OPTIMALIZÁLÁSI MÓDSZEREK

Időbonyolultság

- Futási idő függ a gép hardverétől, csak a hardver megadása mellett értelmezhető
- Műveletigény nem függ a hardvertől
- Definíció:

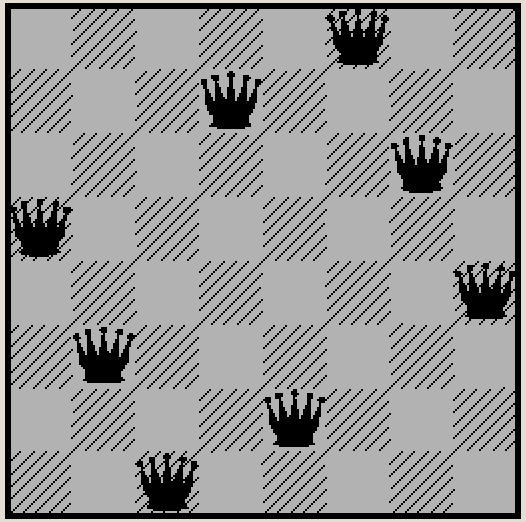
Időbonyolultság a műveletigény változásának trendje abban az esetben, ha az input mérete minden határon túl növekszik.

Időbonyolultság ismétlés

- ***N*** műveletek száma, ***n*** elemszám
- Legyenek $f, g : N \rightarrow R+$ függvények, ahol $N = \{0, 1, \ldots\}$, R+ pedig a nemnegatív valós számok halmaza.
- Azt mondjuk, hogy $f(n) = O(g(n))$, ha létezik olyan $c > 0$ és $n_0 \in N$, hogy $f(n) \leq c \cdot g(n)$ minden $n \geq n_0$ esetén. Ha $f(n) = O(g(n))$ és $g(n) = O(f(n))$ is teljesül, akkor $f(n) =$
- $\Theta(g(n))$.
- Néhány példa:
- • $5n3+6n2+1 = \Theta(n3)$
- • $nk = O(2n)$
- • $\log 2\, n = \Theta(\log 3\, n)$
- • $\log \log n = O(\log n)$

Játékprogramok

- A számítógép feladata
  - Játék szimuláció
  - Játék szereplő



Backtrack algoritmus

8 királynő problémája: Elhelyezhető-e a sakktáblán 8 királynő úgy, hogy ne üsse egyik se a másikat?

Labirintus játékok

Lefedési feladatok

Adatszerkezet

- X[8] tartalmazza minden oszlopban a királynő pozícióját.
- X[5,3,1,7,2,8,6,4]

Backtrack algoritmus- alapalgoritmus

```
Eljárás
 I:=1
 Ciklus amíg I>=1 és I<=N
   Ha VAN_JÓ_ESET(I) akkor I:=I+1 : X(I):=0
                    különben I:=I-1
   Elágazás vége
 Ciklus vége
 VAN:=I>N
Eljárás vége.
```

Backtrack algoritmus II.- részletek

```
VAN_JÓ_ESET(I):
  Ciklus
    X(I):=X(I)+1
  amíg X(I)<=M(I) és ROSSZ_ESET(I,X,(I))
  Ciklus vége
  VAN_JÓ_ESET:=X(I)<=M(I)
Eljárás vége.

ROSSZ_ESET(I,X(I)):
  J:=1
  Ciklus amíg J<I és (J,X(J)) nem zárja ki (I,X(I))-t
    J:=J+1
  Ciklus vége
  ROSSZ_ESET:=J<I
Eljárás vége.
```

Backtrack algoritmus III.-részletek

```
Függvény rossz(i):logikai
    rossz := hamis
    Ciklus j := 1-től (i-1)-ig
        Ha (T[j]=T[i]) vagy (T[j]-j=T[i]-i)
             vagy (T[j]+j=T[i]+i)
        akkor
            rossz := igaz
            kiugrás a ciklusból
        Elágazás vége
    Ciklus vége
  Függvény vége
```

Rossz- ütés:

- egymás mellett
- Azonos főátlóban
- Azonos mellékátlóban

Egy éhes egérnek egy labirintusban elhelyeznek egy darab sajtot. Írjunk programot, amely segít az egérnek megkeresni a sajthoz vezető utat!

```
Program SajtKeresés:

   …
   Hely(0):=RajtHely [ahonnan indul az egér, pl. (1,1)]
   i:=1; LépésIrányIndex(1..MaxHossz):=0

   Ciklus amíg i≥1 és Labirintus(Hely(i-1))≠Sajt
      VanJóEset(i, van, j)
      Ha van akkor   LépésIrányIndex(i):=j
                     Hely(i):=Hely(i-1)+Irány(j) [+: vektorösszeadás]
                                     i:+1
                        különben LépésIrányIndex(i):=0
                                     i:-1
      Elágazás vége
   Ciklus vége

   VanÚt:=i≥1

   Ha VanÚt akkor LépésSzám:=i-1

Eljárás vége.
```

Van-e jó eset

Eljárás VanJóEset(**Konstans** i:Egész, **Változó** van:Logikai, j:Egész):

j:=LépésIrányIndex(i)+1

Ciklus amíg j≤4 **és** (RosszEset(i,j) **vagy**
Labirintus(Hely(i-1)**+**Irány(j))=Fal
[a Labirintusmátrix Hely(i-1)**+**Irány(j)
koordinátájú pontja Fal értékű-e]

j:+1

Ciklus vége

van:=j≤4

Eljárás vége.

Rossz eset

```
Függvény RosszEset(Konstans i,j:Egész): Logikai
  k:=0
  Ciklus amíg k<i és Hely(k)≠Hely(i-1)+Irány(j)
    k:+1
  Ciklus vége
  RosszEset:=k<i
Függvény vége.
```

```
Program Lefedés:
  pillHossz:=0; H(0):=0; Szl(0):=0
  i:=1; Szl(1..N):=0
  Ciklus amíg i≥1 és pillHossz≠szakaszHossz
    VanJóEset(i, van, j)
    Ha van akkor  Szl(i):=j; pillHossz:+H(j);      i:+1
        különben Szl(i):=0; pillHossz:-H(Szl(i-1)); i:-1
  Ciklus vége
  Lefedhető:=pillHossz=szakaszHossz
Program vége.

Eljárás VanJóEset(Konstans i:Egész, Változó van:Logikai, j:Egész):
  j:=Szl(i)+1
  Ciklus amíg j≤N és (RosszEset(i,j) vagy
                 pillHossz+H(j)>szakaszHossz)
    j:+1
  Ciklus vége
  van:=j≤N
Eljárás vége.
```

Lefedhető-e egy adott hosszúságú szakasz egyszeresen a H(1),..., H(N) hosszúságú kisebb szakaszokkal?

Lefedés folyt.

- **Függvény** RosszEset(**Konstans** i,j:Egész): Logikai
- k:=1
- **Ciklus amíg** k<i **és** Szl(k)≠j
- k:+1
- **Ciklus vége**
- RosszEset:=k<i
- **Függvény vége.**
-

Dinamikus programozás

- a feladatot részfeladatokra való osztással oldja meg
- akkor alkalmazható, ha a részproblémák nem függetlenek, azaz közös részproblémáik vannak
- minden egyes részfeladatot és annak minden részfeladatát pontosan egyszer oldja meg
- elkerüli az ismételt számítást, mivel a részfeladatok eredményeit tároljuk

Megoldás

- A dinamikus programozást optimalizálási feladatok megoldására használjuk. Ilyen feladatoknak sok megengedett megoldása lehet, ezek közül elegendő egy optimális (minimális vagy maximális) értékűt megtalálni. Az algoritmus kifejlesztése négy lépésre bontható:
- Jellemezzük az optimális megoldás szerkezetét.
- Rekurzív módon definiáljuk az optimális megoldás értékét.
- Kiszámítjuk az optimális megoldás értékét alulról felfelé történő módon.
- A kiszámított információk alapján megszerkesztünk egy optimális megoldást.

Probléma

- Hátizsák probléma: a kincses szigetről szeretnék a hátizsákomban titokban minél nagyobb értékű kincset elvinni.
- Bemenő adat: $S_1..s_m$ súlyok, $v_1..v_m$ értékek
- Kimenő adat: $I \subseteq \{1,..m\}$ részhalmaz, amelyre teljesül $\sum_{i \in I} s_i \leq b$ és $\sum_{i \in I} v_i \geq k$

Megoldás - táblázat

- V[i,a] a maximális elérhető érték az s1..si súlyokkal és a súlykorláttalM+1 sorból és b+1 oszlopból áll

0………………a…………………..b

	0	0	0	0	0
0	0				
	0				
i	0				
	0				
m	0				
	0				

$$v(i,a) = \max\{v(i-1,a); vi + v(i-1,a-si)\}$$

Számpélda

súly	érték
5	2
3	8
1	2
2	1
2	2
4	4
3	1
2	5

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	8	0	2	0	0	10	0
2	0	8	10	2	0	0	10	12
2	1	8	10	9	11	3	10	12
2	2	8	10	10	12	11	13	12
2	2	8	10	10	12	12	14	14
2	2	8	10	10	12	12	14	14
2	5	8	10	13	15	15	17	17

Megoldás kikeresése

súly	érték
5	2
3	8
1	2
2	1
2	2
4	4
3	1
2	5

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	8	0	2	0	0	10	0
2	0	8	10	2	4	0	10	12
2	1	8	10	9	11	0	10	12
2	2	8	10	10	12	11	13	12
2	2	8	10	10	12	12	14	14
2	2	8	10	10	12	12	14	14
2	5	8	10	13	15	15	17	17

Megoldás

súly	érték
5	2
3	8
1	2
2	1
2	2
4	4
3	1
2	5

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	8	0	2	0	0	10	0
2	0	8	10	2	0	0	10	12
2	1	8	10	9	11	3	10	12
2	2	8	10	10	12	11	13	12
2	2	8	10	10	12	12	14	14
2	2	8	10	10	12	12	14	14
2	5	8	10	13	15	15	17	17

Pénzváltás. Feladat: a lehető legkevesebb bankjeggyel (érmével) fizessünk ki egy összeget.

- Mohó stratégia: mindig a legnagyobb címlettel próbálkozunk.
- Ha a lehetséges címletek: 1,5,8,10, a mohó megoldás nem mindig optimális.
- 13=10+1+1+1, az optimális 13=8+5.
- Ha a lehetséges címletek: 1,5,10,25, a mohó megoldás mindig optimális.

Probléma I.

- Egy N szintes épület szintjeit fehér (F), piros (P) és zöld (Z) színnel festhetjük ki. Fehér emeletet csak piros emelet követhet, zöld emeletet pedig nem követhet piros! Készíts programot, amely megadja, hogy az épület hányféleképpen színezhető ki!

Megoldás:

- F→P
- P→F,P,Z
- Z→F,Z
- Nézzük fordítva; ez szükséges a rekurzióhoz. Kérdés, hogy mi a megelőzője?
- P előtt lehet F,P
- F előtt lehet P, Z
- Z előtt lehet P, Z

Megoldás: Dinamikus programozás – részmegoldások tárolása (táblázatban)

1	**1**	**1**
2	2	2
2+2=4	2+2=4	2+2=4
4+4=8	4+4=8	4+4=8
8+8=16	8+8=16	8+8=16
2^N	2^N	2^N

Megoldás: $3*2^N$

Mohó algoritmus

- 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.

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.

Pénzváltás. Feladat: a lehető legkevesebb bankjeggyel (érmével) fizessünk ki egy összeget.

- Mohó stratégia: mindig a legnagyobb címlettel próbálkozunk.
- Ha a lehetséges címletek: 1,5,8,10, a mohó megoldás nem mindig optimális.
- 13=10+1+1+1, az optimális 13=8+5.
- Ha a lehetséges címletek: 1,5,10,25, a mohó megoldás mindig optimális.

Összefoglalás

- Nyers erő módszer – brute force
- Backtrack
- Mohó algoritmus
- Dinamikus programozás

Köszönöm a figyelmet!