Mohó algoritmusok

Optimalizálási probléma megoldására szolgáló algoritmus sokszor olyan lépések sorozatából áll, ahol minden lépésben adott halmazból választhatunk. Ezt gyakran dinamikus programozás alapján oldjuk meg, de előfordul, hogy a dinamikus programozás túl sok esetet vizsgál annak érdekében, hogy az optimális választást meghatározza.

A mohó algoritmus mindig az adott lépésben optimálisnak látszó választást teszi. Vagyis, a lokális optimumot választja abban a reményben, hogy ez globális optimumhoz fog majd vezetni.

Mohó algoritmus nem mindig ad optimális megoldást, azonban sok probléma megoldható mohó algoritmussal.

Eseménykiválasztási probléma

Tegyük fel, hogy adott események egy $S = \{a_1, a_2, \ldots, a_n\}$ n elemű halmaza, amelyek egy közös erőforrást, például egy előadótermet kívánnak használni. A teremben egy időben csak egy esemény folyhat. Az a_i esemény az s_i kezdő időpont és az f_i befejezési időpont által adott, ahol $s_i < f_i$. A cél egy maximális halmazát kiválasztani kompatibilis eseményeknek (i,j kompatibilis, ha $s_i \geq f_j$ vagy $s_j \geq f_i$).

Feltesszük, hogy az S eseményhalmaz elemeit a befejezési idejük szerint nemcsökkenő sorrendbe rendeztünk.

Példa:

1. táblázat. Az eseménykiválasztási probléma egy inputja

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Mohó algoritmus

```
MOHÓ-ESEMÉNY-KIVÁLASZTÓ(s,f)
n:=hossz[s]
Letesit(A:Halmaz)
Bovit(A,1)
i:=1
for m=2 to n
   {if s[m]>=f[i]
      then {Bovit(A,m)
            i:=m}}
return A
```

Példa: $A = \{1, 4, 8, 11\}$

A helyesség igazolása

1. Lemma Létezik olyan optimális megoldás, amely az 1 eseménnyel kezdődik.

Bizonyítás Legyen A egy tetszőleges optimális megoldás, legyen k a legkisebb indexű esemény. Ha $k = 1$, akkor az állítás nyilvánvaló, egyébként legyen A' az a halmaz, amit A-ból kapunk úgy, hogy k-t kicseréljük 1-re. A Lemma teljesül, mert A' is optimális.

2. Lemma Ha A tartalmazza 1-et és A optimális megoldása az S problémának, akkor $A \setminus \{1\}$ optimális megoldása az S' problémának, ahol $S' = \{i \in S : s_i \geq f_1\}$

Bizonyítás Ha nem lenne optimális megoldása S'-nek, akkor egy jobb megoldást kiegészítve 1-el, A-nál jobb megoldását kapnánk az S problémának.

A mohó megoldó stratégia elemei

1. Fogalmazzuk meg az optimalizációs feladatot úgy, hogy minden egyes választás hatására egy megoldandó részprobléma keletkezzék.

2. Bizonyítsuk be, hogy mindig van olyan optimális megoldása az eredeti problémának, amely tartalmazza a mohó választást, tehát a mohó választás mindig biztonságos.

3. Mutassuk meg, hogy a mohó választással olyan részprobléma keletkezik, amelynek egy optimális megoldásához hozzávéve a mohó választást, az eredeti probléma egy optimális megoldását kapjuk.

Kapcsolat a dinamikus programozással

A feladat megoldható lenne dinamikus programozással is.

Legyen $S_{i,j} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$, tehát $S_{i,j}$ azokat az S-beli eseményeket tartalmazza, amelyek a_i befejeződése után kezdődnek, és befejeződnek a_j kezdete előtt, azaz kompatibilisek mind a_i-vel mind pedig a_j-vel.

Tegyük fel, hogy $A_{i,j}$ egy optimális megoldása az $S_{i,j}$ részproblémának és $a_k \in A_{i,j}$. Ekkor az $A_{i,k}$ megoldás optimális megoldása kell legyen az $S_{i,k}$ részproblémának, és az $A_{k,j}$ megoldás optimális megoldása kell legyen az $S_{k,j}$ részproblémának.

Legyen $c[i,j]$ az $S_{i,j}$ részprobléma optimális megoldásában az események száma, ekkor

$c[i,j] = 0$, ha $S_{i,j} = 0$ és

$c[i,j] = \max_{i<k<j, a_k \in S_{i,j}} \{c[i,k] + c[k,j] + 1\}$ egyébként.

A rekurziónak a mohó megoldás mindig optimális megoldását adja.

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.

A töredékes változat abban különbözik az előzőtől, hogy a tárgyak töredéke is választható, nem kell 0-1 bináris választást tenni.

Megoldó algoritmusok

A töredékes hátizsák feladatot optimálisan megoldja egy mohó algoritmus. A feladat megoldásához számítsuk ki minden tárgyra a $f[i]/s[i]$ fontosság per súly hányadost. A mohó stratégia szerint mindig a legnagyobb hányadosú tárgyból választunk amennyit csak lehet. Ha elfogyott, de még nem telt meg a hátizsák, akkor a következő legnagyobb hányadosú tárgyból választunk amennyit csak lehet, és így tovább, amíg a hátizsák meg nem telik. Mivel a tárgyakat az érték per súly hányados szerint kell rendeznie, a mohó algoritmus futási ideje $O(n \log n)$ lesz.

Ez a mohó algoritmus nem ad feltétlenül optimális megoldást a 0-1 hátizsák feladatra. Ezt igazolja a következő példa: $s = [10,20,30]$, $f = [60,100,120]$, $S = 50$.

Egy ütemezési feladat

Feladat Vegyük azt az ütemezési modellt, ahol adottak az $\{1,\ldots,n\}$ munkák továbbá a j munkának van egy p_j végrehajtási ideje és egy w_j súlya. A cél a munkáknak egy olyan sorrendjét megadni, amely sorrendre a befejezési idők súlyozott összege minimális.

Példa $(p_1 = 2, w_1 = 2)$, $(p_2 = 4, w_2 = 3)$, $(p_3 = 2, w_3 = 1)$.

Ekkor az $1,2,3$ sorrendre a befejezési idők $2,6,8$ és a célfüggvény $2 \cdot 2 + 6 \cdot 3 + 8 \cdot 1 = 30$.

A $2,1,3$ sorrendre a befejezési idők $4,6,8$ és a célfüggvény $4 \cdot 3 + 6 \cdot 2 + 8 \cdot 1 = 32$.

Mohó megoldások:

- Elsőnek azt hajtsuk végre, akinek kicsi a végrehajtási ideje.
- Elsőnek azt hajtsuk végre, akinek nagy a súlya.

Egyik sem ad optimális algoritmust.

Tétel A munkákat a p_j/w_j érték szerint monoton növekvő sorrendbe rendezve egy optimális sorrendet kapunk.

Lemma Van olyan optimális megoldás, amelyben az elsőnek végrehajtott munkára minimális a p_j/w_j érték.

Bizonyítás: Vegyük azt az optimális megoldást, amelyben a legkorábban szerepel olyan munka, amelyre a p_j/w_j érték minimális. Legyen ez a megoldás OPT és tegyük fel, hogy az i-edik munka az első olyan, amire a p_j/w_j érték minimális. Ha $i = 1$, akkor a lemma teljesül.

Ellenkező esetben legyen k és r OPT i-1-edik és i-edik munkája. Vegyük azt az OPT' megoldást, amelyet úgy kapunk OPT-ból, hogy felcseréljük k és r sorrendjét. Ekkor OPT célfüggvényértékéből kivonva OPT' célfüggvényértékét a

$$w_k p_k + w_r(p_k + p_r) - w_r p_r - w_k(p_r + p_k) = w_r p_k - w_k p_r > 0$$

értéket kapjuk, ami ellentmond OPT optimalitásának.

Lemma Egy optimális megoldás tetszőleges végszelete optimális megoldása annak az ütemezési feladatnak, amelyben az input a végszeletben szereplő munkák halmaza.

Bizonyítás Ha nem lenne optimális megoldás, akkor egy jobb megoldást kiegészítve a hiányzó kezdőszelettel, a kiinduló megoldásnál jobb megoldását kapnánk az eredeti problémának.

Egy másik ütemezési feladat

Feladat Vegyük azt az ütemezési problémát, ahol egy gép van, minden munkának van egy végrehajtási ideje és egy határideje. A cél a maximális késedelem (befejezési időből kivont határidő) minimalizálása.

Mohó algoritmus: a munkákat határidő szerint rendezzük és ebben a sorrendben hajtjuk végre.

Tétel A Mohó algoritmus optimális megoldást ad.

Lemma Van olyan optimális megoldás, amelyben az elsőnek végrehajtott munkának a legkisebb a határideje.

Bizonyítás: Tekintsünk egy olyan ütemezést legyen S, amely nem egy legkisebb határidővel rendelkező munkát ütemez elsőnek. Legyen a legkisebb határidővel rendelkező munka j.

Tegyük fel, hogy S valamilyen $i_1, i_2, \ldots, i_k, j$ munkasorrenddel kezdődik. Vegyük S'-t, amelyet úgy kapunk S-ből, hogy a munkák ezen kezdeti sorrendjét kicseréljük a $j, i_1, i_2, \ldots, i_k$ sorrendre. Az új ütemezésre az $i_1, \ldots, i_k$ munkák később fejeződnek be, viszont a befejezési idejük nem lesz nagyobb, mint j-nek az S ütemezésben, ezért a késedelmük sem, hiszen j határideje minimális volt. Következésképpen az így kapott megoldás is optimális lesz, amivel a lemmát igazoltuk.

Lefedés

Feladat: Adott a valós számegyenesen valós pontoknak egy $x_1, \ldots, x_n$ halmaza. Adjunk meg hatékony algoritmust, amely megad minimális számú egységnyi hosszú intervallumot, amelyek tartalmazzák a pontokat. Igazoljuk az algoritmus helyességét!

Megoldás: Legyen a P_i részprobléma az a feladat, hogy az $x_i, \ldots, x_n$ pontokat fedjük le minimális számú egységnyi intervallummal.

A P_i részprobléma esetén a mohó választás az, hogy a lehető legnagyobb kezdőponttal rendelkező intervallumot vesszük, ami tartalmazza az x_i pontot, azaz az $[x_i, x_i + 1]$ intervallumot.

Tegyük fel, hogy ez az intervallum az $x_i, \ldots, x_{j-1}$ pontokat fedi le, és a maradék pontokra az optimális megoldást vesszük. Ekkor a P_i probléma így kapott megoldásának a költsége $1 + OPT(P_j)$.

Másrészt ez valóban $OPT(P_i)$ hiszen P_i minden megoldásában le kell fedni az x_i pontot, és bárhogy választunk lefedő intervallumot az $x_j, \ldots, x_n$ pontok szabadon maradnak.

```
M:=[x_1,x_1+1]
x:=x_1+1
for i:=2 to n do
    if x_i>x then{
       M:=M U {[x_i,x_i+1]}
       x=x_i+1
      }
```