Dinamikus programozás – „Oszd meg, és uralkodj!” – Mohó stratégia

Melyiket válasszuk?

Dinamikus programozás vagy „Oszd meg, és uralkodj!”

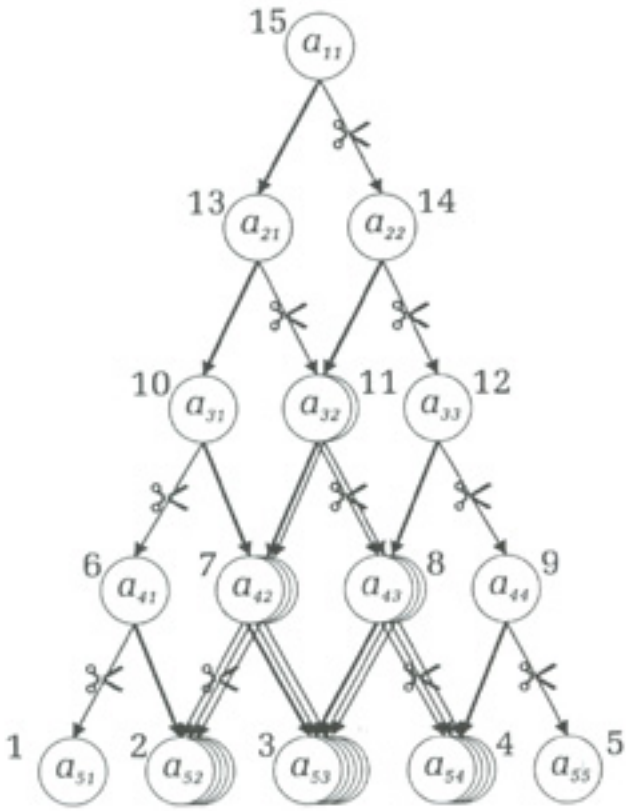
Háromszögfeladat rekurzívan:

$$
\begin{aligned}
c_{nj} &:= a_{nj} && \textbf{ha } 1 \le j \le n \\
c_{ij} &:= a_{ij} + \max\{c_{i+1,j},\, c_{i+1,j+1}\}, && \textbf{ha } n > i \ge 1,\ 1 \le j \le i
\end{aligned}
$$

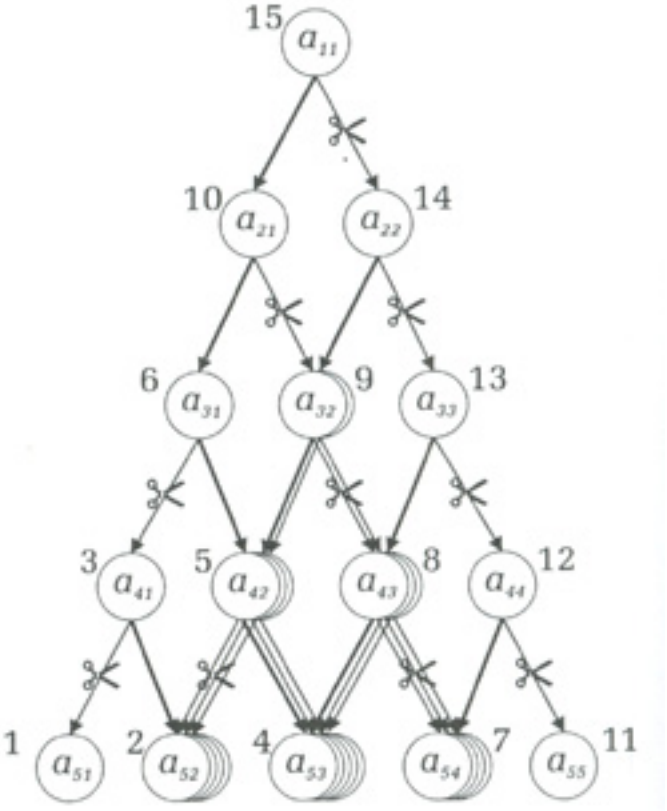
HSZ-REK(i, j)

1. **if** $c_{i,j} = -1$
2. **then if** $i = n$
3. **then** $c_{i,j} := a_{i,j}$
4. **else** $c_{i,j} := a_{i,j} + \max\Big(\text{HSZ-REK}(i+1, j),\ \text{HSZ-REK}(i+1, j+1)\Big)$
5. **return** $c_{i,j}$

Kezdetben minden $c_{ij} := -1$, **az eljárás hívása** $k :=$ HSZ-REK$(1, 1)$.



nem rekurzív



rekurzív

Mátrixszorzás rekurzívan

$$c_{ij} = \begin{cases} 0 & \textbf{ha } i = j \\ \min\left\{c_{ik} + c_{k+1,j} + d_{i-1}d_k d_j\right\} & \textbf{ha } n > i > j,\ i \le k < j \le n. \end{cases}$$

M-REK(i, j)

```
1.  if c_{i,j} = -1
2.     then c_{i,j} := ∞
3.          for k := i to j - 1
4.              do t := M-REK(i, k) + M-REK(k + 1, j) + d_{i-1} d_k d_j
5.                 if t < c_{i,j}
6.                    then c_{i,j} := t
7.                         c_{j,i} := k
8.  return c_{i,j}
```

Hívás előtt minden c_{ij}-t a főátló fölött -1-re állítunk, a főátló elemeit pedig 0-ra. A hívás $k :=$ M-REK$(1, n)$. A zárójelezés ugyanúgy történik.

Szövegátalakítás

$a_1a_2\ldots a_n$ **szöveg átalakítása** $b_1b_2\ldots b_m$ **szöveggé minimális művelettel: betűcsere, betű beillesztrése, betű törlése.**

c_{ij} **minimális műveletszám (távolság)** $a_1a_2\ldots a_i$ **átalakítására** $b_1b_2\ldots b_j$**-vé.**

$$c_{ij} = \begin{cases} i & \textbf{ha } i = \overline{0,n}, j = 0 \\ j & \textbf{ha } i = 0, j = \overline{1,m} \\ \min\Big(c_{i-1,j-1} + [a_i \neq b_j], & \\ \quad c_{i-1,j} + 1, & \\ \quad c_{i,j-1} + 1\Big) & \textbf{ha } i = \overline{1,n}, j = \overline{1,m} \end{cases}$$

ahol $[feltétel] = \begin{cases} 1 & \textbf{ha } feltétel \textbf{ igaz} \\ 0 & \textbf{ha } feltétel \textbf{ hamis} \end{cases}$

iteratív megoldás

ITERATÍV(a, b)

```
1.  n := hossz[a]
2.  m := hossz[b]
3.  for i := 0 to n
4.       do c_{i,0} := i
5.  for j := 0 to m
6.       do c_{0,j} := j
7.  for i := 1 to n
8.       do for j := 1 to m
9.                do x := c_{i-1,j-1}
10.                  if a_i ≠ b_j
11.                     then x := x + 1
12.                  c_{i,j} := min(x, c_{i-1,j} + 1, c_{i,j-1} + 1)
13. return C
```

rekurzív megoldás

TÁVOLSÁG(i, j)

1. **if** $c_{i,j} = -1$
2. **then if** $i = 0$
3. **then** $c_{i,j} := j$
4. **if** $j = 0$
5. **then** $c_{i,j} := i$
6. **if** $(i > 0)$ **és** $(j > 0)$
7. **then** $x :=$ TÁVOLSÁG$(i-1, j-1)$
8. **if** $a_i \neq b_j$
9. **then** $x := x + 1$
10. $c_{i,j} := \min\Big(x,\ $TÁVOLSÁG$(i-1, j) + 1,$ TÁVOLSÁG$(i, j-1) + 1\Big)$
11. **return** $c_{i,j}$

Hívás: minden i és j értékre beállítjuk: $c_{i,j} := -1$. Majd: $k :=$ TÁVOLSÁG(n, m).

Például:

$a =' POLINOMIÁLIS'$

$b =' EXPONENCIÁLIS'$

C:

0	**1**	**2**	3	4	5	6	7	8	9	10	11	12	13
1	1	2	**2**	3	4	5	6	7	8	9	10	11	12
2	2	2	3	**2**	3	4	5	6	7	8	9	10	11
3	3	3	3	3	**3**	4	5	6	7	8	8	9	10
4	4	4	4	4	4	**4**	5	6	6	7	8	8	9
5	5	5	5	5	4	5	**4**	5	6	7	8	9	9
6	6	6	6	5	5	5	**5**	5	6	7	8	9	10
7	7	7	7	6	6	6	6	**6**	6	7	8	9	10
8	8	8	8	7	7	7	7	7	**6**	7	8	8	9
9	9	9	9	8	8	8	8	8	7	**6**	7	8	9
10	10	10	10	9	9	9	9	9	8	7	**6**	7	8
11	11	11	11	10	10	10	10	10	9	8	7	**6**	7
12	12	12	12	11	11	11	11	11	10	9	8	7	**6**

$-$	$-$	P	O	L	I	N	O	M	I	$Á$	L	I	S
|	|			|	|		|	|				
E	X	P	O	N	E	N	$-$	C	I	$Á$	L	I	S

Haladás a C mátrixban:

- átlósan: vagy csere vagy marad
- lefele: törlés
- vízszintesen: beszúrás

NYOMKÖVETÉS(i, j)

```
1.  if (i > 0) és (j > 0)
2.     then d := c_{i-1,j-1}
3.          if a_i ≠ b_j
4.             then d := d + 1
5.          if c_{i,j} = d
6.             then NYOMKÖVETÉS(i − 1, j − 1)
7.             else if c_{i,j} = c_{i-1,j} + 1
8.                     then NYOMKÖVETÉS(i − 1, j)
9.                     else NYOMKÖVETÉS(i, j − 1)
10.    else if i > 0
11.            then NYOMKÖVETÉS(i − 1, j)
12.    else if j > 0
13.            then NYOMKÖVETÉS(i, j − 1)
14. kiír i, j
```

Hívás: NYOMKÖVETÉS(n, m)

Eredmény példánkban: $(0,0)$, $(0,1)$, $(0,2)$, $(1,3)$, $(2,4)$, $(3,5)$, $(4,6)$, $(5,7)$, $(6,7)$, $(7,8)$, $(8,9)$, $(9,10)$, $(10,11)$, $(11,12)$, $(12,13)$

Egyéb feladatok, amelyek inkább dinamikus megközelítésűek, mint "oszd meg, és uralkodj" típusúak.

- Kombinációk

$$C_n^k = \binom{n}{k} = \begin{cases} 1, & \textbf{ha } k = 0 \textbf{ vagy } k = n \\ C_{n-1}^k + C_{n-1}^{k-1} & \textbf{ha } 0 < k < n. \end{cases}$$

KOMBINÁCIÓ(n, k)

1. if $c_{nk} = -1$
2. then if $(k = 0)$ vagy $(k = n)$
3. then $c_{nk} := 1$
4. else $c_{nk} :=$ KOMBINÁCIÓ$(n-1, k)$ + KOMBINÁCIÓ$(n-1, k-1)$
5. return c_{nk}

- Fibonacci-számok
- Ackermann-függvény

Egér és sajt a labirintusban

Az egértől szélességi bejárást folytatunk, amíg eljutunk a sajthoz. A C mátrixban őrizzük a távolságokat. A Q sorban őrizzük a szomszédok koordinátáit.
Az $m \times n$-es labirintus falai 1-esek, a folyosók 0-k. Az egér kordinátái: (xe, ye), a sajt koordinátái: (xs, ys).

EGÉR-SAJT

```
1.  for i := 1 to m
2.      do for j := 1 to n
3.             do if a_{i,j} = 0
4.                    then c_{i,j} := -1
5.  c_{xe,ye} := 0
6.  BETESZ(xe, ye)
7.  while (Q nem üres)
8.        do KIVESZ(x, y)
9.           for ∀(i, j) szomszédjára (x, y)-nak
10.             if (a_{ij} = 0) és (c_{ij} = -1)
11.                then c_{i,j} := c_{x,y} + 1
12.                     BETESZ(i, j)
13.                     if (i = xs) és (j = ys)
14.                     then exit while
15. if Q üres
16.    then return Nincs megoldás
17.    else  return c_{xs,ys}
```

$m = 6, n = 7$

0	0	0	0	0	0	0
s	1	0	0	1	0	0
0	1	0	0	0	0	0
0	0	0	1	0	1	0
0	0	e	0	0	0	0
0	0	0	0	0	0	0

Legrövidebb út hossza: 5.

-1	5	4	5	-1	-1	-1
s	0	3	4	0	-1	-1
4	0	2	3	4	5	-1
3	2	1	0	3	0	-1
2	1	e	1	2	3	4
3	2	1	2	3	4	5

Mohó stratégia vagy dinamikus programozás

A hátizsák kapacitása G, ezt kell megtölteni a $g_1, g_2, \ldots, g_n$ tömegű, $a_1, a_2, \ldots, a_n$ értékű (árú) tárgyakkal, hogy ne szakadjon el, és az összérték maximális legyen.

c_i $(i = 1, 2, \ldots, G)$ őrzi az i tömeggel megtöltött zsák értékét.
d_{ij} $(i = 0, 1, 2, \ldots, G, j = 1, 2, \ldots, n)$ őrzi az i tömeggel megtöltött zsák esetében az optimális megoldást ($d_{ij} = 1$, ha a j-edik tárgy benne van zsákban, és 0 különben).

Kezdetben minden $c_i = -1$ $(i = 1, 2, \ldots, G)$ és $d_{ij} = 0$ $(i = 0, 1, 2, \ldots, G, j = 1, 2, \ldots, n)$.

Az i-edik tárgy akkor kerül a megoldásba, ha nélküle az $s - g_i$ tömeg esetében optimális részmegoldás van.

Hátizsák

```
0.  c_0 = 0
1.  for s := 1 to G
2.      do for i := 1 to n
3.             do if s − g_i ≥ 0
4.                    then if (c_{s−g_i} ≠ −1) és (d_{s−g_i,i} = 0)
                                  és (c_{s−g_i} + a_i > c_s)
5.                         then c_s := c_{s−g_i} + a_i
6.                              k := i
7.         if c_s ≠ −1
8.            then for j := 1 to n
9.                    do d_{s,j} := d_{s−g_k,j}
10.                d_{s,k} := 1
11. return c, d
```

Eredmény: a legnagyobb $c[s]$ és d-nek a s-edik sora.

Példa:

súlyok:	5	4	5	3	4	$G = 15$
értékek (árak):	10	20	6	3	30	
optimum:	60					
megoldás:	(1,1,0,0,1)					

A d mátrix és a c vektor elemei:

d	1	2	3	4	5	c elemei
0	0	0	0	0	0	0
1	0	0	0	0	0	−1
2	0	0	0	0	0	−1
3	0	0	0	1	0	3
4	0	0	0	0	1	30
5	1	0	0	0	0	10
6	1	0	0	0	0	−1
7	0	0	0	1	1	33
8	0	1	0	0	1	50
9	1	0	0	0	1	40
10	1	0	1	0	0	16
11	0	1	0	1	1	53
12	1	0	0	1	1	43
13	1	1	0	0	1	60
14	1	0	1	0	1	46
15	0	0	1	0	0	−1

Megoldás:
Max. = 60
1 1 0 0 1

Az ECN programozási verseny 2009 egyik feladata

Legkevesebb palindrom

Adott egy $a_1 a_2 \dots a_n$ füzér. Bontsuk fel minimális számú palindrommá!

Legyen c_{ij} az $a_i a_{i+1} \dots a_j$ füzér palindromjainak minimális száma.

Ekkor $C = (c_{ij})$ ahol :

$$c_{ij} = \begin{cases} 1, & \textbf{ha } i \geq j \\ 1, & \textbf{ha } a_i = a_j \textbf{ és } c_{i+1,j-1} = 1 \\ \min(c_{i,k} + c_{k+1,j}), & \textbf{különben} \end{cases}$$

c_{1n} megadja a palindromok minimális számát

Például:

$aaba$ felbontása: a, aba

$$C = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$abaabbaaaa$ **felbontása:** $aba, abba, aaa$

$$C = \begin{pmatrix} 1 & 2 & 1 & 2 & 2 & 3 & 2 & 3 & 3 & 3 \\ 1 & 1 & 2 & 2 & 1 & 2 & 3 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

MinPalindrom(a)

1. $n := hossz[a]$
2. **for** $i := 1$ **to** n
3. **do for** $j := 1$ **to** i
4. **do** $c_{i,j} := 1$
5. **for** $i := n - 1$ **downto** 1
6. **do for** $j := i + 1$ **to** n
7. **do if** $(a_i = a_j)$ **és** $(c_{i+1,j-1} = 1)$
8. **then** $c_{ij} := 1$
9. **else** $c_{ij} := \min_{k=i,\ldots,j-1} \left(c_{i,k} + c_{k+1,j} \right)$
10. **return** C

A palindromok meghatározása

PALINDROMOK(a, C)

0. $n := hossz[a]$
1. $i := 1$
2. **while** $i \leq n$ és $c_{in} \neq 1$
3. **do** $k := i$
4. **while** $c_{i,n} \neq c_{i,k} + c_{k+1,n}$
5. **do** $k := k + 1$
6. kiír $a_i a_{i+1} \ldots a_k$
7. $i := k + 1$
8. **if** $i \leq n$
9. **then** kiír $a_i a_{i+1} \ldots a_n$
10. **return**

Keressük meg az összes megoldást!

abbcbbcbbbcbb

A C mátrix:

1	2	2	3	3	2	3	3	2	3	3	3	4
1	1	1	2	2	1	2	2	1	2	2	2	3
1	1	1	2	1	2	2	1	2	2	3	2	2
1	1	1	1	2	2	1	2	2	2	3	3	2
1	1	1	1	1	1	2	2	1	2	2	2	1
1	1	1	1	1	1	2	1	2	2	2	1	2
1	1	1	1	1	1	1	2	2	2	1	2	2
1	1	1	1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	1	1	1	2	2	1
1	1	1	1	1	1	1	1	1	1	2	1	2
1	1	1	1	1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

Összes felbontás:

a b bcbbcb bbcbb
a bb c bbcbbbcbb
a bbcbb cbbbc bb
a bbcbbcbb bcb b