Metoda Greedy

Metoda de programare **Greedy** se aplică problemelor de optimizare. Aceasta metoda constă în faptul că se construieşte soluția optimă pas cu pas, la fiecare pas fiind selectat în soluție elementul care pare „**cel mai bun/cel mai optim**" la momentul respectiv, în speranța că această alegere locală va conduce la **optimul globa**l.

Algoritmii Greedy sunt foarte eficienți, dar nu conduc în mod necesar la o soluție optimă. Şi nici nu este posibilă formularea unui criteriu general conform căruia să putem stabili exact dacă metoda Greedy rezolvă sau nu o anumită problemă de optimizare. Din acest motiv, orice algoritm Greedy trebuie însoțit de o demonstrație a corectitudinii sale . Demonstrația faptului că o anumită problemă are proprietatea alegerii Greedy se face de obicei prin inducție matematică.

Metoda **Greedy** se aplică problemelor pentru care se dă o **mulţime A** cu **n** elemente şi pentru care trebuie determinată o **submulţime** a sa**, S** cu **m** elemente, care îndeplinesc anumite condiţii, numite si **condiții de optim**. Algoritmul în limbaj natural al metodei de programare Greedy are următoarea structură:

Algoritm Greedy:

- **se dă o mulţime A**
- **se cere o submulţime S din mulțimea A care să:**
- **îndeplinească anumite condiţii interne (să fie acceptabilă)**
- **fie optimală (să realizeze un maxim sau un minim).**

Principiul metodei Greedy:

- se **iniţializează** mulţimea soluţiilor S cu mulţimea vidă, S=Ø
- la fiecare pas se alege un anumit element **x∈A** (cel mai promiţător element la momentul respectiv) care poate conduce la o soluţie optimă
- se verifică dacă elementul ales poate fi adăugat la mulţimea soluţiilor:
- **dacă** da **atunci**
- va fi adăugat şi mulţimea soluţiilor devine S=S∪{x} - un element introdus în mulţimea S nu va mai putea fi eliminat
- **altfel**
- el nu se mai testează ulterior
- procedeul continuă, până când au fost determinate toate elementele din mulţimea soluţiilor

Aplicații:

1. Sumă maximă – produs maxim

Se dă o mulţime A={a_1, a_2, . . ., a_n} cu elemente reale. Să se determine o submulţime a lui S astfel încât

- **suma elementelor submulţimii să fie maximă.**
- **produsul elementelor submulțimii să fie maxim**

Exemplu: dacă **N=10** şi elementele mulțimii A sunt **(-3, 7, 9, -2, 19, 20, -10, 15, -20, -5).** Rezultatul aşteptat este:

- **suma maximă** are valoarea **70** si se obține adunând doar valorile strict pozitive : **7+9+19+20+15**
- **produsul maxim** are valoarea **1077300000** si se obține înmulțind cele mai mici numere negative (se vor înmulți un număr par de valori negative) si toate numerele strict pozitive : **(-20)*(-10)*(-5)*(-3)*7*9*15*19*20**

suma elementelor submulţimii să fie maximă

```
#include <iostream>
#include <fstream>

using namespace std;

ifstream f("suma.in");

int a[100];
int n;

void citire( int a[],int &n)
{
	f>>n;
	for(int i=1;i<=n;i++)
	f>>a[i];
}

void afisare (int a[], int n)
{
	cout<<"Multimea A-{ ";
	for(int i=1;i<=n;i++)
	cout<<a[i]<<" ";
	cout<<"}"<<endl;
}

int main()
{
	 int i,sumaxim=0;
	citire(a,n);
	afisare(a,n);
```

```
	cout<<"Submultimea de suma maxima este S-{";
	for(i=1;i<=n;i++)
		if(a[i]>=0)
		{
			cout<<a[i]<<" ";
			sumaxim=sumaxim+a[i];
		}
	cout<<"}"<<endl;
	cout<<"Valoarea sumei elemetelor este :"<<sumaxim;
	return 0;
}
```

produsul elementelor submulțimii sa fie maxim

```
#include <iostream>
#include <fstream>
using namespace std;
int a[50];
int n;
ifstream f("numere.in");

void citire(int a[], int &n)
{
	f>>n;
	for(int i=1;i<=n;i++)
	f>>a[i];
}
void afisare(int a[], int n)
{
	cout<<"Multimea A={ ";
	for(int i=1;i<=n;i++)
		cout<<a[i]<<" ";
	cout<<"}"<<endl;
}

void sortare(int a[], int n)
{
	int aux,ok;
	do
	{
		ok=1;
		for(int i=1;i<n;i++)
		if(a[i]>a[i+1])
			{ aux=a[i];a[i]=a[i+1];a[i+1]=aux;ok=0;}
	}while(ok==0);
}

//determin pozitia primului numar strict pozitiv din multimea a
int primapoz(int a[], int n)
{
```

```
	for(int i=1;i<=n;i++)
		if(a[i]>0)
			return i;
	return 0;
}

void greedy (int a[], int n)
{
	int pmax=1;
	cout<<"Submultimea de produs maxim este B={ ";
	//inmultesc toate numere strict pozitive
	for(int i=primapoz(a,n);i<=n;i++)
	{
		cout<<a[i]<<" ";
		pmax=pmax*a[i];
	}
	//inmultesc un numar par de numere negative
	if(primapoz(a,n)%2==0)
		for(int i=1;i<=primapoz(a,n)-primapoz(a,n)%2;i++)
		{
			cout<<a[i]<<" ";
			pmax=pmax*a[i];
		}
	cout<<" }"<<endl;
	cout<<"Produsul elementelor este: "<<pmax;
}
int main()
{
	citire(a,n);
	afisare(a,n);
	sortare(a,n);
	afisare(a,n);
	greedy(a,n);
	return 0;
}
```

2. Problema spectacolelor

Managerul artistic al unui festival trebuie să selecteze o mulțime cât mai amplă de spectacole ce pot fi jucate în singură sală pe care o are la dispoziție. Știind că i s-au propus **n≤100** spectacole şi pentru fiecare spectacol **i** i-a fost anunțat intervalul în care se poate desfăşura **[si, fi)** (**si** reprezintă ora şi minutul de început, iar **fi** ora şi minutul de final al spectacolului i)

Scrieți un program care să permită spectatorilor vizionarea unui număr cât mai mare de spectacole.

De exemplu, dacă vom citi n=5 şi următorii timpi:
12 30 16 30
15 0 18 0
10 0 18 30
18 0 20 45
12 15 13 0

Spectacolele selectate sunt: **5 2 4**.

```
#include <iostream>

#include <fstream>

using namespace std;

struct timp { int ora,minut;};

struct spectacol
{
	timp inc; // ora de inceput
	timp sf; // ora de sfarsit
	int nr; // nr de spectacole
};
spectacol a[100]; // memorez spectacolele intr-un vector de structuri
int n;
ifstream f("spectacol.in");

void citire (spectacol a[], int &n)
{
	f>>n;
	for(int i=1;i<=n;i++)
	{
		f>>a[i].inc.ora>>a[i].inc.minut;
		f>>a[i].sf.ora>>a[i].sf.minut;
		a[i].nr=i;
	}
}

void afisare (spectacol a[], int n)
```

```
{
	for(int i=1;i<=n;i++)
	{
		cout<<"Spectacolul "<<a[i].nr<<" incepe la "<<a[i].inc.ora<<":"<<a[i].inc.minut;
		cout<<" si se termina la "<<a[i].sf.ora<<":"<<a[i].sf.minut;
		cout<<endl;
	}
}


// orodonarea celor n spectacole dupa ora de sfarsit
// la ore de sfarsit egale se ordoneaza dupa minut

void sortare (spectacol a[], int n)
{
	int ok;// presupun sortat
	spectacol aux;
	do
	{
		ok=1;
		for(int i=1;i<n;i++)
		if(a[i].sf.ora>a[i+1].sf.ora)//primul criteriu
		{
			aux=a[i];
			a[i]=a[i+1];
			a[i+1]=aux;
			ok=0;
		}
		else
			if(a[i].sf.ora==a[i+1].sf.ora)//al doilea criteriu
		if(a[i].sf.minut>a[i+1].sf.minut)
		{
			aux=a[i];
			a[i]=a[i+1];
			a[i+1]=aux;
			ok=0;
		}
	}while(ok==0);
}
void greedy(spectacol a[], int n)
{
	cout<<"Lista de spectacole selectate este:"<<endl;
	int nr=1;//cate spectacole am ales
	cout<<"Spectacolul "<<a[1].nr<<" incepe la "<<a[1].inc.ora<<":"<<a[1].inc.minut;
	cout<<" si se termina la "<<a[1].sf.ora<<":"<<a[1].sf.minut;
	cout<<endl;
	int ora_sf=a[1].sf.ora;
	int min_sf=a[1].sf.minut;
	for(int i=2;i<=n;i++)
		if(a[i].inc.ora>=ora_sf && a[i].inc.minut>=min_sf )
```

```
                {
                        nr++;
                        cout<<"Spectacolul "<<a[i].nr<<" incepe la ";
                        cout<<a[i].inc.ora<<":"<<a[i].inc.minut;
                        cout<<" si se termina la "<<a[i].sf.ora<<":"<<a[i].sf.minut;
                        cout<<endl;
                        //pastrez ora de sfarsit a ultimului spectacol selectat
                        ora_sf=a[i].sf.ora;
                        min_sf=a[i].sf.minut;
                }
        cout<<"Au fost selectate maximum "<<nr<<" spectacole !";
}

int main()
{
        citire(a,n);
        cout<<"Lista de spectacole initiala este:"<<endl;
        afisare(a,n);
        sortare(a,n);
        cout<<"Lista de spectacole ordonate este:"<<endl;
        afisare(a,n);
        cout<<endl;
        greedy(a,n);
        return 0;
}
```

3. Problema continuă a rucsacului

Un hoţ neprevăzător are la dispoziție un singur rucsac cu care poate transporta o greutate maximă **Gmax**. Hoţul are de ales din **n≤50** obiecte şi, evident, intenționează să obțină un **câştig maxim** în urma singurului transport pe care îl poate face. Cunoscând pentru fiecare obiect **i** greutatea **gi** şi câştigul **ci** pe care hotul l-ar obține transportând obiectul respectiv în întregime, scrieți un program care să determine o încărcare optimală a rucsacului. Obiectele nu pot fi secționate

De exemplu, pentru **n=5, GMax=100** şi greutățile-câștigurile următoare:
(1000 120), (500 20), (400 200), (1000 100), (25 1) se va afişa pe ecran:

```
5 obiecte
100 greutate maxima a rucsacului
Initial:
Lista de obiecte este:
Obiectul 1 are greutatea 1000 si costul de: 120
Obiectul 2 are greutatea 500 si costul de: 20
Obiectul 3 are greutatea 400 si costul de: 200
Obiectul 4 are greutatea 1000 si costul de: 100
Obiectul 5 are greutatea 25 si costul de: 1


Dupa sortare:
Lista de obiecte este:
Obiectul 3 are greutatea 400 si costul de: 200
Obiectul 1 are greutatea 1000 si costul de: 120
Obiectul 4 are greutatea 1000 si costul de: 100
Obiectul 2 are greutatea 500 si costul de: 20
Obiectul 5 are greutatea 25 si costul de: 1


Greedy:

Obiectul cu numarul: :5 greutatea=  25 cost=  1
Cost maxim= 1
```

Iar pentru, pentru **n=5, GMax=1000** şi greutățile-câștigurile următoare:
(1000 120), (500 20), (400 200), (1000 100), (25 1) se va afişa pe ecran:

```
5 obiecte
1000 greutate maxima a rucsacului
Initial:
Lista de obiecte este:
Obiectul 1 are greutatea 1000 si costul de: 120
Obiectul 2 are greutatea 500 si costul de: 20
Obiectul 3 are greutatea 400 si costul de: 200
Obiectul 4 are greutatea 1000 si costul de: 100
Obiectul 5 are greutatea 25 si costul de: 1
```

Dupa sortare:
Lista de obiecte este:
Obiectul 3 are greutatea 400 si costul de: 200
Obiectul 1 are greutatea 1000 si costul de: 120
Obiectul 4 are greutatea 1000 si costul de: 100
Obiectul 2 are greutatea 500 si costul de: 20
Obiectul 5 are greutatea 25 si costul de: 1

Greedy:

Obiectul cu numarul: :3 greutatea= 400 cost= 200
Obiectul cu numarul: :2 greutatea= 500 cost= 20
Obiectul cu numarul: :5 greutatea= 25 cost= 1
Cost maxim= 221

```
#include <iostream>
#include <fstream>

using namespace std;

ifstream f("rucsac.in");

struct obiect
{
    float greutate;
    int numar;
    int cost;
};
obiect a[100];

int gmax,n;

void citire (obiect a[],int &n)
{
    f>>n;
    cout<<n<<" obiecte\n";
    f>>gmax;
    cout<<gmax<<" greutate maxima a
```

```
rucsacului\n";
     for(int i=1;i<=n;i++)
     {
          f>>a[i].greutate;
          f>>a[i].cost;
          a[i].numar=i;
     }
}

void afisare (obiect a[],int n)
{
     cout<<"Lista de obiecte este: "<<endl;
     for(int i=1;i<=n;i++)
          cout<<"Obiectul "<<a[i].numar<<" are
greutatea "<<a[i].greutate<<" si costul de:
"<<a[i].cost<<endl;
     cout<<endl;
}

void sortare(obiect a[],int n)
{
     int ok;
     obiect aux;
     do
          {
               ok=1;
               for(int i=1;i<=n-1;i++)
                    if(a[i].cost<a[i+1].cost)

{aux=a[i];a[i]=a[i+1];a[i+1]=aux;ok=0;}
                    else
                         if(a[i].cost==a[i+1].cost)

if(a[i].greutate>a[i+1].greutate)
                                        {
aux=a[i];a[i]=a[i+1];a[i+1]=aux;ok=0;}
          }
     while(ok==0);
```

```
}

void greedy(obiect a[],int n)
{
     int s=0;
     for(int i=1;i<=n;i++)
          if(a[i].greutate<=gmax)
          {
               cout<<"\nObiectul cu numarul:
:"<<a[i].numar<<" greutatea=
"<<a[i].greutate<<" cost=  "<<a[i].cost;
               s=s+a[i].cost;
               gmax=gmax-a[i].greutate;
          }
     cout<<"\nCost maxim= "<<s;
}

int main()
{
     citire(a,n);
     cout<<"Initial:\n";
     afisare(a,n);
     sortare(a,n);
     cout<<endl;
     cout<<"Dupa sortare:\n";
     afisare(a,n);
     cout<<endl;
     cout<<"Greedy: \n";
     greedy(a,n);
     return 0;
}
```

4. Maximizarea valorilor unei expresii

Se dau n numere întregi, nenule $b_1, b_2, \ldots, b_n$ şi m numere întregi nenule $a_1, a_2, \ldots, a_m$, cu m<=n. Să se determine o submulţime a mulţimii B={ $b_1, b_2, \ldots, b_n$ } care să maximizeze valoarea expresiei

$E= a_1* x_1 + a_2 * x_2+\ldots+ a_m * x_m$

ştiind că n mai mare sau egal cu **m** şi că x_i aparţine mulţimii { **$b_1, b_2, \ldots, b_n$** }

Exemplu: Pentru m=5 şi mulţimea A={5, 9, -4, 2, -3}, si n=8 şi mulţimea B={6, 1, -2, 4, -3, -5, -1, 8} expresia este:

$E= 5*x_1 + 9*x_2 - 4*x_3 + 2*x_4 - 3*x_5$

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("expresie.in");
int a[100],b[100],n,m;
int v[100],k;

void citire(int a[], int &m, int b[],int &n)
{
	f>>m>>n;
	for(int i=1;i<=m;i++)
		f>>a[i];
	for(int i=1;i<=n;i++)
		f>>b[i];
}

// functie de afisare ale ni vect V cu K elemente
// va fii folosita pentru un vector cu k elemente
//ch va contine o litera care reprezinta denumirea, sau denumirea numelui vectorului care se afiseaza;
void afisare(int v[],int k,char ch)
{
	cout<<"Elementele vectorului "<<ch<<" sunt: "<<endl;
	for(int i=1;i<=k;i++)
	cout<<v[i]<<" ";
	cout<<endl;
}

void ordonare(int v[], int k)
{
	int ok,aux;
	do
	{
		ok=1;
		for(int i=1;i<=k-1;i++)
			if(v[i]>v[i+1])
```

```
                                        {aux=v[i];v[i]=v[i+1];v[i+1]=aux;ok=0;}
        }while(ok==0);
}

void greedy(int a[],int m,int b[],int n)
{
        int e=0,i=1,j;
        //inmultesc perechi de numere negative
        while(a[i]<0 && b[i]<0 && i<=m)
        {
                cout<<a[i]<<" "<<b[i]<<endl;
                e=e+a[i]*b[i];
                i++;
        }
        //inmultesc perechi de numere pozitive
        i=m;
        j=n;
        while(a[i]>0 && b[j]>0 && i>=1)
        {
                e=e+a[i]*b[j];
                cout<<a[i]<<" "<<b[j]<<endl;
                i--;j--;
        }
        cout<<endl<<"Valoarea maxima a expresiei E, este: "<<e;
}

int main()
{
        citire(a,m,b,n);
        afisare(a,m,'a');
        afisare(b,n,'b');
        ordonare(a,m);
        ordonare(b,n);
        cout<<endl<<"Dupa sortare"<<endl;
        afisare(a,m,'a');
        afisare(b,n,'b');
        cout<<endl;
        greedy(a,m,b,n);
        return 0;
}
```

5. Problema comisului voiajor

Un comis voiajor trebuie să viziteze un număr n de oraşe şi să se întoarcă în oraşul de plecare astfel încât să nu treacă de două ori prin acelaşi loc. Cunoscând legăturile dintre oraşe şi distanţele corespunzătoare, să se afişeze traseul de lungime minimă pe care să-l parcurgă comisul voiajor.

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
ifstream f("comis.in");
//declararea variabilelor
int a[20][20];//matricea costurilor
int v[20];//vectorul vizitat
int n;//nr de orase
int ns;//numarul orasului de start
int cost;//costul traseului
int t[20];//vector care pastreaza traseul
// t[i] este un nr de la 1 la n
// cu conditia:
// 1. a[t[i]][t[i+1]] != Infinit
// 2. a[t[1]][t[n]] != Infinit
//construirea matricei costurilor
void citire(int a[20][20], int &n, int &ns)
{
int x,y,i,j;
f>>n;
f>>ns;
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(i==j)//daca sunt pe d.p
a[i][j]=0;
else
a[i][j]=100000;
while(f>>x>>y>>cost)
a[x][y]=a[y][x]=cost;
}
void afisare (int a[20][20], int n)
{
cout<<"Matricea costurilor este:"<<endl;
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n;j++)
if(a[i][j]==100000)
{
cout.width(5);
cout<<'.'<<" ";
```

```
}
else
{ cout.width(5);
cout<<a[i][j]<<" ";
}
cout<<endl;
}
}
//functia care determina daca am vizitat toate nodurile
int vizitat (int v[20], int n)
{
for (int i=1;i<=n;i++)
if(v[i]==0)
return 1;
return 0;
}
int prim_oras_nevizitat(int a[20][20], int n, int x)
{
// minimul de pe linia x != 0 si nevizitat
int minim=100000;
int pmin=0; //pozitia minimului
for (int i=1;i<=n;i++)
if(a[x][i]>0 && a[x][i]<100000)
if(a[x][i]<minim && v[i]==0)
{
minim=a[x][i];
pmin=i;
}
return pmin;
}
void greedy (int a[20][20], int n, int ns)
{
int x; // un nod de pe traseu
int cost=0; //costul traseului initial cu 0
int gata=1; //presupun ca am solutie
int ultim=1; //pastreaza pozitia ultimului nod din traseu
t[1]=ns; //memorez nodul de start ca prim nod in solutie
v[ns]=1; //marchez vizitat nodul de start
while(vizitat(v,n)!=0 && gata)
{
//determin primul nod nevizitat care se leaga cu t[ultim]
// si se afla la distanta minima
x=prim_oras_nevizitat(a,n,t[ultim]);
if(x!=0)
{
v[x]=1;//marchez vizitat orasul x
ultim++;
t[ultim]=x;//adaug orasul X la solutie
cost=cost+a[t[ultim-1]][t[ultim]];
}
```

```
else
gata=0;
}
//afisare solutie
if(gata==0)
cout<<"Nu exista solutie!";
else
if(a[t[ultim]][t[1]]>0 and a[t[ultim]][t[1]]<100000)
{
cout<<endl<<"Traseul este :";
for(int i=1;i<=n;i++)
cout<<t[i]<<" ";
cout<<t[1]<<endl;
cost=cost+a[t[ultim]][t[1]];
cout<<"Costul minim este: "<<cost;
}
}
int main()
{
citire(a,n,ns);
afisare(a,n);
greedy(a,n,ns);
return 0;
}
```