Algoritmusok gyakorlat, 2020

5. hét, október 5.

5. gyakorlat

Egy mohó algoritmus minden lépésben egy lokálisan legjobb döntést hoz, de figyelmen kívül hagyja ennek a döntésnek a hatását a későbbi eseményekre.

,,optimálisnak tűnő döntés"

Nem minden probléma oldható meg optimálisan mohó algoritmussal!

5. gyakorlat

Részproblémára bontáskor az a cél, hogy a mohó választás *egyetlen* részproblémát eredményezzen, amelynek optimális megoldásából következik az eredeti probléma optimális megoldása.

5. gyakorlat

1. Fogalmazzuk meg az optimalizálási feladatot úgy, hogy minden egyes döntés hatására *egy* megoldandó részprobléma keletkezzen.
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 megoldásá*t kapjuk. (=„optimális részstruktúrák”)

5. gyakorlat

Oldjuk meg a hátizsák feladatot, ha a hátizsák kapacitása 12, a tárgyak súlyai és értékei:

(3, 5), (4, 4), (2, 5), (6, 5), (5, 4)

Mi a megoldás töredékes hátizsák feladat esetében?

5. gyakorlat

Adott n elem, mindegyiknek van súlya és értéke. Pakoljuk ezeket az elemeket egy W méretű hátizsákba úgy, hogy a benne lévő elemek összértéke maximális legyen! Két változatot vizsgálunk, az úgynevezett 0-1 probléma esetén minden elemet vagy beleteszünk, vagy kihagyunk. A töredékes változatnál az egyes elemek valamekkora részét is választhatjuk.

5. gyakorlat

0-1 probléma megoldása dinamikus programozással:

Egy adott részfeladat optimális megoldásának kiszámításához minden elemre két esetet különböztetünk meg.

1. eset: Az elem szerepel az optimális részhalmazban.
2. eset: Az elem nem szerepel az optimális részhalmazban.

Ezért az „n” elem esetén elérhető maximális érték a következő két érték maximuma.

1. Az n-edik elem értékéhez hozzáadjuk azt az előző n-1 elem által kapott maximális értéket, amit W-ből az n-edik elem súlyát kivonva kapunk.
2. Az előző n-1 elem és a W súly által elérthető maximális érték (az n-edik elem nélkül).

5. gyakorlat

A tárgyak súlyai (w) és értékei (v):

(3, 5), (4, 4), (2, 5), (6, 5), (5, 4)

$$T[i, j] = \max\{ T[i-1, j], v_i + T[i-1, j - w_i,]\}$$

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

5. gyakorlat

A tárgyak súlyai (w) és értékei (v):
(3, 5), (4, 4), (2, 5), (6, 5), (5, 4)

Mohó algoritmus: vegyük mindig a legnagyobb v/w értékkel bíró elemet!

V	5	4	5	5	4
W	3	4	2	6	5
v/w	5/3	1	5/2	5/6	4/5

0-1 probléma esetén lehet hogy nem optimális!

5. gyakorlat

A tárgyak súlyai (w) és értékei (v):
(3, 5), (4, 4), (2, 5), (6, 5), (5, 4)

Mohó algoritmus: vegyük mindig a legnagyobb v/w értékkel bíró elemet!

V	5	4	5	5	4
W	3	4	2	6	5
v/w	5/3	1	5/2	5/6	4/5

Töredékes probléma esetén optimális.

Megoldás: 16.5=5+5+4+5/2.

5. gyakorlat

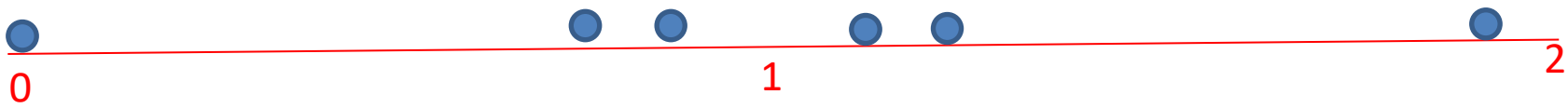
Adott a való számegyenesen valós számok x_1, x_2, x_n halmaza. Adjunk egy hatékony algoritmust, amely megad minimális számú egységnyi hosszú intervallumot, amelyek tartalmazzák a pontokat!

5. gyakorlat

Első mohó próbálkozás:

Fedjük le mindig mohó módon a legtöbb pontot ami lehetséges!

Nem feltétlen optimális:

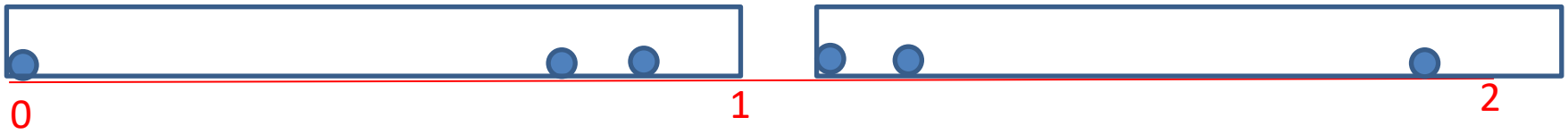


A fenti megoldás erre a példára 3-at ad, míg az optimális megoldás 2.

5. gyakorlat

Második mohó próbálkozás:

Mohó módon mindig azt az intervallumot vegyük egy részprobléma esetén, aminek a kezdőpontja a legnagyobb és tartalmazza a részproblémában az első számot! A számok mindig nagyság szerint rendezettek.



A fenti algoritmus optimális megoldást ad.

Ötlet: mindig van olyan optimális megoldás, aminél az első intervallum az első pontnál kezdődik.

5. gyakorlat

Egy hosszú utazást tervezünk autóval. Az autó tankjában *n* liter benzin fér el. Van egy térképünk, amely ábrázolja az úton levő benzinkutakat. Tervezzünk hatékony algoritmust, amely megadja azon benzinkutakat, amelyeknél megállva minimális számú megállással megoldható az utazás.

Példa: *n*= 40, az egymás utáni kutak távolsága (literben): 8, 12, 15, 18, 10, 15, 10, 12, 7

5. gyakorlat

Legyenek a K_{l-1},…, K_0 a benzinkutak az út során elérhető sorrendben és K_l a kiindulási pontunk.

A P_i részprobléma legyen az a feladat, hogy a K_i pontból, ahol tele a tank, jussunk el a célba minimális számú megállással. A P_i részprobléma esetén a mohó választás az, hogy a lehető legutolsó benzinkútnál állunk meg, azaz az utolsó olyan kútnál, amely nincs messzebb, mint n.

Tegyük fel, hogy ez az út a K_i,…, K_j pontokat fedi le, és a maradék útra az optimális megoldást vesszük. Ekkor a P_i probléma így kapott megoldásának költsége 1 + OPT(P_j). Másrészt ez valóban OPT(P_i), hiszen P_i minden megoldásában tankolni kell a K_{i+1},…, K_j kutak valamelyikénél es bárhogy választunk kutat, legalább a K_j-től kezdődő utat meg kell tennünk. Tehát mindig van olyan optimális megoldás, ami az utolsó olyan kútnál áll meg, ahol még nem fogy el a benzin.

A tankolási helyek a példában: 3,5,8

5. gyakorlat

Vegyük a pénzváltási probléma azon változatát, ahol a lehető legkevesebb érmére akarunk felváltani *n* forintot.

a) Tervezzünk mohó algoritmust, ha a felhasználható érmék 1, 5, 10 es 25 forintok. Igazoljuk, hogy az algoritmus optimális!

b) Adjunk meg olyan érmesorozatot, amelyre a mohó algoritmus nem optimális!

Házi feladat

5. gyakorlat

ZH témák

- Algoritmusok elemzése
- Oszd-meg-és-uralkodj, rekurzió:
  - nem független részfeladatok
- Dinamikus Programozás:
  - ismétlődő részfeladatok
  - tipikusan optimalizálási feladatok

5. gyakorlat

- Magyarázatok, megoldások:
- Gelle Kitti: Algoritmusok és adatszerkezetek I. gyakorlati jegyzet.
- http://www.inf.u-szeged.hu/~kgelle/sites/default/files/upload/alga-gyak-05.pdf