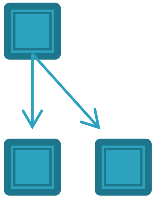
Trasee de sumă maximă

- Se consideră un triunghi de numere naturale **t** cu n linii.

Să se determine cea mai mare sumă pe care o putem forma dacă ne deplasăm în triunghi și adunăm numerele din celulele de pe traseu, regulile de deplasare fiind următoarele:

- pornim de la numărul de pe prima linie
- din celula (i,j) putem merge doar în (i+1,j) sau (i+1,j+1).



Să se indice și un traseu de sumă maximă

Trasee de sumă maximă

- **Exemplu**

1

6 **2**

1 2 **10**

3 4 **7** 2

Câte astfel de trasee există?

Trasee de sumă maximă

- **Exemplu**

```
1
6 2
1 2 10
3 4  7 2
```

- Se pot construi în total 2^{n-1} astfel de trasee (avem 2 opțiuni la fiecare pas)

Trasee de sumă maximă

- **Exemplu**

  1

  6 **2**

  1 2 **10**

  3 4 **7** 2

- **Greedy** – nu obţinem soluţia optimă

  1

  6 2

  1 **2** 10

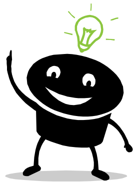
  3 4 **7** 2

Trasee de sumă maximă

- Unde este mai bine să facem primul pas din traseu, pornind din (1,1): în (2,1) sau (2,2)?

Trasee de sumă maximă

- **Unde este mai bine să facem primul pas din traseu, pornind din (1,1): în (2,1) sau (2,2)?**



Dacă am ști deja care este suma maximă care se poate obține pornind din (2,1) respectiv din (2,2), am ști unde să facem primul pas.

$\Rightarrow$ Subprobleme utile:

Suma maximă care se poate obține pornind din celula (i,j) – verifică principiu de optimalitate

Trasee de sumă maximă

- **Principiu de optimalitate:**

  Dacă

  $(i_1, j_1), (i_2, j_2), \dots ,(i_n, j_n)$

  este un traseu optim care **începe** din celula (i_1, j_1), atunci

  $(i_2, j_2), \dots ,(i_n, j_n)$

  este un traseu optim **dacă pornim din celula (i_2, j_2)**

Subşir crescător de lungime maximă

Principiu de optimalitate

↓

Subprobleme:

Calculăm pentru o poziție (i, j) suma maximă pe care o putem obține **dacă pornim din celula (i, j)**

↓

Subproblemele se suprapun
memorăm rezultatele într-o matrice

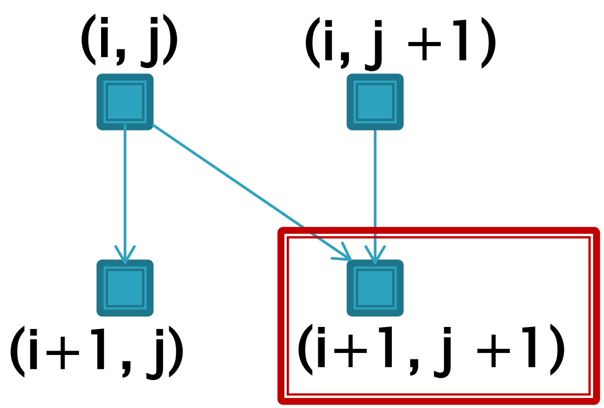
Trasee de sumă maximă

- **Subprobleme:**

Calculăm pentru o poziție (i, j) suma maximă pe care o putem obține **dacă pornim din celula (i, j)**

$$\text{suma}(i,j) = t[i][j] + \max(\text{suma}(i+1, j), \text{suma}(i+1, j+1))$$

Subproblemele se suprapun, o soluție recursivă fără memiozare (tip “Divide et Impera”) este ineficientă



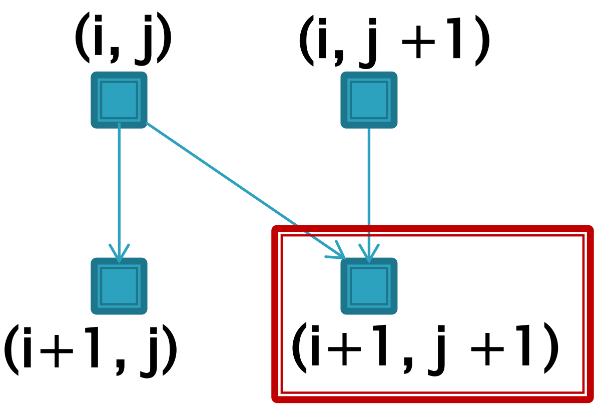
Trasee de sumă maximă

- **Subprobleme:**

Calculăm pentru o poziție (i, j) suma maximă pe care o putem obține **dacă pornim din celula (i, j)**

$$\text{suma}(i,j) = t[i][j] + \max(\text{suma}(i+1, j), \text{suma}(i+1, j+1))$$

Subproblemele se suprapun, o soluție recursivă fără memiozare (tip “Divide et Impera”) este ineficientă



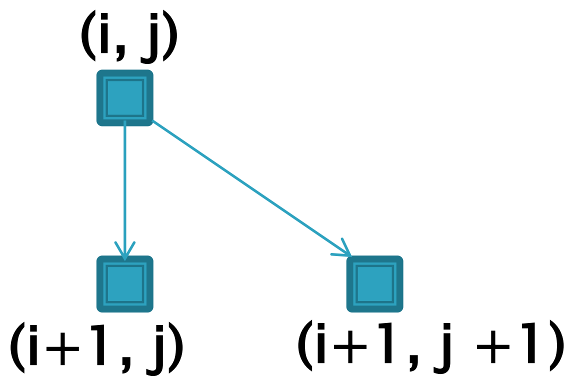
Memorăm rezultatele subproblemelor într-o matrice

Trasee de sumă maximă

- **Subprobleme:**

`s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

$$s[i][j] = t[i][j] + \max(s[i+1][j], s[i+1][j+1])$$



Trasee de sumă maximă

- **Subproblemă:**

`s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

Trasee de sumă maximă

- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă**

Trasee de sumă maximă

- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă** `s[1][1]`

Trasee de sumă maximă

- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă** `s[1][1]`
- **Știm direct**

Trasee de sumă maximă

- Subproblemă:

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- Soluție problemă `s[1][1]`

- Știm direct `s[n][j] = t[n][j], j=1,2,…,n`

Trasee de sumă maximă

- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă** `s[1][1]`
- **Știm direct** `s[n][j] = t[n][j], j=1,2,…,n`
- **Relație de recurență**

`s[i][j] = t[i][j]+ max{s[i+1][j],s[i+1][j+1]}`

Trasee de sumă maximă

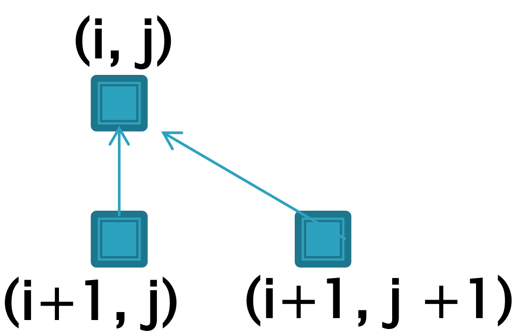
- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă** `s[1][1]`
- **Ştim direct** `s[n][j] = t[n][j], j=1,2,…,n`
- **Relație de recurență**

`s[i][j] = t[i][j]+ max{s[i+1][j],s[i+1][j+1]}`

- **Ordinea de rezolvare a recurenţelor**



Trasee de sumă maximă

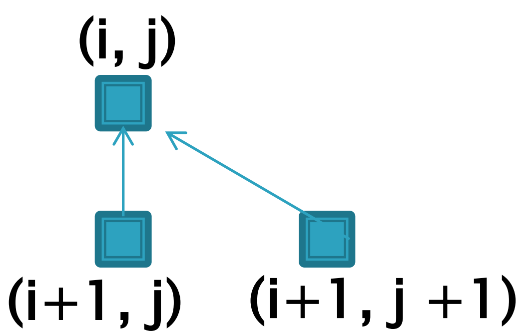
- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține dacă **pornim din celula (i, j)**

- **Soluție problemă** `s[1][1]`
- **Știm direct** `s[n][j] = t[n][j], j=1,2,…,n`
- **Relație de recurență**

`s[i][j] = t[i][j]+ max{s[i+1][j],s[i+1][j+1]}`

- **Ordinea de rezolvare a recurenţelor**



de la ultima linie către prima

Trasee de sumă maximă

- **Pentru a memora și un traseu**

  `u[i][j]` = coloana pe care ne deplasăm din celula (i, j) pe linia i+1 într-un traseu optim

sau

reconstituim traseul folosind relația de recurență = ne deplasăm mereu în celula permisă de pe linia următoare cu s (!!nu t) maxim

Trasee de sumă maximă

- **Exemplu**

```
1
6 2
1 2 10
3 4  7 2                3   4   7   2
    t                       s
```

Trasee de sumă maximă

- **Exemplu**

1	
6 2	
1 2 10	5
3 4 7 2	3 4 7 2
t	**s**

Trasee de sumă maximă

- **Exemplu**

```
1
6 2
1 2 10
3 4  7 2
    t
```

```
5   9
3   4   7   2
    s
```

Trasee de sumă maximă

- **Exemplu**

1								
6	2							
1	2	10			5	9	17	
3	4	7	2		3	4	7	2
	t					**s**		

Trasee de sumă maximă

- **Exemplu**

```
1
6 2                         15 19
1 2 10                      5   9 17
3 4  7 2                    3   4   7   2
      t                          s
```

Trasee de sumă maximă

- **Exemplu**

t					s			
1					20			
6	2				15	19		
1	2	10			5	9	17	
3	4	7	2		3	4	7	2

Trasee de sumă maximă

- **traseu**

1	**20**
6 2	15 19
1 2 10	5 9 17
3 4 7 2	3 4 7 2
t	**s**

Trasee de sumă maximă

- **traseu**

1			
6	2		
1	2	10	
3	4	7	2

t

20			
15	**19**		
5	9	17	
3	4	7	2

s

Trasee de sumă maximă

- **traseu**

```
1              20
6 2            15 19
1 2 10         5  9  17
3 4  7 2       3  4  7  2
   t              s
```

Trasee de sumă maximă

- **traseu**

1			
6	2		
1	2	10	
3	4	7	2

t

20			
15	**19**		
5	9	**17**	
3	4	**7**	2

s

Trasee de sumă maximă

- Implementare – v.sursa py
- Complexitate – $O(n^2)$

Trasee de sumă maximă

- Principiu de optimalitate – <u>Altă variantă</u>

Trasee de sumă maximă

- **Principiu de optimalitate – <u>Altă variantă</u>**

Dacă

$$(i_1=1,\ j_1=1),\ (i_2,\ j_2),\ \ldots\ ,(i_n,\ j_n)$$

este un traseu optim atunci

$$(i_1,\ j_1),\ \ldots\ ,(i_k,\ j_k)$$

este un traseu optim ...

Trasee de sumă maximă

- **Principiu de optimalitate – <u>Altă variantă</u>**

  Dacă

  $$(i_1=1, j_1=1), (i_2, j_2), \dots ,(i_n, j_n)$$

  este un traseu optim atunci

  $$(i_1, j_1), \dots ,(i_k, j_k)$$

  este un traseu optim **pentru a ajunge** în celula (i_k, j_k) pornind din (i_1, j_1)

- **Subproblemă:**

  Calculăm pentru o poziție (i, j) suma maximă pe care o putem obține dacă **ajungem în celula (i, j) pornind din (1,1)**

Trasee de sumă maximă

- **Subproblemă:**

  `s[i][j]` = suma maximă pe care o putem obține **pornind din (1,1)** și ajungând în **celula (i, j)**

- **Soluție problemă**
- **Ştim direct**
- **Relație de recurență**
- **Ordinea de calcul** – **temă**