Proiect individual la informatică

Tehnica greedy

Realizat:Gherganov Alexandr

**Despre tehnică:**

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. Ea lucrează după următotii pași:

1. Stabilirea unei condiții pentru elementele introduse.

2. Selectarea pas cu pas a elementelor ce satisfac această condiție.

3. Submulțimea finală, elementele căreia au îndeplinit condițiile, și este răspunsul.

**Avantaje:**

* Are o aplicație foarte largă, fiind utilizată la rezolvarea multor tipuri de probleme.

**Dezavantaje:**

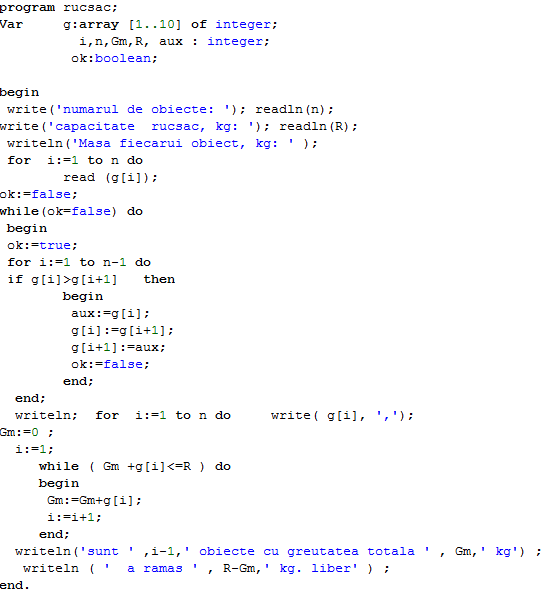
* Algoritmii Greedy nu conduc în mod necesar la o solutie optimă, ci doar la cea mai avantajoasă, care, în dependență de caz, nu mereu e cea mai bună.

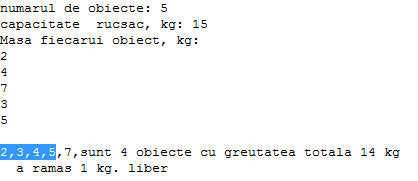
**Exemplu:**

Metoda dată este bună pentru găsirea celei mai optime soluții. Ea oferă nr. maxim de soluții posibile. Vorbind în limaj colocvial, tehnica greedy oferă cantitate, nu calitate.

Iată un program, care rezolvă o problemă legată de capacitatea unui rucsac:

Avem un rucsac, capacitatea lui maximă fiin introdusă de noi de la tastieră, un număr anumit de obiecte, dar și masa fiecăruia, la fel introduse de noi. Astfel, programul va determina care obiecte trebuie introduse în rucsac, astfel ca nr. lor să fie maxim.





Astfel, aici avem rezultatul programului. El a selectat obiectele în așa mod, ca nr. lor în rucsac să fie maxim.

Programul a lucrat într-un anumit mod. Astfel, în primul rând el a ordonat masele obiectelor în ordine crescătoare, adică 2, 3, 4, 5, 7 kg. Apoi, pe rând, el a început să scadă din capacitatea rucsacului masa fiecărui obiect, în ordine crescătoare.

15-2=13

13-3=10

10-4=6

6-5=1

1-7 -acest obiect deja nu mai încape.

Astfel, dacă introducem obiectele de 2, 3, 4 și 5kg, nr de obiecte în rucsac va fi maxim.

În continuare, mai avem niște probleme rezolvate prin această metodă:

**1. SUMA MAXIMĂ**

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.

Observaţie : evident că pentru a maximiza suma unui şir de numere acestea trebuie să fie, în primul rând, pozitive. Deci condiţia de alegere a unui element din şir ca să facă parte din mulţimea soluţie este ca acesta să fie pozitiv. Dacă, în plus, am adăuga şi condiţia suplimentară ca mulţimea soluţie să conţină un număr dat, m, de elemente (m<=n) atunci apare necesară ordonarea elementelor din mulţimea X în ordine descrescătoare, astfel ca la fiecare alegere să adăugăm la soluţie un element cu valoare maximă. În acest caz algoritmul se termină când în mulţimea soluţie au fost introduse numărul cerut de elemente.

Pentru rezolvarea problemei reprezentăm atât mulţimea X cât şi mulţimea soluţiilor S sub forma a doi vectori de numere reale. Alegerea unui element din X se face in ordine, de la 1 la n. Funcţia POSIBIL(B, x) se reduce la comparaţia x[i]>0, iar 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.

**2. K 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.

Metoda Greedy pare atât de simplă încât, la început, ne miră faptul că a fost evidenţiată ca tehnică. La o analiză atentă, vom observa însă, că lucrurile nu stau chiar aşa. Exemplele prezentate sunt didactice (le rezolvăm şi fără să ştim că există această tehnică), ele nu au alt rol decât de a evidenţia caracteristicile tehnicii.

NU întotdeauna există un algoritm de tip Greedy care găseşte soluţia optimă.

Există probleme pentru care nu se cunosc astfel de algoritmi. Mai mult, pentru cele mai multe probleme, nu se cunosc algoritmi Greedy.

**3. Problema banilor**

Este o problemă pentru care metoda Greedy determină soluţia optimă

*Scrieţi un program, care afişează modalitatea de plată, a unei sume întregi S de lei (S<20000), folosind un număr minim de bancnote. 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 linii, pe fiecare fiind indicate numărul de bancnote respectiv de 1, 5, 10, 50, 100, 200 şi 500 de lei.*

*Date de Intrare:* Fişierul text *BANI.IN* şi de la tastatură se citeşte suma *S*.

*Date de Ieşire:* Dacă e posibil de plătit această sumă *S*, atunci pe ecran se va afişa valoarea bancnotei şi numărul de bancnote respective utilizate la plată. Bancnotele nefolosite la plata sumei nu sunt afişate. Dacă nu este posibil de efectuat plata cu bancnotele indicate – afişaţi mesajul corespunzător.

(Menţionăm, că probleme asemănătoare, sunt mai multe. De obicei se presupune că dispunem de un număr nelimitat de bancnote de fiecare fel. Această problemă ne limitează însă numărul de bancnote de o anumită valoare.)

Ideea algoritmului de rezolvare a acestei probleme constă în faptul că trebuie să începem eliberarea restului de la cea mai mare bancnotă. Există 2 variante:

* dacă suma necesară e mai mare ca produsul dintre numărul de bancnote şi nominalul atunci se iau toate bancnotele de acest nominal,
* dacă nu – atunci se iau atâtea bancnote, câte „încap”, număr care se află prin împărţirea sumei rămase la nominal.

Pentru rezolvare se foloseşte un tablou cu 3 rânduri şi 7 coloane (pentru fiecare nominal câte o coloană). În primul rând al tabloului vom păstra nominalul bancnotelor, în al doilea rând - numărul bancnotelor citite din fişier şi în al treilea rând - numărul bancnotelor folosite la plată, practic ceea ce trebuie aflat. Calculul se va începe cu coloana a 7, adică începem de la sfârşit.

Program problema\_banilor;

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.

4. Problema spectacolelor

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.

Pentru descrierea algoritmului convenim că spectacolele sunt codificate cu numere întregi din mulţimea {1,2,…N} iar ora de început şi sfârşit al fiecărui spectacol este exprimată în minute scurse de la miezul nopţii

O planificare optimă a spectacolelor presupune alegerea unui număr maxim k de spectacole i1, i2,...,ik unde i1, i2,...,ik∈{1,2,…N}, şi care îndeplinesc condiţia că spectacolul ij+1 începe după terminarea spectacolului ij.

Vom construi o soluţie după următorul algoritm:

P1. Sortăm spectacolele după ora terminării lor;

P2 . Primul spectacol programat este cel care se termină cel mai devreme;

P3. 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;

P4. Dacă tentativa de mai sus a eşuat (nu am găsit un astfel de spectacol) algoritmul se termină; altfel se programează spectacolul găsit şi algoritmul se reia de la P3.

Algoritmul poate fi descris folosind diferite structuri de date:

* tablouri cu două linii şi N coloane în care să memorăm ora de început şi sfârşit al spectacolului de pe coloana j
* un vector de înregistrări în care să memorăm atât numărul de ordine al spectacolului cât şi ora de început şi sfârşit al lui.

Vom aborda algoritmul folosind a doua variantă.

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 ;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.