Algoritmusok és adatszerkezetek gyakorlat – 05

Mohó algoritmusok

Mohó algoritmus

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 mohó algoritmussal!

Ha mégis megoldható, akkor 2 tulajdonságot biztosan el tudunk róla mondani:

- **Optimális részstruktúra:** Egy feladat optimális részstruktúrájú, ha a probléma egy optimális megoldása önmagán belül a részfeladatok optimális megoldásait tartalmazza.
- **Mohó választás tulajdonság:** lokálisan optimális választások a globálisan optimális megoldáshoz vezetnek

Egy mohó algoritmus tervezésének lépései:

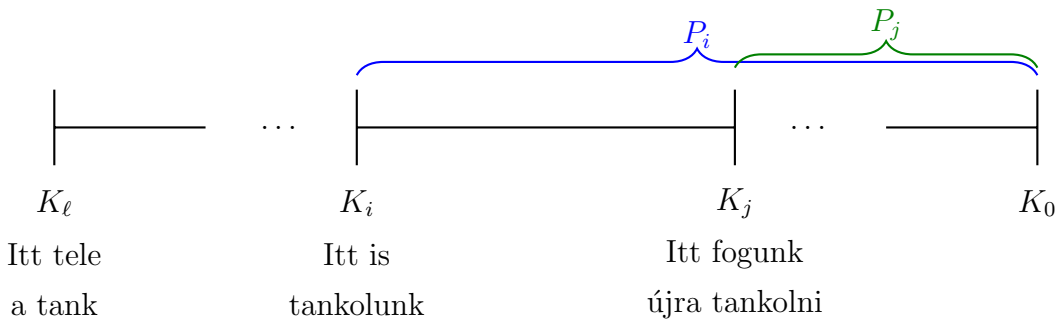
1. Fogalmazzuk meg az **optimalizációs feladat**ot úgy, hogy minden egyes döntés hatására egy megoldandó részprobléma keletkezzen.
2. **Bizonyítsuk be**, hogy mindig van olyan optimális megoldása az eredeti problémának, amely tartalmazza a mohó választást, tehát a mohó választás mindig biztonságos.
3. Mutassuk meg, hogy a mohó választással olyan **részprobléma** keletkezik, amelynek egy optimális megoldásához hozzávéve a mohó választást, az eredeti probléma egy optimális megoldását kapjuk.

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

Megoldás

Legyenek a $K_{\ell-1}, \ldots, K_0$ a benzinkutak az út során elérhető sorrendben és K_ℓ a kiindulási pontunk.

A P_i részprobléma 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 $K_i, \ldots, K_j$ 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 $K_{i+1}, \ldots, K_j$ kutak valamelyikénél, és 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.

2. Feladat 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 és 25 forintok. Igazoljuk, hogy az algoritmus optimális!

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

Megoldás

a) Ebben az esetben a $P(i)$ részprobléma az i forint optimális felváltása. A mohó stratégia az, hogy mindig a lehető legnagyobb pénzérmét használjuk a váltáshoz, amely még nem nagyobb az aktuális felváltandó összegnél.

Ebben az esetben meg kell mutatnunk, hogy mindig szükség van a lehető legnagyobb érme használatára. Először nézzük meg, hogy egy optimális megoldásban melyik érméből mennyi lehet maximálisan (pl esetanalízissel):

- Ha van a megoldásban 5 db 1-es, akkor ahelyett jobb az 1 db 5-ös $\Rightarrow$ egy optimális megoldásban max 4 db 1-es van
- Ha van benne 2 db 5-ös, akkor ahelyett jobb az 1 db 10-es $\Rightarrow$ egy optimális megoldásban max 1 db 5-ös van
- Ha van benne 3 db 10-es, akkor ahelyett jobb a $25+5 \Rightarrow$ egy optimális megoldásban max 2 db 10-es van

Most már meg tudjuk mutatni, hogy bármilyen i felváltandó összeg esetén vagy olyan a felbontás, hogy tudunk belőle jobbat csinálni a fentiek szerint, vagy használja a lehető legnagyobb érmét:

- Tfh a felváltandó pénz, $i \geq 25$. Ekkor összesen max 29 lesz ,,apróban" egy optimális megoldásban (hiszen tudjuk, hogy max 2×10, 1×5 és 4×1 lehet egy optimális felbontásban), de ha van köztük 2 db 10-es és 1 db 5-ös, akkor ahelyett egy 25-ös jobb $\Rightarrow$ Tehát ezek egyszerre nem lehetnek, azaz akkor vagy egy 5-ös hiányzik (és akkor ,,apróban" max 24 van egy ilyen optimális megoldásban), vagy egy 10-es (akkor pedig 19), tehát ,,apróból" csak kevesebb, mint 25 lehet egy optimális megoldásban $\Rightarrow$ ha ≥ 25 pénzt akarunk optimálisan legkevesebb érmére felváltani, akkor 25-öst be kell válasszunk
- Ha $25 > i \geq 10$, akkor szintén nem lehet benne csak max 1 db 5-ös, és max 4 db 1-es. Így ha $i \geq 10$, akkor az ,,apróból" csak 9 jön ki max $\Rightarrow$ ez pedig azt jelenti, hogy egy optimális felbontásban használni kell a 10-es érmét
- Ha $10 > i \geq 5$, akkor a max 4 db 1-es miatt kell használni az 5-ös érmét
- Ha $5 > i > 0$, akkor az pedig csak egyesekből jöhet ki.

b) Legyenek az érmék $1, 4, 5$. Ekkor $n = 8$-ra a mohó eredmény $8 = 5 + 1 + 1 + 1$, az optimális $8 = 4 + 4$.

3. Feladat Mi az optimális Huffman kódja az alábbi abc-nek: $f(a) = 3$, $f(b) = 4$, $f(c) = 6$, $f(d) = 17$, $f(e) = 21$, $f(f) = 29$?

Megoldás

Ahhoz, hogy a kódokat meg tudjuk mondani, fel kell építenünk a hozzá tartozó fát! Konstruáljuk meg tehát először ezt!

A	B	C	D	E	F
3	4	6	17	21	29

Sor feltöltése

Tehát a betűk Huffman kódjai:

- A: 0010
- B: 0011
- C: 000
- D: 01
- E: 10
- F: 11

Gyakorlás: Add meg a következő szöveg betűinek Huffman kódjait! A gyakoriságok megadásához szerencsénkre elég a segítőkész szövegre hallgatni.

> *This sentence contains three a's, three c's, two d's, twenty-six e's, five f's, three g's, eight h's, thirteen i's, two l's, sixteen n's, nine o's, six r's, twenty-seven s's, twenty-two t's, two u's, five v's, eight w's, four x's, five y's, and only one z.*

A megoldást ebben a jegyzetben, a 6. oldalon találod.