Tehnica Greedy

Referat elaborate de: Furculiță Andreea,
clasa a XI-a “C”

Profesor: Guțu Maria

2020

Cuprins

Aspecte teoretice

Rezolvarea problemelor cu ajutorul tehnicii Greedy implică efectuarea la fiecare etapă a alegerii ce pare a fi cea mai bună la momentul dat. Cu alte cuvinte, tehnica dată face o alegere local-optimală în speranța că aceasta va duce la obținerea unei soluții optime la nivel global [5]. Astfel, în cazul problemelor pentru care această metodă este utilă, soluțiile optimale ale acestora conțin soluții optimale pentru sub-probleme [7].

În cadrul tehnicii Greedy, deciziile sunt luate în baza informației disponibile la moment, fără a se lua în considerare efectul deciziei curente în viitor. Aceasta construiește soluția pas cu pas, alegând pasul următor, așa încât acesta să aducă un beneficiu imediat. În cazul acestei abordări nu se reconsideră niciodată alegerile efectuate [6].

Metoda Greedy presupune că problemele pe care trebuie să le rezolvăm au următoarea structură:

- se dă o mulţime $A=\{a_1, a_2, ..., a_n\}$ formată din n elemente;
- se cere să determinăm o submulţime B, $B \subseteq A$, care îndeplineşte anumite condiţii pentru a fi acceptată ca soluţie.

Problemele de acest tip pot fi rezolvate prin metoda trierii, generând consecutiv cele 2^n submulţimi A_i ale mulţimii A. Dezavantajul metodei trierii constă în faptul că timpul cerut de algoritmii respectivi este foarte mare. Pentru a evita trierea tuturor submulţimilor, în metoda Greedy se utilizează un criteriu care asigură alegerea directă a elementelor necesare din mulţimea A.

Schema generală a unui algoritm bazat pe metoda Greedy poate fi redată cu ajutorul unui ciclu:

```
while ExistaElemente do
begin
AlegeUnElement(x);
IncludeElementul(x);
end;
```

Funcţia ExistaElemente returnează valoarea *true* dacă în mulţimea A există elemente care satisfac criteriul (regula) de selecţie. Procedura AlegeUnElement extrage din mulţimea A un astfel de element x, iar

procedura IncludeElementul înscrie elementul selectat în submulţimea B. Iniţial B este o mulţime vidă [4]. Cu toate acestea, Greedy reprezintă o tehnică şi nu un algoritm, de aceea structura programelor variază.

Avantaje și dezavantaje

Avantaje

- Găsirea soluţiei este destul de uşoară cu un algortm în baza metodei Greedy pentru problemă [2].
- Alegerea următorului element este, de obicei, de asemenea, uşoară [1].
- Dacă se poate demonstra că un algoritm bazat pe metoda Greedy dă randament global optim pentru o anumită clasă de probleme, de obicei, acesta devine metoda aleasă, pentru că este mai rapidă decât alte metode de optimizare [7].
- Analizarea timpului necesar pentru execuţia algoritmilor în baza metodei Greedy este mai uşoară decât în cazul altor tehnici [2].
- Chiar dacă tehnica Greedy nu prezintă soluţia perfectă pentru majoritatea problemelor, utilizarea acesteia în combinaţie cu alte tehnici a produs algoritmi optimali pentru mai multe probleme importante din informatică [5].

Dezavantaje

- Algoritmii în baza tehnicii Greedy pot da greş în găsirea soluţiei optime globale mai ales pentru că nu operează exhaustiv pe toate datele. Angajamentele pe care şi le iau pentru anumite alegeri ce par optime la moment îi pot împiedica să găsească cele mai bune soluţii globale mai târziu [7].
- Metoda Greedy poate fi aplicată numai atunci când din enunţul problemei poate fi dedusă regula care asigură selecţia directă a elementelor necesare din mulţimea iniţială [4].
- În cazul tehnicii Greedy, trebuie să munceşti foarte mult pentru a înţelege problemele de corectitudine. Chiar şi având un algoritm corect, este greu de demonstrat de ce acesta este corect. Demonstrarea corectitudinii unui algoritm în baza acestei metode este mai mult o artă decât o ştiinţă şi implică multă creativitate [5].

- Alegerile efectuate nu se reconsideră, astfel încât dacă algoritmul ia o cale greșită, acesta nu va mai reveni la cea corectă [3].

Exemple de programe

1. Să se scrie un program ce determină primele 5 cele mai mari numere dintr-un tablou format din 10 numere cuprinse între 0 și 10000. Toate numele de intrare sunt diferite.

```
Program Exemplul_1;
var a,b: array[1..100] of 0..10000;
n, m, x, j: integer;
function Exista: boolean;
begin
Exista:= (n>=5) and (m<=5);
end;
procedure Maxim(var element : integer);
var i, max, maxPrec : integer;
begin
max:= -32000;
maxPrec := element;
for i:=1 to n do begin
if (a[i] > max) and (a[i] < maxPrec) then max:= a[i];
end;
element:= max;
end;
procedure IncludeElementul(element : integer);
begin
b[m]:= element;
inc(m);
end;
procedure afisare;
var i: integer;
begin
for i:=1 to m-1 do write(b[i], ' ');
end;
begin
m:=1;
```

```
x:=10001;
writeln('Dati numarul de elemente'); readln(n);
writeln('Dati elementele');
for j:=1 to n do readln(a[j]);
while Exista do
begin
Maxim(x);
IncludeElementul(x);
end;
afisare;
end.
```

2. Se consideră mulţimea A={a_1, a_2,..., a_i,..., a_n}, elementele căreia sunt numere reale, iar cel puţin unul din ele satisface condiţia $a_i<0$. Elaboraţi un program care determină o submulţime B, $B \subseteq A$, astfel încât suma elementelor din B să fie minimă. Calculaţi această sumă.

```
Program Exemplul_2;
const nmax=1000;
var A: array [1..nmax] of real;
n: 1..nmax;
B: array [1..nmax] of real; m : 0..nmax;
x, s : 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('Dati n='); readln(n); writeln('Dati elementele multimii A:');
for i:=1 to n do read(A[i]);
writeln;
m:=0;
while ExistaElemente do
begin
AlegeUnElement(x);
IncludeElementul(x);
end;
writeln('Elementele multimii B:');
for i:=1 to m do writeln(B[i]);
s:=0;
for i:=1 to m do s:=s+B[i];
writeln('Suma elementelor multimii B este ', s);
end.
```

3. O persoană dispune de timpul T pentru a lua parte la nişte activităţi interesante şi vrea să reuşească să se implice în cât mai multe activităţi. Se dă un tablou A cu numere integer, în care fiecare element indică timpul necesar unei activităţi. Să se calculeze numărul maxim de activităţi la care poate lua parte persoana în timpul disponibil [5].

```
Program Exemplul_3;
type vector = array [1..100] of integer;
var T, n, t_curent, nr_activ, i:integer;
   A: vector;
procedure BubbleSort(j:integer; var A:vector);
   var i,k: integer;
   begin
   for k:=j-1 downto 1 do
   for i:=2 to j do
   if A[i]<A[i-1] then begin
      A[i]:=A[i-1]+A[i];
      A[i-1]:=A[i]-A[i-1];
      A[i]:=A[i]-A[i-1];
      end;
```

```
end;

begin
   writeln('Dati timpul disponibil'); readln(T);
   writeln('Dati nr de activitati posibile'); readln(n);
   writeln('Dati timpul necesar fiecarei activitati');
for i:=1 to n do readln(A[i]);
BubbleSort(n, A);
for i:=1 to n do write(A[i], ' ');
t_curent:=0;
nr_activ:=0;
for i:=1 to n do begin
   t_curent:=t_curent+A[i];
   if t_curent>T then break;
   inc(nr_activ);
end;
writeln('Numarul maxim posibil de activitati este: '); writeln(nr_activ);
end.
```

4. Se dă o listă de cuvinte. Să se afișeze în ordine descrescătoare după greutatea cuvintelor acele cuvinte ce conţin vocale. Un cuvânt este mai greu, dacă suma greutăţilor literelor lui este mai mare. Greutatea literei este determinată de numărul de ordine al acesteia în alfabet. Notă: Cel puţin un cuvânt trebuie să conţină cel puţin o vocală.

```
Program Exemplul_4;
var a,b : array[1..100] of string;
n, m, j : integer;
x : string;
function eVocala(c: char) : boolean;
begin
eVocala := (c = 'a') or (c = 'e') or (c = 'u') or (c = 'i') or (c = 'o');
end;
function ContineVocale(cuvant : string) : boolean;
var i : integer;
result1 : boolean;
begin
result1 := false;
for i:=1 to Length(cuvant) do
```

```
if eVocala(cuvant[i]) then begin result1 := true; break; end;
ContineVocale := result1;
end;
function dejaInclus(cuvant : string) : boolean;
var i : integer;
result1 : boolean;
begin
result1 := false;
for i:=1 to m-1 do begin
if b[i] = cuvant then begin result1 := true; break; end;
end;
dejaInclus := result1;
end;
function ExistaElementeNoi: boolean;
var i : integer;
result1 : boolean;
begin
result1 := false;
for i:=1 to n do
if ContineVocale(a[i]) then
begin
if not dejaInclus(a[i]) then
begin
result1 := true;
break;
end;
end;
ExistaElementeNoi:= result1;
end;
function cantareste(cuvant : string) : integer;
var i, greutate : integer;
begin
greutate := 0;
for i:=1 to Length(cuvant) do begin  greutate := greutate + ord(cuvant[i]);
end;
cantareste := greutate;
end;
```

```
procedure AlegeCuvantulGreu(var cuvant: string);
var i, max, greutate : integer;
begin
max := 0;
for i:=1 to n do if ContineVocale(a[i]) and not dejaInclus(a[i]) then
begin
greutate := cantareste(a[i]);
if greutate > max then begin
max:= greutate;
cuvant:= a[i];
end;
end;
end;
procedure IncludeCuvantul(cuvant: string);
begin
b[m] := cuvant;
inc(m);
end;
procedure afisare();
var i : integer;
begin
for i:=1 to m-1 do write(b[i], ' ');
end;
begin
m := 1;
x:=' ';
writeln('Dati numarul de cuvinte'); readln(n);
for j:=1 to n do readln(a[j]);
while ExistaElementeNoi do
begin
AlegeCuvantulGreu(x);
IncludeCuvantul(x);
end;
afisare;
end.
```

5. Să se scrie un program ce determină primele 5 cele mai mici numere dintr-un tablou format din 10 numere cuprinse între 0 și 10000. Toate numele introduse trebuie să fie diferite.

```
Program Exemplul_5;
var a,b: array[1..100] of 0..10000;
n, m, x, j: integer;
function Exista: boolean;
begin
Exista:= (n>=5) and (m<=5);
end;
procedure Minim(var element : integer);
var i, min, minPrec : integer;
begin
min:=32000;
minPrec := element;
for i:=1 to n do begin
if (a[i] < min) and (a[i] > minPrec) then min:= a[i];
end;
element:= min;
end;
procedure IncludeElementul(element: integer);
begin
b[m]:= element;
inc(m);
end;
procedure afisare;
var i: integer;
begin
for i:=1 to m-1 do write(b[i], ' ');
end;
begin
m:=1;
x:=-1;
writeln('Dati numarul de elemente'); readln(n);
writeln('Dati elementele');
for j:=1 to n do readln(a[j]);
while Exista do
```

```
begin
Minim(x);
IncludeElementul(x);
end;
afisare;
end.
```

Ex.7 p.125 [4]

```
Program Ex_7;
var  n, m, x, y, perete_x, perete_y: integer;
a: array[1..102, 1..102] of integer;
F: TextFile;
pas: string;
procedure citire;
var i, j : integer;
 begin
   Assignfile(F, 'hrube.txt');
   Reset(F);
   read(f, n);
   read(f, m);
   for i:=2 to n+1 do
     for j:=2 to m+1 do
       Read(F, a[i, j]);
   Closefile(F);
 end;
 procedure incercuieste_cu_pereti;
 var i, j : integer;
 begin
   for i:=1 to n+2 do
   begin
     a[i, 1] := 1;
     a[i, m+2] := 1;
   end;
   for j:=1 to m+2 do
   begin
     a[1, j] := 1;
     a[n+2, j] := 1;
   end;
```

```
end;
procedure afisare;
var i, j: integer;
begin
   for i:=1 to n+2 do
   begin
    for j:=1 to m+2 do
      write(a[i, j], ' ');
   writeln;
 end;
end;
procedure gaseste_inceputul;
var i, j: integer;
begin
   for i:=2 to n+1 do
    for j:=2 to m+1 do
      if a[i, j] = 2 then
      begin
        x := i;
        y := j;
      end;
end;
function exista_elemente: boolean;
begin
 exista_elemente:= (a[x-1, y] = 0)
              or (a[x+1, y] = 0)
              or (a[x, y-1] = 0)
              or (a[x, y+1] = 0);
end;
function perete_continuu(x1, y1: integer): boolean;
begin
 perete_continuu := (abs(perete_x - x1) < 2) and (abs(perete_y - y1)< 2);
end;
procedure alege_un_element();
var i, j: integer;
begin
 {
```

Alegerea e bazata pe 2 conditii:

1. ne tinem tot timpul de peretele drept;
2. Ordinea directiilor preferate este: SUS, DREAPTA, JOS, STANGA.

}

// (putem pasi acolo sau e iesire) and (peretele va fi pe dreapta si e acelasi perete ca cel din pasul precedent)

if ((a[x-1, y] = 0) or (a[x-1, y] = 3)) and (a[x, y+1] = 1) and (perete_continuu(x, y+1)) then begin x := x-1; pas := 'SUS'; perete_x := x; perete_y := y+1; end

else if ((a[x, y+1] = 0) or (a[x, y+1] = 3)) and (a[x+1, y] = 1) and (perete_continuu(x+1, y)) then begin y := y+1; pas := 'DREAPTA'; perete_x := x+1; perete_y := y; end

else if ((a[x+1, y] = 0) or (a[x+1, y] = 3)) and (a[x, y-1] = 1) and (perete_continuu(x, y-1)) then begin x := x+1; pas := 'JOS'; perete_x := x; perete_y := y-1; end

else if ((a[x, y-1] = 0) or (a[x, y-1] = 3)) and (a[x-1, y] = 1) and (perete_continuu(x-1, y)) then begin y := y-1; pas := 'STANGA'; perete_x := x-1; perete_y := y; end

// (putem pasi acolo sau e iesire) and (peretele va ramane in urma, dreapta, deci vom coti - ocolim)

else if ((a[x, y+1] = 0) or (a[x, y+1] = 3)) and (a[x+1, y+1] = 1) and (perete_continuu(x+1, y+1)) then begin y := y+1; pas := 'DREAPTA'; perete_x := x+1; perete_y := y+1; end

else if ((a[x+1, y] = 0) or (a[x+1, y] = 3)) and (a[x+1, y-1] = 1) and (perete_continuu(x+1, y-1)) then begin x := x+1; pas := 'JOS'; perete_x := x+1; perete_y := y-1; end

else if ((a[x, y-1] = 0) or (a[x, y-1] = 3)) and (a[x-1, y-1] = 1) and (perete_continuu(x-1, y-1)) then begin y := y-1; pas := 'STANGA'; perete_x := x-1; perete_y := y-1; end

else if ((a[x-1, y] = 0) or (a[x-1, y] = 3)) and (a[x-1, y+1] = 1) and (perete_continuu(x-1, y+1)) then begin x := x-1; pas := 'SUS'; perete_x := x-1; perete_y := y+1; end

// (putem pasi acolo sau e iesire) and (avem pereti din 3 parti, ne intoarcem la 180 de grade) perete_x si perete_y nu trebuiesc verificati, deja au fost in iteratia precedenta

else if ((a[x+1, y] = 0) or (a[x+1, y] = 3)) and (a[x-1, y] = 1) and

```
(a[x, y-1] = 1) then begin x := x+1; pas := 'JOS'; perete_x := x; perete_y := y-1;
end
   else if ((a[x, y-1] = 0) or (a[x, y-1] = 3)) and (a[x, y+1] = 1) and
(a[x-1, y] = 1) then begin y := y-1;  pas := 'STANGA'; perete_x := x-1; perete_y
:= y; end
   else if ((a[x-1, y] = 0) or (a[x-1, y] = 3)) and (a[x+1, y] = 1) and
(a[x, y-1] = 1) then begin x := x-1; pas := 'SUS'; perete_x := x; perete_y := y+1;
end
   else if ((a[x, y+1] = 0) or (a[x, y+1] = 3)) and (a[x, y-1] = 1) and
(a[x+1, y] = 1) then begin y := y+1; pas := 'DREAPTA'; perete_x := x+1;
perete_y := y; end
   else pas := '???';
   writeln(pas);
 end;
 procedure gaseste_calea;
 var i, j : integer;
 begin
   Assignfile(F, 'hrube_out.txt');
   rewrite(F);
   while exista_elemente do
   begin
    alege_un_element;
    // include_elementul:
    writeln(f, pas);
    if (a[x, y] = 3) or (pas = '???') then break;
   end;
   Closefile(F);
 end;
begin
   citire;
   incercuieste_cu_pereti;
   afisare;
   x := 0;
   y := 0;
   gaseste_inceputul;
   writeln('Inceputul este ', x-1, ', ', y-1);
   perete_x := x; perete_y := y;
```

```
  gaseste_calea;
end.
```

Concluzie

Tehnica Greedy constă în determinarea soluției finale efectuând la fiecare etapă alegerea optimă la momentul dat. Această tehnică prezintă atât avantaje, cât și dezavantaje.

Pe de o parte, găsirea soluției prin această metodă este destul de simplă, iar algoritmii în baza acesteia sunt mai rapizi decât în cazul altor tehnici.

Pe de altă parte, în multe cazuri, tehnica Greedy eșuează în găsirea soluției optime. Aceasta nu operează cu toate datele, iar alegerile efectuate nu pot fi reconsiderate. De asemenea, metoda Greedy poate fi aplicată numai atunci când din enunțul problemei se poate deduce criteriul ce asigură selecția directă a elementelor necesare.

Astfel, tehnica Greedy este eficientă doar în cazul în care la baza soluției finale se află soluții optime ale sub-problemelor.

Bibliografie

1. http://bluehawk.monmouth.edu/~rclayton/web-pages/f08-306/greed.html
2. https://developerinsider.co/introduction-to-greedy-algorithms/
3. https://www.educba.com/what-is-a-greedy-algorithm/
4. Gremalschi Anatol. “Informatica. Manual pentru clasa a 11-a”
5. https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/
6. https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_greedy_method.htm
7. https://en.wikipedia.org/wiki/Greedy_algorithm