Lucrare practică

Tehnica Greedy

Bordei Dan-Nicolae

Clasa a XI-a C

Introducere

Un algoritm, atât în matematică cât şi informatică este o metodă sau o procedură de calcul, alcătuită din paşii elementari necesari pentru rezolvarea unei probleme sau categorii de probleme. De obicei algoritmii se implementează în mod concret prin programarea adecvată a unui calculator, sau a mai multora.

Multe probleme de o importanță practică pot fi rezolvare cu ajutorul unor metode standard denumite thenici de programare: Recursia; Trierea; Metoda reluării; Tehnica Greedy, etc.

Tehnica Greedy

Aspecte teoretice

Metoda de programare Greedy se aplică problemelor de optimizare. Aceasta metoda constă în faptul că se construieşte solutia optimă pas cu pas, la fiecare pas fiind selectat în solutie elementul care pare „cel mai bun/cel mai optim" la momentul respectiv, în speranta că această alegere locală va conduce la optimul global[1].

Această metodă presupune că problemele pe care trebuie să le rezolvăm au următoarea structură: – se dă o mulţime A={a1, a2, ..., an} formată din n elemente; – se cere să determinăm o submulţime B, B⊆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 2n submulţimi Ai ale mulţimii A[5].

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.[2]

Algoritmii Greedy sunt foarte eficienți, dar 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 acest motiv, orice algoritm Greedy trebuie însoțit de o demonstrație a corectitudinii sale . Demonstrația faptului că o anumită

problemă poate fi alcătuită cu metod Greedy se face de obicei prin inducție matematică[5].

Metoda Greedy se aplică problemelor pentru care se dă o mulţime A cu n elemente şi pentru care trebuie determinată o submulţime a sa, S cu m elemente, care îndeplinesc anumite condiţii, numite si conditii de optim. Algoritmul in limbaj natural al metodei de programare Greedy are urmatoarea structura:

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)[4].

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.

Exemple de probleme/ Ex.7, pg:125

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.
```

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('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.
```

3.
```
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.[2]
```

4. Program bani;

```
type tablou=array[1..3,1..7] of integer;
var s,ss,i : integer;
a:tablou;
f:text;
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;
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;
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.
```

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.[6]
```

Concluzii

Un algoritm greedy este un algoritm simplu, intuitiv, care este utilizat în problemele de optimizare. Algoritmul face alegerea optimă la fiecare pas, deoarece încearcă să găsească modalitatea globală optimă de a rezolva întreaga problemă. Algoritmii greedy sunt destul de eficienți în unele probleme, însă destul de rar sunt situații în care o problema are toate particularitățile necesare pentru a putea fi folosită tehnica greedy.

Bibliografie

1. Manual
2. https://forum.lazarus.freepascal.org/index.php?topic=24264.15

3. https://books.google.md/books?id=F9TIo4yuchoC&pg=PA171&lpg=PA171&dq=greedy+technique+programs+pascal&source=bl&ots=Pr1soH8ljW&sig=ACfU3U2zCuwCzoA7P-Y90X_Idp2dV6psZA&hl=en&sa=X&ved=2ahUKEwjdu8jhvMDpAhUOjosKHX37BpMQ6AEwAnoECAYQAQ#v=onepage&q=greedy%20technique%20programs%20pascal&f=false
4. geeksforgeeks.org/greedy-algorithms/
5. https://www.tutorialspoint.com/data_structures_algorithms/greedy_algorithms.htm
6. https://tpascalblog.wordpress.com/