Programozás I.

Óbudai Egyetem, Neumann János Informatikai Kar

Dinamikus programozás

- A dinamikus programozás (DP) bevezetése
- Példa: 0/1 hátizsák probléma
- Dinamikus programozás jellemzői
- Optimális részstruktúra ÉS átfedő részproblémák
- Példa: Leghosszabb közös részsorozat (LCS)

Hallgatói tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, az előadásokon és gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint az előadásokon, és gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

A dinamikus programozás áttekintése

- A dinamikus programozás (DP) széles körben használt optimalizációs problémák megoldására:
 - ütemezési feladatok
 - legrövidebb út keresése
 - utazó ügynök
 - hasonló fehérjesorozatok, vagy aminósavak keresése
 - string-feladatok
 - elválasztási problémák
 - dinamikus idő-vetemítés
 - sztereó képfeldolgozás
 - csomagolási probléma, stb.
- A problémákat részproblémákra bontjuk és ezek megoldásait kombináljuk, hogy a nagyobb probléma megoldását megtaláljuk.
- Az "oszd meg és uralkodj" típusú megoldási módszerrel szemben, itt a részproblémák között kapcsolatok, átfedések állhatnak fent (átfedő részproblémák) ÉS optimális részstruktúrákból építkezünk.
- Az elnevezés matematikai optimalizálásra utal.



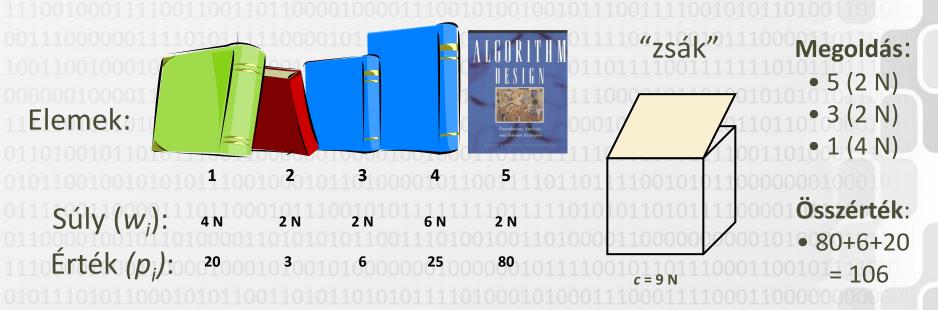
0/1 hátizsák probléma

0/1 knapsack problem

- "A thief is robbing the King's treasury, but he can only carry a load weighing at most W…"
- Adott: S halmaz n elemmel és egy súlyhatár c, minden i elemnek
 - $-p_i$ pozitív egész értéke van (*profit*)
 - w_i és pozitív egész súlya
- Cél: úgy válasszuk ki az elemeket, hogy az összérték maximális legyen, de az összsúly kisebb legyen, mint c.
 - Tjelölje azokat az elemeket, amit kiválasztunk, $T \subseteq S$
 - **Cél**: összérték maximalizálása $\sum_{i \in T} p_i$
 - **Feltétel**: a súlyhatáron belül kell maradni $\sum_{i \in T} w_i \leq c$

0/1 hátizsák probléma: példa

- Adott: n elemű S halmaz, minden elem
 - p_i pozitív értékű és w_i pozitív súlyú
- Cél: válasszunk ki elemeket, hogy az összérték maximális legyen, de az összsúly ne haladja meg c-t.
- Összesen 2ⁿ eset lehet! Minden eset kipróbálása: "brute force"



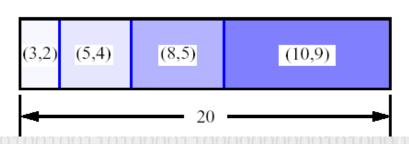
0/1 hátizsák probléma, első próbálkozás

- S_i: A halmaz elemeinek azonosítója 1-től i-ig (i <= n).
- Definiálja F[i] = a legjobb kiválasztást S_i -ből, azaz az elemek 1-től i-ig.
- Legyen pl. S = {(3,2),(5,4),(8,5),(4,3),(10,9)} érték-súly párok, és c = 20 kapacitás

A legjobb S_4 , ha csak négyet veszünk ki: Összsúly: 14, érték: 20



A legjobb S_5 : az 5. benne van, de a 4. nem



• Rossz hír: S_4 nem része az S_5 optimális megoldásnak

0/1 hátizsák probléma, más megközelítés

- S_i: Az elemek halmaza 1-től i-ig, i <= n.
- Definiálja F[i, x] = az S; halmazból a legjobb kiválasztást, ahol a súly legfeljebb x további paraméter, és a 0 lehetséges súlyt is figyelembe vesszük: x = 0, 1, 2, ..., c
- Ez optimális részprobléma megoldáshoz vezet.
- Az S_i legjobb kiválasztás legfeljebb x súllyal, két esetet jelenthet:
 - Ha $w_i > x$, akkor az i. elemet nem lehet hozzávenni, mert súlya nagyobb, mint az aktuális határ
 - Egyébként: ha a legjobb S_{i-1} részhalmaz súlya x w_i és ehhez vagy jön i, vagy nem eredményez nagyobb értéket

$$F[i,x] = \begin{cases} F[i-1,x] & \text{ha } w_i > x \\ \max\{F[i-1,x], F[i-1,x-w_i] + p_i\} & \text{egy\'ebk\'ent} \end{cases}$$

0/1 hátizsák algoritmus

• F[i, x] rekurzív formula:

$$F[i,x] = \begin{cases} F[i-1,x] & \text{ha } w_i > x \\ \max\{F[i-1,x], F[i-1,x-w_i] + p_i\} & \text{egyébként} \end{cases}$$

- F[i, x] = az 1-től i-ig tartó elemekből a legjobb kiválasztás, ahol az összsúly legfeljebb x
- Alapeset: i = 0, nem került elem kiválasztásra, azaz ha nem veszünk figyelembe elemet a halmazból, akkor az összérték 0.
- A feladat megoldása: a legnagyobb érték az utolsó sorban (n), utolsó oszlopban (c)
- Futási idő: O(n×c).
 Nem 2ⁿ ideig tart.
- Kivételesen a tömb indexeit 0-tól indítjuk, hogy az előzőkkel összhangban legyünk.

0/1 hátizsák algoritmus

Függvény vége

```
Függvény 0-1Knapsack(S, n, c)
    Input: S halmaz elemei p_i értékkel, w_i súllyal i = 1 ... n; valamint a max. súly c
    Output: a legjobb részhalmaz értéke (F[n, c]), hogy teljesül: összsúly \leq c
    Mivel 0 elem, vagy 0 súly esetén az összérték 0, ezért kivételesen 0-tól indexelünk
    Ciklus x \leftarrow 0-tól c-ig
      F[0,x] \leftarrow 0
    Ciklus vége
    Ciklus i \leftarrow 1-tól n-ig
          F[i, 0] \leftarrow 0
    Ciklus vége
    Ciklus i \leftarrow 1-től n-ig
        Ciklus x \leftarrow 1-től c-ig
            Ha w_i \le x, akkor
                F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)
            Különben
                F[i,x] \leftarrow F[i-1,x]
            Elágazás vége
        Ciklus vége
    Ciklus vége
    return F[n, c]
```

0/1 hátizsák példa

- n = 4 az adatok száma
- c = 5 kapacitás (maximális összsúly)
- Elemek: i súly (w_i) , profit (p_i)
 - **1**110101**2**01001001001100**3**1
 - 2 3 4
 - 311110041011111100101150
 - 140001005100010011101006

i \ x	0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2	3	4	5 = c
0001010	1010100001	0.00011101	0.01011101	1011111000	1011111011	0
1000111	10011001101	10000100001	11001010010	0101110011	1100101011	01001101
2010001	01101011111	01001011111	0010111000	11011111111	1001011100	OUTTOITE
3	10001100001	00111000100	11101000000	11111100001		
4 = n	01101010110 10111001100	10010111010 00001000010	11000001000 01000110100	10100010011 11111100010	1100101101 0011110011	10100001 01000114

Ciklus $x \leftarrow 0$ -tól c-ig

 $F[0,x] \leftarrow 0$

Ciklus vége

i\x	0	001011001101 010100010010	2	3	4	5 = c
0 001010	101010000	010 10 0001110	1 0 0101	1101101111	10001011111	1011001111
1000111	0	011000010000	11100101	0010010111	00111100101	101101001101
2010001	01 010000	000100101111	00010011	0110110110	1111001111	1110101110111
3	0	010011100010	01110100	0000111110		
4 = n	0	1 015010111101 000500100001	00100001	0100 1010001	0011 1110010 . 00100011110	001101000001

Ciklus i \leftarrow 1-tól n-ig

 $F[i,0] \leftarrow 0$

Ciklus vége

w_i ; p_i	
1: (2; 3)	
2: (3; 4)	
3: (4; 5)	
4: (5; 6)	

i\x	0	001011001101 010100010010	2	3	4	5 = c
0001010	101010000	010 100001110	1 0 010111	011011111	0001 0 111110	011000111110
1000111	0	0	1110010100	0100101110	01111001010	011010011010
2010001	010000	000100101111	0001001110	101101111	1110010111	110101101110
3	0	010011100010	0111010000	00011111100	00101101001	
4 = n	0	0000001011101	.0010000010 .0010001101	.001111100	01000111100	0110100001

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i,x] \leftarrow F[i-1,x]$$

$$i = 1$$

$$p_i = 3$$

$$w_i = 2$$

$$x = 1$$

$$x - w_i = -1$$

$w_i; p_i$	
1: (2; 3)	
2: (3; 4)	
3: (4; 5)	
4: (5; 6)	

i\x	0	10011011	2	3	4	5 = <i>c</i>
0001010	10 0 00	00011101	0 0101110	1011111000	1011111011	0111110
1000111	0	0	3	0101110011	1100101011	010011010
2010001	01000000	00100101111	001011100	1101101111	001111110	1011011100
3	0 100001	100111000100	011101000000	1111100001		010101102
4 = n	0,001100	000001000010	001000011010 00100011010	0100010011 01111100010	0011110011	01000011

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 1$$

$$p_i = 3$$

$$w_i = 2$$

$$x = 2$$

$$x - w_i = 0$$

w _i ; p _i	
1: (2; 3)	
2: (3; 4) 3: (4; 5) 4: (5; 6)	

i\x	0	010100110	2	3	4	5 = c
0001010	101010000	010 000	01 00101	11011011111	10001 0 11111	01100111110
1 000111	0	0	3	3	00111100101	011010011010
2010001	0 0 0000	0001001011	10001011	01101101101	1111001011	110101101110
3	0	01001110001	.001110100			
4 = n	0 0 1	00000010000	0101100000 0100100011	0100 10100010	001000111110	011010000

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 1$$

$$p_i = 3$$

$$w_i = 2$$

$$x = 3$$

$$x - w_i = 1$$

$w_i; p_i$
1: (2; 3)
2: (3; 4)
3: (4; 5)
4: (5; 6)

i\x	0	01010011011	2	3	4	5 = c
0001010	1010100000	010 000011101	0010-0	01111100	01011111011	0111110
1 000111	0	0	3	3	3	010011010
2010001	01 0100000	0001001011010	00101110110	110110111	10011111110	1011011100
3	0 10000	0100111000100				010101102
4 = n	0,00110	0000001000010	01000001000	1111110001	00011110011	0100001113

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 1$$

$$p_i = 3$$

$$w_i = 2$$

$$x = 4$$

$$x - w_i = 2$$

w _i ; p _i
1: (2; 3)
2: (3; 4) 3: (4; 5) 4: (5; 6)

i\x	0	0101000100	2	3	4	5 = c
0001010	101010000	010 10000111	001011	10110111	0001011111	10110 0 111111
1 000111	0	0	3	3	3	3
2010001	0 0000	00001001011	01000100110	1101101101	11110011111	1110101110111
3	0	01001110001				
4 = n	0	.0000001001110)1011000001)1001000110)100 00100010	0011 1110010. 001000111110	00110100001

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 1$$

$$p_i = 3$$

$$w_i = 2$$

$$x = 5$$

$$x - w_i = 3$$

01	w_i ; p_i	
30	1: (2; 3)	╛
90	2: (3; 4)	
11	3: (4; 5)	٦
nn	4: (5; 6)	ľ
0.4		
JI		

i\x	0	10011	2	3	4	5 = c
0001010	101010000	010 0 0 0 0 1 1	101 001011	1011011111	0001011111	01100111110
1000111	0	0	3	3	3	3
2010001	01010000	000 0 1011	01000100110	1101101101	.111001011	110101101110
J	011000		10011101000			10101010110
4 = n	0	000000101110	01001000001 01001000110	10011111100	001000011110	0110100001

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i,x] \leftarrow F[i-1,x]$$

$$i = 2$$

$$p_i = 4$$

$$w_i = 3$$

$$x = 1$$

$$x - w_i = -2$$

$W_i; p_i$	
	-11
2: (3; 4)	
3: (4; 5)	
4: (5; 6)	ľ
	1: (2; 3) 2: (3; 4) 3: (4; 5)

i\x	0	10011011	2	3	4	5 = c
000101010	1010000010	000011101	00101110	1011111000	1011111011	0111110
1 00011111	0	0	3 1	3	3	3
20100010	101000000	0)1011010	30010110	1101101111	001111110	10110111
J	0 1000010	0111000100	11101000000	1111100001		
4 = n	0	00010111010	01000011010	11111100010	0011110011	010000111

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i,x] \leftarrow F[i-1,x]$$

$$i = 2$$

$$p_i = 4$$

$$w_i = 3$$

$$x = 2$$

$$x - w_i = -1$$

01	w _i ; p _i	
20	1: (2; 3)	
90	2: (3; 4)	
11	3: (4; 5)	
00	4: (5; 6)	
01		

i\x	0	100110	2	3	4	5 = c
0001010	101010000	010 0000111	01 001011	1011011111	10001011111	01100111110
1000111	0 —	0	3	3	3	3
2010001	011010000	000 0010110	10 3 00110	110 40 1101	.1110011111	110101101110
3 100001	10011000	01001110001	0011101000	00001111100	00010110100	1010101010
4 = n	0	00000010000	1001000001 1001000110	10011111100	001000011110	0011010000111

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 2$$

$$p_i = 4$$

$$w_i = 3$$

$$x = 3$$

$$x - w_i = 0$$

01	w _i ; p _i
000	1: (2; 3)
100	2: (3; 4)
11	3: (4; 5)
.00	4: (5; 6)
01	

i\x	0	0010110011011 010100010010010	2	3	4	5 = c
0001010	101010000	0101000011101	00101110	101111100	01011111011	0 1111
1000111	0	0	3 0101001	3	3	3
2010001	01 01 00000	000 001011010	3.0011011	40110111	4 11110	10110111
3	0	0100111000100	01110100000	0111110000	10110100101	010101102
4 = n	0	00000010000010	001000011010	01111110001	00011110011	01000011

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 2$$

$$p_i = 4$$

$$w_i = 3$$

$$x = 4$$

$$x - w_i = 1$$

01	w _i ; p _i
000	1: (2; 3)
100	2: (3; 4)
11	3: (4; 5)
.00	4: (5; 6)
01	

i\x	0	01010001001	2	3	4	5 = c
0001010	101010000	010 0000111	01: 0 0101110	101111100	00101111101	10011111
1000111	0	011 0 01000	3 —	3	3	3
2010001	01 010000	000 0010110	10030011011	40110111	104111111	7
3	0	01001110001	001110100000	001111110000	1110101010	101010110
4 = n	0	00000010000	101100000100 1001000011010	01111110001	.0001111001	10100001

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 2$$

$$p_i = 4$$

$$w_i = 3$$

$$x = 5$$

$$x - w_i = 2$$

1	w_i ; p_i	
0	1: (2; 3)	ı
U	2: (3; 4)	
1	3: (4; 5)	
0	4: (5; 6)	1
6		
1		
		- 1

i\x	0	0101001101	2	3	4	5 = c
0001010	101010000	010100001110	1 0 0 1 0 1 1 1	10110111111	00010111110	011000111110
91000111	0	0	3	3	3	3
2010001	011010000	000 0010 101	0 3 00 101	10 401 1011	11 411111	10 7110111
3	0	0 0 0	3	4	00101101001	10101010110
4 = n	1 0 0011	.000000100001	00100001101	LOC11111000	01000111100	01101000011

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i,x] \leftarrow F[i-1,x]$$

$$i = 3$$

 $p_i = 5$
 $w_i = 4$
 $x = 1..3$
 $x - w_i = -3..-1$

1	w_i ; p_i	
0	1: (2; 3)	ŀ
0	2: (3; 4)	
1	3: (4; 5)	
O	4: (5; 6)	7
1		
74		

i\x	0	100110	2	3	4	5 = c
0001010	101010000	010 000111	01:00101	11011011111	10001 0 11111	0110 0 11111
1 0001111	0	0	13	3	3	3
2010001	01101000	0	1003 0011	01101401101	4 1 1 1 1	110 7110111
3	10	0	3	4	5	10101010110
4 = n	0	00000010000	100100001	010011111100	0100011110	0110100001

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

Elágazás vége

i = 3 $p_i = 5$ $w_i = 4$ x = 4 $x - w_i = 0$

01	w_i ; p_i
30	1: (2; 3)
10	2: (3; 4)
11	3: (4; 5)
00	4: (5; 6)
01	

i\x	0	010100110	2	3	4	5 = c
0001010	101010000	010 10 00011:	101: 00101	1101 011111	0001 0 11111	0110 0 11111
01000111	0	0	3	3	300101	3
2010001	01 010000	000 010110	30011	0110 401101	111 411111	110 7110 11
3	0	010 011000	0	0000 4 11100	0	101 7 10 10
4 = n	0 0011	.000000101110	0100100001	010011111100	0100011110	01101000011

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

$$i = 3$$

$$p_i = 5$$

$$w_i = 4$$

$$x = 5$$

$$x - w_i = 1$$

1	w_i ; p_i
	1: (2; 3)
U	2: (3; 4)
1	3: (4; 5)
'n	4: (5; 6)
U	
1	

i\x	0	01010011	2	3	4	5 = c
0001010	101010000	0101000011	101 00101	1101101111	10001 0 11111	011 0 11111
1000111	0	0	3	3	3	3
2010001	01 010000	000 001011	0100 3 0011	0110 4 0110	1111 4 1111	110 7110111
3	0	010 0 1000	3	4	5 5	101 7 10110
4 = n	0	0000	3	0100 41110	5	01101000011

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i-1, x], F[i-1, x-w_i] + p_i)$$

Különben

$$F[i,x] \leftarrow F[i-1,x]$$

$$i = 4$$

 $p_i = 6$
 $w_i = 5$
 $x = 1..4$
 $x - w_i = -4..-1$

1	w_i ; p_i
	1: (2; 3)
	2: (3; 4)
1	3: (4; 5)
	4: (5; 6)
U	
1	

i\x	0	01010001001	2	3	4	5 = c
0001010	101010000	010 000111	01:001011	1011011111	10001 0 11111	0110011111
1000111	0	011 0001000	3	3	3	3
2010001	011010000	000 0010110	1003 00110	1401101	111041111	110 7110111
3	1 0	010 0110001	0	1	J	7 10 10
4 = n	0	000 0 10000	3	4	511110	001107000

Ha
$$w_i \le x$$
, akkor

$$F[i, x] \leftarrow \max(F[i - 1, x], F[i - 1, x - w_i] + p_i)$$

Különben

$$F[i, x] \leftarrow F[i - 1, x]$$

Elágazás vége

i = 4 $p_i = 6$ $w_i = 5$ x = 5 $x - w_i = 0$

Ciklus vége

- Az algoritmus a maximális összsúlyt vette figyelembe úgy, hogy a zsákba tehető F[n, c] érték a lehető legnagyobb legyen
- Az elemek kiolvasásához egy visszafele haladó algoritmus szükséges, amely a táblázatot használja
- S kimenet a kiválasztott elemek indexeit tartalmazza a kiolvasás végén:

```
Eljárás Kiolvas (F[n, c], S)

S \leftarrow \emptyset

i \leftarrow n, x \leftarrow c

Ciklus amíg (i > 0) és (x > 0)

Ha F[i, x] \neq F[i - 1, x]

// Jelöljük meg az i. elemet, hogy a zsákban van

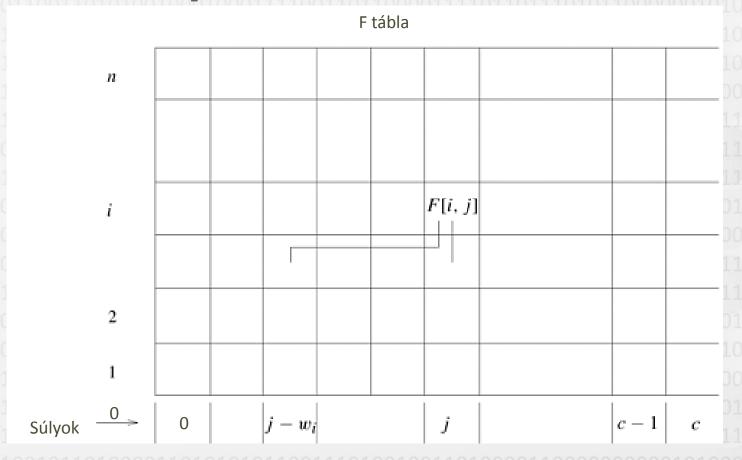
S \leftarrow S \cup \{i\}

x \leftarrow x - w_i

Elágazás vége

i \leftarrow i - 1
```

0/1 hátizsák probléma



A 0/1 hátizsák probléma F táblájának számítása. Az F[i, j] meghatározásához szükséges kommunikáció az i - 1 sorban található elemekkel, amelyek tartalmazzák F[i - 1, j] és F[i - 1, j - w_i] elemeket.

Az i értéke ebben a táblázatban lentről felfelé nő!

DINAMIKUS PROGRAMOZAS Dr. Vámossy Zoltán, OE NIK, 2012 vamossy.zoltan@nik.uni-obuda.hu 31

A dinamikus programozás áttekintése

- A problémamegoldás menete:
 - Az optimális megoldás jellemzése.
 - Részproblémákra osztás úgy, hogy az összetevőktől való függés "körmentes" legyen
 - A részproblémák optimális megoldása rekurzívan.
 - Az optimális megoldások felhasználása az eredeti feladat optimális megoldásának megtalálásához: egy részprobléma megoldását gyakran egy, vagy több az előző szintű részproblémák függvényeként adják meg.
- A rekurzív DP kifejezést funkcionális egyenletnek, vagy optimalizációs egyenletnek is nevezzük.
- A dinamikus programozás (DP) során az (átfedő) részmegoldások eredményeit gyakran táblázatokban tároljuk, hogy azokat – szemben a rekurzív megközelítésekkel – ne kelljen újra és újra kiszámolni (memoization, NEM memorization).

Dinamikus programozás

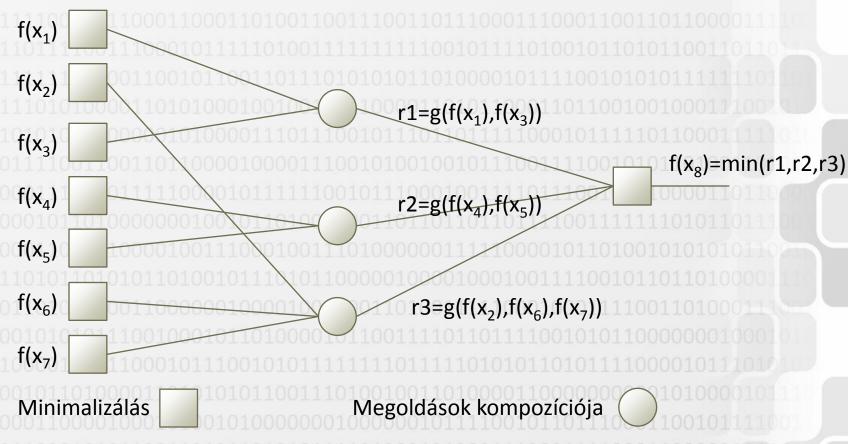
- Eleget kell tennünk bizonyos feltételeknek
- A DP formátumú probléma megoldását tipikusan a lehetséges megoldások minimumaként vagy maximumaként fejezzük ki.
 - Ha r az x_1 , x_2 ,..., x_l részproblémák kompozíciójából meghatározott megoldás költségét jelenti, akkor r a következő alakban írható fel

$$f = g(f(x_1), f(x_2), \dots, f(x_l)).$$

Itt g az un. kompozíciós függvény.

Ha minden probléma optimális megoldása az átfedő részfeladatok optimális megoldásának optimális kompozíciójaként kerül meghatározásra (azaz optimális részstruktúra) és a minimum (vagy maximum) érték kiválasztásra kerül, akkor DP formátumú megoldásról beszélünk. (Metamódszer, nem konkrét algoritmus.)

Dinamikus programozás: példa



Az $f(x_8)$ megoldás meghatározásának kompozíciója és számítása részfeladatok megoldásából.

LEGHOSSZABB KÖZÖS RÉSZSOROZAT (LCS) Dr. Vámossy Zoltán, OE NIK, 2012 vamossy.zoltan@nik.uni-obuda.hu 35

Leghosszabb közös részsorozat (LCS)

- Adott egy $X = \langle x_1, x_2, ..., x_n \rangle$ sorozat; az X részsorozatát kapjuk, ha valahány elemet törlünk belőle.
- Cél: Adott két sorozat: $X = \langle x_1, x_2, ..., x_n \rangle$ és $Y = \langle y_1, y_2, ..., y_m \rangle$, keressük meg a leghosszabb sorozatot, ami részsorozata egyaránt X-nek és Y-nak.
- Példa: ha X = <c, a, d, b, r, z> és Y = <a, s, b, z>, a leghosszabb közös részsorozata (longest common subsequence) X-nek és Y-nak <a, b, z>.
- Feladat: ha X = <A, B, C, B, D, A, B> és Y = <B, D, C, A, B, A>, mi
 X-nek és Y-nak a leghosszabb közös részsorozata?

Leghosszabb közös részsorozat (Longest-Common-Subsequence: LCS)

Stratégia:

- 1. Az LCS hosszát vizsgáljuk
- 2. Utána magát az LCS-t határozzuk meg
- Nem a közös részsorozatokat, hanem az X és Y "elejét" tekintjük
- Jelölje F[i, j] az X első i elemének és Y első j elemének leghosszabb közös részsorozat hosszát. Az LCS célja, hogy megtaláljuk F[n, m] értékét (és a sorozat elemeit).
- Ekkor igaz, hogy:

$$F[i, j] = \begin{cases} 0, & ha \ i = 0, vagy \ j = 0 \\ F[i-1, j-1] + 1, & ha \ i, j > 0 \text{ \'es } x_i = y_j \\ \max\{F[i, j-1], F[i-1, j]\}, & ha \ i, j > 0 \text{ \'es } x_i \neq y_j \end{cases}$$

LCS rekurzív algoritmus váz

```
Függvény LCS(X, Y, n, m)
   Input: X és Y sorozat n, illetve m elemmel, valamint i \le n és j \le m
   Output: F[i, j] az X és Y i és j hosszú elejének LCS hossza
   Mivel az alapesetek 0 hosszú részsorozatok, itt is 0-tól indexeljük az n + 1 \times m + 1 tömböt
   Ha (i = 0) vagy (j = 0) akkor
       F[i,j] \leftarrow 0
   Különben
       Ha x_i = y_i, akkor
           F[i,j] \leftarrow LCS(X, Y, i-1, j-1) + 1
       Különben
           F[i,j] \leftarrow \max(LCS(X, Y, i-1, j), LCS(X, Y, i, j-1))
       Elágazás vége
   Elágazás vége
   return F[i,j]
Függvény vége
```

Futási idő: exponenciális

LCS dinamikus programozással

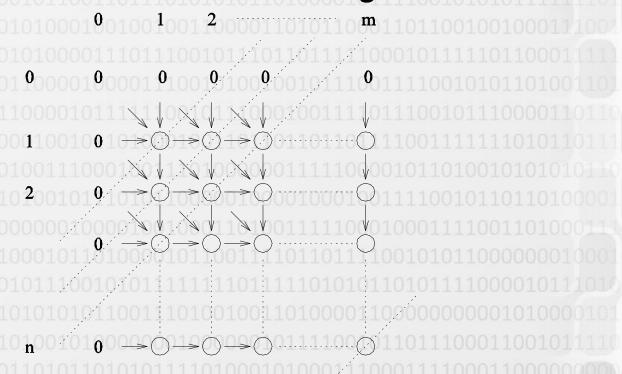
```
Eljárás LCS(X, Y, n, m, F)
   Input: X és Y sorozat n, illetve m elemmel, valamint i \le n és j \le m
   Output: F[n+1, m+1] az X és Y közös részsorozatainak hosszait tartalmazó tömb
   Mivel az alapesetek 0 hosszú részsorozatok, itt is 0-tól indexeljük az n+1\times m+1 tömböt
   Ciklus j \leftarrow 0-tól m-ig
   Ciklus vége
   Ciklus i \leftarrow 1-tól n-ig
   Ciklus vége
   Ciklus i \leftarrow 1-től n-ig
      Ciklus j \leftarrow 1-től m-ig
          Ha x_i = y_i, akkor 001000010011010011111100010001111
             F[i,j] \leftarrow F[i-1,j-1] + 1
          Különben
             F[i,j] \leftarrow \max(F[i-1,j], F[i,j-1])
          Elágazás vége
      Ciklus vége
   Ciklus vége
```

Futási idő: O(n × m) Dr. Vámossy Zoltán, OE NIK, 2012 vamossy.zoltan@nik.uni-obuda.hu

Eljárás vége

Leghosszabb közös részsorozat

- Az algoritmus kiszámolja a két-dimenziós F táblát (memoization) sor-oszlop sorrendben => O(n*m), konstans idő.
- Az átlós csomópontok mindegyike két részproblémához kapcsolódik, az előző szinthez és az azt megelőző szinthez.



Az LCS táblázat számítási elemei. A számítás a jelzett átlós irányban halad.

Leghosszabb közös részsorozat: példa

 Legyen két aminosavnak a szekvenciája H E A G A W G H E E és P A W H E A E, ahol A: Alanine, E: Glutamic acid, G: Glycine, H: Histidine, P: Proline és W: Tryptophan.

		Н	Ε	Α	G	Α	W	G	Н	Ε	E
) 3	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
Α	0	0	0	1	1	1	1	1	1	1	1
w	0	0	0	1	1	1	2	2	2	2	2
Н	0	1_	1	1	1	1	2	2	3	3	3
E	0	1	2	2	2	2	2	2	3	4	4
A	0	1	2	3	3	3	3	3	3	4	4
E	0	1	2	3	3	3	3	3	3	4	5

- Az LCS: A W H E E.
- Feladat: írjon algoritmust az LCS kiolvasására

Az általános dinamikus programozási technika

- Általában olyan feladatoknál alkalmazzuk, amelyek első ránézésre rengeteg időt vesznek igénybe.
 Részei:
 - Egyszerű részproblémák: a részproblémákat néhány változó függvényeként kell definiálni.
 - Részprobléma optimalitás: a globális optimum a részproblémák optimumaként definiálható.
 - Átfedő, kapcsolódó részproblémák: a részfeladatok nem függetlenek,
 hanem átfedőek (ezért bottom-up konstrukcióban kell feldolgozni).

Megoldás során:

- Rekurziós összefüggés felírása
- Az összefüggés elemzése. Tömbre átírás (a változók diszkrétek, korlátozottak)
- A tömbben a kitöltési irány meghatározása (minden elem esetén definiált legyen az összes olyan elem értéke, amit használ)
- Végső megoldás helyének megadása
- Ha kell, optimális megoldás visszakeresése
- Ellenkező esetben memóriaigény csökkentése (ha lehet)

Felhasznált és javasolt irodalom

• [1] A. Grama, A. Gupta, G. Karypis, V. Kumar Introduction to Parallel Computing

Addison-Wesley, ISBN 0-201-64865-2, 2003, 2nd ed., angol, 636 o.