cl.11”D”

Tehnica Greedy

Buzu Alexandru

Cuprins

Aspecte teoretice

Definiția

Metoda de programare Greedy se aplică problemelor de optimizare. Această metodă constă în faptul că se construieşte 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 speranță că această alegere locală va conduce la optimul global.
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 și condiții de optim[3].

Schema generală

```
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[2].

Componentele unui algoritm *Greedy*

1. O mulțime de candidați, din care se creează o soluție
2. Funcția de selecție, care alege cel mai bun candidat pentru a fi adăugat la soluție
3. O funcție de fezabilitate, care este folosită pentru a determina dacă un candidat poate fi utilizat pentru a contribui la o soluție
4. O funcție obiectiv, care atribuie o valoare unei soluții sau unei soluții parțiale
5. O funcție de soluție, care va indica atunci când s-a descoperit o soluție completă [1].

Principiul de funcționare a metodei *Greedy*

1. Se dă o mulţime A
2. Se cere o submulţime S din multimea A care să:
   - îndeplinească anumite condiţii interne (să fie acceptabilă)
   - fie optimală (să realizeze un maxim sau un minim).
3. Se **iniţializează** mulţimea soluţiilor S cu mulţimea vidă, S=Ø
4. La fiecare pas se alege un anumit element **x∈A** (cel mai promiţător element la momentul respectiv) care poate conduce la o soluţie optimă
5. 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
6. Procedeul continuă, până când au fost determinate toate elementele din mulţimea soluţiilor[3].

Probleme rezolvate

1. Program Maxim[4]

```
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.
```

2. Scrieţi un program, care afişează modalitatea de plată, folosind un număr minim de bancnote, a unei sume întregi S de lei (S<20000). Plata se efectuează folosind bancnote cu valoarea 1, 5, 10, 50, 100, 200 şi 500 de lei. Numărul de bancnote de fiecare valoare se citeşte din fişierul text BANI.IN, care conţine 7 rânduri, în fiecare din care sunt indicate numărul de bancnote respectiv de 1, 5, 10, 50, 100, 200 şi 500 de lei.
Intrare: Fişierul text BANI.IN şi de la tastatură se citeşte suma S.
Ieşire: Dacă e posibil de plătit această sumă S, atunci la ecran se va afişa valoarea bancnotei şi numărul de bancnote respective utilizate la plată. Dacă bancnote de careva valoare nu se folosesc, atunci nu se afişează această valoare. [5].

```
Program V3P7_02;
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; { 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.
```

3. Se consideră n obiecte. Pentru fi ecare obiect i (i=1, 2, ..., n) se cunoaște greutatea g_i și cîștigul c_i care se obţine în urma transportului său la destinaţie. O persoană are un rucsac cu care pot fi transportate unul sau mai multe obiecte greutatea sumară a cărora nu depășește valoarea G_{max}. Elaboraţi un program care determină ce obiecte trebuie să transporte persoana în așa fel încît cîștigul să fie maxim. În caz de necesitate, unele obiecte pot fi tăiate în fragmente mai mici[5].

```
Program rucsac;
type tablou=array [1..4] of real;
      matrice=array [1..10] of tablou;
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 Benzinărie[4]

```
Program p2;
Type benz=record
        ins, sfs:integer;
        ord:integer; end;
Var v:array [1..100] of benz;
      n, ultim, nr:integer;
Procedure citire_clienti;
Var hh, mm, i:integer;
begin
  Write ('n= '); Readln (n);
  for i:=1 to n do begin
     Write ('Clientul cu nr. ',i,'cand este servit? (ora si minutul)');
     Readln (hh, mm);
     v[i].ins:=hh*60+mm;
     Write ('clientul cu nr ', i, ' cand a terminat alimentarea ? ');
     Readln (hh, mm);
     v[i].sfs:=hh*60+mm;
     v[i].ord:=i; end; end;
Procedure afisare_clienti;
Var i:integer;
Begin
  Write (' cand incepe sa fie servit si cand a terminat alimentarea: ');
  for i:=1 to n Do Write ('('v[i].ins,',',v[i].sfs, ',',v[i].ord')');
  Writeln;
end;
Procedure sortare_clienti;
Var i,j:integer; t:benz;
Begin
  for i:=1 to n-1 Do
    for j:=i+1 to n Do
     if (v[j].sfs<v[i].sfs) then
       Begin
         t:=v[i]; v[i]:=v[j];
         v[j]:=t; end; end;
Procedure alg_greedy;
var i:integer;
Begin
  Write ('posibilii clienti, in ordine: ');
  ultim:=1; nr:=1;Write (v[i].ord, ' ');
  for i:=2 to n do
    if (v[i].ins>v[ultim].sfs) then
      begin
        Write (v[i].ord, ' '); ultim:=i; nr:=nr+1;
      end;
  Writeln ('in total se pot alege maxim',nr, 'clienti');End;
begin
  citire_clienti; afisare_clienti;
  sortare_clienti; afisare_clienti; alg_greedy;
END.
```

5. 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[2].

```
Program P153;
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]);
end.
```

Avantaje, dezavantaje și concluzii

Algoritmii Greedy sunt foarte eficienti, dar nu conduc în mod necesar la o solutie optimă. Şi 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 are proprietatea alegerii Greedy se face de obicei prin inductie matematică[3].

Algoritmii greedy dau greș în găsirea soluției optime globale mai ales pentru că nu operează exhaustiv pe toate datele. Ei își pot lua angajamente pentru anumite alegeri prea devreme, ceea ce îi împiedică să găsească cele mai bune soluții globale mai târziu. De exemplu, toți algoritmii greedy de colorare pentru problema colorării grafurilor și pentru toate celelalte probleme NP-complete nu găsesc în mod consistent soluții optime. Cu toate acestea, ele sunt utile, deoarece acestea fac rapid alegeri și de multe ori dau aproximări bune ale soluției optime[1].

Bibliografie

1. https://ro.wikipedia.org/wiki/Algoritm_greedy
2. „Informatica” de Anatol Gremalschi, Editura Știința, 2014
3. https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy
4. https://tpascalblog.wordpress.com/
5. http://dasinika.blogspot.com/2009/04/tehnica-greedy-pentru-problemele-pentru.html