Tehnica Greedy

Referat

PROFESOR: MARIA GUȚU

May 20, 2020

Authored by: Claudiu Bujor

Cuprins:

1.Aspectele teoretice:

Această metodă 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.

În principiu, problemele de acest tip pot fi rezolvate prin metoda trierii, generînd 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 A_i, Ai $\subseteq$A, în metoda ***Greedy*** se utilizează **un criteriu (o regulă)** care asigură alegerea directă a elementelor necesare din mulţimea A. De obicei, criteriile sau regulile de selecţie nu sînt indicate explicit în enunţul problemei şi formularea lor cade în sarcina programatorului. Evident, înabsenţa unor astfel de criterii metoda *Greedy* nu poate fi aplicată.

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.

După cum se vede, în metoda ***Greedy*** soluţia problemei se caută prin testarea consecutivă a elementelor din mulţimea *A* şi prin includerea unora din ele în submulţimea B. Într-un limbaj plastic, submulţimea B încearcă să „înghită”

elementele „gustoase” din mulţimea A, de unde provine şi denumirea metodei (***greedy*** - lacom, hrăpăreţ).

Exemplu. Se consideră mulţimea A={a_1, a_2, ..., a_i, ..., a_n} elementele căreia sînt numere reale, iar cel puţin unul din ele satisface condiţia a_i >0. Elaboraţi un program care determină o submulţime B, B⊆A, astfel încît suma elementelor din B să fi e maximă. De exemplu, pentru A={21,5; -3,4; 0; -12,3; 83,6} avem B={21,5; 83,6}.

Rezolvare. Se observă că dacă o submulţime B, B⊆A, conţine un element b≤0, atunci suma elementelor submulţimii B \{b} este mai mare sau egală cu cea a elementelor din B. Prin urmare, regula de selecţie este foarte simplă: la fiecare pas în submulţimea B se include un element pozitiv arbitrar din mulţimea A.

Menţionăm că procedura **AlegeUnElement** zerografiază componenta vectorului *A* ce conţinea elementul *x*, extras din mulţimea respectivă.

Complexitatea temporală a algoritmilor bazaţi pe metoda ***Greedy*** poate fi evaluată urmînd schema generală de calcul prezentată mai sus. De obicei, timpul cerut de procedurile **ExistaElemente**, **AlegeUnElement** şi **IncludeElementul** este de ordinul *n*. În componenţa ciclului **while** aceste proceduri se execută cel mult de n ori. Prin urmare, algoritmii bazaţi pe metoda ***Greedy*** sînt algoritmi polinomiali. Pentru comparare, amintim că algoritmii bazaţi pe trierea tuturor submulţimilor A_i, A_i ⊆A, sînt algoritmi de ordinul $O(2^n)$, deci exponenţiali. Cu regret, metoda ***Greedy*** poate fi aplicată numai atunci cînd din enunţul problemei poate fi dedusă regula care asigură selecţia directă a elementelor necesare din mulţimea A. **[1]**

Algoritm Greedy:

- **se dă o mulţime A**
- **se cere o submulţime S din multimea A care sa:**
- **să îndeplinească anumite condiţii interne (să fie acceptabilă)**
- **să 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. **[2]**

2.Avantaje/dezavantaje

Dezavantaje

- Algoritmii Greedy nu conduc în mod necesar la o solutie optimă.

- Nu este posibilă formularea unui criteriu general conform căruia să putem stabili exact dacă metoda Greedy rezolvă sau nu o anumită problemă de optimizare.

- Metoda Greedy poate fi aplicată numai atunci cînd din enunţul problemei poate fi dedusă regula care asigură selecţia directă a elementelor necesare din mulţimea A.

Avantaje

- Algoritmii Greedy sunt foarte eficienti

- Poate fi aplicata multor probleme: determinarea celor mai scurte drumuri in grafuri (Dijkstra), determinarea arborelui minimal de acoperire (Prim, Kruskal), codificare arborilor Huffmann, planificarea activitatilor, problema spectacolelor si problema fractionara a rucsacului.

- Cel mai mare avantaj pe care algoritmul Greedy îl are față de alții este că este ușor de implementat și foarte eficient în majoritatea cazurilor.**[3]**

3.Exemple:

1.Program P1;

```
Var weight : Array[1..10000] of Integer;
Var value : Array[1..10000] of Integer;
Var Noviweight : Array[1..10000] of Integer;
Var Novivalue : Array[1..10000] of Integer;
Var n : Integer;
Var i : Integer;
Var j : Integer;
Var val : Int64;
Var wei : Integer;
Var temp : Integer;
Begin
Write('N: ');
Readln(n);
For i := 1 to n do
Begin
Readln(weight);
Readln(value);
End;
Readln(wei);
For i := 2 to n-1 do
```

```
Begin
Noviweight := weight;
Novivalue := value;
For j := i-1 downto 1 do
Begin
if (value[j] <= Novivalue) then
Begin
value[j+1] := value[j];
weight[j+1] := weight[j];
End;
 End;
value[j+1] := Novivalue;
weight[j+1] := Noviweight;
End;
wei := 0;
For i := 1 to n do
Begin
if (i <> 0) then
Begin
if (wei >= weight) then
Begin
temp := wei;
```

```
wei := temp + value * weight;
End;
if (wei < weight) then
Begin
temp := val;
val := temp + value * wei;
wei := 0;
End;
End;
End;
Write('value:', val);
End.
```

5. Program P1;

```
Var n, a1, a2, c:Integer;
Begin
a1:=-MAXINT; (initializam primele 2 numere si n cu o constanta predefinita)
a2:=-MAXINT;
n:=-MAXINT;
While n<>0 Do Begin
```

If (n>a1) Then a1:=n; (daca numarul n este mai mare decat primul cel mai mare numar atunci maximul este n)

If (a2<a1) Then Begin

c:=a1;

a1:=a2;

a2:=c; end; (interschimbare)

Readln (n); end;

Writeln ('a1, ' ',a2');

end. **[4]**

3.Program impartirea_clasei_dupa_sex;

```
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.
```

4. Extragerea numerelor pozitive/negative, aplicând algoritmul Greedy

```
Program sortare_numere;
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.
```

5. 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;
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.
```

4.Concluzie:

În prezența unui anumit criteriu, metoda Greedy „înghite” elementele gustoase din mulțimea A, testând consecutiv toate elementele mulțimii.

Cel mai mare avantaj pe care algoritmul Greedy îl are față de alții este că e ușor de implementat și foarte eficient în majoritatea cazurilor.

Timpul de rezolvare este mic, algorimul Greedy fiind unul polinomial.

Dacă condiția si formularea programului este corectă, algoritmul Greedy va găsi mereu o soluție.

5.Bibliografie:

1. http://ctice.gov.md/manuale-scolare/
2. https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy
3. https://www.educba.com/what-is-a-greedy-algorithm/
4. https://tpascalblog.wordpress.com/