TEHNICA DE PROGRAMARE GREEDY

Proiect realizat de:
Mihaela Frecăuţanu, clasa a XI-a C

INTRODUCERE

Metoda de programare **Greedy** se aplică problemelor de optimizare.

Aceasta metoda 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 speranța că această alegere locală va conduce la **optimul global.**

Un algoritm greedy este potrivit atunci când trebuie să luăm o serie de decizii, și anume pe cea mai conveabilă la un moment dat.

1. Se **iniţializează** mulţimea soluţiilor S egală cu mulţimea vidă, S=Ø

2. La fiecare pas se alege un anumit element **x∈A** (cel mai promiţător element la momentul respectiv) care poate conduce la o soluţie optimă

3. 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.

Această tehnică este folosită într-un mare număr de probleme și aplicații printre care:

- Minimalizarea timpului mediu de așteptare
- Interclasarea optimă a șirurilor ordonate
- Coduri Huffman
- Cele mai scurte drumuri care pleacă din același punct
- Problema comis-voiajorului
- Arbori parțiali de cost minim

EXEMPLE

de programe Pascal

Probleme
rezolvate prin
metoda Greedy

1) Problema continuă a rucsacului

Sarcina: Se consideră n obiecte. Pentru fiecare obiect i (i=1, 2,...,n) se cunoaște greutatea gi și câștigul ci 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 Gmax. Elaboraţi un program care determină ce obiecte trebuie să transporte persoana în așa fel încât câștigul să fie maxim. În caz de necesitate, unele obiecte pot fi tăiate în fragmente mai mici.

Algoritm:

1. Citirea greutății fiecarui obiect;
2. Citirea capacității rucsacului;
3. Iniţializăm obiectele
4. Se ordonează obiectele crescător în funcție de greutatea lor;
5. Se inţializează volumul disponibil cu volumul obiectului;
6. Se verifică dacă fiecare obiect încape în rucsac astfel
7. Dacă a rămas o zonă din rucsac neocupată atunci afişăm volumul rămas neocupat, în caz contrar înseamnă că nu am putut introduce nici un obiect în rucsac.

```
program rucsac;
Var g:array [1..10] of integer;
 i,n,Gm,R, aux : integer;
 ok:boolean;
begin
writeln('nr obiecte'); readln(n);
writeln('capacitate rucsac'); readln(R);
writeln(' Obiectele de luat in rucsac:' );
for i:=1 to n do
 read (g[i]);
ok:=false;
while(ok=false) do
begin
ok:=true;
for i:=1 to n-1 do
if g[i]>g[i+1] then
```

```
begin
aux:=g[i];
g[i]:=g[i+1];
g[i+1]:=aux;
ok:=false;
end;
end;
writeln; for i:=1 to n do write( g[i], '*');
Gm:=0 ;
i:=1;
while ( Gm +g[i]<=R ) do
begin
Gm:=Gm+g[i];
i:=i+1;
end;
writeln('sunt' ,i-1,'obiecte cu greutate', Gm) ;
writeln ( ' a ramas' , R-Gm,' loc liber' );
end.
```

Răspuns:

Окно вывода

```
nr obiecte
7
capacitate rucsac
15
 Obiectele de luat in rucsac:
2
1
7
10
18
19
25

1*2*7*10*18*19*25*sunt 3 obiecte cu greutate 10
 a ramas 5 loc liber
```

2) Problema spectacolelor

Sarcina: Într-un oraş de provincie se organizează un festival de teatru. Oraşul are o singură sală de spectacole, iar la festival şi-au anunţat participarea mai multe trupe. Aşadar, în sală, într-o zi, trebuie planificate N spectacole. Pentru fiecare spectacol se cunoaşte intervalul în care se desfăşoară: [ora_inceput, ora_sfarsit].
Se cere să se planifice un număr maxim de spectacole care, bineînţeles, nu se pot suprapune.

Algoritm:

1. •Sortăm spectacolele după ora terminării lor;
2. •Primul spectacol programat este cel care se termină cel mai devreme;
3. •Alegem primul spectacol dintre cele care urmează în şir după ultimul spectacol programat care îndeplineşte condiţia că începe după ce s-a terminat ultimul spectacol programat;
4. •Dacă tentativa de mai sus a eşuat (nu am găsit un astfel de spectacol) algoritmul se termină; altfel seprogramează spectacolul găsit şi algoritmul se reia de la 3.

```
Program spectacole;
Type spectacol=record
 ora_inc, ora_sf:integer;
ord:integer;
 end;
Var v:array[1..30] of spectacol;
n, ultim, nr:integer;
procedure sortare;
var i,j :integer; aux:spectacol;
begin
 for i:=1 to n-1 do
 for j:=i+1 to n do
 if v[j].ora_sf < v[i].ora_sf then
 begin
 aux:=v[j];
v[j]:=v[i];
v[i]:=aux;
 end;
end;
procedure citire;
var hh, mm, i:integer;
begin
 write('Numarul de spectacole:');
 readln(n);
 for i:=1 to n do
 begin
 write('Spectacolul, i, incepe la:');
 readln(hh,mm);
 v[i]. ora_inc:=hh*60+mm;
 write ('Spectacolul, i, se termina la:');
 readln(hh,mm);
 v[i].ora_sf:=hh*60+mm;
 v[i].ord:=i;
 end;
end;
procedure greedy;
var i:integer;
begin
 writeln('Ordinea spectacolelor este:');
 ultim:=1;
 nr:=1;
 write(v[1].ord,' ');
 for i:=2 to n do
 if v[i].ora_inc>v[ultim].ora_sf then
 begin
 write(v[i].ord,' ');
 ultim:=i;
 Inc(nr);
 end;
writeln('Se pot juca ', nr, ' spectacole');
end;
begin
 citire;
 sortare;
greedy;
end.
```

Răspuns:

Окно вывода

```
Numarul de spectacole:3
Spectacolul, i, incepe la:15
60
Spectacolul, i, se termina la:17
20
Spectacolul, i, incepe la:19
30
Spectacolul, i, se termina la:21
00
Spectacolul, i, incepe la:22
30
Spectacolul, i, se termina la:24
00
Ordinea spectacolelor este:
1 2 3 Se pot juca 3 spectacole
```

3) Problema sumei maxime

Sarcina: Se dă o mulţime X={x1, x2, . . ., xn } cu elemente reale. Să se determine o submulţime a lui X astfel încât suma elementelor submulţimii să fie maximă.

Algoritm:

1. •Reprezentăm atât mulţimea X cât şi mulţimea soluţiilor S sub forma a doi vectori de numere reale;
2. •Alegerea unui element din X se face in ordine, de la 1 la n;
3. •Funcţia POSIBIL(B, x) se reduce la comparaţia x[i]>0;
4. •Procedura ADAUG(B, x) va consta din adăugarea unui element x[i]>0 la vectorul S în funcţie de contorul k.

```
program suma_maxima;
var s,x:array[1..20] of real;
i,k,n:integer;
begin
 write('Numarul de elemente n = ');
 readln(n);
 for i:=1 to n do
 begin
 write('x[',i,']= ');
 readln(x[i]);
 end;
 k:=0;
 for i:=1 to n do
 if x[i]>0 then
 begin
 k:=k+1;
 s[k]:=x[i]
 end;
 for i:=1 to k do
 write(s[i]:5:2,' ');
 readln;
end.
```

Răspuns:

Окно вывода

```
Numarul de elemente n = 5
x[1]= 2
x[2]= 6
x[3]= 7
x[4]= 8
x[5]= 9
 2.00  6.00  7.00  8.00  9.00
```

5) Problema cu submulțimea

Sarcina: Elementele sale sunt numere reale, iar cel puţin unul din ele satisface condiţia ai>0. Elaboraţi un program care determină o submulţime B, astfel încât suma elementelor din B să fie maximă.

```
Program Problema;
{ 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.
```

Răspuns:

Окно вывода

```
Daţi n=6
Daţi elementele mulţimii A:
1
2
91
283
32
375

Elementele mulţimii B:
1
2
91
283
32
375
```

Concluzie:

- Metoda Greedy nu caută să determine toate soluțiile posibile (care ar putea fi prea numeroase) și apoi să aleagă din ele pe cea optimă, ci caută să introducă direct un element x în soluția optimă. Acest lucru duce la eficiența algoritmilor Greedy, însă nu conduce 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. Acest motiv duce la însoțirea fiecărei rezolvări prin metoda Greedy a unei demonstrații matematice(în general prin inducție).

BIBLIOGRAFIE:

- http://www.worldit.info/articole/algoritmica-articole/metoda-greedy
- https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy
- https://www.scribd.com/doc/43454385/Metoda-Greedy#download
- https://www.infoarena.ro/metoda-greedy-si-problema-fractionara-a-rucsacului
- *Anatol Gremalschi*, Informatică Manual pentru clasa a 11-a, Știința, 2014