**PROIECT IDIVIDUAL**

**LA INFORMATICĂ**

**TEMA:** TEHNICA GREEDY

A REALIZAT:Stoianov Victor, clasa a XI-a ”C”

A VERIFICAT: Maria Guțu

**IPLT,,Spiru Haret”**

Informaţie generală:

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 acesti algoritmi clacheaza, cum ar fi problema comisului voiajor sau problemele NP-complete.

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.

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 eliminate  
**altfel** el nu se mai testează ulterior   
-procedeul continuă, până când au fost determinate toate elementele din mulţimea soluţiilor

Exemple de probleme:

1**. SUMA MAXIMĂ**

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

**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**.

**Problema spectacolelor**

Managerul artistic al unui festival trebuie sa selecteze o multime cat mai ampla de spectacole ce pot fi jucate in singura sala pe care o are la dispozitie.Stiind ca i s-au propus n spectacole si pentru fiecare spectacol i-a fost anuntat intervalul in care se poate desfasura [Si, Fi] (Si reprezinta ora si minutul de inceput, iar Fi ora si minutul de final al spectacolului i), scrieti un program care sa permita spectatorilor vizionarea unui numar cat mai mare de spectacole.

**Date de intrare**

Pe prima linie a fisierului de intrare *spectacole.in* se afla numarul n, numarul de spectacole propus. Pe urmatoarele n linii se vor afla 4 valori, primele doua reprezentand ora si minutul inceperii spectacolului curent, iar ultimele doua reprezentand ora si minutul terminarii spectacolului.

**Date de iesire**

Fisierul de iesire *spectacole.out* contine o singura linie, pe aceasta vor fi scrise numerele de ordine ale spectacolelor care indeplinesc solutia problemei, printr-un spatiu.

Restrictii

* n <= 100
* Exemplu

|  |  |
| --- | --- |
| **spectacole.in** | **spectacole.out** |
| 5  12 30 16 30  15 0 18 0  10 0 18 30 18 0 20 45 12 15 13 0 | 5 2 4 |

**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**.

**3. DIVIZORI NATURALI**Fiind dat numărul natural **k > 1**, se cere să se determine cel mai mic număr natural **n** având exact **k** divizori naturali proprii (diferiţi de 1 şi n).

**program** k\_divizori\_naturali;

**var** v:boolean;

k,n,s,i:integer;

**procedure** VERIF(n,k:integer;**var** v:boolean);

**var** j,i:integer;

**begin**

i:=0;

**for** j:=2 **to** n-1 **do**

**if** n **mod** j = 0 **then**

i:=i+1;

**if** i = k **then**

v:=true

**else**

v:=false;

**end**;

**begin**

write('Numarul de divizori k > 1 ');

readln(k);

write('Cel mai mic numar care are exact ',k,' divizori este ');

n:=k+2;

s:=0;

**while** s = 0 **do**

**begin**

VERIF(n,k,v);

**if** v = true **then**

**begin**

write(n);

s:=1;

**end**;

n:=n+1;

**end**;

readln;

**end**.

**Problema Stațiilor**

Staţii Patronul unei companii private de transport în comun a primit de la primăria oraşului aprobarea de a putea folosi o parte din staţiile Regiei Locale de Transport în Comun. Staţiile disponibile sunt situate de-a lungul arterei principale a oraşului. El se hotărăşte să introducă o cursă rapidă care să străbată oraşul, de la un capăt la celălalt, pe artera principală. Pentru început se ocupă de staţiile situate de aceeaşi parte a drumului. Patronul are o dilemă: dacă opririle vor fi prea dese, atunci străbaterea oraşului va dura prea mult şi va plictisi călătorii, iar dacă staţiile sunt prea rare, călătorii vor fi prea puţini. De aceea, criteriile după care patronul stabileşte staţiile în care va opri cursa rapidă sunt:

• Între două staţii alăturate să fie cel puţin x metri.

• Numărul total de staţii să fie maxim.

**Date de intrare**.

Vom considera staţiile situate pe aceeaşi parte a arterei principale numerotate în ordine, dintr-un capăt până în celălalt cu **1, 2, …, n**. Prima linie a fişierului de intrare **STATII.IN** conţine un număr natural n, reprezentând numărul de staţii situate pe artera principală. Următoarea linie conţine **n – 1** numere întregi ai**, i = 1, 2, …, n – 1** cu semnificaţia: ai este distanţa dintre staţia i şi staţia **i + 1**. Aceste numere vor fi separate prin câte un spaţiu. Date de ieşire Fişierul de ieşire **STATII.OUT** va conţine două linii. Pe prima linie se va scrie un număr întreg k care reprezintă numărul maxim de staţii alese de patron, iar pe a doua linie se vor afla k numere, reprezentând numerele de ordine ale acestor staţii. Numerele vor fi scrise în ordine crescătoare. Restricţii şi precizări **• 1 ≤ n ≤ 1000; • 1 ≤ ai ≤ 2000, i = 1, 2, ..., n – 1; • 1 ≤ k ≤ 1000;** • Dacă există mai multe soluţii, în fişier se va scrie una singură.

**Program P1;**

**. . . . .**

**Procedure P(var x,y,z:real);**

**Var j:integer;**

**Begin**

**For j:=2 to n do**

**if a[j-1] +x>=z then x:=y;**

**else x:=x+a[j-1];**

**end;**

**begin**

**. . . . .**

**End.**

**5.Se consideră mulţimea A={a1, a2, ..., ai, ..., an}**

Elementele sale sînt numere reale, iar cel puţin unul din ele satisface condiţia ai>0. Elaboraţi un program care determină o submulţime B, 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, 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 ﬁ ecare pas în submulţimea B se include un element pozitiv arbitrar din mulţimea A.

**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**.

**Concluzie:** Metoda Greedyesteuna din cele mai directe tehnici de proiectare a algoritmilor carese aplicala o varietate larga de probleme.

**Date bibliografice:**

Manual clasa a-XI-a.

<http://www.worldit.info/articole/algoritmica-articole/metoda-greedy/>

<http://info.ghibuoradea.ro/documents/09_Greedy_pregatireOJI.pdf>

<https://ru.scribd.com/document/371897213/Metoda-Greedy>