

CURS 13 (ampl.)

METODA PROGRAMĂRII DINAMICE ♥

R. Bellman ≈ 50

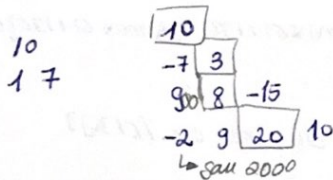
Condiții:

C1: Condiția de subproblemă / substructură optimă = pb. se poate descompune în mai multe subp., iar sol. sa se obț. combinând sol. sub. sale.
 optim global optime locale

C2: Condiția de superpozabilitate: (sol.) subps. se suprapun

MEMOIZARE = str. de date adăugate în care mem. sol. sub. pt. a evita calculul inutil de recursiv

1. Suma maximă într-un triunghi de numere



$$T_{i,j} \rightarrow \begin{cases} T_{i+1,j} \\ T_{i+1,j+1} \end{cases}$$

Vari. Greedy: rezolvă cu 900 sau 2000

Vari. Backtracking: 0,0,0,...,0 ← a trasee că parcurge Backul

0,0,0,...,1

0,0,0,...,1,0

0,0,0,...,1,1

...

1,1,1,...,1

$n-1$ valori binare = 2^{n-1} șiruri ! ca la ASE

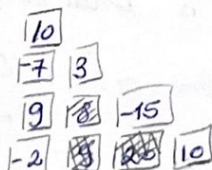
Varianta D&I:

$$s_{\max}(T, i, j) = \begin{cases} T[i][j] + \max\{s_{\max}(T, i+1, j), s_{\max}(T, i+1, j+1)\} & \text{dacă } 0 \leq i = n-2 \\ T[i][j], & \text{dacă } i = n-1 \end{cases}$$

↓ memoizate
 $s_{\max}(T, i, j)$

000
 001
 010
 011
 100
 101
 111
 $\left. \vphantom{\begin{matrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 111 \end{matrix}} \right\} 2^3 = 8 \checkmark$

subps. se suprapun



face fix ce face și backtrackingul
 deci 2^{n-1}

$$s_{\max}[i][j] = -1 \text{ dacă cu } s_{\max}[i+1][j] \text{ și } s_{\max}[i+1][j+1]$$

$$T = \begin{pmatrix} 10 \\ -7 & 3 \\ 9 & 8 & -15 \\ -2 & 9 & 20 & 10 \end{pmatrix} \quad s_{\max} = \begin{pmatrix} 41 \rightarrow \text{soluția} \\ +21 & 31 \\ 18 & 28 & +5 \\ -2 & 9 & 20 & 10 \end{pmatrix} \rightarrow \text{traseul e de sus în jos urmând maximul}$$

$$\text{Complexitate: } O(n^2) \text{ (adică } \frac{n^2}{2})$$

pe j în range (m) :

$$s_{\max}[n-1][j] = T[n-1][j]$$

pe i în range $(n-2, -1, -1)$:

pe j în range $(i+1)$:

$$s_{\max}[i][j] = T[i][j] + \max(s_{\max}[i+1][j], s_{\max}[i+1][j+1])$$

$s_{\max}[i][j]$ = suma maximă pe un traseu care se termină cu $T[i][j]$

$$\begin{pmatrix} 10 \\ 3 & 13 \\ 12 & 21 & -2 \\ 10 & 30 & 41 & 8 \end{pmatrix} \leftarrow \text{depinde de cele de înainte}$$

41
↳ Soluția

$$s_{\max}[i][j] = T[i][j] + \underbrace{s_{\max}(i+1, j), s_{\max}(i+1, j+1)}_{\text{încinte}} \text{ depinde de cele de înainte}$$

2. Subșirul cresc. maximal

$$L = [5, 9, 1, 7, 3, 5, 10, 8, 12, 4, 6]$$

$l_{\max}[i]$ = lungimea maximă a unui subșir cresc. care începe cu $L[i]$

$$l_{\max}[i] = \begin{cases} 1 + \max(l_{\max}[j]), & L[j] \geq L[i], \quad 0 \leq i \leq n-2 \\ 1, & i = n-1 \end{cases}$$

$i+1 \leq j \leq n-1$

def.

$f_{max} = [7, 3, 5, 3, 4, 3, 2, 2, 1, 2, 1]$ ← sol. este maximul din lista
succesi = $[1, 3, 5, 6, 8, 8, -1, 10, -1]$

mă duc pe pot. pe care le am ..