Programozás I.

Mohó algoritmusok

Sergyán Szabolcs
sergyan.szabolcs@nik.uni-obuda.hu

Óbudai Egyetem
Neumann János Informatikai Kar

2012. december 3.

1. Postai ügyintézés
2. Mohó stratégia
3. Kincsek begyűjtése
4. 0-1 hátizsák probléma
5. Tankolási pontok választása
6. Ütemezési feladatok
   - Esemény kiválasztási probléma
   - Esemény elkülönítési probléma
   - Ütemezés késés minimalizálással

Mohó algoritmusok

1. Postai ügyintézés
2. Mohó stratégia
3. Kincsek begyűjtése
4. 0-1 hátizsák probléma
5. Tankolási pontok választása
6. Ütemezési feladatok
   - Esemény kiválasztási probléma
   - Esemény elkülönítési probléma
   - Ütemezés késés minimalizálással

Postai pénztáros

Feladat

Egy nyugdíjas elmegy a postára, hogy felvegye 79.845 Ft-os nyugdíját. Hogyan tudja a postai pénztáros kifizetni neki ezt az összeget úgy, hogy a kifizetéshez a lehető legkevesebb darab papír pénzt, illetve pénzérmét használja?

Megjegyzések

- Rendelkezésre álló papír pénzek és pénzérmék: 20.000, 10.000, 5.000, 2.000, 1.000, 500, 200, 100, 50, 20, 10 és 5 Ft-os.
- Tegyük fel, hogy minden címletből korlátlan mennyiség van a postán.
- Az nem tekinthető megoldásnak, ha a nyugdíjasnak vissza kell adnia. (Pl.: a pénztáros ad 80.000 Ft-ot 4 db 20.000 Ft-os címlettel, majd a nyugdíjas visszaad 155 Ft-ot.

Postai pénztáros

Megoldás

A pénztáros az alábbi címleteket fogja adni:

3 db	20.000 Ft-os	=	60.000 Ft
1 db	10.000 Ft-os	=	10.000 Ft
1 db	5.000 Ft-os	=	5.000 Ft
2 db	2.000 Ft-os	=	4.000 Ft
1 db	500 Ft-os	=	500 Ft
1 db	200 Ft-os	=	200 Ft
1 db	100 Ft-os	=	100Ft
2 db	20 Ft-os	=	40 Ft
1 db	5 Ft-os	=	5 Ft
			79.845 Ft

Ez összesen **13 db** papír pénzt, illetve pénzérmét jelent.

Postai pénztáros

Kérdés

Megoldható a feladat kevesebb címlettel is?

Válasz

Nem.
(Ezt nem fogjuk pontosan bizonyítani, de az eddigi tapasztalataink alapján tudjuk, hogy egy optimális megoldást adtunk meg.)

Postai pénztáros

Algoritmus

Bemenet: x - kifizetendő összeg
Kimenet: S - papírpénzek és pénzérmék halmaza

Függvény Pénzkifizetés(x)
    $S \leftarrow \emptyset$
    Ciklus amíg $x \neq 0$
        $k \leftarrow$ a legnagyobb egész, melyre $c_k \leq x$
        Ha $k = 0$ **akkor**
            return nincs megoldás
        Elágazás vége
        $x \leftarrow x - c_k$
        $S \leftarrow S \cup \{k\}$
    Ciklus vége
    return S
Függvény vége

$c_1 < c_2 < \ldots < c_n$ jelöli a rendelkezésre álló címleteket.

Postai pénztáros

- A bemutatott algoritmus egy mohó algoritmus, mert mindig a legnagyobb választható címletet választja ki.
- A mohó algoritmus ebben a példában optimális megoldást ad.

Levélfeladás

Feladat

- Nyugdíjasunk szeretne feladni egy levelet, mely 1400 Ft-ba kerül.
- Hogyan lehet ezt megtenni, ha a lehető legkevesebb számú bélyeget szeretnénk a borítékra tenni.

Megjegyzés

Rendelkezésre álló bélyegcímletek:

- 3.500 Ft
- 1.000 Ft
- 700 Ft
- 340 Ft
- 210 Ft
- 100 Ft
- 10 Ft

Levélfeladás

Megoldás

Ha a mohó algoritmust használjuk oly módon, hogy mindig a lehető legnagyobb címletű bélyeget ragasztjuk fel, akkor az alábbi megoldást kapjuk:

0 db	3.500 Ft-os	=	0 Ft
1 db	1.000 Ft-os	=	1.000 Ft
0 db	700 Ft-os	=	0 Ft
1 db	340 Ft-os	=	340 Ft
0 db	210 Ft-os	=	0 Ft
0 db	100 Ft-os	=	0 Ft
6 db	10 Ft-os	=	60 Ft
			1.400 Ft-os

Az algoritmus szerint 8 db bélyegre van szükségünk.

Megoldás elemzése

- Könnyen látható, hogy a mohó algoritmus által adott megoldás most nem optimális.
- Az optimális megoldás 2 db 700 Ft-os bélyeg választása lenne.

Mohó algoritmusok

Mohó stratégia

- A mohó algoritmus egy probléma optimális megoldását úgy alkotja meg, hogy választások sorozatát hajtja végre.
- Minden döntési pontban azt az esetet választja az algoritmus, mely az adott pillanatban éppen optimálisnak tűnik.

- Két fontos alkotóeleme van az olyan problémáknak, melyek mohó algoritmussal megoldhatók:
  - Mohó-választási tulajdonság
  - Optimális részstruktúrák tulajdonság

Mohó-választási tulajdonság

- Globális optimális tulajdonság elérhető lokális optimum választásával
  - Itt van különbség a mohó algoritmusok és a dinamikus programozás között
  - Dinamikus programozásnál minden lépésben választunk, de a választás függ(het) a részproblémák megoldásától.
  - Mohó esetben viszont az adott pillanatban legjobbnak tűnő választást hajtjuk végre, majd utána oldjuk meg a választás hatására fellépő részproblémákat.
  - A dinamikus programozás alulról-felfelé haladva ad megoldást, a mohó stratégia viszont felülfől-lefelé halad, így egymás után végrehajt mohó választásokat, melyekkel folyamatosan redukálja a probléma méretét.

Optimális részstruktúrák tulajdonság

- Egy probléma teljesíti az *optimális részstruktúra tulajdonságot*, ha az optimális megoldás felépíthető a részproblémák optimális megoldásából.
- Ez az alkotóelem kulcsfontosságú, mind a dinamikus programozás, mind a mohó stratégia alkalmazhatóságának vizsgálatánál.

Kincsek begyűjtése

Feladat

- Egy téglalap egész koordinátájú rácspontjaiban különböző értékű kincsek vannak elhelyezve.
- Menjünk el a bal alsó rácspontból a jobb felső rácspontba úgy, hogy közben a lehető legtöbb kincset gyűjtük be!
- Bejárásunk során csak jobbra és felfelé haladhatunk.

C

i \ j	1	2	3	4
4	1	5	3	6
3	11	2	15	1
2	6	10	20	9
1	1	3	4	8

Megoldás dinamikus programozással

Definiálunk egy T mátrixot, mely értékei az egyes rácspontokba való eljutás során maximálisan begyűjthető kincsek mennyiségét adják meg.

$$T[i][j] = \begin{cases} C[i][j], & \text{ha } i = 1,\ j = 1 \\ C[i][j] + T[i-1][j], & \text{ha } i \neq 1,\ j = 1 \\ C[i][j] + T[i][j-1], & \text{ha } i = 1,\ j \neq 1 \\ C[i][j] + \max\{T[i-1][j], T[i][j-1]\}, & \text{ha } i \neq 1,\ j \neq 1 \end{cases}$$

C

i \ j	1	2	3	4
4	1	5	3	6
3	11	2	15	1
2	6	10	20	9
1	1	3	4	8

T

i \ j	1	2	3	4
4	19	24	55	61
3	18	16	52	53
2	7	17	37	46
1	1	4	8	16

Megoldás mohó algoritmussal

- Elindulunk a bal alsó sarokból
- Megvizsgáljuk, hogy a jobb- vagy a felső szomszéd értéke nagyobb-e, és arra megyünk tovább

$i \leftarrow 1$; $j \leftarrow 1$
Ciklus amíg $i < 4$ és $j < 4$
    Ha $C[i+1][j] > C[i][j+1]$ **akkor**
        $i \leftarrow i + 1$
    Különben
        $j \leftarrow j + 1$
    Elágazás vége
Ciklus vége
Ciklus amíg $i < 4$
    $i \leftarrow i + 1$
Ciklus vége
Ciklus amíg $j < 4$
    $j \leftarrow j + 1$
Ciklus vége

Kincsek begyűjtése

Megoldás mohó algoritmussal

C

i \ j	1	2	3	4
4	1	5	3	6
3	11	2	15	1
2	6	10	20	9
1	1	3	4	8

A bejárás során 44 kincset lehet begyűjteni.

Megoldás dinamikus programozással

C

i \ j	1	2	3	4
4	1	5	3	6
3	11	2	15	1
2	6	10	20	9
1	1	3	4	8

A bejárás során 61 kincset lehet begyűjteni.

0-1 hátizsák probléma

Vizsgáljuk meg, hogy a dinamikus programozásnál megismert 0-1 hátizsák probléma megoldható-e mohó stratégiával.

A probléma megfogalmazása

- Adott n darab tárgy
- Az i-edik tárgy értéke: p_i
- Az i-edik tárgy súlya: w_i
- Kiválasztandó a tárgyak olyan részhalmaza, melyek értékének összege a lehető legnagyobb, miközben a súlyuk összege nem haladja meg a hátizsák c kapacitását.

0-1 hátizsák probléma

Megoldás mohó algoritmussal

Első körben el kell döntenünk, hogy milyen jellemző alapján akarunk mohó stratégiát folytatni:

- Érték alapján
- Súly alapján
- Érték/súly arány alapján

Tekintsük pl. az érték/súly arány alapú megközelítést.
Legyenek a tárgyak érték/súly arány alapján csökkenő módon rendezettek, azaz

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_n}{w_n}.$$

0-1 hátizsák probléma

Algoritmus

Bemenet: n - tárgyak darabszáma; p - tárgyak értékeinek tömbje; w - tárgyak súlyainak tömbje; c - hátizsák kapacitása
Kimenet: A - kiválasztott tárgyak sorszámainak halmaza

Függvény Mohó0-1HátizsákProbléma(n,p,w,c)
  $u \leftarrow c$
  $A \leftarrow \emptyset$
  $i \leftarrow 1$
  Ciklus amíg $u > 0$ és $i \leq n$
    Ha $w_i \leq u$ **akkor**
      $A \leftarrow A \cup \{i\}$
      $u \leftarrow u - w_i$
    Elágazás vége
    $i \leftarrow i + 1$
  Ciklus vége
  return(A)
Függvény vége

0-1 hátizsák probléma

Optimális megoldást ad a mohó stratégia?

Tekintsük az alábbi példát:

- Három tárgyunk van: $p_1 = 60$, $w_1 = 10$; $p_2 = 100$, $w_2 = 20$; $p_3 = 120$, $w_3 = 30$.
- A hátizsákunk mérete $c = 50$ egységnyi.
- A mohó stratégia alapján az 1. és 2. tárgyat fogjuk kiválasztani, így tele rakjuk a zsákot és összesen 160 értékű tárgyat viszünk el.
- Ha viszont a 2. és 3. vagy akár az 1. és 3. választottuk volna, akkor a zsákunkba azok is beférnének, de nagyobb lenne az elvitt érték.

A mohó stratégia itt nem ad optimális megoldást!

Tankolási pontok választása

Feladat

- Madridból szeretnénk Moszkvába utazni autóval
- Az útvonalat ismerjük, nem feladat annak a módosítása
- Tudjuk, hogy utunk során hol vannak benzinkutak
- Autónk benzintankjának méretét és kocsink fogyasztását is jól ismerjük
- Határozzuk meg, hogy hol kell megállnunk tankolni, ha minimális számú tankolással szeretnénk az utat teljesíteni.

Tankolási pontok választása

Megoldási ötlet

1. A kiindulási ponttól menjünk el a legtávolabbi benzinkútig, ameddig még van elegendő benzinünk
2. Ott tankoljunk tele
3. Ismét menjünk el a legtávolabbi benzinkútig, ameddig még van elegendő benzinünk
4. Ott tankoljunk tele
5. $\vdots$

Folytassuk addig az algoritmust, amíg el nem jutunk a célállomásig!

Megjegyzés:

- Ha valahol két egymást követő benzinkút távolsága nagyobb, mint a teli tankkal megtehető távolság, akkor nem teljesíthető az utazás.

Tankolási pontok választása

Megoldás

Jelölések:

- L jelöli az utunk hosszát
- A benzinkutak távolságát a kiindulási ponttól b_i-k jelölik, az alábbiak szerint:
$$0 = b_0 < b_1 < b_2 < \ldots < b_n = L$$
- S jelöli az algoritmus által megadott megállási pontok halmazát
- x jelöli az aktuális helyünket (távolságunkat a kiindulási ponttól)
- C jelöli a teli tankkal megtehető maximális távolságot

Megoldás

Algoritmus:

$S \leftarrow \{0\}$
$x \leftarrow 0$
Ciklus amíg $x \neq b_n$
    $p \leftarrow$ a legnagyobb index, melyre $b_p \leq x + C$
    Ha $b_p = x$ **akkor**
        return "nincs megoldás"
    Elágazás vége
    $x \leftarrow b_p$
    $S \leftarrow S \cup \{p\}$
Ciklus vége

Mohó algoritmusok

Esemény kiválasztási probléma

Erőforrás ütemezést szeretnénk megvalósítani egymással versengő feladatok között.

Feladat megfogalmazása

- Adott *események* egy $S = \{1, 2, \ldots, n\}$ halmaza, amelyek egy közös erőforrást kívánnak használni
- Minden eseményre ismert
  - a *kezdő időpontja*: s_i
  - a *befejező időpontja*: f_i (ahol $s_i \leq f_i$)
- Ha az i eseményt kiválasztjuk, az esemény az $[s_i, f_i[$ intervallumot foglalja el
- Az i és j események *kompatibilisek*, ha az $[s_i, f_i[$ és $[s_j, f_j[$ intervallumok nem fedik egymást, azaz $s_i \geq f_j$ vagy $s_j \geq f_i$
- Kiválasztandó kölcsönösen kompatibilis eseményeknek egy legnagyobb elemszámú halmaza

Esemény kiválasztási probléma

Megoldás

Tegyük fel, hogy az események a befejező időpontjaik szerint rendezettek, azaz

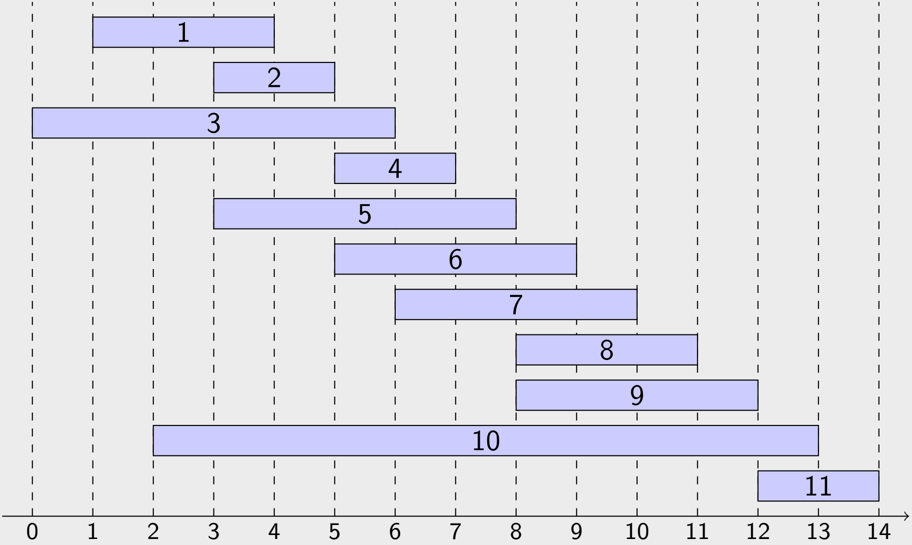
$$f_1 \leq f_2 \leq \ldots \leq f_n$$

Bemenet: Kezdő időpontok tömbje (s), befejező időpontok tömbje (f), tömbök mérete (n)
Kimenet: Kiválasztott feladatok indexeinek halmaza (A)

Függvény MohóEseményKiválasztó(s,f,n)
    $A \leftarrow \{1\}$
    $j \leftarrow 1$
    Ciklus $i \leftarrow 2$**-től** n**-ig**
        Ha $s_i \geq f_j$ **akkor**
            $A \leftarrow A \cup \{i\}$
            $j \leftarrow i$
        Elágazás vége
    Ciklus vége
    return(A)
Függvény vége

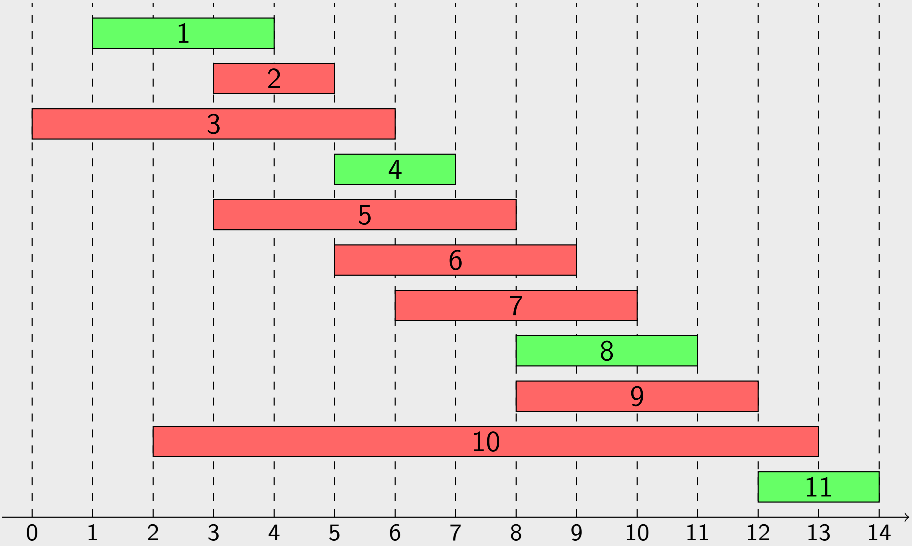
Esemény kiválasztási probléma

Példa



Esemény kiválasztási probléma

Példa megoldása



Esemény kiválasztási probléma

Optimális megoldást ad a mohó algoritmus?

1. A mohó algoritmus kiválasztja az 1 eseményt. Ez az esemény biztos benne van az optimális megoldásban is?
    - Tegyük fel, hogy $A \subset S$ egy optimális megoldás, és legyenek az A-beli események a befejezési idő szerint rendezettek.
    - Tegyük fel, hogy A-ban az első esemény k.
    - Ha $k = 1$, akkor A a mohó választással kezdődik.
    - Ha $k \neq 1$, akkor legyen $B = (A \setminus \{k\}) \cup \{1\}$.
    - Mivel $f_1 \leq f_k$, így a B-beli események nem átfedők, és B ugyanannyi elemet tartalmaz, mint A, így B is optimális.
    - Tehát mindig létezik olyan optimális ütemezés, mely a mohó választással kezdődik.

Optimális megoldást ad a mohó algoritmus?

2. Az 1 esemény mohó választása után a probléma redukálódik azon esemény kiválasztási problémára, amely az S halmaz azon elemeit tartalmazza, amelyek kompatibilisek az 1 eseménnyel.
   - Kérdés, hogy $A' = A \setminus \{1\}$ optimális megoldása-e az $S' = \{i \in S : s_i \geq f_1\}$ eseményeket tartalmazó problémának.
   - Ha találnánk olyan B' megoldását az S' problémának, amely több eseményt tartalmazna, mint A', akkor az 1 eseményt hozzáadva B'-höz az S probléma olyan B megoldásához jutnánk, amely több eseményt tartalmaz, mint A.
   - Ez viszont ellentmond A optimalitásának.
   - Tehát minden mohó választás után olyan problémánk marad, mint amilyen az eredeti is volt.

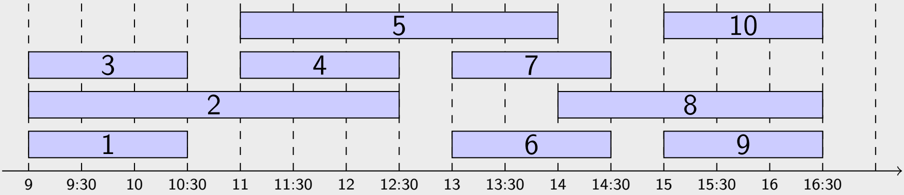
Esemény elkülönítési probléma

Feladat megfogalmazása

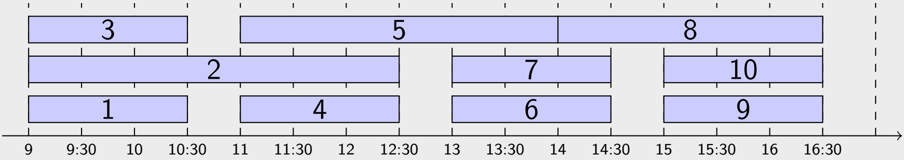
- Adott n darab esemény, melyeknek ismerjük a kezdési és befejezési időpontját. (Az i-edik esemény kezdő időpontja: s_i, befejezési időpontja pedig: f_i.)
- Különítsük el az eseményeket oly módon előadó termekbe, hogy az egy előadó terembe tartozó események kompatibilisek legyenek egymással, viszont a lehető legkevesebb előadó termet használjuk.

Esemény elkülönítési probléma

Egy példa



Kevesebb előadó terem is elég lenne



Esemény elkülönítési probléma

Algoritmus

Tegyük fel, hogy az események a kezdő időpontjaik szerint rendezettek, azaz

$$s_1 \leq s_2 \leq \ldots \leq s_n.$$

Függvény MohóEseményElkülönítő
    $d \leftarrow 0$
    Ciklus $i \leftarrow 1$-**től** n-**ig**
        $k \leftarrow 1$
        Ciklus amíg $k \leq d$ és az i-edik esemény nem kompatibilis a k-adik előadóban lévő eseményekkel
            $k \leftarrow k + 1$
        Ciklus vége
        Ha $k \leq d$ **akkor**
            i-edik esemény legyen a k-adik előadóban
        Különben
            $d \leftarrow d + 1$
            Legyen a d-edik előadó is használatba véve
            i-edik esemény legyen a d-edik előadóban
        Elágazás vége
    Ciklus vége
    return(előadók és hozzájuk rendelt események listája)
Függvény vége

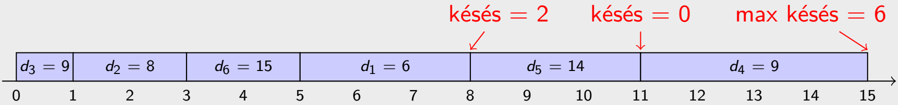
Ütemezés késés minimalizálással

Feladat

- Adottak egyetlen erőforrást igénylő feladatok
- Ismerjük az i-edik feladat elvégzéséhez szükséges t_i időt, illetve a feladat d_i határidejét
- Az i-edik feladat késése: $l_i = \max\{0, f_i - d_i\}$
- Cél: minimalizálni a késések maximumát,
  azaz minimalizálni a $\max\{l_i\}_{i=1,\dots,n}$ kifejezést

Példa

	1	**2**	**3**	**4**	**5**	**6**
$\mathbf{t_i}$	3	2	1	4	3	2
$\mathbf{d_i}$	6	8	9	9	14	15



Ütemezés késés minimalizálással

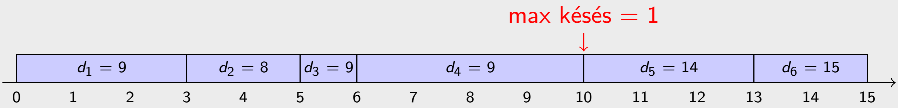
Algoritmus

Tegyük fel, hogy az események a határidejük szerint rendezettek, azaz

$$d_1 \leq d_2 \leq \ldots \leq d_n$$

Függvény MohóKésésMinimalizáló
    $\tau \leftarrow 0$
    Ciklus $i \leftarrow 1$**-től** n**-ig**
        $s_i \leftarrow \tau$
        $f_i \leftarrow \tau + t_i$
        $\tau \leftarrow \tau + t_i$
    Ciklus vége
    return(s,f)
Függvény vége

Megoldás



Felhasznált irodalom

- T.H. Cormen, C.E. Leiserson, R.L. Rivest: Algoritmusok. Műszaki Könyvkiadó, 1999
- J. Kleinberg, É. Tardos: Algorithm Design. (Előadás prezentációk)