Metoda Greedy

CIOBANU BOGDAN
CLASA 11C

PROFESOR: GUȚU MARIA

Cuprins:

1. Aspecte teoretice

1.1 Noţiuni generale

Metoda Greedy este una dintre cele mai directe tehnici de proiectare a algoritmilor care poate fi aplicată la o gamă largă de probleme. În general, această metodă se aplică problemelor de optimizare. Majoritatea acestor probleme constau în determinarea unei submulţimi B, a unei mulţimi A cu n elementecare să îndeplinească anumite condiţii pentru a fi acceptată. Orice astfel de submulţime care respectă aceste restricţii se numeşte soluţie posibilă. Din mulţimea tuturor soluţiilor posibile se doreşte determinarea unei soluţii care maximizează sau minimizează o funcţie de cost. O soluţie posibilă care realizează acest lucru se numeşte soluţie optimă. Considerăm că soluţiile posibile au următoarea proprietate: dacă B este o soluţie posibilă, atunci orice submulţime a sa este soluţie posibilă.Specificul acestei metode constă în faptul că se construieşte soluţia optimă pas cu pas, la fiecare pas fiind selectat (sau "înghiţit") în soluţie elementul care pare "cel mai bun" la momentul respectiv, însperanţa că va duce la soluţia optimă globală.

1.2 Schema

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;
```

1.3 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. Avantaje și dezavantaje

2.1 Avantaje

Metoda Greedy are si avantaje: 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. Dintre acestea, articolul le trateaza numai pe ultimele doua pentru a da un exemplu cat mai bun a modului de functionare si aplicare a algoritmilor Greedy.

2.2 Dezavantaje

·Daca nu formulam atent problema pentru a nu face pasi repetitivi, cautarea poate oscila la infinit

· Are unele programe neoptimizate

3. Exemple

```
1.  Program bani;
type tablou=array[1..3,1..7] of integer;
var s,ss,i : integer;  a:tablou;  f:text;
{In primul rind al tabelului vom pastra nominalul bancnotelor}
{In al doilea rind - numarul bancnotelor citite din fisier}
{In al treilea rind - numarul bancnotelor obtinute la schimb}
Procedure Afisare(sa:integer);
begin writeln('suma ',s);
if sa<>0
then writeln('nu poate fi transformata cu bancnotele date ')
else
begin writeln('se plateste cu urmatoarele bancnote');
for i:=1 to 7 do
if a[3,i]<>0
then writeln('bancnote de ',a[1,i]:6,' sau folosit ',a[3,i]);
end;
end; { Afisare }
Procedure calcul(var sa:integer);
```

```
var nb:integer;
begin
i:=7;
while (i>=1) and (sa>0) do
begin nb:=sa div a[1,i];
if nb<>0 then if nb>= a[2,i]
then a[3,i]:=a[2,i]
else a[3,i]:=nb;
sa:=sa-a[3,i]*a[1,i];
i:=i-1;
end;
end; { calcul }
begin
a[1,1]:=1; a[1,2]:=5; a[1,3]:=10; a[1,4]:=50;
a[1,5]:=100; a[1,6]:=200; a[1,7]:=500;
assign (f,'bani.in'); reset(f);
for i:=1 to 7 do readln(f,a[2,i]);
write ('introduceti suma de lei S ');readln(s);
ss:=s; calcul(ss); Afisare(ss);
end.
```

2. Program P153;

```
{ 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 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 mulțimii B:');
for i:=1 to m do writeln(B[i]);
readln;
end.
```

3.
```
type tablou=array [1..4] of real;
matrice=array [1..10] of tablou;

{In prima coloana se inscrie costul,
In a II - greutatea, in a III - eficienta,
si in a IV - a cata parte se ia
tabloul c il folosim la sortare, "al treilea pahar"}

var a:matrice; c:tablou; f:text;
```

```
loc,n,g,i,j:integer; max,castig,dg:real;
begin
assign (f,'rucsac.txt'); reset (f);
readln(f,n,g);
for i:=1 to n do
begin readln(f,a[i,1],a[i,2]);
a[i,3]:=a[i,1]/a[i,2]; a[i,4]:=0;
end;
{sortam tabloul dupa eficienta}
for i:=1 to n-1 do
begin max:=a[i,3];loc:=i;
for j:=i+1 to n do
if a[j,3]>max then begin max:=a[j,3]; loc:=j; end;
c:=a[i]; a[i]:=a[loc]; a[loc]:=c;
end;
{Aflam cat din fiecare obiect se pune in rucsac si calculam
castigul}
castig:=0;
i:=1; dg:=g;
writeln ('greutatea ','costul ','eficienta ','rucsac');
while (i<=n) and (dg>0) do
begin;
if dg>=a[i,2]
then begin castig:=castig+a[i,1];
dg:=dg-a[i,2]; a[i,4]:=1;
end
else begin castig:=castig+dg*a[i,3];
a[i,4]:=dg/a[i,2];dg:=0;
end;
writeln (a[i,1]:6:2,a[i,2]:8:2,a[i,3]:12:2,a[i,4]:10:2);
i:=i+1;
end;
```

```
writeln ('greutatea rucsacului este ',g-dg:0:2);
writeln ('costul este ',castig:0:2);
end.
```

4.
```
program Rucsac;
const max=5;
var C,G,X: array [1..max] of Real;
n,i,j:Integer; GG,GGr,aux:Real;
begin
Write(Nr. obiecte = ');
ReadLn (n);
For i:=1 to n do
begin
Write ('C[',i,']=');
ReadLN (c[i]);
Write ('G[',i,']=');
ReadLn (G[i]);
end;
Write('Greut. max. = ');
ReadLn (GG);
for i:=1 to n-1 do
for j:=i+1 to n do
if C[j]/G[j]>C[i]/G[i] then
begin
aux:=C[j]; C[j]:=C[i];
```

```
C[i]:=aux; aux:=G[j];
G[j]:=G[i]; G[i]:=aux;
end;
WriteLN('Am ordonat . . .');
for i:=1 to n do
WriteLN('C[',i,']=',C[i] :5:2,
'G[',i,]=' G[i] :5:2,
'   ‘,C[i]/G[i] :5:2);
GGr:=GG; i:=1;
While (i<=n) do
if GGr > G[i] then
begin
X[i]:=1;
GGr:=GGr-G[i]; i:=i+1
end
else
begin
X[i]:=GGr/G[i];
For j:=i+1 to n do X[j]:=0
I:=n+1
end;
for i:=1 to n do
WriteLn (‘X[‘,I,’]=’X[i] :5:2);
ReadLn
```

```
end.
```

5.

```
Program P153;
{ 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(’Daţi n=’); readln(n);
writeln(’Daţi elementele mulţimii A:’);
for i:=1 to n do read(A[i]);
writeln;
m:=0;
while ExistaElemente do
begin
AlegeUnElement(x);
IncludeElementul(x);
end;
writeln(’Elementele mulţimii B:’);
for i:=1 to m do writeln(B[i]);
readln;
end.
```

4. Problema 7 pagina 125

```
Program Hrube;
{ Clasele 07-09 }
const nmax=100; mmax=100;
```

```
type Directie = (NORD, SUD, EST, VEST);
var A : array[1..nmax, 1..mmax] of integer;
n, m : integer; { dimensiunile campului }
p, q : integer; { coordonatele intrarii }
r, s : integer; { coordonatele iesirii }
DI : Directie; { orientarea initiala a robotului }
Iesire : text;
procedure Citeste;
{ Citeste datele de intrare }
var i, j : integer;
Intrare : text;
begin
assign(Intrare, 'HRUBE.IN');
reset(Intrare);
readln(Intrare, n, m);
for i:=1 to n do
begin
for j:=1 to m do
begin
read(Intrare, A[i, j]);
if A[i, j]=2 then begin p:=i; q:=j; A[i, j]:=0; end;
if A[i, j]=3 then begin r:=i; s:=j; A[i, j]:=0; end;
end; { for }
readln(Intrare);
```

```
end; { for }
close(Intrare);
if q=1 then DI:=EST;
if q=m then DI:=VEST;
if p=1 then DI:=SUD;
if p=n then DI:=NORD;
end; { Citeste }
procedure Mergi(x, y : integer; D : Directie);
label 1;
begin
repeat
if D=NORD then
begin
if A[x, y+1]=0 then
begin writeln(Iesire, 'DREAPTA'); y:=y+1; D:=EST; goto 1; end;
if A[x-1, y]=0 then
begin writeln(Iesire, 'SUS'); x:=x-1; D:=NORD; goto 1; end;
if A[x, y-1]=0 then
begin writeln(Iesire, 'STANGA'); y:=y-1; D:=VEST; goto 1; end;
D:=SUD; goto 1;
end; { Nord }
if D=SUD then
```

```
begin
if A[x, y-1]=0 then
begin writeln(Iesire, 'STANGA'); y:=y-1; D:=VEST; goto 1;
end;
if A[x+1, y]=0 then
begin writeln(Iesire, 'JOS'); x:=x+1; D:=SUD; goto 1; end;
if A[x, y+1]=0 then
begin writeln(Iesire, 'DREAPTA'); y:=y+1; D:=EST; goto 1;
end;
D:=NORD; goto 1;
end; { SUD }
if D=EST then
begin
if A[x+1, y]=0 then
begin writeln(Iesire, 'JOS'); x:=x+1; D:=SUD; goto 1; end;
if A[x, y+1]=0 then
begin writeln(Iesire, 'DREAPTA'); y:=y+1; D:=EST; goto 1;
end;
if A[x-1, y]=0 then
begin writeln(Iesire, 'SUS'); x:=x-1; D:=NORD; goto 1; end;
D:=VEST; goto 1;
end; { EST }
if D=VEST then
begin
if A[x-1, y]=0 then
```

```
begin writeln(Iesire, 'SUS'); x:=x-1; D:=NORD; goto 1; end;
if A[x, y-1]=0 then
begin writeln(Iesire, 'STANGA'); y:=y-1; D:=VEST; goto 1; end;
if A[x+1, y]=0 then
begin writeln(Iesire, 'JOS'); x:=x+1; D:=SUD; goto 1; end;
D:=EST; goto 1;
end; { VEST }
until ((x=r) and (y=s));
end; { Mergi }
begin
Citeste;
assign(Iesire, 'HRUBE.OUT');
rewrite(Iesire);
Mergi(p, q, DI);
close(Iesire);
end.
```

5. Concluzie

Algoritmii Greedy sunt caracterizati de metoda lor de functionare: la fiecare pas se alege cel mai bun candidat posibil, dupa evaluarea tuturor acestora. Metoda determina intotdeauna o singura solutie, asigurand un optim local, dar nu intotdeauna si global. Tehnica Greedy este una de optimizare, ruland mai rapid decat un Backtraking, dar nefiind intotdeauna cea mai buna. Cand nu aveti o idee mai buna legata de o problema, in timpul unui concurs, o implementare Greedy ar putea aduce in jur de

30% din punctaj. Exista situatii in care algoritmii clacheaza, cum ar fi problema comisului voiajor, sau problemele NP-complete.cum ar fi problema comisului voiajor, sau problemele NP-complete.

6. Bibliografie

1. https://www.slideshare.net/BalanVeronica/metoda-greedy1

2. https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy

3. https://prezi.com/ys_iuqvyoxsi/tehnica-greedy/

4. http://dasinika.blogspot.com/2009/04/tehnica-greedy-pentru-problemele-pentru.html

5. http://paulborza.blogspot.com/2010/08/metoda-greedy-in-limbajul-de-programare.html

6. Manual de informatică, clasa 11