**LAB1**

| static void MaxMinLinii(int[][] numbers, int length)  {  for(int i=0; i<length; i++)  {  //Resetam max si minim, dupa terminarea fiecarei linii.  int max=-99999999;  int min=99999999;    for(int j=0; j<length; j++)  {  if(numbers[i][j]>max) max=numbers[i][j];  if(numbers[i][j]<min) min=numbers[i][j];  }    System.out.println("Max Linia " + i + ": " + max);  System.out.println("Min Linia " + i + ": " + min);  }  } |
| --- |

**PRODUS SCALAR MAXIM**

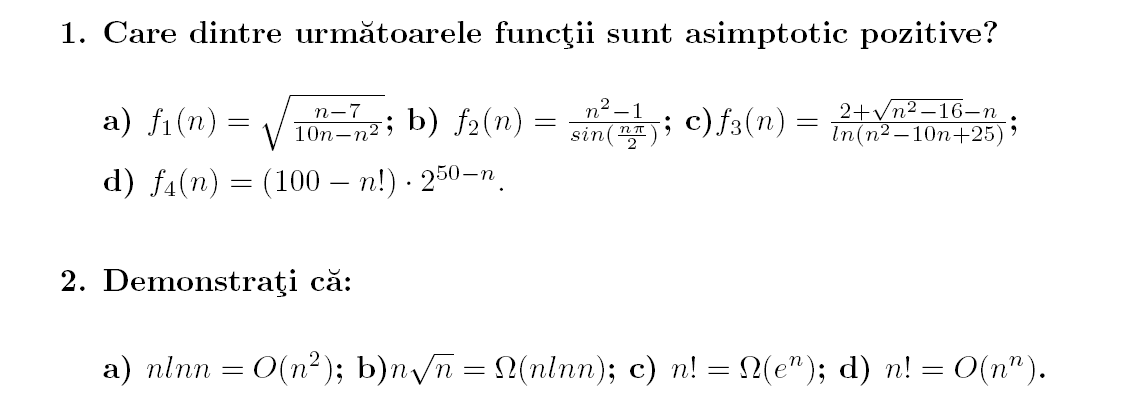
| static void SortareCrescatoare(int a[], int n)  {  int temp;    //Sortare de complexitate O(n^2), are 2 for-uri  for(int i=1; i<=n; i++)  {  for(int j=1; j<=n; j++)  {  //Cu interschimbare  if(a[i]<a[j])  {  temp=a[i];  a[i]=a[j];  a[j]=temp;  }  }  }  }  static void MAXIM2(int a[],int b[], int n, int s)  {  SortareCrescatoare(a,n);  SortareCrescatoare(b,n);  s=0;    for(int i=1; i<=n; i++){  s=s+a[i]\*b[i];  }    AFISARE(a,b,n,s);  }  //Sortând ambele tablouri **în ordine crescătoare**, se asociază **cei mai mici termeni cu cei mai mici termeni** și **cei mai mari termeni cu cei mai mari termeni**.  Acest lucru **maximizează produsul scalar**, ceea ce este corect. | static void AFISARE(int a[], int b[], int n,int s)  {  System.out.println("Suma="+s);  for(int i=1; i<=n; i++) System.out.print(a[i] + " ");    System.out.println();    for(int i=1; i<=n; i++) System.out.print(b[i] + " ");  }  public static void main(String[] args)  {  //Produs Scalar Maxim/Minim  int n,s=0,i,a[], b[];  Scanner cin=new Scanner(System.in);  System.out.print("Nr elemente=");  n=cin.nextInt();  a=new int [n+1];  b=new int [n+1];  System.out.print("Elementele din a[]: ");  for(i=1;i<=n;i++) a[i]=cin.nextInt();    System.out.print("Elementele din b[]: ");  for(i=1;i<=n;i++) b[i]=cin.nextInt();  MAXIM2(a,b,n,s);  } |
| --- | --- |

**PRODUS SCALAR MINIM**

Dacă dorim produsul scalar minim, atunci vem înmulții cel mai mare element dintr-un vector cu cel mai mic element din celălalt vector.

* Asta s-ar face sortând un vector crescător și celălalt descrescător.

| static void MINIM1(int a[],int b[], int n, int s)  {  s=0;  for(int i=1; i<=n; i++)  {  int k=i;  int m=a[i];  for(int j=i+1; j<=n; j++)  {  if(a[j]<m)  {  k=j;  m=a[j];  }  }  //Interschimbam  int temp=a[i];  a[i]=a[k];  a[k]=temp;    k=i;  m=b[i];    for(int j=i+1; j<=n; j++)  {  if(b[j]>m)  {  k=j;  m=b[j];  }  }  temp=b[i];  b[i]=b[k];  b[k]=temp;  s=s+a[i]\*b[i];  }  AFISARE(a,b,n,s);  } | import java.util.Scanner;  public class MinProdusScalar{  static void AFISARE(int a[], int b[], int n,int s)  {  System.out.println("Suma="+s);  for(int i=1; i<=n; i++) System.out.print(a[i] + " ");    System.out.println();    for(int i=1; i<=n; i++) System.out.print(b[i] + " ");  }  **//AICI AR VENII FUNCȚIILE. din stanga**  public static void main(String[] args)  {  //Produs Scalar Maxim/Minim  int n,s=0,i,a[], b[];  Scanner cin=new Scanner(System.in);  System.out.print("Nr elemente=");  n=cin.nextInt();  a=new int [n+1];  b=new int [n+1];  System.out.print("Elementele din a[]: ");  for(i=1;i<=n;i++) a[i]=cin.nextInt();    System.out.print("Elementele din b[]: ");  for(i=1;i<=n;i++) b[i]=cin.nextInt();  MINIM1(a,b,n,s);  }  } |
| --- | --- |



1. **Care dintre urmatoarele functii sunt asimptotic pozitive?**

**a)** **(cand n este un numar mare)**

* Numaratorul (n−7) este aproximativ n
* Numitorul 10n - n^2 este dominat de termenul −n^2
* Rezulta aproximativ ca fractia este -1/n, iar radacina devine radical din -1/n care nu este definit.
* Deci **NU** este asimptotic pozitiva.

#### **b)**

* Numaratorul (n^2-1) este pozitiv pentru n>1.
* Numitorul sinus oscilează între -1 și 1, semnul depinde de n.
* Deoare sin ia valori negative pentru unele valori ale lui n, functia va fii uneori negativa, deci **NU** este asimptotic pozitiva.

#### **c)​​**

* Numărătorul este echivalent cu radical din n^2, adica aproximativ n.
* Numitorul este pozitiv, deoarece, ce se află sub (ln) este > 0 cand n este numar mare.
* Numitorul si Numaratoru sunt pozitivi => **Funcția este Asimptotic pozitiva**

#### **d)**

* Factorialul devine negativ, cand n>100
* Deci funcția nu e definită pentru n>100 => **NU** este asimptotic pozitiva

**2. Demonstrati ca:**

**a)**

* Comparam n ln(n) si n^2.
* ln(n) creste mult mai incet decat n.
* Astfel, n ln(n) < n^2 (cand n are valori mari)

**b)**

* Definitia notatiei Omega (Ω) înseamnă ca exista constante c>0 și n0>0 astfel incat f(n) >= c\* g(n).
* Comparam n radical din n CU n ln(n):
* n sqrt(n) este aproximativ = cu n^1.5
* n ln(n) creste mai lent decat n^1.5
* Deci n sqrt(n) este mereu > n ln(n) (pentru valori suficient de mari)

**c)**

* n factorial = 1\*2\*3\*4\*....\*n
* e^n = e\*e\*e\*....\*e (de n ori)
* Pentru n>3, toti termeni n! > e^n deoarece factorial creste mai repede decat e^n.

**d)**

* n factorial = 1\*2\*3\*4\*....\*n
* n^n = n\*n\*n\*...\*n
* n! este clar mai mic decat n^n, pana și pentru n=2

**Avem o multime nr int. Determina o submultime a acestuia cu suma maxima!**

| import java.util.Scanner;  class Main {  public static Scanner cin = new Scanner(System.in);  public static void AFISARE(int sumaMax, int n, int[] numere, int[] aparitii){  System.out.println("Suma maxima este -> " + sumaMax);  System.out.println("Compusa din submultimea:");  System.out.print("{");  for(int i=1; i<=n; i++){  if(aparitii[i] == 1) System.out.print(numere[i] + " ");  }  System.out.println("}");  }  public static void SUBMULTIME\_SUMA\_MAXIMA(int n, int[] numere, int sumaMax, int[] aparitii){  sumaMax=0;  for(int i=1; i<=n; i++){  //Daca e nr negativ, sau 0, sarim peste el.  if(numere[i] <= 0) continue;  sumaMax += numere[i];  aparitii[i]=1;//numărul curent apare în submulțime  }  AFISARE(sumaMax, n, numere, aparitii);  }  public static void main(String[] args) {  int n, sumaMax=0; int[] numere, aparitii;  System.out.print("n = ");  n = cin.nextInt();  numere = new int[n+1];  aparitii = new int[n+1];  System.out.print("Dati " + n + " numere: ");  for (int i=1; i<=n; i++){  numere[i] = cin.nextInt();  }  SUBMULTIME\_SUMA\_MAXIMA(n, numere, sumaMax, aparitii);  }  }  /\* Exemplu de apelare:  n = 5  Dati 5 numere: 0 3 -2 5 -4  Rezultat:  Suma maxima este -> 8  Compusa din submultimea: {3 5 } \*/ |
| --- |

**L3**

**Memorarea optima a textelor pe benzi**

Explicație:

* Definirea clasei Text – Aceasta conține textul și lungimea sa calculată automat.
* Crearea unui array de obiecte Text – Se introduc mai multe texte de diferite lungimi.
* Sortarea array-ului – Se folosește Arrays.sort() cu un Comparator care sortează după lungime.
* Afișarea rezultatului – Se afișează textele în ordinea optimă. (de la lungime minima la lungime maxima)
* Această soluție minimizează timpul mediu de citire conform cerințelor problemei.

| import java.util.Arrays;  import java.util.Comparator;  class Text {  String content;  int length;  public Text(String content) {  this.content = content;  this.length = content.length();  }  }  public class MagneticTapeStorage {  public static void main(String[] args) {  Text[] texts = {  new Text("Text text text text 4"),  new Text("Text 1"),  new Text("Text text text 3"),  new Text("Text text 2"),  new Text("Text text text text text 5 "),  };  // Sortare după lungimea textelor (O(n log n))  Arrays.sort(texts, Comparator.comparingInt(t -> t.length));  // Afisare ordine optima de memorare  System.out.println("Ordinea optima de memorare pe banda:");  for (Text text : texts) {  System.out.println(text.content + " (" + text.length + " octeți)");  }  }  } |
| --- |

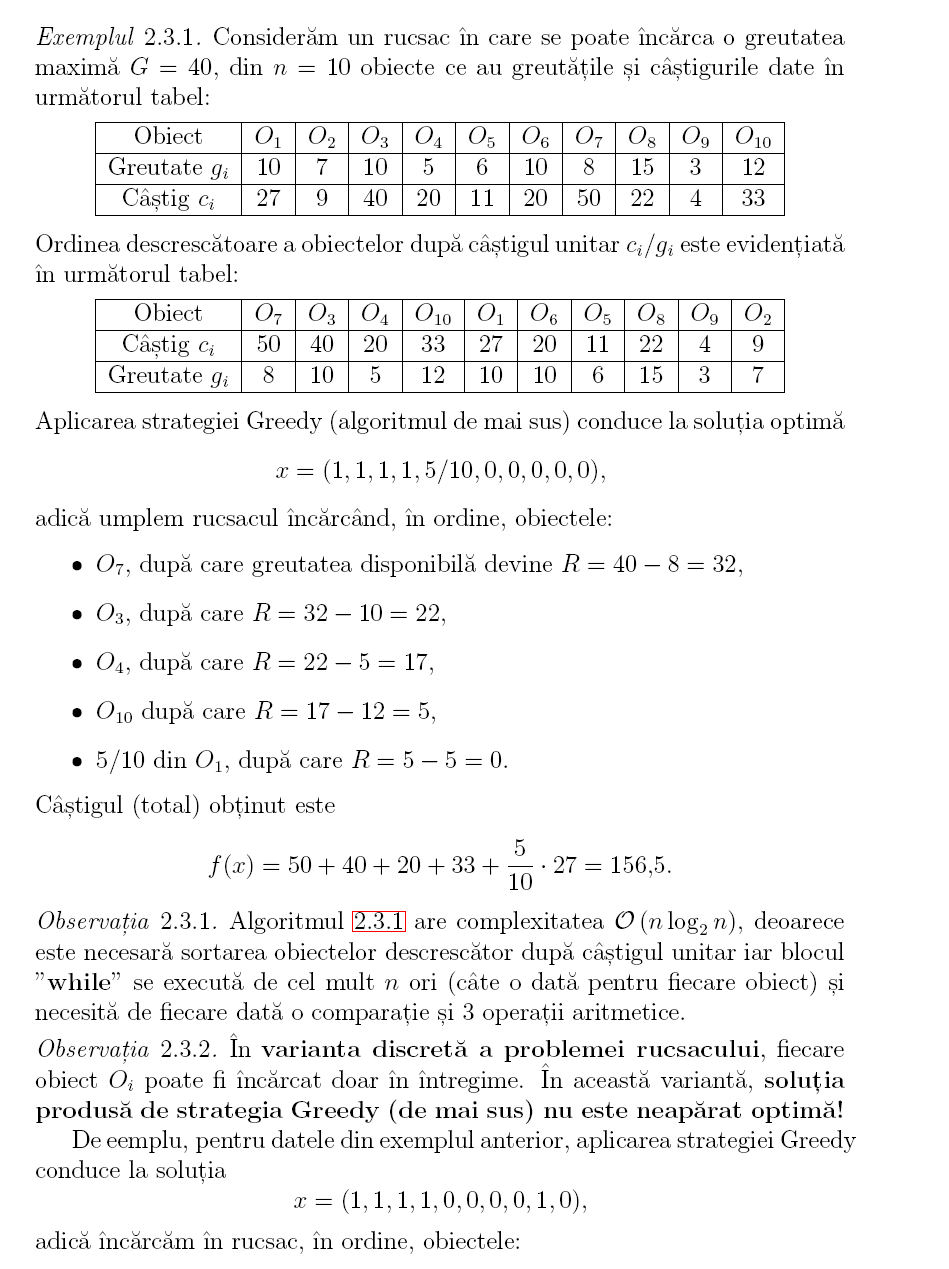
**//(La fiecare pas) La citirea unui text de pe banda, trebuie citite și textele aflate înaintea lui.**

//Acest program va lua textele si le va memora in ordinea lungimii lor, deoarece cele mai scurte texte au cea mai mica memorie. De aceea ordinea conteaza. (aceasta sortare face timpul mediu de citire să fie minim)

**Problema Rucsacului**

| import java.util.Scanner;  class L4Rucsac {      public static void SORTARE(double g[], double c[], int n) {  for (int i = 1; i <= n - 1; i++) {  for (int j = i + 1; j <= n; j++) {  if (c[i] / g[i] < c[j] / g[j]) {  double aux = g[i];  g[i] = g[j];  g[j] = aux;  aux = c[i];  c[i] = c[j];  c[j] = aux;  }  }  }  }    public static void CITIRE(double g[], double c[], int n)  {  Scanner cin = new Scanner(System.in);  System.out.println("Citire " + n +  " Greutati:");  for(int i=1; i<=n; i++)g[i]=cin.nextDouble();    System.out.println("Citire " + n +  " Costuri:");  for(int i=1; i<=n; i++)c[i]=cin.nextDouble();  }    public static void AFISARE(double g[], double c[], int n)  {  System.out.println("c[]");  for(int i=1; i<=n; i++)System.out.print(c[i] + " ");  System.out.println("");  System.out.println("g[]");  for(int i=1; i<=n; i++)System.out.print(g[i] + " ");  }    public static void RUCSAC(double G, double g[], double C, double c[],  int n, double x[])  {  C=0;  for(int i=1; i<=n; i++)x[i]=0;  double R=G;  int i=1;    while(R>0)  {  if(g[i]<=R)  {  x[i]=1;  C=C+c[i];  R=R-g[i]; | i++;  }  else{  x[i]=R/g[i];  C=C+x[i]\*c[i];  R=0;  }  }    System.out.println("Ordine optima:");  for(i=1; i<=n; i++) System.out.print(x[i] + " ");  }      public static void main(String[] args) {  double g[], c[], x[], C=0, G=40; int n;    Scanner cin = new Scanner(System.in);  System.out.print("n=");  n = cin.nextInt();    System.out.print("G:");  G = cin.nextDouble(); //40    g= new double[n+1];  c= new double[n+1];  x= new double[n+1];    CITIRE(g,c,n);  SORTARE(g,c,n);  //AFISARE(g, c, n);  RUCSAC(G,g,C,c,n,x);  }  }  **/\* REZULTAT:**  **n=10**  **G:40**  **Citire 10 Greutati:**  **10 7 10 5 6 10 8 15 3 12**  **Citire 10 Costuri:**  **27 9 40 20 11 20 50 22 4 33**  **Ordine optima:**  **1.0 1.0 1.0 1.0 0.5 0.0 0.0 0.0 0.0 0.0**  **\*/** |
| --- | --- |

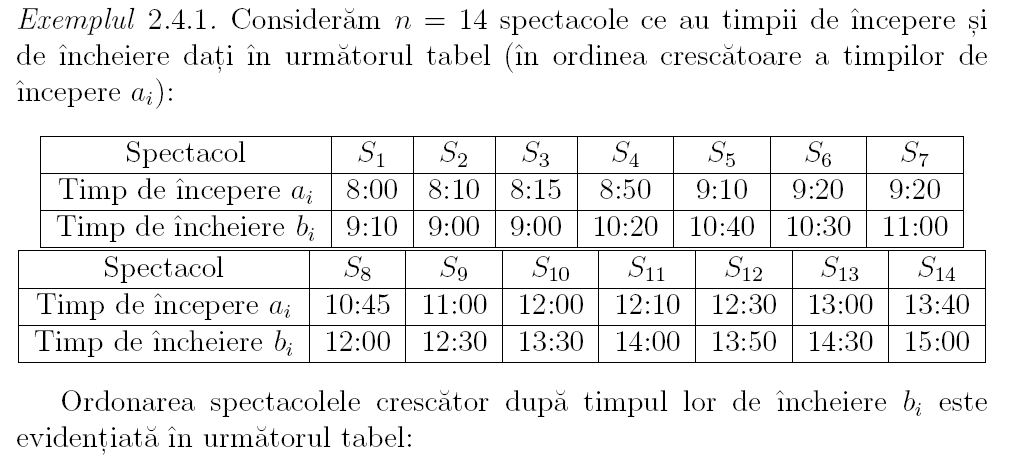
| **O7 = 50/8 = 6.25** | **O3 = 40/10 = 4** | **O4 = 20/5 = 4** | **O10 = 33/12 = 2.75** | **O1 = 27/10 = 2.7** |
| --- | --- | --- | --- | --- |
| **O6 = 20/10 = 2** | **O5 = 11/6 = 1.83** | **O8 = 22/15 = 1.46** | **O9 = 4/3 = 1.33** | **O2 = 1.28** |

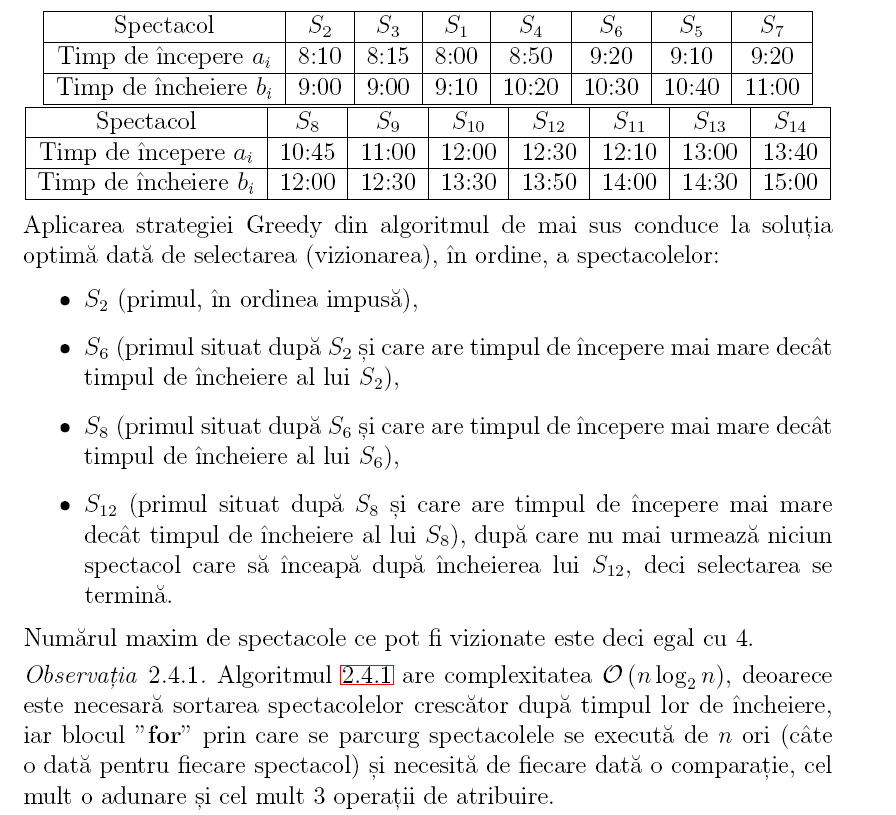
****

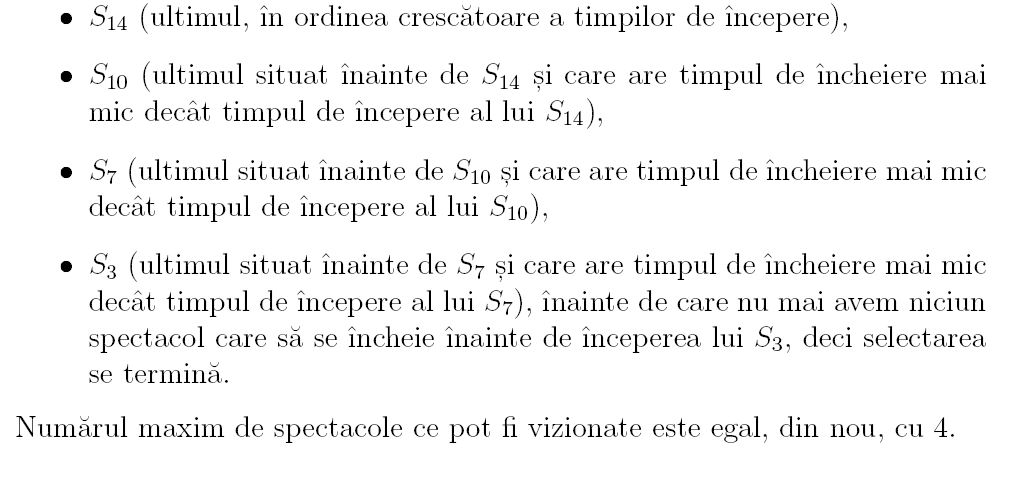
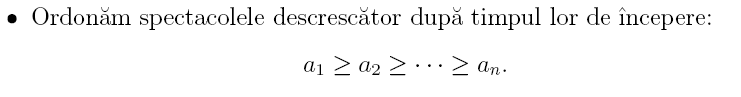
**Nr. Spectacole**

| import java.util.Scanner;  class L4Spectacole {      public static void SORTARE(double a[], double b[], int n) {  for (int i = 1; i <= n - 1; i++) {  for (int j = i + 1; j <= n; j++) {  if (b[i] > b[j]) {  double aux = a[i];  a[i] = a[j];  a[j] = aux;  aux = b[i];  b[i] = b[j];  b[j] = aux;  }  }  }  }    public static void CITIRE(double a[], double b[], int n)  {  Scanner cin = new Scanner(System.in);  System.out.println(n + " Timpuri de Incepere:");  for(int i=1; i<=n; i++)a[i]=cin.nextDouble();    System.out.println(n + " Timpuri de Incheiere:");  for(int i=1; i<=n; i++)b[i]=cin.nextDouble();  cin.close();  }    public static void AFISARE(double a[], double b[], int n)  {  System.out.println("Lista Sortata: Timpuri Incepere:");  for(int i=1; i<=n; i++)System.out.print(a[i] + " ");  System.out.println("");  System.out.println("Lista Sortata: Timpuri Incheiere:");  for(int i=1; i<=n; i++)System.out.print(b[i] + " ");  } | public static void SPECTACOLE(double a[], double b[], double c[], int n, int m)  {  m=0;  for(int i=1; i<=n; i++) c[i]=0;  double t=a[1]-1;    for(int i=1; i<=n; i++){  if(a[i]>t)  {  c[i]=1;  m++;  t=b[i];  }  }    System.out.println("Nr. spectacole vizionate: " + m);  System.out.println("Lista vizionarii:");  for(int i=1; i<=n; i++) if(c[i]==1) System.out.println("Incepe la: " + a[i] + " si se termina la: " + b[i]);  }      public static void main(String[] args) {  double a[], b[], c[]; int n, m=0;    Scanner cin = new Scanner(System.in);  System.out.println("");  System.out.print(" Nr spectacole=");  n = cin.nextInt();    a= new double[n+1];  b= new double[n+1];  c= new double[n+1];      CITIRE(a,b,n);  SORTARE(a,b,n);  //AFISARE(a, b, n);  SPECTACOLE(a,b,c,n,m);  cin.close();  }  }  /\*  N=14  Timp Incepere:  8 8.1 8.15 8.50 9.10 9.20 9.20 10.45 11 12 12.1 12.3 13 13.4  Timp Incheiere:  9.1 9 9 10.2 10.4 10.3 11 12 12.3 13.3 14 13.5 14.3 15  \*/ |
| --- | --- |

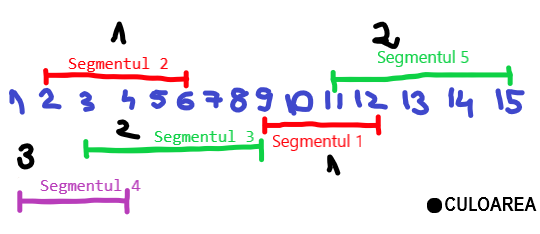
* **Ordonam spectacolele crescător după timpul lor de iıncheiere: b1 <= b2 <=... <= bn**
* Selectam primul. Spectacolul curent S(i) selectează dacă **timp începere>timp încheiere al spectacolului selectat.**

****

****

****

Nr minim de culori necesare pentru a colora “n” segmente dar 2 segmente care se intersectează sa fie colorate diferit.



Segmentul 1, 2 - Culoarea 1

Segmentul 3 - Culoarea 2

Segmentul 4,5 - Culoarea 3

**COLORAREA STEAGULUI TRICOLOR**

| import java.util.Scanner;  class Main {  static Scanner cin = new Scanner(System.in);  public static void COLORARE\_STEAG\_TRICOLOR(int n, String[] c){  //"i" reprezinta culoarea pusa pe prima pozitie.  //"j" reprezinta culoare pusa pe a 3-a pozitie.  //Dar toate culorile steagului trebuie sa fie diferite.  for(int i=1; i<=n; i++){  for(int j=1; j<=n; j++) {  //n-1 este penultima culoare (punem la mijloc)  if(i!=j && i!=n-1 && j!=n-1) System.out.println(c[i] + " " + c[n-1] + " " + c[j] );  //"n" este ultima culoare (punem la mijloc) | if(i!=j && i!=n && j!=n)System.out.println(c[i] + " " + c[n] + " " + c[j] );  //Fac cu 2 if-uri ca sa putem afisa toate solutiile.  }  }  }  public static void main(String[] args) {  int n=4; //n- nr culori  String[] c = new String[n+1];//culorile  c[1] = "rosu";  c[2] = "albastru";  c[3] = "galben";  c[4] = "alb";  COLORARE\_STEAG\_TRICOLOR(n,c);  }  } |
| --- | --- |

**Colorarea grafurilor**

| import java.util.Scanner;  class Main {  static Scanner cin = new Scanner(System.in);  static int i,muchii=10,n=6, m=5, a[][];  public static void CreareMatriceAdiacenta(int a[][], int muchie)  {  int x=0,y=0;  //System.out.println("Dati muchia " + i + ": " + "Intre nodurile: ");  System.out.println("Dati muchiile: ");  for(int i=1; i<=muchie; i++)  {  x = cin.nextInt();  y = cin.nextInt();  a[x][y]++;  a[y][x]++;  }  }  public static void AfisareMatrice(int a[][], int n)  {  System.out.println("Matrice Adiacenta:");  for(int i=1; i<=n; i++)  {  for(int j=1; j<=n; j++) System.out.print(a[i][j] + " ");  System.out.println();  }  System.out.println("- - - - - -");  }  public static void AFISARE(int vector[], int n)  {  for(int i=1; i<=n; i++) System.out.print(vector[i] + " ");  System.out.println();  } | public static void COLORARE(int a[][], int n, int m){  int k=1, x[];  x = new int[n+1];  x[1]=0;  while(k>0){  if(x[k]<m){  x[k]= x[k]+1;  if(VALID(x,k)==1)  {  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=0;  }  }  }  else{  k--;  }  }  }  public static int VALID(int x[], int k){  for(int i=1; i<=k-1; i++)  {  if(a[i][k]>=1 && x[i]==x[k])return 0;  }  return 1;  }  public static void main(String[] args) {  a = new int[n+1][n+1];  //System.out.println("Nr noduri=");  //cin.nextInt(n);  //System.out.println("Nr muschii=");  //cin.nextInt(muchii);  CreareMatriceAdiacenta(a, muchii);  AfisareMatrice(a, n);  //Asumam ca avem m=5 culori, n=6 noduri, m=10 muchii  //1 2 1 3 1 5 1 6 2 3 2 5 3 4 3 6 4 5 5 6  COLORARE(a,n,m);  }  } |
| --- | --- |

x = (x1, x2, . . . , xn), (asa arata fiecare solutia, unde xi reprezinta culoarea)

**Dame**

| import java.util.Scanner;  class HelloWorld {  public static Scanner cin = new Scanner(System.in);        public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++){  System.out.print(x[i] + " ");  }  System.out.println();  }    public static void DAME(int n){  int k=1;  int[] x;  x = new int[n+1];  x[1]=0;    while(k>0){  if(x[k]<n){  x[k]++;  if(VALID(x,k)==1){  if(k==n){  AFISARE(x,n);  k--;  }  else{  k++;  x[k]=0;  }  }  }  else k--;  }    }    public static int VALID(int[] x, int k){  for(int i=1; i<=k-1; i++){  int pozitiv = Math.abs(x[k]-x[i]);  if(x[k]==x[i] || k-i==pozitiv) return 0;  }  return 1;  }  public static void main(String[] args) {  int n;  System.out.print("n=");  n = cin.nextInt();  DAME(n);  }  } | * x = (1, 3, 5, 2, 4) * NU aceeași coloana (i!=j) * NU pe aceeași diagonală |i-j| != xi-xj * Pentru n = 2, problema nu are soluție, deoarece două dame așezate pe linii și coloane diferite se află pe aceeași diagonală, deci se atacă reciproc. * Nici pentru n = 3, problema nu are soluție, deoarece orice două dame așezate pe aceeași culoare se atacă reciproc. |
| --- | --- |

**Nebuni pe Tabla Sah**

| **//PE NEGRU**  import java.util.Scanner;  class HelloWorld {  public static Scanner cin = new Scanner(System.in);  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n-1; i++){  System.out.print(x[i] + " ");  }  System.out.println();  }    public static void NEBUNI1(int n){  int k=1, ult=2;  int[] x;  x = new int[n+1];  x[1]=0;    while(k>0){  if(x[k]<ult){  x[k]++;  if(VALID(x,k)==1){  if(k==n-1)AFISARE(x,n);  else{  k++;  if(k<=n/2)  {  x[k]=0;  ult=2\*k;  }  else{  x[k]=2\*k-n;  ult=n;  }  }  }  }  else{  k--;  if(k<=n/2)ult=2\*k;  else ult=n;  }  }    }    public static int VALID(int[] x, int k){  for(int i=1; i<=k-1; i++){  if(x[k]-x[i]==k-i) return 0;  }  return 1;  }  public static void main(String[] args) {  int n;  System.out.print("n=");  n = cin.nextInt();  NEBUNI1(n);  }  } | **//PE ALB**  class HelloWorld {  public static Scanner cin = new Scanner(System.in);  //afisare functie  public static void NEBUNI2(int n){  int k=2, ult=3;  int[] y;  y = new int[n+1];  y[2]=0;    while(k>1){  if(y[k]<ult){  y[k]++;  if(VALID(y,k)==1){  if(k==n-1)  {  y[1]=1;  AFISARE(y,n);  y[1]=n;  AFISARE(y,n);  }  else{  k++;  if(k<=(n+1)/2)  {  y[k]=0;  ult=2\*k-1;  }  else{  y[k]=2\*k-n-1;  ult=n;  }  }  }  }  else{  k--;  if(k<=n/2)ult=2\*k;  else ult=n;  }  }  }    public static int VALID(int[] y, int k){  if(y[k]==k)return 0;    for(int i=2; i<=k-1; i++){  if(y[k]-y[i]==k-i) return 0;  }  return 1;  }  public static void main(String[] args) {  int n; System.out.print("n=");  n = cin.nextInt();  NEBUNI2(n);  }  } |
| --- | --- |

| * x = (1, 4, 2, 5) | * Pe alb, primul nebun este pus ori pe poziția 1 ori pe poziția 5. * **Nu se pot afla pe aceeași diagonală.** * xₖ reprezintă linia pe care se află nebunul de pe diagonala Dₖ, pentru k = 1, n − 1. * **IMPORTANT:** Numărul maxim de nebuni ce pot fi așezați pe o tablă de șah de dimensiune n×n, n ≥ 2, astfel încât oricare doi nebuni să nu se atace reciproc este = **n − 1**. * Negru: Direcția stanga-sus -> dreapta-jos |
| --- | --- |

| **CALUL pe Tabla De Șah** (merge ca un **“L”** adică 2 poziții într-o direcție, și apoi 1 poziție în alta directie)  // Verificăm dacă noua poziție este validă (dacă nu am ieșit din tabla si mai poate fii vizitata)  //Spre exemplu x = x + dx[i]  //Si y = y + dy[i].  //Pentru i=1, calul încearca sa mearga 2 poziții la dreapta, si 1 pozitie in sus. (daca poate)  Stocam Poziția curentă în vectorul de pași  // Dacă nu am găsit mișcări valide), afisam solutia partiala.  // Dar doar daca tabla este și goala. (altfel înseamnă ca ne-am blocat.) |
| --- |

### **Algorithm prin backtracking**

1. Se încearcă să atribuim o culoare primului nod.
2. Se verifică dacă alegerea este validă (adică nu intră în conflict cu culorile nodurilor adiacente).
3. Dacă este validă, se trece la următorul nod și se repetă procesul.
4. Dacă un nod nu poate primi nicio culoare validă, se face **backtracking**: se revine la nodul anterior și se încearcă o altă culoare.
5. Se repetă procesul până când:
   * Toate nodurile au fost colorate (soluție validă).
   * S-au epuizat toate opțiunile (nu există soluție cu mm culori).

## **Problema colorarii hartilor**

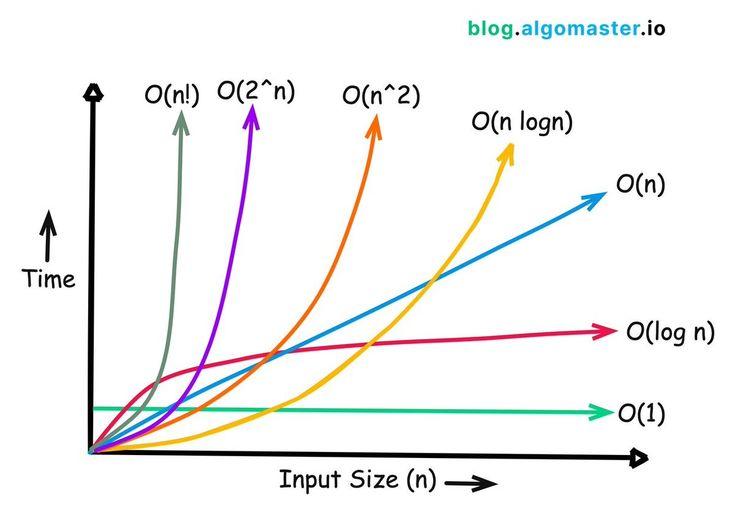
Se dă o hartă formată din mai multe țări și un număr de culori disponibile. Se cere să se coloreze fiecare țară cu una dintre aceste culori, astfel încât **două țări vecine (care au frontieră comună) să nu aibă aceeași culoare**.

### **Modelarea problemei**

Putem reprezenta această problemă sub forma unui **graf neorientat**, unde:

* **Fiecare țară este un nod al grafului**.
* **Există o muchie între două noduri dacă și numai dacă țările respective au frontieră comună**.

Astfel, problema colorarii hartilor devine o **problemă de colorare a unui graf**, unde trebuie să atribuim fiecărui nod o culoare astfel încât **nodurile conectate prin muchii să aibă culori diferite**.



**2. Metoda Greedy**

* Metoda Greedy nu caută să determine toate soluțiile posibile și apoi să aleagă pe cea optimă conform criteriului de optimizare dat (ceea ce ar necesita în general un timp de calcul și spațiu de memorie mari), ci constă în a alege pe rând câte un element, urmând să-l "înghită" eventual în soluția optimă. De aici vine și numele metodei (Greedy = lacom).
* Astfel, dacă trebuie determinat maximul unei funcții de cost depinzând de a1, ..., an, ideea generală a metodei este de a alege la fiecare pas acel element care face să crească cât mai mult valoarea acestei funcții. Din acest motiv metoda se mai numește și a **optimului local.**
* **Optimul global** se obține prin alegeri succesive, la fiecare pas, ale optimului local, ceea ce permite rezolvarea problemelor fără revenire la deciziile anterioare (așa cum se întâmplă la metoda backtracking).
* În general metoda Greedy oferă o soluție posibilă și nu întotdeauna soluția optimă. De aceea, dacă problema cere soluția optimă, algoritmul trebuie să fie însoțit și de justificarea faptului că soluția generată este optimă.

### 

### **🔹 Complexitate**

1. **Timp de execuție și complexitate**
   * Se măsoară în funcție de **numărul de operații** efectuate în raport cu dimensiunea intrării (n).
   * Se exprimă în **notația Big O** care oferă o estimare a performanței în cel mai rău caz.
2. **Notația Big-O**  
   Reprezintă o limită superioară pentru creșterea numărului de operații în raport cu n.  
   Exemple:
   * **O(1)** – Timp constant (ex: accesarea unui element într-un array)
   * **O(log n)** – Logarithmic (ex: căutare binară)
   * **O(n)** – Liniar (ex: parcurgerea unui array)
   * **O(n log n)** – Quasilinear (ex: sortarea prin Merge Sort)
   * **O(n²), O(2ⁿ), O(n!)** – Complexități mai mari (ex: algoritmi de forță brută)
3. **Cel mai bun, mediu și cel mai rău caz**
   * **Best-case (Ω)** – Cel mai favorabil scenariu
   * **Average-case (Θ)** – Cazul mediu
   * **Worst-case (O)** – Cel mai rău caz (folosit de obicei pentru analiză)

**Pattern-uri standard**

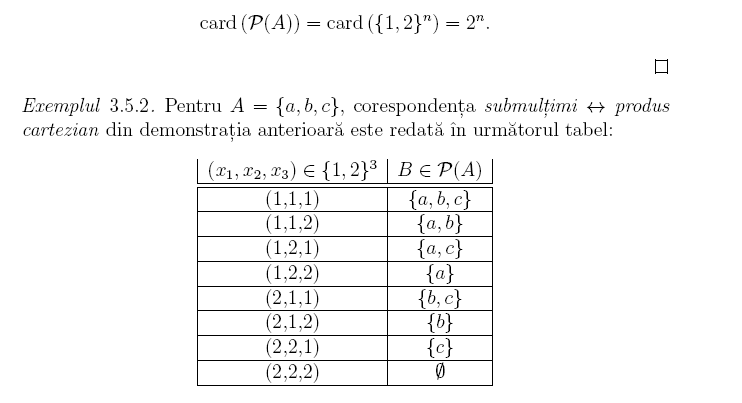
* Sortare Bubble Sort ⟶ O(n²)
* Căutare binară ⟶ O(log n)
* DFS/BFS într-un graf ⟶ O(V + E)

| **PRODUS CARTEZIAN**  import java.io.Reader;  import java.io.StringReader;  import java.util.Scanner;  class Main {  static Scanner cin = new Scanner(System.in);  static public int[] m;  static public char[][] a;  public static void PRODUS\_CARTEZIAN(int n, int[] m){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while(k>0)  {  if(x[k]<m[k]){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=0;  }  }  else{  k--;  }  }  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(a[i][x[i]]);  System.out.println();  }  public static void SETEAZA\_LUNGIMI\_SIRURI(int n){  m = new int[n+1];  a = new char[n+1][];  System.out.print("Citeste toate lungimile de sir 'm': ");  for(int i=1; i<=n; i++){  m[i] = cin.nextInt();  a[i] = new char[m[i]+1];  }  //Goleste caracterul \n  cin.nextLine();  }  public static void CITESTE\_SIRURI(int n){  for(int i=1; i<=n; i++){  System.out.print("Citeste Sir" + i + " de lungime " + m[i] + ": ");  String line;  do{  line = cin.nextLine();  if(line.length() < m[i])System.out.println("Linie prea scurta! Incearca din nou.");  }  while(line.length() < m[i]);  line = line.substring(0, m[i]); //Taie toate caracterele peste lungimea maxima.  for(int j=0; j<m[i]; j++){  a[i][j+1] = line.charAt(j); //Stocheaza la pozitia corecta.  // "j" incepe de la 0 deoarece String incepe tot de la 0.  }  }  }  public static void main(String[] args) {  int n;  System.out.print("n=");  n=cin.nextInt();  SETEAZA\_LUNGIMI\_SIRURI(n);  CITESTE\_SIRURI(n);  PRODUS\_CARTEZIAN(n,m);  }  } | Exemplu:  n=3 //nr siruri  Citește toate lungimile de șir 'm': 221  Citeste Sir1 de lungime 2: ab  Citeste Sir2 de lungime 2: +-  Citeste Sir3 de lungime 1: с  Rezultat:  a+c  a-c  b+c  b-c |
| --- | --- |

**SUBMULTIMI**

| import java.util.Scanner;  class Main {  public static void SUBMULTIMI\_STANDARD(int n){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while(k>0)  {  if(x[k]<2){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=0;  }  }  else k--;  }  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)if(x[i]==1) System.out.print(i + " ");  System.out.println();  }  public static void main(String[] args) {  int n;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  SUBMULTIMI\_STANDARD(n);  }  } | import java.util.Scanner;  class Main {  static public int[] a;  public static void SUBMULTIMI\_OARECARE(int[] a, int n){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while(k>0)  {  if(x[k]<2){  x[k]++;  if(k==n)AFISARE(x,a,n);  else{  k++;  x[k]=0;  }  }  else k--;  }  }  public static void AFISARE(int[] x, int[] a, int n){  for(int i=1; i<=n; i++)if(x[i]==1) System.out.print(a[i] + " ");  System.out.println();  }  public static void main(String[] args) {  int n;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  a = new int[n+1];  System.out.print("Citeste toate nr oarecare: ");  for(int i=1; i<=n; i++){  a[i] = cin.nextInt();  }  SUBMULTIMI\_OARECARE(a,n);  }  } |
| --- | --- |

Deci A are 2^3 = 8 submultimi



**Aranjamente (cu repetitie)**

Nr cuvinte de lungime n peste un alfabet cu m litere = m

Pentru m = 2 și n = 3, aranjamente cu repetiție sunt, în ordine lexicografică:

(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2), (2, 1, 1), (2, 1, 2), (2, 2, 1), (2, 2, 2).

| import java.util.Scanner;  class Main {  public static void ARANJAMENTE\_CU\_REPETITII\_STANDARD(int m, int n){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while (k > 0) {  if (x[k] < m) {  x[k]++;  if (k == n) AFISARE(x, n);  else {  k++;  x[k] = 0; // Reset  }  } else {  k--; // Backtrack  }  }  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }  public static void main(String[] args) {  int n, m;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  System.out.print("m=");  m=cin.nextInt();  ARANJAMENTE\_CU\_REPETITII\_STANDARD(m,n);  }  } | import java.util.Scanner;  class Main {  static int[] a;  public static void ARANJAMENTE\_CU\_REPETITII\_OARECARE(int m, int n){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while (k > 0) {  if (x[k] <= m) {  x[k]++;  if (k == n) AFISARE(x, n);  else {  k++;  x[k] = 0; // Reset  }  } else {  k--; // Backtrack  }  }  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(a[x[i]] + " ");  System.out.println();  }  public static void main(String[] args) {  int n, m;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  a = new int[n+1];  System.out.print("m=");  m=cin.nextInt();  System.out.print("Numere oarecare=");  for(int i=1; i<=n; i++){  a[i] = cin.nextInt();  }  ARANJAMENTE\_CU\_REPETITII\_OARECARE(m,n);  }  } |
| --- | --- |

**Aranjamente**

Nr cuvinte de lungime n **(cu litere distincte)** pt un alfabet cu m litere este [m]n

Pt m=4, n=2, aranjamentele sunt in ordine lexicografica:

(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3).

Deci avem [4]2 = 4 · 3 = 12 aranjamente.

Formula aranjamente [m]n =

| import java.util.Scanner; //**STANDARD**  class Main {  public static void ARANJAMENTE\_STANDARD(int m, int n){  int k=1;  int[] x;  x= new int[m+1];  x[1]=0;  while(k>0)  {  if(x[k]<m){  x[k]++;  if(VALID(x,k))  {  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=0;  }  }  }  else k--;  }  }  public static boolean VALID(int[] x, int k){  for(int i=1;i<=k-1;i++){  if(x[k]==x[i])return false;  }  return true;  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }  public static void main(String[] args) {  int n, m;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  System.out.print("m=");  m=cin.nextInt();  ARANJAMENTE\_STANDARD(m,n);  }  } | import java.util.Scanner; //**OARECARE**  class Main {  static int[] a;  public static void ARANJAMENTE1(int m, int n){  int k=1;  int[] x;  x= new int[m+1];  x[1]=0;  while(k>0)  {  if(x[k]<m){  x[k]++;  if(VALID(x,k))  {  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=0;  }  }  }  else k--;  }  }  public static boolean VALID(int[] x, int k){  for(int i=1;i<=k-1;i++){  if(x[k]==x[i])return false;  }  return true;  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(a[x[i]] + " ");  System.out.println();  }  public static void main(String[] args) {  int n, m;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  System.out.print("m=");  m=cin.nextInt();  a = new int[m+1];  System.out.print("Numere oarecare=");  for(int i=1; i<=m; i++)a[i]=cin.nextInt();  ARANJAMENTE1(m,n);  }  } |
| --- | --- |

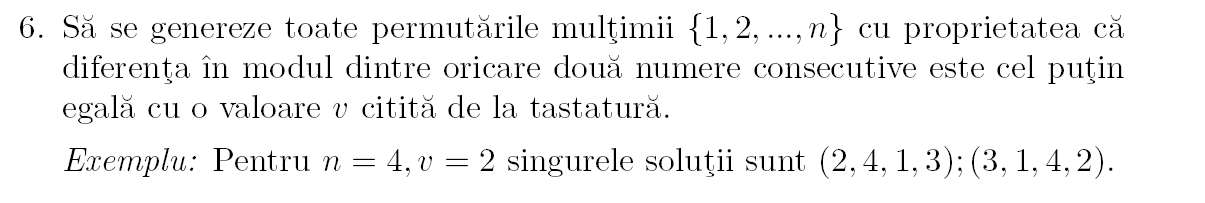
**Permutari**

Nr cuvinte de lungime ce conțin exact o data fiecare litera dintr-un alfabet cu “n” litere = n! (0! = 1)

n=3 => (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)

| import java.util.Scanner; //**STANDARD**  class Main {    public static void PERMUTARI1(int n){  int k=1;  int x[];  x= new int[n+1];  x[1]=0;    while(k>0)  {  if(x[k]<n){  x[k]++;  if(VALID(x,k)==1){  if(k==n){  AFISARE(x,n);  k--;  }  else{  k++;  x[k]=0;  }  }  }  else k--;  }  }    public static int VALID(int[] x, int k){  for(int i=1; i<=k-1; i++)if(x[k]==x[i]) return 0;  return 1;  }    public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }    public static void main(String[] args) {  int n;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();    PERMUTARI1(n);  }  } | import java.util.Scanner; //**OARECARE**  class Main {  static public int y[];    public static void PERMUTARI1(int n){  int k=1;  int x[];  x= new int[n+1];  x[1]=0;    while(k>0)  {  if(x[k]<n){  x[k]++;  if(VALID(x,k)==1){  if(k==n){  AFISARE(x,n);  k--;  }  else{  k++;  x[k]=0;  }  }  }  else k--;  }  }    public static int VALID(int[] x, int k){  for(int i=1; i<=k-1; i++)if(x[k]==x[i]) return 0;  return 1;  }    public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(y[x[i]] + " ");  System.out.println();  }    public static void main(String[] args) {  int n;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();    y = new int[n+1];  System.out.print("Citeste toate nr oarecare: ");  for(int i=1; i<=n; i++){  y[i] = cin.nextInt();  }    PERMUTARI1(n);  }  } |
| --- | --- |

Formula combinari [m]n =

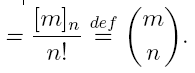


| import java.util.Scanner;  class Main {  static public int[] y;  static public int v;  public static void PERMUTARI\_OPTIONAL(int n){  int k=1;  int[] x;  x= new int[n+1];  x[1]=0;  while(k>0)  {  if(x[k]<n){  x[k]++;  if(VALID(x,k)==1){  if(k==n){  AFISARE(x,n);  k--;  }  else{  k++;  x[k]=0;  }  }  }  else k--;  }  }  public static int VALID(int[] x, int k){  for(int i=1; i<=k-1; i++) if(x[k]==x[i]) return 0;  if(Math.abs(y[x[k]] - y[x[k-1]]) < v) return 0;  return 1;  }  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(y[x[i]] + " ");  System.out.println();  }  public static void main(String[] args) {  int n;  Scanner cin = new Scanner(System.in);  System.out.print("n=");  n=cin.nextInt();  y = new int[n+1];  System.out.print("Citeste toate nr oarecare: ");  for(int i=1; i<=n; i++){  y[i] = cin.nextInt();  }  System.out.print("v="); v=cin.nextInt();  PERMUTARI\_OPTIONAL(n);}} |
| --- |

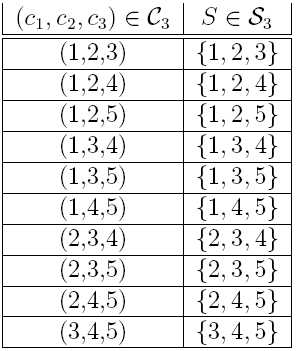
**Combinari**

Nr cuvinte STRICT crescătoare de lungime “n” cu alfabet “m” litere =

= nr submultimi cu “n” nr ale unei mulțimi cu “m” nr



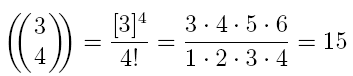
m=5 n=3



| class Main {  public static Scanner cin;    public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }    public static void COMBINARI(int m, int n){  int k=1;  int[] x = new int[n+1];  x[1]=0;    while(k>0){  if(x[k]<m-n+k){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=x[k-1];  }  }  else{  k--;  }  }  }    public static void main(String[] args) {  int m,n;  cin = new Scanner(System.in);    System.out.print("n=");  n = cin.nextInt();  System.out.print("m=");  m = cin.nextInt();  COMBINARI(m,n); }} | class Main {  public static Scanner cin = new Scanner(System.in);  public static int[] a;    public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(a[x[i]] + " ");  System.out.println();  }    public static void CITIRE(int[] a, int n){  for(int i=1; i<=n; i++)a[i] = cin.nextInt();}    public static void COMBINARI(int m, int n){  int k=1;  int[] x = new int[m+1];  x[1]=0;    while(k>0){  if(x[k]<m-n+k){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=x[k-1];  }  }  else{  k--;  }  }  }  public static void main(String[] args) {  int m,n;  System.out.print("n=");  n = cin.nextInt();  System.out.print("m=");  m = cin.nextInt();  a = new int[m+1]; CITIRE(a, m); COMBINARI(m,n);}} |
| --- | --- |

**Combinari (cu repetitie)**

Nr cuvinte STRICT crescătoare de lungime “n” cu alfabet **(ordonat)** “m” litere =

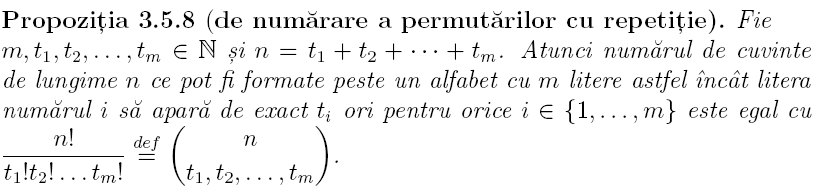
(1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 1, 3), (1, 1, 2, 2), (1, 1, 2, 3),

(1, 1, 3, 3), (1, 2, 2, 2), (1, 2, 2, 3), (1, 2, 3, 3), (1, 3, 3, 3),

(2, 2, 2, 2), (2, 2, 2, 3), (2, 2, 3, 3), (2, 3, 3, 3), (3, 3, 3, 3).

| import java.util.Scanner;  class Main {  public static Scanner cin;  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }  public static void COMBINARI\_REPETITIE(int m, int n){  int k=1;  int[] x = new int[n+1];  x[1]=0;  while(k>0){  if(x[k]<m){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=x[k-1]-1;  }  }  else{  k--;  }  }  }  public static void main(String[] args) {  int m,n;  cin = new Scanner(System.in);  System.out.print("n=");  n = cin.nextInt();  System.out.print("m=");  m = cin.nextInt();  COMBINARI\_REPETITIE(m,n);  }  } | import java.util.Scanner;  class Main {  public static Scanner cin = new Scanner(System.in);  public static int[] a;  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(a[x[i]] + " ");  System.out.println();  }  public static void CITIRE(int[] a, int n){  for(int i=1; i<=n; i++)a[i] = cin.nextInt();  }  public static void COMBINARI\_REPETITIE\_OARECARE(int m, int n){  int k=1;  int[] x = new int[n+1];  x[1]=0;  while(k>0){  if(x[k]<m){  x[k]++;  if(k==n)AFISARE(x,n);  else{  k++;  x[k]=x[k-1]-1;  }  }  else{  k--;  }  }  }  public static void main(String[] args) {  int m,n;  System.out.print("n=");  n = cin.nextInt();  System.out.print("m=");  m = cin.nextInt();  a = new int[m+1];  CITIRE(a, m);  COMBINARI\_REPETITIE\_OARECARE(m,n);  }  } |
| --- | --- |

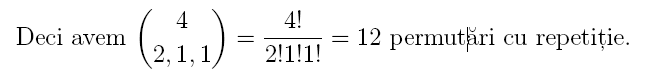
**Permutari (cu repetitie)**



Pentru m = 3, t1 = 2, t2 = t3 = 1, deci n = 4, permutarile cu repetitie sunt, in ordine lexicografica:

(1, 1, 2, 3), (1, 1, 3, 2), (1, 2, 1, 3), (1, 2, 3, 1), (1, 3, 1, 2), (1, 3, 2, 1),

(2, 1, 1, 3), (2, 1, 3, 1), (2, 3, 1, 1), (3, 1, 1, 2), (3, 1, 2, 1), (3, 2, 1, 1).

(deci litera 1 apare de 2 ori, litera 2 si 3 apare 1 data)

| import java.util.Scanner; **//DOAR STANDARD**  class Main {  public static Scanner cin;  public static void AFISARE(int[] x, int n){  for(int i=1; i<=n; i++)System.out.print(x[i] + " ");  System.out.println();  }  public static void CITESTE(int[] t, int n){  for(int i=1; i<=n; i++)t[i] = cin.nextInt();  }  public static void PERMUTARI\_CU\_REPETITIE\_STANDARD(int n, int[] t, int m){  n=0;  for(int i=1; i<=m; i++)n=n+t[i];  int k=1;  int[] x = new int[n+1];  x[1]=0;  while(k>0){  if(x[k]<m){  x[k]++;  int p = t[x[k]];  if(VALID(p,x,k)){  if(k==n){  AFISARE(x,n);  k--;  }  else{  k++;  x[k]=0;  }  }  }  else{  k--;  }  }  } | //**DOAR STANDARD**  public static boolean VALID(int p, int[] x, int k){  int q=0;  if(p==0)return false;  for(int i=1; i<=k-1; i++){  if(x[i]==x[k]){  q++;  if(q>=p) return false;  }  }  return true;  }  public static void main(String[] args) {  int m,n; int[] t;  cin = new Scanner(System.in);  System.out.print("n=");  n = cin.nextInt();  System.out.print("m=");  m = cin.nextInt();  t = new int[m+1];  CITESTE(t,m);  PERMUTARI\_CU\_REPETITIE\_STANDARD(n,t,m);  }  } |
| --- | --- |