



TEHNICA DE PROGRAMARE GREEDY

Proiect realizat de:
Mihaela Frecăuțanu, clasa a XI-a C

INTRODUCERE

Metoda de programare **Greedy** se aplică problemelor de optimizare.

Aceasta metoda constă în faptul că se construiește 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 global**.

Un algoritm greedy este potrivit atunci când trebuie să luăm o serie de decizii, și anume pe cea mai conveabilă la un moment dat.

1

Se **inițializează** mulțimea soluțiilor S egală cu mulțimea vidă, $S=\emptyset$

2

La fiecare pas se alege un anumit element $x \in A$ (cel mai promițător element la momentul respectiv) care poate conduce la o soluție optimă

3

Se verifică dacă elementul ales poate fi adăugat la mulțimea soluțiilor:

- **dacă** da **atunci** va fi adăugat și mulțimea soluțiilor devine $S=S \cup \{x\}$ - un element introdus în mulțimea S nu va mai putea fi eliminat
- **Altfel** el nu se mai testează ulterior

Procedeul continuă, până când au fost determinate toate elementele din mulțimea soluțiilor.

Această tehnică este folosită într-un mare număr de probleme și aplicații printre care:

Minimalizarea timpului mediu de așteptare

Interclasarea optimă a șirurilor ordonate

Coduri Huffman

Cele mai scurte drumuri care pleacă din același punct

Problema comis-voiajorului

Arbori parțiali de cost minim

EXEMPLE

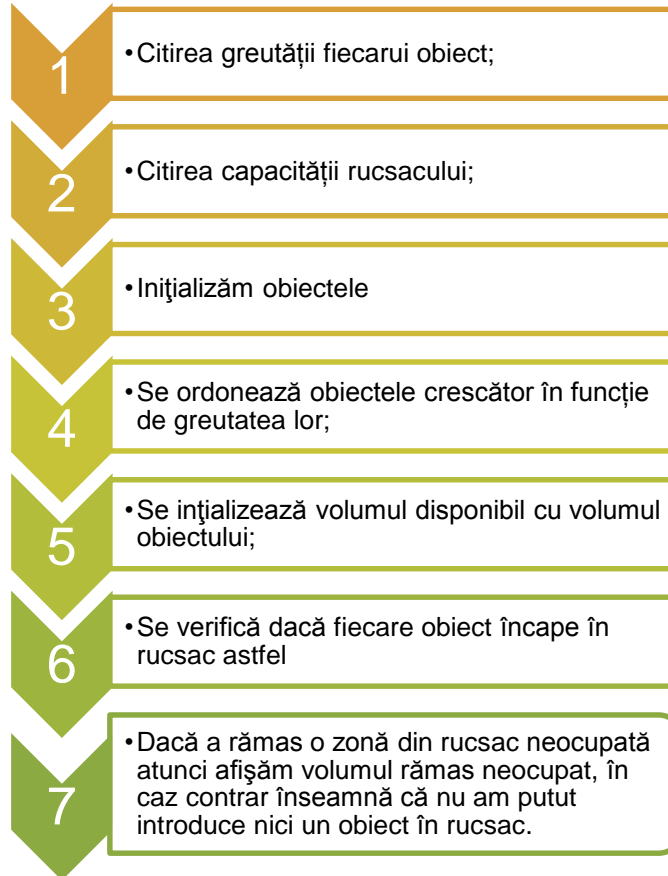
de programe Pascal

Probleme
rezolvate prin
metoda Greedy

1) Problema continuă a rucsacului

Sarcina: Se consideră n obiecte. Pentru fiecare obiect i ($i=1, 2, \dots, n$) se cunoaște greutatea g_i și câștigul c_i care se obține în urma transportului său la destinație. O persoană are un rucsac cu care pot fi transportate unul sau mai multe obiecte, greutatea sumară a cărora nu depășește valoarea G_{\max} . Elaborați un program care determină ce obiecte trebuie să transporte persoana în așa fel încât câștigul să fie maxim. În caz de necesitate, unele obiecte pot fi tăiate în fragmente mai mici.

Algoritm:



```

program rucsac;
Var g:array [1..10] of integer;
    i,n,Gm,R, aux : integer;
    ok:boolean;
begin
writeln('nr obiecte'); readln(n);
writeln('capacitate rucsac'); readln(R);
writeln(' Obiectele de luat in rucsac:' );
for i:=1 to n do
    read (g[i]);
ok:=false;
while(ok=false) do
begin
ok:=true;
for i:=1 to n-1 do
if g[i]>g[i+1] then

```

```

begin
aux:=g[i];
g[i]:=g[i+1];
g[i+1]:=aux;
ok:=false;
end;
end;
writeln; for i:=1 to n do write( g[i], ' ');
Gm:=0 ;
i:=1;
while ( Gm +g[i]<=R ) do
begin
Gm:=Gm+g[i];
i:=i+1;
end;
writeln('sunt' ,i-1,'obiecte cu greutate', Gm) ;
writeln ( ' a ramas' , R-Gm,' loc liber' );
end.

```

Răspuns:

Окно вывода

```

nr obiecte
7
capacitate rucsac
15
Obiectele de luat in rucsac:
2
1
7
10
18
19
25

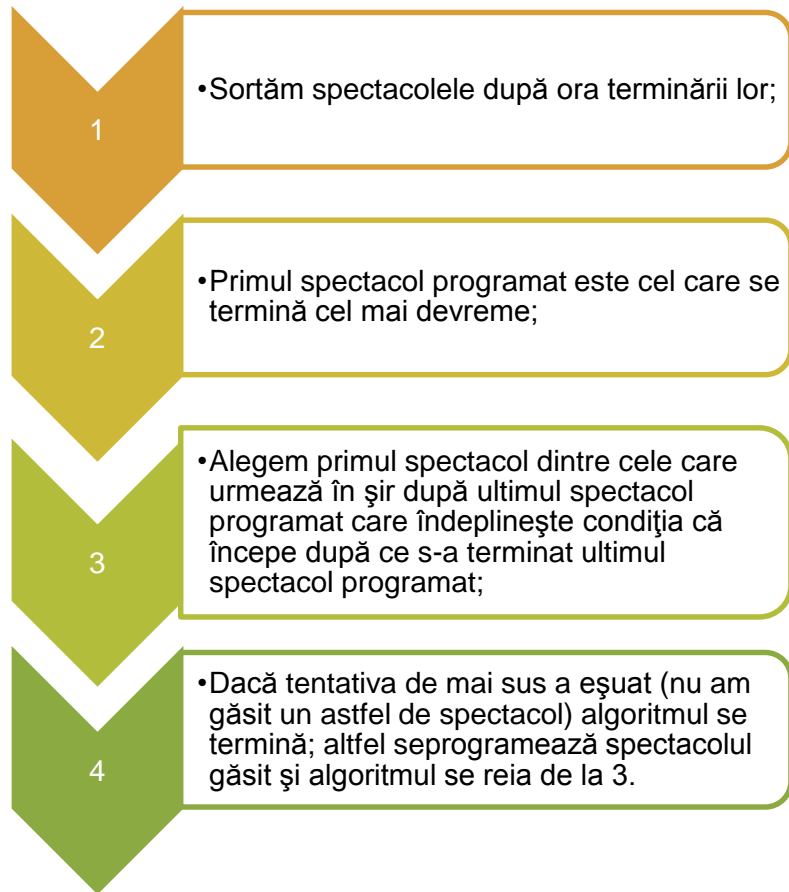
1*2*7*10*18*19*25*sunt 3 obiecte cu greutate 10
a ramas 5 loc liber

```

2) Problema spectacolelor

Algoritm:

Sarcina: Într-un oraș de provincie se organizează un festival de teatru. Orașul are o singură sală de spectacole, iar la festival și-au anunțat participarea mai multe trupe. Așadar, în sală, într-o zi, trebuie planificate N spectacole. Pentru fiecare spectacol se cunoaște intervalul în care se desfășoară: [ora_inceput, ora_sfarsit]. Se cere să se planifice un număr maxim de spectacole care, bineînțeles, nu se pot suprapune.




```

Program spectacole;
Type spectacol=record
  ora_inc, ora_sf:integer;
ord:integer;
end;
Var v:array[1..30] of spectacol;
n, ultim, nr:integer;
procedure sortare;
var i,j :integer; aux:spectacol;
begin
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if v[j].ora_sf < v[i].ora_sf then
        begin
          aux:=v[j];
v[j]:=v[i];
v[i]:=aux;
        end;
      end;
    end;
  procedure citire;
  var hh, mm, i:integer;
  begin
    write('Numarul de spectacole:');
    readln(n);
    for i:=1 to n do
      begin
        write('Spectacolul, i, incepe la:');
        readln(hh,mm);
        v[i]. ora_inc:=hh*60+mm;
        write ('Spectacolul, i, se termina la:');

```

```

        readln(hh,mm);
        v[i].ora_sf:=hh*60+mm;
        v[i].ord:=i;
      end;
    end;
  procedure greedy;
  var i:integer;
  begin
    writeln('Ordinea spectacolelor este:');
    ultim:=1;
    nr:=1;
    write(v[1].ord, ' ');
    for i:=2 to n do
      if v[i].ora_inc>v[ultim].ora_sf then
        begin
          write(v[i].ord, ' ');
          ultim:=i;
          Inc(nr);
        end;
    writeln('Se pot juca ', nr, ' spectacole');
  end;
  begin
    citire;
    sortare;
    greedy;
  end.

```

Răspuns:

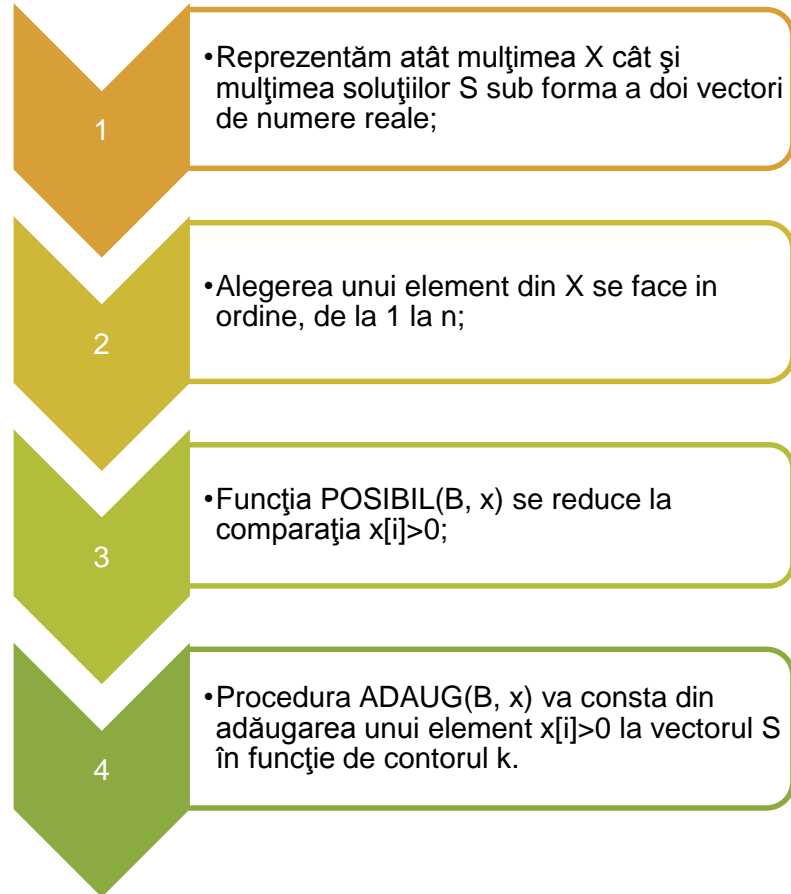
Окно вывода

```
Numarul de spectacole:3  
Spectacolul, i, incepe la:15  
60  
Spectacolul, i, se termina la:17  
20  
Spectacolul, i, incepe la:19  
30  
Spectacolul, i, se termina la:21  
00  
Spectacolul, i, incepe la:22  
30  
Spectacolul, i, se termina la:24  
00  
Ordinea spectacolelor este:  
1 2 3 Se pot juca 3 spectacole
```

3) Problema sumei maxime

Sarcina: Se dă o mulțime $X=\{x_1, x_2, \dots, x_n\}$ cu elemente reale. Să se determine o submulțime a lui X astfel încât suma elementelor submulțimii să fie maximă.

Algoritm:



```
program suma_maxima;  
var s,x:array[1..20] of real;  
i,k,n:integer;  
begin  
  write('Numarul de elemente n = ');  
  readln(n);  
  for i:=1 to n do  
    begin  
      write('x[' ,i, ']= ');  
      readln(x[i]);  
    end;  
    k:=0;  
    for i:=1 to n do  
      if x[i]>0 then  
        begin  
          k:=k+1;  
          s[k]:=x[i]  
        end;  
        for i:=1 to k do  
          write(s[i]:5:2, ' ');  
          readln;  
        end.
```

Răspuns:

Окно вывода

```
Numarul de elemente n = 5  
x[1]= 2  
x[2]= 6  
x[3]= 7  
x[4]= 8  
x[5]= 9  
2.00  6.00  7.00  8.00  9.00
```

4) Problema cu submulțimea

Sarcina: Elementele sale sunt numere reale, iar cel puțin unul din ele satisface condiția $a_i > 0$. Elaborați un program care determină o submulțime B, astfel încât suma elementelor din B să fie maximă.

```
Program Problema;
{ Tehnica Greedy }
const nmax=1000;
var A : array [1..nmax] of real;
n : 1..nmax;
B : array [1..nmax] of real;
m : 0..nmax;
x : real;
i : 1..nmax;
Function ExistaElemente : boolean;
var i : integer;
begin
ExistaElemente:=false;
for i:=1 to n do
if A[i]>0 then ExistaElemente:=true;
end; { ExistaElemente }
procedure AlegeUnElement(var x : real);
var i : integer;
begin
i:=1;
while A[i]<=0 do i:=i+1;
x:=A[i];
A[i]:=0;
end; { AlegeUnElement }
procedure IncludeElementul(x : real);
begin
```

```
m:=m+1;
B[m]:=x;
end; { IncludeElementul }
begin
write('Dați n='); readln(n);
writeln('Dați elementele mulțimii A:');
for i:=1 to n do read(A[i]);
writeln;
m:=0;
while ExistaElemente do
begin
AlegeUnElement(x);
IncludeElementul(x);
end;
writeln('Elementele mulțimii B:');
for i:=1 to m do writeln(B[i]);
readln;
end.
```

Răspuns:

Окно вывода

```
Dați n=6
Dați elementele mulțimii A:
1
2
91
283
32
375
```

```
Elementele mulțimii B:
1
2
91
283
32
375
```

5) Problema divizorilor naturali

Sarcina: Fiind dat numărul natural $k > 1$, se cere să se determine cel mai mic număr natural n având exact k divizori naturali proprii (diferiți de 1 și n).

```
program k_divizori_naturali;
var v:boolean;
    k,n,s,i:integer;
procedure VERIF(n,k:integer;var v:boolean);
var j,i:integer;
begin
    i:=0;
    for j:=2 to n-1 do
        if n mod j = 0 then
            i:=i+1;
        if i = k then
            v:=true
        else
            v:=false;
    end;
begin
    write('Numarul de divizori k > 1 ');
```

```
readln(k);
write('Cel mai mic numar care are exact ',k,' divizori este ');
n:=k+2;
s:=0;
while s = 0 do
begin
    VERIF(n,k,v);
    if v = true then
        begin
            write(n);
            s:=1;
        end;
        n:=n+1;
    end;
    readln;
end.
```

Răspuns:

Окно вывода

Numarul de divizori k > 1 4

Cel mai mic numar care are exact 4 divizori este 12



Concluzie:

- Metoda Greedy nu caută să determine toate soluțiile posibile (care ar putea fi prea numeroase) și apoi să aleagă din ele pe cea optimă, ci caută să introducă direct un element x în soluția optimă. Acest lucru duce la eficiența algorimilor Greedy, însă nu conduce în mod necesar la o soluție optimă și 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. Acest motiv duce la însoțirea fiecărei rezolvări prin metoda Greedy a unei demonstrații matematice (în general prin inducție).



BIBLIOGRAFIE:

- ▶ <http://www.worldit.info/articole/algoritmica-articole/metoda-greedy>
- ▶ <https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy>
- ▶ <https://www.scribd.com/doc/43454385/Metoda-Greedy#download>
- ▶ <https://www.infoarena.ro/metoda-greedy-si-problema-fractionara-a-rucsacului>
- ▶ *Anatol Gremalschi*, Informatică Manual pentru clasa a 11-a, Știința, 2014