




# TEHNICA GREEDY



Goriuc Patricia  
IPLT SPIRU HARET

## Cuprins

1. Aspecte teoretice .....	2
Noțiuni.....	2
Structura algoritmului Greedy .....	2
Schemă .....	3
Avantaje .....	3
Dezavantaje.....	4
2. Exemple de probleme .....	4
Program P1;.....	4
Program P2;.....	5
Program P3;.....	8
Program P4;.....	9
Program P5;.....	10
3. Exemple de probleme din cotidian .....	11
4. Concluzie .....	12
5. Bibliografie .....	13

# 1.Aspecte teoretice

## Noțiuni

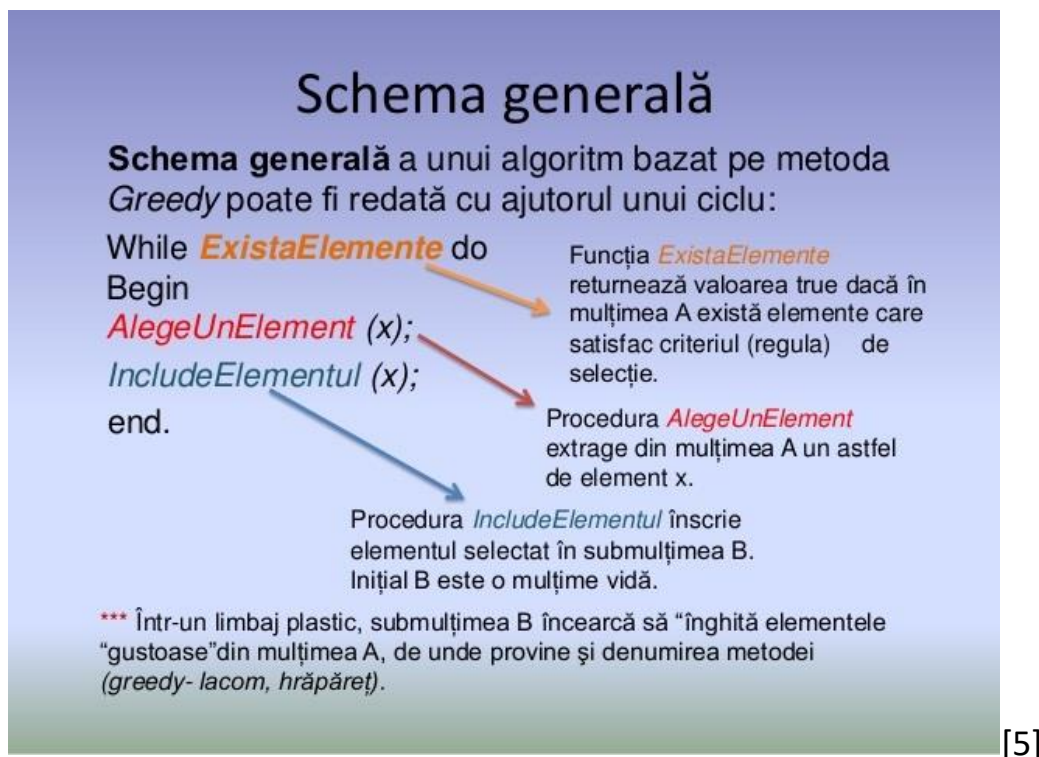
Metoda de programare Greedy se aplică problemelor de optimizare. Aceasta metoda constă în faptul că se construiește soluția optimă pas cu pas, la fiecare pas fiind selectat în soluție elementul care pare „cel mai bun/cel mai optim” la momentul respectiv, în speranța că această alegere locală va conduce la optimul global. Algoritmii Greedy nu conduc 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. Din această cauză, orice algoritm Greedy trebuie însoțit de o demonstrație a corectitudinii sale. Demonstrația se face de obicei prin inducție matematică. Majoritatea acestor probleme constau în determinarea unei submulțimi  $B$ , a unei mulțimi  $A$  cu  $n$  elemente care să îndeplinească anumite condiții pentru a fi acceptată. Orice astfel de submulțime care respectă aceste restricții se numește soluție posibilă. Din mulțimea tuturor soluțiilor posibile se dorește determinarea unei soluții care maximizează sau minimizează o funcție de cost, numită soluție optimă. Considerăm că soluțiile posibile au următoarea proprietate: dacă  $B$  este o soluție posibilă, atunci orice submulțime a sa este soluție posibilă [1] [2].

## Structura algoritmului Greedy

- se inițializează mulțimea soluțiilor  $S$  cu mulțimea vidă,  $S=\emptyset$
- 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

- 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 [2].

## Schemă



## Avantaje

- Timpul de rezolvare este mic, algoritmul Greedy fiind unul polinomial.
- Dacă condiția și formularea programului este corectă, algoritmul Greedy va găsi mereu o soluție. [3]

## Dezavantaje

- Nu toate problemele au o condiție bine definită, astfel algoritmul Greedy nu este aplicabil acestora.
- Deși algoritmul Greedy va găsi o soluție în cazul că condiția și formularea programului sunt valide, această soluție poate să nu fie optimală problemei.[3]

## 2.Exemple de probleme

### Program P1;

```
type coins = array[1..5] of integer;
var
x,i:integer;
a,b:coins;
begin
a[1]:=50; a[2]:=25; a[3]:=10; a[4]:=5; a[5]:=1;
write('Introduceti numarul de banuti (rest) : ');
readln(x);
i:=1;
while x>0 do begin
if x-a[i]>=0 then begin
x:=x-a[i];
inc(b[i]);
end else begin
inc(i);
end;
writeln();
writeln('Pentru a intoarce acest rest aveti nevoie de urmatorul set de banuti : ');
writeln();
for i:=1 to 5 do writeln(a[i], 'x ', b[i]);
end.
```

## Program P2;

```
type
data = record
  name : string;
  artist : string;
  genre : string;
end;
tab = array[1..100] of data;
var
  a,a_copy,b:tab;
  i,n,n1:integer;
  input:string;
  x:char;
  {-----}
function checkGenre(var a:tab):boolean;
  var i:integer;
  begin
    checkGenre:=False;
    i:=1;
    while (a[i].genre<>input) and (i<=n) do inc(i);
    if (i<=n) and (a[i].genre=input) then checkGenre:=True;
  end;
procedure includeItem_Genre(var a,b:tab; var x:integer);
  var i:integer;
  begin
    i:=1;
    while (i<n) and (a[i].genre<>input) do inc(i);
    inc(x);
    b[x].artist:=a[i].artist;
    a[i].artist:='N/A';
    b[x].genre:=a[i].genre;
    a[i].genre:='N/A';
    b[x].name:=a[i].name;
    a[i].name:='N/A';
  end;
  {-----}
procedure recovery();
  var i:integer;
```

```

begin
a_copy:=a;
for i:=1 to n1 do begin
b[i].artist:="";
b[i].genre:="";
b[i].name:="";
end;
n1:=0;
end;
{-----}
function checkArtist(var a:tab):boolean;
var i:integer;
begin
checkArtist:=False;
i:=1;
while (a[i].artist<>input) and (i<=n) do inc(i);
if (i<=n) and (a[i].artist=input) then checkArtist:=True;
end;
procedure includeItem_Artist(var a,b:tab; var x:integer);
var i:integer;
begin
i:=1;
while (i<n) and (a[i].artist<>input) do inc(i);
inc(x);
b[x].artist:=a[i].artist;
a[i].artist:='N/A';
b[x].genre:=a[i].genre;
a[i].genre:='N/A';
b[x].name:=a[i].name;
a[i].name:='N/A';
end;
procedure filterArtist();
var i:integer;
begin
write('Numele artistului : '); readln(input);
if checkArtist(a_copy) then includeItem_Artist(a_copy,b,n1);
end;
procedure filterGenre();
var i:integer;

```

```

begin
write('Denumirea genului de muzica : '); readln(input);
if (checkGenre(a_copy) = true) then includeItem_Genre(a_copy,b,n1);
end;
begin
i:=0;
while x<>'N' do begin
inc(i);
writeln(i,' '); write(' Arist : '); readln(a[i].artist);
write(' Genre : '); readln(a[i].genre); write(' Name : ');
readln(a[i].name);
writeln('continue list creation?');
writeln('-----Y/N-----');
readln(x);
end;
n:=i;
a_copy:=a;
while x<>'X' do begin
writeln(); writeln();
writeln('G - Filtrare dupa gen de muzica'); writeln('A - Filtrare dupa artist');
writeln('L - Afisarea listei originale'); writeln('N - Adaugare elemente');
writeln('X - Iesire din program'); readln(x);
if x = 'G' then begin
filterGenre();
for i:=1 to n1 do writeln(' ',b[i].artist,' - ',b[i].name);
recovery();
end else
if x = 'A' then begin
filterArtist();
for i:=1 to n1 do writeln(' ',b[i].name,' - ',b[i].genre);
recovery();
end else
if x = 'L' then for i:=1 to n do writeln(' ',i,'# ',a[i].name,' -
',a[i].artist,' : ',a[i].genre)
else if x = 'N' then begin
x:=' ';
i:=n;
while x<>'N' do begin
writeln();

```



```

inc(i);
writeln(i,': ');
write(' Arist : ');
readln(a[i].artist);
write(' Genre : ');
readln(a[i].genre);
write(' Name : ');
readln(a[i].name);
writeln('continue list creation?');
writeln('-----Y/N-----');
readln(x);
end;
n:=i;
end;
end;
end.

```

### Program P3;

```

type
data = record
  name : string;
  gender : char;
end;
tab = array[1..100] of data;
var
a,b:tab;
i,n,n1:integer;
x:char;
function checkFemale(var a:tab):boolean;
  var i:integer;
  begin
    checkFemale:=False;i:=1;
    while (a[i].gender<>'F') and (i<=n) do inc(i);
    if (i<=n) and (a[i].gender='F') then checkFemale:=True;
  end;
procedure extractFemale(var a,b:tab; var x:integer);
  var i:integer;
  begin
    i:=1;

```

```

while (i<=n) and (a[i].gender<>'F') do inc(i);inc(x);
b[x].gender:=a[i].gender;
a[i].gender:='-';
b[x].name:=a[i].name;
a[i].name:='N/A'
end;
begin
i:=0;
while x<>'N' do begin
inc(i);
writeln(i,' '); write(' nume : ');readln(a[i].name,x);
write(' sex[M/F] : ');readln(a[i].gender);
writeln('continue list creation?'); writeln('-----Y/N-----'); readln(x);
end;
n:=i;
while checkFemale(a)=true do extractFemale(a,b,n1);writeln('----- Lista Fetelor -----');
for i:=1 to n1 do writeln(b[i].name);
writeln('-----Lista Initiala-----');
for i:=1 to n do writeln(a[i].name);
end.

```

## Program P4;

```

type tab = array[1..100] of integer;
var i,j,n:integer;
a,b:tab;
x:char;
procedure pos(var a:tab; var j:integer);
var i:integer;
begin
for i:=1 to n do if a[i]>0 then begin
inc(j); b[j]:=a[i];
end; end;
procedure neg(var a:tab; var j:integer);
var i:integer;
begin
for i:=1 to n do if a[i]<0 then begin
inc(j); b[j]:=a[i];
end; end; procedure reset();

```

```

var i:integer;
begin
for i:=1 to j do b[i]:=0;j:=0;
end;
begin
write('numarul de elemente al tabelului : '); readln(n);writeln();
for i:=1 to n do begin
write(i,'# : '); readln(a[i]);
end;
while x<>'X' do begin
writeln('-----'); writeln('P - Extragerea elementelor pozitive');
writeln('N - Extragerea elementelor negative'); writeln('X - Iesire din program');
writeln(); readln(x);
if x = 'P' then begin
pos(a,j);
for i:=1 to j do writeln(i,'# ',b[i]); reset();
end else
if x = 'N' then begin
neg(a,j);
for i:=1 to j do writeln(i,'# ',b[i]);reset();
end; end;
end.

```

## Program P5;

```

type
data = record
ID : string;
terminal : string;
end;
tab = array[1..100] of data;
var
a,b:tab;
i,n,n1:integer;
input:string;
x:char;
function checkTerminal(var a:tab):boolean;
var i:integer;
begin
checkTerminal:=False;

```

```

i:=1;
while (a[i].terminal<>input) and (i<=n) do inc(i);
if (i<=n) and (a[i].terminal=input) then checkTerminal:=True;
end;
procedure extractVehicle(var a,b:tab; var x:integer);
var i:integer;
begin
i:=1;
while (i<=n) and (a[i].terminal<>input) do inc(i);
inc(x);
b[x].terminal:=a[i].terminal; a[i].terminal:='-';
b[x].ID:=a[i].ID; a[i].ID:='N/A'
end;
begin
i:=0;
while x<>'N' do begin
inc(i);
writeln(i,' ');
write(' Numerele Inmatriculare : ');readln(a[i].ID,x);
write(' Gara Carei Apratine Vehiculul : ');readln(a[i].terminal);
writeln('continue list creation?');writeln('-----Y/N-----');readln(x);
end;
n:=i;
write('Introduceti numele garii a carei vehicule doriti sa fie afisate :');readln(input);
while checkTerminal(a)=true do extractVehicle(a,b,n1);
writeln;writeln;writeln;
writeln('----- Lista Vehiculelor al garii ',input,'-----');
for i:=1 to n1 do writeln(b[i].ID);
end.

```

### 3.Exemple de probleme din cotidian

- Plata unei sume în monede de mai multe tipuri.
- Alegerea cât mai multor feluri de mâncare întâlnite în lista unui meniu din restaurant.
- Alegerea unui număr maxim de obiecte care pot încăpea într-un rucsac.

- Problema selectării activităților este caracteristică acestui tip de probleme, în cazul în care obiectivul este de a alege numărul maxim de activități care nu intră în conflict unele cu altele.
- Într-un calculator Macintosh jocul Crystal Quest are obiectivul de a colecta cristale. Jocul are un mod demo, în acest caz jocul folosește un algoritm greedy pentru a merge la fiecare cristal. Inteligența artificială însă nu ține cont de obstacole, deci modul demonstrativ de multe ori se termină repede [5].

## 4. Concluzie

În concluzie putem spune că în prezența unui anumit criteriu, metoda Greedy „înghite” elementele gustoase din mulțimea A, testând consecutiv toate elementele mulțimii[4]. Metoda determină întotdeauna o singură soluție, asigurând un optim local, dar nu întotdeauna și global. Tehnica Greedy este una de optimizare, rulând mai rapid decât un Backtracking, dar nefiind întotdeauna cea mai bună[1]. Metoda Greedy este foarte eficientă atunci când dorim să aflăm rezultatul optim în cât mai scurt timp posibil, deoarece algoritmi sunt polinomiali. Cu regret, aceasta poate fi aplicată numai atunci când din enunțul problemei poate fi dedusă regula care asigură selecția directă a elementelor necesare din mulțimea dată[7].

## 5. Bibiliografie

- 1) <https://www.slideshare.net/BalanVeronica/metoda-greedy1> [1]
- 2) <https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy> [2]
- 3) [https://www.researchgate.net/figure/Advantages-and-drawbacks-of-genetic-and-greedy-algorithms\\_tbl2\\_221472685](https://www.researchgate.net/figure/Advantages-and-drawbacks-of-genetic-and-greedy-algorithms_tbl2_221472685) [3]
- 4) <https://www.slideshare.net/BalanVeronica/tehnica-greedy> [4]
- 5) <http://timofti7.simplesite.com/435052889> [5]
- 6) [https://en.wikipedia.org/wiki/Greedy\\_algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) [6]
- 7) <http://caterinamacovenco.blogspot.com/p/metoda-greedy.html> [7]