

# Metoda Programării Dinamice

-3-

# Problema discretă a rucsacului



Se consideră un rucsac de capacitate (greutate) maximă  $G$  (număr natural) și  $n$  obiecte caracterizate prin:

- greutatea lor (numere naturale)  $g_1, \dots, g_n$ ;
- câștigurile  $v_1, \dots, v_n$  obținute la încărcarea lor în totalitate în rucsac.

Un obiect **nu poate fi fracționat**.

Se cere o modalitate de încărcare de obiecte în rucsac, astfel încât câștigul total să fie maxim.

# Problema discretă a rucsacului

## Caz particular

Date  $n$  obiecte cu ponderile  $w_1, w_2, \dots, w_n$  și o limită  $W$ , să se selecteze o submulțime de obiecte cu suma ponderilor maxime, fără a depăși însă ponderea  $W$



# Problema discretă a rucsacului

## Caz particular

Date  $n$  obiecte cu ponderile  $w_1, w_2, \dots, w_n$  și o limită  $W$ , să se selecteze o submulțime de obiecte cu suma ponderilor maxime, fără a depăși însă ponderea  $W$

## Interpretări

- Submulțime de sumă maximă mai mică sau egală cu o valoare  $M$  dată (v. Greedy)
- $n$  activități cu duratele  $w_1, w_2, \dots, w_n$  necesită o resursă. Știind că timpul maxim de funcționare a resursei este  $W$ , să se selecteze o submulțime de activități care țin resursa ocupată un timp cât mai lung (maxim)

# Problema discretă a rucsacului

## Exemplu:

$G = 8$

$n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

# Problema discretă a rucsacului

## Exemplu:

$G = 8$

$n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

**Greedy** – în ordinea descrescătoare a raportului  $v/g$

- Alege întâi obiectul 4 de greutate 6
- Nu se mai poate pune nici un alt obiect întreg în rucsac
- Câștigul Greedy: 18

# Problema discretă a rucsacului

## Exemplu:

$$G = 8$$

$$n = 4 \text{ obiecte}$$

$$g: \quad 3 \quad 4 \quad 4 \quad 6$$

$$v: \quad 3 \quad 9 \quad 10 \quad 18$$

**Greedy** – în ordinea descrescătoare a raportului  $v/g$

- Alege întâi obiectul 4 de greutate 6
- Nu se mai poate pune nici un alt obiect întreg în rucsac
- Câștigul Greedy: 18

**Soluția optimă:**

- Alegem obiectele 2 și 3
- Câștigul total  $10 + 9 = 19$

# Problema discretă a rucsacului

## ► Principiu de optimalitate

Dacă  $S$  este soluție optimă pentru greutatea  $g$  și obiectele  $\{1, 2, \dots, n\}$  care

- conține  $n$
- nu conține  $n$



# Problema discretă a rucsacului

## ► Principiu de optimalitate

Dacă  $S$  este soluție optimă pentru greutatea  $g$  și obiectele  $\{1, 2, \dots, n\}$  care

- **conține  $n$**  atunci  $S - n$  este soluție optimă pentru greutatea  $g - g_n$  și obiectele  $\{1, 2, \dots, n-1\}$
- **nu conține  $n$**  atunci  $S$  este soluție optimă pentru greutatea  $g$  și obiectele  $\{1, 2, \dots, n-1\}$

# Problema discretă a rucsacului

- ▶ Subproblemă
- ▶ Soluție
- ▶ Știm direct
- ▶ Relație de recurență
- ▶ Ordinea de calcul
- ▶ Afișarea obiectelor din soluția optima

# Problema discretă a rucsacului

```
for(int g = 0; g<= G;g++)    c[0][g]= 0;

for(int i = 1; i<= n;i++){
    c[i][0]=0;
    for(int gr = 1;gr<= G;gr++){
        if (g[i]<= gr)
            if (v[i]+c[i-1][gr-g[i]]>c[i-1][gr])
                c[i][gr]=v[i]+c[i-1][gr-g[i]];
            else
                c[i][gr]=c[i-1][gr];
        else
            c[i][gr]=c[i-1][gr];
    }
}

System.out.println("Castigul total "+c[n][G]);
```

# Problema discretă a rucsacului

- ▶ **Afișarea obiectelor**

- din relația de recurență

# Problema discretă a rucsacului

## Afișarea obiectelor – recursiv

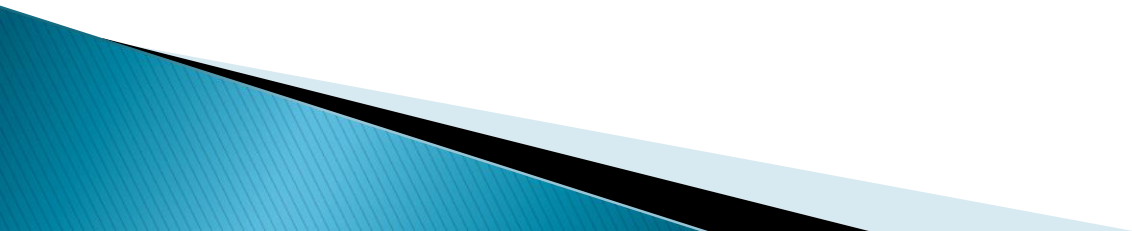
```
void afis(int i,int gr){
    if(i==0 || gr==0)
        return;
    if((g[i]<=gr)&&(c[i][gr]==v[i]+c[i-1][gr-g[i]])){
        afis(i-1,gr-g[i]);
        System.out.print(i+" ");
    }
    else
        afis(i-1,gr);
}

afis(n,G);
```

# Problema discretă a rucsacului

## Afișarea obiectelor – nerecursiv

```
gr = G;
i = n;
while (gr > 0 && i > 0) {
    if ((g[i] <= gr) && (c[i][gr] == v[i] + c[i-1][gr - g[i]])) {
        System.out.print(i + " ");
        gr = gr - g[i];
    }
    i--;
}
```



# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$c:$

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0									
0									
0									
0									

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$c:$

	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0
	0	0	0	3					
	0								
	0								
	0								



# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$c:$

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	3	3	3	3
0									
0									
0									

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$C:$

0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0
0	0	0	3	3	3	3	3	3
0	0	0	3	9				
0								
0								

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$C:$

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	3	3	3	3
0	0	0	0	3	9	9	9	12	
0									
0									

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

C:

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

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g$ : 3 4 4 6

$v$ : 3 9 10 18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$C$ :

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

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

$C:$

0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0
0	0	0	3	3	3	3	3	3
0	0	0	3	9	9	9	12	12
0	0	0	3	10	10	10	13	19
0	0	0	3	10	10	18	18	19

# Problema discretă a rucsacului

► **Exemplu**  $G = 8$ ,  $n = 4$  obiecte

$g:$     3    4    4    6

$v:$     3    9    10    18

$$c[i][g] = \begin{cases} c[i-1][g], & \text{daca } g_i > g \\ \max\{v_i + c[i-1][g-g_i], c[i-1][g]\}, & \text{altfel} \end{cases}$$

**Traseu:**

	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0
	0	0	0	3	3	3	3	3	3
C:	0	0	0	3	9	9	9	12	12
	0	0	0	3	10	10	10	13	19
	0	0	0	3	10	10	18	18	19

# Problema discretă a rucsacului

- ▶  $O(nG)$



# Alte aplicații

## ► **Distanțe de editare. Alinierea secvențelor**

Putem măsura similaritatea între secvențe (ADN) prin

- **Elemente comune** – cel mai lung subșir comun pentru două secvențe
  - **Distanțe de editare** – numărul minim de inserări și modificări (eventual și ștergeri) de caractere necesar pentru transforma prima secvență în cea de a doua
- + aplicații în procese de căutare de cuvinte – sugestii de cuvinte similare

# Alinierea secvențelor

## ▶ Exemplu

**Aliniere** – punerea pozițiilor (caracterelor) din cele două secvențe a și b în corespondență 1 la 1, cu posibilitatea de a insera spații (păstrând ordinea literelor)

a = AGGGCT      b = AGGCA

AGGGCT

AGG-CA

**penalizarea** = penalizarea spațiului + penalizarea pentru diferența T/A

# Alinierea secvențelor

## ▶ Exemplu

**Aliniere** – punerea pozițiilor (caracterelor) din cele două secvențe a și b în corespondență 1 la 1, cu posibilitatea de a insera spații (păstrând ordinea literelor)

**a = AGGGCT      b = AGGCA**

**AGGGCT**

**AGG-CA**

**penalizarea** = penalizarea spațiului + penalizarea pentru diferența T/A

sau

**AGGGCT-**

**AGG-C-A**

**penalizarea** = 3\*penalizarea spațiului

# Alinierea secvențelor

Date două secvențe,  $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$  și  $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\ldots\mathbf{y}_m$

aliniem secvențele inserând în ele caracterul ‘ ‘ astfel încât secvențele să devină de aceeași lungime și penalizând pozițiile pe care diferă secvențele obținute.

## ► Formulare echivalentă:

**Aliniere** = formarea de perechi  $(x_i, y_j)$  astfel încât fiecare caracter apare în cel mult o pereche și nu există perechi încrucișate:

– dacă avem perechile  $(x_i, y_j)$  și  $(x_k, y_t)$  și  $i < k \Rightarrow j < t$

AGGGCT

AGG-CA

# Alinierea secvențelor

**Scorul (penalizarea) alinierii** = suma penalizărilor alinierilor de caractere diferite și alinierilor caracter-spațiu (**scorul Needleman-Wunsch**).

- ▶ Penalizări diferite pentru diferențe de litere, spațiu (de exemplu diferența A-G poate fi mai grava decât A-T)
- ▶ Notatii:
  - $p_{\text{spatiu}}$
  - $p_{XY}$  – penalizarea alinierii caracterului X cu caracterul Y

# Alinierea secvențelor

- ▶ ADN – alfabet A,C,G,T
- ▶ **Asemănări ADN** – poate semnifica apropiere în arborele genealogic
- ▶ **Esențial să fie rapizi** – se aplică pentru volum mare de date

# Alinierea secvențelor

## ▶ Principiu de optimalitate:

$x_1 x_2 \dots x_n$

$y_1 y_2 \dots y_m$

alinieare cu penalizare minimă

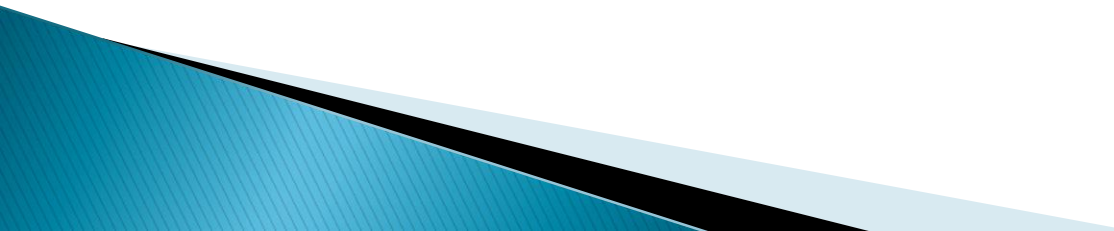
Evidențiem ultima pereche din aliniere



### Cazuri

- $x_n$  aliniat cu  $y_m$
- $x_n$  aliniat cu spațiu
- $y_m$  aliniat cu spațiu

# Alinierea secvențelor

- ▶ Subprobleme
  - ▶ Știm direct
  - ▶ Relație de
  - ▶ Ordinea de calcul
  - ▶ Determinarea unei soluții
- 



# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**                      **TCAG-**      **scor 6**

C:

	0	1	2	3	4
0	0	2	4	6	8
2					
4					
6					
8					

$j * p_{\text{spațiu}}$

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**                      **TCAG-**      **scor 6**

	0	1	2	3	4	
	0	2	4	6	8	<b>G</b>
<b>C:</b>	2	1				<b>T</b>
	4					
	6					
	8					

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**                      **TCAG-**      **scor 6**

	0	1	2	3	4	
	0	2	4	6	8	G
C:	2	1	3			TC
	4					
	6					
	8					

aliniem G cu C, rămâne de aliniat secvența vidă cu T cu cost  $c[0][1]$   
aliniem - cu C, rămâne de aliniat secvența G cu T cu cost  $c[1][1]$   
aliniem G cu -, rămâne de aliniat secvența vidă cu TC cu cost  $c[0][2]$

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**              **TCAG-**      **scor 6**

	0	1	2	3	4	
	0	2	4	6	8	<b>G</b>
<b>C:</b>	2	1	3	5		<b>TCA</b>
	4					
	6					
	8					

alinie G cu A, rămâne de aliniat secvența vidă cu TC cu cost  $c[0][2]$

alinie - cu A, rămâne de aliniat secvența G cu TC cu cost  $c[1][2]$

alinie G cu -, rămâne de aliniat secvența vidă cu TCA cu cost  
 $c[0][3]$

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**                      **TCAG-**      **scor 6**

C:

	0	1	2	3	4
0	0	2	4	6	8
2	2	1	3	5	6
4	4	3	2	3	5
6	6	4	4	5	4
8	8	6	4	5	6

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC**      **->**      **G-ATC**  
**TCAG**                      **TCAG-**      **scor 6**

**Soluția:**

C:

	0	1	2	3	4
0	0	2	4	6	8
2	2	1	3	5	6
4	4	3	2	3	5
6	6	4	4	5	4
8	8	6	4	5	6

# Alinierea secvențelor

## ► Exemplu

$$p_{\text{spațiu}} = 2,$$

$$p_{AC} = p_{GT} = 1$$

$$p_{XY} = 3 \text{ pentru } X \neq Y \text{ în rest}$$

**GATC      ->      G-ATC**  
**TCAG                      TCAG-      scor 6**

**Soluția:**

C:

	0	1	2	3	4	
	0	2	4	6	8	↑ aliniere GT
	2	1	3	5	6	↑ aliniere -C
	4	3	2	3	5	↑ aliniere AA
	6	4	4	5	4	↑ aliniere TG
	8	6	4	5	6	↑ aliniere C-

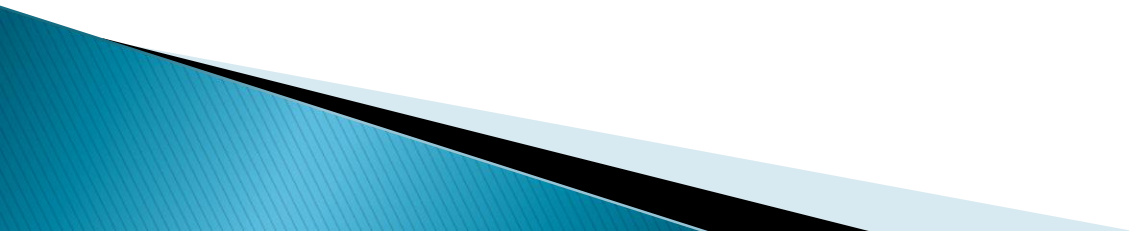
# Distanța de ediare – Levenstein

- ▶ **Similar (generalizare)**

carte

antet

- ▶ **Laborator**





# Alte probleme de numărare

 Numărul de șiruri binare de lungime  $n$  care nu conțin două valori egale cu 1 pe poziții consecutive

000

001

010

100

101

# Probleme de numărare

- ▶ Numărul de șiruri binare de lungime  $n$  care nu conțin două valori egale cu 1 pe poziții consecutive



Analizăm structura unui șir soluție evidențiind primul element

- Începe cu 0
- Începe cu 1

# Probleme de numărare

- ▶ Numărul de șiruri binare de lungime  $n$  care nu conțin două valori egale cu 1 pe poziții consecutive



Analizăm structura unui șir soluție evidențiind primul element

- Începe cu 0 – poate continua cu orice șir binar valid de lungime  $n-1$
- Începe cu 1 – poate continua cu orice șir binar valid de lungime  $n-1$  care începe cu 0

# Probleme de numărare

- Numărul de șiruri binare de lungime  $n$  care nu conțin două valori egale cu 1 pe poziții consecutive
- **Subprobleme**
  - $Nr0[i]$  – numărul de șiruri binare valide care încep cu 0
  - $Nr1[i]$  – numărul de șiruri binare valide care încep cu 1
- **Recurențe**
- **Soluție**

# Probleme de numărare



► Numărul de șiruri de lungime  $n$  peste alfabetul  $\{1,2,3\}$  care respectă constrângerile:

- orice 1 are pe pozițiile alăturate în stânga și dreapta valoarea 3
- orice 2 are pe poziția din stânga (**altă variantă: pe cel puțin una dintre pozițiile din stânga**) valoarea 3

3132331332 – DA

313223133 – NU/**DA**

132331332 – NU

Temă

# Probleme de numărare



► Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)

# Probleme de numărare

- ▶ Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)



Analizăm structura unei permutări soluție  $x_1 x_2 \dots x_n$  evidențiind ultimul element  $x_n$

- $n$
- $n-1$
- $n-2$
- ...
- $1$

# Probleme de numărare

- ▶ Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)



Analizăm structura unei permutări soluție  $x_1 x_2 \dots x_n$  evidențiind ultimul element  $x_n$

- $n$  – în  $x_1 x_2 \dots x_{n-1}$  sunt  $k$  inversiuni, deoarece nu există inversiune de forma  $(x, n)$



# Probleme de numărare

- ▶ Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)



Analizăm structura unei permutări soluție  $x_1 x_2 \dots x_n$  evidențiind ultimul element  $x_n$

- $n$  – în  $x_1 x_2 \dots x_{n-1}$  sunt  $k$  inversiuni, deoarece nu există inversiune de forma  $(x, n)$
- $n-1$  – în  $x_1 x_2 \dots x_{n-1}$  sunt  $k-1$  inversiuni, deoarece unica inversiune determinată de  $x_n = n-1$  este  $(n, n-1)$
- ...  $n-k \dots 1$

# Probleme de numărare

- ▶ Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)

- ▶ **Subprobleme**

- $Nr[i][t]$  – numărul de permutări cu  $i$  elemente având  $t$  inversiuni,  $i \leq n, t \leq k$

- ▶ **Recurențe**

$$Nr[i][t] = Nr[i-1][t] + Nr[i-1][t-1] + \dots + Nr[i-1][\max\{t-i+1, 0\}]$$
$$t \leq i(i-1)/2$$

- ▶ **Soluție**  $Nr[n][k]$

# Probleme de numărare

- ▶ Numărul permutări ale mulțimii  $\{1, 2, \dots, n\}$  care au exact  $k$  inversiuni ( $n, k$  date)

➤ Mahonian numbers

$Nr[n][k] = T(n, k) =$  coeficientul lui  $x^k$  din produsul

$$\prod_{i=0}^{n-1} (1 + x + \dots + x^i)$$

$$Nr[n][k] = Nr[n][\binom{n}{2} - k]$$

1										
1	1									
1	2	2	1							
1	3	5	6	5	3	1				
1	4	9	15	20	22	20	15	9	4	1

# Suplimentar

**Horia Georgescu. Tehnici de programare. Editura  
Universității din București 2005**

# Verificarea apartenenței unui cuvânt la limbajul generat de o gramatică independentă de context



Fie  $G = (N, T, S, P)$  o gramatică independentă de context și  $w \in T^*$ .

Se cere să se determine dacă  $w \in L(G)$ .

- ▶ Gramatica este în forma normală a lui Chomsky: producțiile au numai formele  $A \rightarrow BC$  și  $A \rightarrow a$ , cu  $A, B \in N$  și  $a \in T$ .

# Verificarea apartenenței unui cuvânt la limbajul generat de o gramatică independentă de context

$$w = a_1 a_2 \dots a_n$$

$$(a_1 a_2 \dots a_k) (a_{k+1} \dots a_n)$$



B



C

există  $A \rightarrow BC$  în  $P$

# Verificarea apartenenței unui cuvânt la limbajul generat de o gramatică independentă de context

$$w = a_1 a_2 \dots a_n$$

**Subproblemă:**  $M(i,j) = \{ A \in N \mid A \Rightarrow a_i \dots a_j \}$ , unde  $\Rightarrow$  semnifică derivare în oricâți pași

# Verificarea apartenenței unui cuvânt la limbajul generat de o gramatică independentă de context

$$w = a_1 a_2 \dots a_n$$

**Subproblemă:**  $M(i,j) = \{ A \in N \mid A \Rightarrow a_i \dots a_j \}$ , unde  $\Rightarrow$  semnifică derivare în oricâți pași

**Soluție:**  $w \in L(G) \Leftrightarrow S \in M(1,n)$

**Știm:**  $M(i,i) = \{ A \in N \mid A \rightarrow a_i \in P \}$

**Recurențe:**  $M(i,j) = \{ A \in N \mid \exists k \in i..j-1, \text{ astfel încât } \exists B \in M(i,k) \text{ și } C \in M(k+1,j) \text{ cu } A \rightarrow BC \in P \}$



# Verificarea apartenenței unui cuvânt la limbajul generat de o gramatică independentă de context

**Recurențe:**  $M(i,j) = \{ A \in N \mid \exists k \in i..j-1, \text{ astfel încât } \exists B \in M(i,k) \text{ și } C \in M(k+1,j) \text{ cu } A \rightarrow BC \in P \}$

```
M(i,j) ← ∅  
for k=i,j-1  
  for toți B∈M(i,k)  
    for toți C∈M(k+1,j)  
      if A→BC ∈ P  
        M(i,j) ← M(i,j) ∪ {A}
```

# Descompunerea unui dreptunghi în pătrate



- ▶ Se consideră un dreptunghi cu laturile de  $m$ , respectiv  $n$  unități ( $m < n$ ). Asupra sa se pot face tăieturi *complete* pe orizontală sau verticală. Se cere numărul minim de pătrate (cu laturi numere întregi) în care poate fi descompus dreptunghiul.

# Descompunerea unui dreptunghi în pătrate

**Exemplu .** Un dreptunghi 5 x 6 poate fi descompus în două pătrate de latură 3 și 3 pătrate de latură 2.

