megoldásához hozzávéve a mohó választást, az eredeti (S, M) probléma egy optimális megoldását kapjuk. Az nyilvánvaló, hogy az $(m_2, d_2), \ldots, (m_k, d_k)$ beosztás egy (nem feltétlenül optimális) megoldása az (S', M') problémának. Tehát (S', M') optimális megoldásához tartozó összhaszon legalább $\sum_{i=2}^{k} P[m_i]$. Ha (S', M') optimális megoldása ennél több összhasznot eredményezne, akkor a mohó választás hozzá-vételével az eredeti (S, M) probléma jobb megoldását kapnánk, mint az optimális. Ezzel bebizonyítottuk, hogy (S', M') egy optimális megoldásához hozzávéve a mohó választást, az eredeti (S, M) probléma egy optimális megoldását kapjuk.
Megvalósítás.

```
{ Globalis programelemek az EgyUtemez eljarashoz :
  const
    MaxN = ???  ;{ a munkák max. száma }
    MaxH = ???  ;{ a max. határidő }
  type
    Index = 1..MaxN;
    Harido = array [Index] Of Integer;
    Beosztas = array [1..MaxN] Of 0..MaxH;
    Haszon   = array [Index] Of Word;
 }
procedure EgyUtemez(Const H : Harido;
                    Const P : Haszon;
                          N : Word;
                    var   K : Word;
                    var MaxP: Word;
                    var   B : Beosztas );
{Bemenet: Feltesszük, hogy a munkák a hasznuk szerint nemnövekvően
 rendezettek: P[i]>=P[i+1],i=1..N-1
 Kimenet: B[i]=j>0 esetén az i. munkát a j. napra ütemezzük,
          ha az i. munkát nem választjuk, akkor B[i]=0 }

var
  i,j : Integer;
  Szabad : array[1..MaxH] of boolean;{a még szabad napok nyilvántartására}
begin{ EgyUtemez }
 M:=0; MaxP:=0;
  for i := 1 to MaxH do
    Szabad[i]:=True;       {kezdetben minden nap szabad}
  for i:=1 to N do               {nincs beosztott munka}
    B[i]:=0;
  for i:=1 to N do begin
    j := H[i];
    while (j>0) and (not Szabad[j]) do
      Dec(j);
    if j > 0 then begin {van szabad nap i. határidejéig}
      B[i]:=j;     {beválasztjuk az utolsó szabad napra}
      Szabad[j]:=False;         {a választás bejegyzése}
      Inc(K);
      MaxP:=MaxP+P[i];
    end;{if}
  end{for i};
end{EgyUtemez};
```

Az algoritmus futási ideje a legrosszabb esetben a munkák száma szorozva a napok számával. A mohó választás nem mindig fejezhető ki olyan egyszerűen, hogy egy halmazból adott rendezés szerint a legkisebb elemet kell választani. Erre mutat példát a következő feladat és annak megoldása.

5.3. Feladat: Darabolás

Adott egy fémrúd, amelyet megadott számú és hosszúságú darabokra kell felvágni. A darabok hosszát milliméterben kifejezett értékek adják meg. Olyan vágógéppel kell a feladatot megoldani, amely egyszerre csak egy vágást tud végezni. A vágások tetszőleges sorrendben elvégezhetőek. Egy vágás költsége megegyezik annak a darabnak a hosszával, amit éppen (két darabra) vágunk. A célunk optimalizálni a műveletsor teljes költséget.
Készítsünk programot, amely kiszámítja a vágási műveletsor optimális összköltségét és megad egy olyan vágási sorrendet, amely optimális költséget eredményez.

Bemenet

A `be.txt` darabol szöveges állomány első sora egy egész számot tartalmaz, a darabok n számát $(0 < n \leq 1000)$. A második sor n darab pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, a darabok hosszát. A második sorban szereplő számok nem nagyobbak, mint 1000.

Kimenet

A `be.txt` darabol szöveges állomány első sorába egyetlen számot, a vágási műveletsor optimális összköltségét kell írni! A további $n-1$ sor mindegyikébe két egész számot kell írni, egy szóközzel elválasztva. Az első szám legyen az adott lépésben kettévágott rúd hossza, a második szám pedig az egyik keletkező darab hossza. Minden sor csak olyan hosszúságú darab kettévágását tartalmazhatja, amelyből a korábbi lépések során több keletkezett, mint az azóta elvégzett lépések által felhasználtak száma.

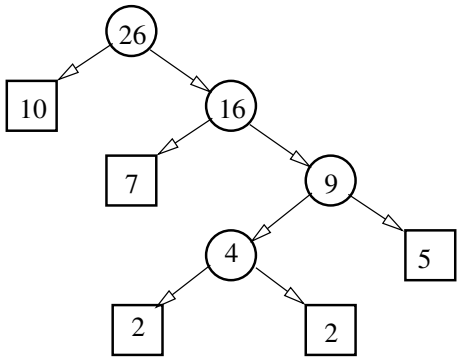
Példa bemenet és kimenet

bemenet

```
5
2 5 2 7 10
```

kimenet

```
55
26 10
16 7
9 4
4 2
```



3. ábra. A példa megoldásának ábrázolása bináris fával.

Elemezzük az optimális megoldás szerkezetét. Vegyük észre, hogy minden darabolás, így az optimális is leírható egy bináris fával. A fa levelei tartalmazzák a bemenetként kapott darabok hosszait, és minden belső pontja annak a darabnak a hosszát, amelyből vágással a két fiú-pontban lévő darab keletkezett, azaz a két fiának az összegét. Példánk esetén a fa a következőképpen néz ki.

A darabolás összköltsége is kifejezhető a fával, nevezetesen, az összköltség éppen a fa belső (nem levél) pontjaiban található számok összege. Fordítva is igaz, minden ilyen fa egy darabolást ír le. A fa költségén a fa belső pontjaiban lévő számok összegét értjük. Tehát keressük az optimális megoldást, mint egy darabolási fát, tehát azt, amelynek a költsége minimális. A darabolási fa költsége kifejezhető a következőképpen. Legyenek $d_1, \ldots, d_n$ a vágandó darabok hosszai és legyen m_i a d_i darabot tartalmazó levélpont mélysége (a fa gyökerétől vett távolsága) a fában. Ezekkel a jelölésekkel a fa költsége:

$$\sum_{i=1}^{n} m_i * d_i$$

Az optimális fára a következő két állítás teljesül.

5.2. lemma. *A két legkisebb értéket tartalmazó levélpont mélysége a legnagyobb, és testvérek.*
Bizonyítás. *Ha az állítás nem teljesülne, akkor a két legmélyebb testvér levélpontot felcserélve a két legkisebb értéket tartalmazó levéllel, kisebb költségű fát kapnánk.* ■

5.3. lemma. *Legyen d_u és d_v a két legkisebb darab. Ha az optimális fában töröljük a d_u-t és d_v-t tartalmazó levélpontot, akkor olyan fát kapunk, amely optimális arra a bemenetre, amely d_u és d_v helyett a $d_u + d_v$ darabot tartalmazza.*
Bizonyítás. *A két levél törlésével kapott fa nyilván darabolási fa lesz a módosított bemenetre, amelynek költsége*

$$\sum_{i=1}^{n} m_i * d_i - (d_u + d_v)$$

Legyen F egy optimális darabolási fa a módosított bemenetre és legyen a költsége K. Ha a $d_u + d_v$ darabot tartalmazó levélponthoz hozzávesszük bal fiúként a d_u értéket tartalmazó, jobb fiúként pedig a d_v értéket tartalmazó új levelet, akkor egy olyan fát kapunk, amely darabolási fa lesz az eredeti bemenetre, költsége pedig $K + d_u + d_v$. Ez azonban nem lehet kisebb, mint az optimális darabolási fa költsége az eredeti bemenetre, tehát

$$\sum_{i=1}^{n} m_i * d_i \leq K + (d_u + d_v)$$

$$\sum_{i=1}^{n} m_i * d_i - (d_u + d_v) \leq K \leq \sum_{i=1}^{n} m_i * d_i - (d_u + d_v)$$

Ami az állítás bizonyítását jelenti. ■

Most már megfogalmazhatjuk a mohó stratégiánkat. Építsük fel a darabolási fát úgy, hogy lépésenként a két legkisebb értéket tartalmazó pontot egy új pont két fiává tesszük, és az új pontba a két fiúban lévő érték összegét írjuk. Az 1. Állítás igazolja a mohó választási tulajdonságot, a 2. Állítás pedig az optimális részproblémák tulajdonságot, tehát korrekt algoritmust kapunk.
Megvalósítás. A mohó választás megvalósítására prioritási sort alkalmazunk.

```
1  procedure Darabol(   var D:Darabok;
2                           N:Word;
3                       var F:Fa;
4                   var Kolts:Word);
5  var x,y,z,i:Word;
6  begin{Darabol}
7    for i:=1 to N do begin                    {inicializálás}
8      SorBa(i);
9      F[i].bal:=0;F[i].jobb:=0;
10   end{for i};
11   for i:=1 to N-1 do begin{}
12     x:=SorBol;              {kivesszük a prisoritási sorból}
13     y:=SorBol;                      {a két legkisebb elemet}
14     z:=i+N;                            {új pont létesítése}
15     D[z]:=D[x]+D[y];                    {az új pont értéke}
16     SorBa(z);              {berakjuk a sorba az új fa-pontot}
17     F[z].bal:=x;                 {az új pont két fia x és y}
18     F[z].jobb:=y;
19     Kolts:=Kolts+D[z];
20   end{for i};
21 end{Darabol};

22 {A feladatban megkövetelt kimenetet a fa bejárásával állítjuk elő.}
23  procedure KiIr;
24  var  KiF:Text;
25   Procedure Bejar(p:Word);
26   begin{Bejar}
27     if F[p].bal=0 then exit;
28       WriteLn(KiF, D[p]:1,' ',D[F[p].bal]:1);
29     if F[p].bal<>0 then Bejar(F[p].bal);
```