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: $s = [10, 20, 30]$, $f = [60, 100, 120]$, $S = 50$.

Huffman kód

A Huffman-kód széles körben használt és nagyon hatékony módszer adatállományok tömörítésére. Az elérhető megtakarítás 20%-tól 90%-ig terjedhet, a tömörítendő adatállomány sajátosságainak függvényében. A kódolandó adatállományt karaktersorozatnak tekintjük. A Huffman féle mohó algoritmus egy táblázatot használ az egyes karakterek előfordulási gyakoriságára, hogy meghatározza, hogyan lehet a karaktereket optimálisan ábrázolni bináris jelsorozattal.

Kódolási módszerek

2. táblázat. Karakterek kódolása

karakter	a	b	c	d	e	f
gyakoriság	45	13	12	16	9	5
fix hosszú kód	000	001	010	011	100	101
változó hosszú kód	0	101	100	111	1101	1100

Kézenfekvő megoldás fix hosszú kódokat használni, ekkor a betűk kódolásához 3 bit kell, így a teljes kód hossza: 300.

A változó hosszú kód alkalmazása tekintélyes megtakarítást eredményez, ha gyakori karaktereknek rövid, ritkán előforduló karaktereknek hosszabb kódszavat feleltetünk meg. A fenti példában szereplő kód esetén a kódoláshoz szükséges bitek száma: $(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) = 224$.

A kódot vissza is kell tudnunk fejteni. A továbbiakban csak olyan kódszavakat tekintünk, amelyekre igaz, hogy egyik sem kezdőszelete a másiknak. Az ilyen kódolást prefix-kódnak vagy prefix mentes kódnak nevezzük.

Kódok ábrázolása

Az olyan bináris fa, amelynek levelei a kódolandó karakterek, egy alkalmas ábrázolása a prefixmentes kódoknak. Ekkor egy karakter kódját a fa gyökerétől az adott karakterig vezető út ábrázolja, a 0 azt jelenti, hogy balra megyünk, az 1 pedig, hogy jobbra megyünk az úton a fában.

Ekkor minden karakter kódjának hossza a fában a megfelelő levél mélysége, így a teljes állomány kódolásának várható hossza egy adott T fára:

$$B_T = \sum_{c \in C} f(c) d_T(c),$$

ahol C a karakterek halmaza, $f(c)$ a c karakter gyakorisága, $d_T(c)$ pedig a c-nek megfelelő levél mélysége a T fában.

Huffman kód algoritmusa

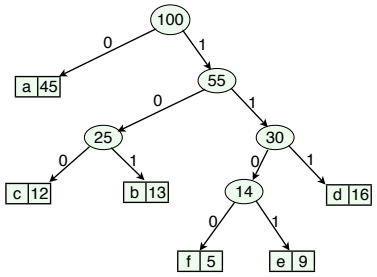
A kód fájának megszerkesztéséhez egy prioritási sort használunk, amely a fa pontjait az f érték alapján tárolja. Eredményképpen a fa gyökerét adja az algoritmus vissza.

```
HUFFMAN(C)
n:=|C|
Letesit(Q:PriSor)
for (c in C)
     Sorba(Q,c)
for i=1 to n-1
  {új z csúcs létesítése
   Sorbol(Q,x)
   bal[z]:=x
   Sorbol(Q,y)
   jobb[z]:=y
   f[z]:=f[x]+ f[y]
   Sorba(Q,z)}
Sorbol(Q,x)
Return x
```

Az algoritmus futási ideje $O(n \log n)$.

Megjegyzés: A fenti pszeudókód által kapott algoritmus nem alkalmas arra, hogy az egyes pontók kódját hatékonyan megkapjuk belőle. Egy hatékony megvalósítás során, a fa konstruálásánál, az adot pont fiainál meg kell jegyeznünk, hogy bal vagy jobb fiúk.

Példa:



1. ábra.

Helyesség

1. Lemma Legyen C tetszőleges karakter halmaz, és legyen $f[c]$ a $c \in C$ karakter gyakorisága. Legyen x és y a két legkisebb gyakoriságú karakter C -ben. Ekkor létezik olyan optimális prefix-kód, amely esetén az x-hez és y-hoz tartozó kódszó hossza megegyezik, és a két kódszó csak az utolsó bitben különbözik.

Bizonyítás: A bizonyítás alapötlete az, hogy vegyünk egy optimális prefix-kódot ábrázoló T fát és módosítsuk úgy, hogy a fában x és y a két legmélyebben lévő testvércsúcs legyen. Ha ezt meg tudjuk tenni, akkor a hozzájuk tartozó kódszavak valóban azonos hosszúságúak lesznek és csak az utolsó bitben különböznek.

Legyen a és b a T fában a két legmélyebb testvércsúcs. Az általánosság megszorítása nélkül feltehetjük, hogy $f[a] \leq f[b]$ és $f[x] \leq f[y]$. Mivel $f[x] \leq f[y]$ a két legkisebb gyakoriság, ezért azt kapjuk, hogy $f[x] \leq f[a]$ és $f[y] \leq f[b]$.

Cseréljük ki az a és x pontokat T-ben, legyen a kapott fa T'. Majd cseréljük ki a b és y pontokat T'-ben a kapott fa legyen T".

Ekkor kiszámolva a költségeket:
$B(T) - B(T') = (f[a] - f[x])(d_T(a) - d_T(x)) \geq 0$
$B(T') - B(T") = (f[b] - f[y])(d_{T'}(b) - d_{T'}(y)) \geq 0.$

Következésképpen T" is optimális.

A konstrukció teljesíti az optimális részproblémák tulajdonságot.

2. Lemma Legyen C tetszőleges ábécé, és minden $c \in C$ karakter gyakorisága $f[c]$. Legyen x és y a két legkisebb gyakoriságú karakter C-ben. Tekintsük azt a C' ábécét, amelyet C-ből úgy kapunk, hogy eltávolítjuk az x és y karaktert, majd hozzáadunk egy új z karaktert, tehát $C' = C \setminus \{x,y\} \cup \{z\}$. Az f gyakoriságok C'-re megegyeznek a C-beli gyakoriságokkal, kivéve z esetét, amelyre f[z] = f[x]+ f[y]. Legyen T' olyan fa, amely optimális prefix-kódját ábrázolja a C' ábécének. Ekkor az a T fa, amelyet úgy kapunk, hogy a z levélcsúcshoz hozzákapcsoljuk gyerek csúcsként x-et és y-t, olyan fa lesz, amely a C ábécé optimális prefix-kódját ábrázolja.

Bizonyítás: A formulák behelyettesítésével adódik, hogy $B(T) = B(T') + f[x] + f[y]$.

Ezt követően indirekt bizonyítunk. Tegyük fel, hogy T nem optimális prefix-kódfa a C ábécére. Ekkor létezik olyan T" kódfa C-re, hogy $B(T") < B(T)$. Az általánosság megszorítása nélkül (az 1. lemma alapján) feltehetjük, hogy x és y testvérek. Legyen T^* az a fa, amelyet T"-ből úgy kapunk, hogy eltávolítjuk az x és y csúcsokat, és ezek közös z szülőjének gyakorisága az $f[z] = f[x] + f[y]$ érték lesz.

Ekkor $B(T^*) = B(T") - f[x] - f[y] < B(T) - f[x] - f[y] = B(T')$, ami ellentmondás.

Kiskérdések a ZH utáni hétre

- Eseményválasztás algoritmusa
- Huffman kód algoritmusa
- Huffman kód algoritmusának végrehajtása, az optimális fa felépítése, a kódok megadása egy megadott példán