**Noţiuni generale**

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. Majoritatea acestor probleme constau în determinarea unei submulţimi B, a unei mulţimi A cu n elementecare să îndeplinească anumite condiţii pentru a fi acceptată. Orice astfel de submulţime care respectă aceste restricţii se numeşte soluţie posibilă. Din mulţimea tuturor soluţiilor posibile se doreşte determinarea unei soluţii care maximizează sau minimizează o funcţie de cost. O soluţie posibilă care realizează acest lucru se numeşte soluţie optimă. Considerăm că soluţiile posibile au următoarea proprietate: dacă B este o soluţie posibilă, atunci orice submulţime a sa este soluţie posibilă.Specificul acestei metode constă în faptul că se construieşte soluţia optimă pas cu pas, la fiecare pas fiind selectat (sau "înghiţit") în soluţie elementul care pare "cel mai bun" la momentul respectiv, însperanţa că va duce la soluţia optimă globală.

**Schema**

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;

**Exemplu 1**

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.

**Rezolvare**

Program bani; 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.

**Concluzie**

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 algoritmii 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.

**Exemplu 2**

Fiind data o harta cu n tari, se cere o solutie de colorare a hartii, utilizand cel mult patru culori, astfel incat doua tari ce au frontiera comuna sa fie colorate diferit. Este demonstrat faptul ca sunt suficiente numai patru culori pentru ca orice harta sa poata fi colorata.

Exemplu:

colorare.in 6

alb verde galben rosu

1 2 1 3 1 4 1 5 1 6 2 3 2 5 2 6 3 4 3 5 4 5 5 6

colorare.out

1 alb 2 verde 3 galben 4 verde 5 rosu 6 galben

**Rezolvare**

#include <fstream>

using namespace std;

ifstream fin("colorare.in");

ofstream fout("colorare.out");

int A[101][101];// A[i][j]==1 pt tari vecine, 0 altfel

int X[101],n;//X[i]=indicele culorii tarii i

char C[5][21];//culorile

void afisare()

{

for(int i=1;i<=n;i++)

fout<<i<<" "<<C[X[i]]<<"\n";//afisez tara si culoarea

fout<<endl;

}

int valid(int k)

{//1 daca tara k poate fi colorata cu X[k], 0 altfel

for(int i=1;i<k;i++)//pt tarile colorate deja

if(A[i][k]==1 && X[i]==X[k]) return 0;//exita tara invecinata care are aceeasi culoare=>0

return 1;

}

int alege(int k)

{//alege culoarea minima valida

for(int i=1;i<=4;i++)//am 4 culori

{

X[k]=i;//plasez culoarea

if(valid(k)) return i;//daca e buna o returnez

}

return 0;//daca nu putem colora

}

void colorare()

{//plaseaza in X[i] culoarea tarii i

for(int i=1;i<=n;i++)

X[i]=alege(i);//culoarea minima

}

int main()

{

int t1,t2;

fin>>n;//citesc nr de tari

for(int i=1;i<=4;i++)

fin>>C[i];//citesc culorile

while(fin>>t1>>t2)//pereche de tari vecine

{

A[t1][t2]=1;//pun 1 in matrice

A[t2][t1]=1;

}

colorare();

afisare();

return 0;

**Exemplu 3**

Fiind data o tabla de sah de dimensiunea nxn si un cal in coltul stanga sus al acesteia, se cere sa se deplaseze calul pe tabla astfel incat sa treaca o singura data prin fiecare patrat al tablei.

Solutia va fi afisata ca o matrice nxn in care sunt numerotate sariturile calului.

Exemplu, pentru n=11

1 26 85 30 23 28 99 120 21 104 101

84 31 24 27 86 119 22 103 100 121 20

25 2 83 80 29 98 107 118 115 102 105

32 81 34 87 108 79 116 97 106 19 114

3 36 69 82 75 88 109 112 117 96 93

64 33 76 35 70 111 78 95 92 113 18

37 4 63 68 77 74 89 110 57 94 91

40 65 38 71 62 67 56 73 90 17 54

5 44 41 66 49 72 61 58 55 14 11

42 39 46 7 60 51 48 9 12 53 16

45 6 43 50 47 8 59 52 15 10 13

**Rezolvare**

#include<fstream>

#include<iomanip>

using namespace std;

const int dx[8]={-1,1,2,2,1,-1,-2,-2};

const int dy[8]={-2,-2,-1,1,2,2,1,-1};

int a[201][201],n,gata;

ifstream fin("cal.in");

ofstream fout("cal.out");

void afis()

{ int i,j;

fout<<"n="<<n<<endl;

for(i=1;i<=n;i++)

{ for(j=1;j<=n;j++) fout<<setw(4)<<a[i][j]<<" ";

fout<<endl;

}

fout<<endl;

gata=1;

}

int inside(int i,int j)

{

return i>=1 && i<=n && j>=1 && j<=n;

}

int nm(int i, int j)//calculeaza in cate pozitii poate merge din i,j

{

int inou,jnou,k,x=0;

for(k=0;k<8;k++)

{ inou=i+dx[k];

jnou=j+dy[k];

if(inside(inou,jnou) && a[inou][jnou]==0) x++;

}

return x;

}

void calgreedy(int i, int j, int pas)

{ int k,v,min=9,inou,jnou,ii,jj;

if(!gata)

{

a[i][j]=pas;

if (pas==n\*n) afis();

else for(k=0;k<8;k++)

{ inou=i+dx[k];

jnou=j+dy[k];

if (inside(inou,jnou) && a[inou][jnou]==0)

{ //cauta pozitia cel mai greu accesibila

v=nm(inou,jnou);

if(v<min) { min=v; ii=inou; jj=jnou;}

}

}

if(min!=9) calgreedy(ii,jj,pas+1);

a[i][j]=0;

}

}

int main()

{ fin>>n;

calgreedy(1,1,1);

fout.close();

return 0;

}

**Exemplu 4**

Problema rucsacului (cazul continuu)   
O persoana are un rucsac cu care poate transporta o greutate maxima g. Persoana are la dispozitie n obiecte pentru care stie greutatea si castigul obtinut daca transporta obiectul.   
Fiecare obiect poate fi transportat integral sau taiat.   
Sa se precizeze ce obiecte alege persoana si in ce proportie le ia astfel incat castigul total sa fie maxim si sa nu se depaseasca greutatea maxima a rucsacului.   
Exemplu:   
g=3 n=3   
obiectele(greutate,castig):   
2 2   
1 4   
3 6   
Solutie(greutate, castig, raport taiere):   
1,4,1   
3,6,0.6667 (al doilea obiect se ia in raport de 2/3)   
castig maxim=8

**Rezolvare**

#include<fstream>

#include<iomanip>

using namespace std;

ifstream fin("date.in");

ofstream fout("date.out");

struct obiect

{

float g,c,r;//greutate, castig, raportul cat se ia din obiect

};

void citire(float &g, int &n, obiect a[])

{

fin>>g>>n;

for(int i=1;i<=n;i++)

{ fin>>a[i].g>>a[i].c;

a[i].r=0;//initial nu se foloseste obiectul

}

}

void ordonare(int n, obiect a[])//ordonare dupa castig/greutate

{

int i,j;

obiect aux;

for(i=1;i<n;i++)

for(j=i+1;j<=n;j++)

if(a[i].c/a[i].g<a[j].c/a[j].g)

{

aux=a[i]; a[i]=a[j]; a[j]=aux;

}

}

void afisare(int n, obiect a[])

{

float c=0;

for(int i=1;i<=n;i++)

{ fout<<a[i].g<<","<<a[i].c<<","<<setprecision(4)<<a[i].r<<"\n";

c=c+a[i].c\*a[i].r;

}

fout<<"castig maxim="<<c;

}

void greedy(float g, int n, obiect a[])

{

obiect s[100];

int i,k;

float sp=0;

k=0; i=1;

while(sp<g)

{

if(sp+a[i].g<=g)

{ sp=sp+a[i].g;

s[++k]=a[i];

s[k].r=1;//obiect intreg

}

else

{

s[++k]=a[i];

s[k].r=(g-sp)/a[i].g;//obiect fractionat

sp=g;

}

i++;

}

afisare(k,s);

}

int main()

{

int n;

obiect a[100];

float g;

citire(g,n,a);

ordonare(n,a);

greedy(g,n,a);

fin.close();

fout.close();

}

**Exemplu 5**

Problema programarii spectacolelor   
Intr-o sala de spectacol trebuie planificate n spectacole intr-o zi. Pentru fiecare spectacol se cunosc ora de inceput si ora de terminare (numere intregi).   
Sa se planifice un numar maxim de spectacole astfel inct sa nu fie doua spectacole care sa se suprapuna.   
Exemplu:   
date.in   
7   
2 4   
8 11   
5 6   
5 8   
3 7   
7 8   
9 12   
date.out:   
2,4 5,6 7,8 9,12

**Rezolvare**

#include<fstream>

using namespace std;

ifstream fin("date.in");

ofstream fout("date.out");

struct spectacol

{

int s,d;

};

void citire(int &n, spectacol a[])

{

fin>>n;

for(int i=1;i<=n;i++)

fin>>a[i].s>>a[i].d;

}

void ordonare(int n, spectacol a[])

{

int i,j;

spectacol aux;

for(i=1;i<n;i++)

for(j=i+1;j<=n;j++)

if(a[i].d>a[j].d)

{

aux=a[i]; a[i]=a[j]; a[j]=aux;

}

}

void afisare(int n, spectacol a[])

{

for(int i=1;i<=n;i++)

fout<<a[i].s<<","<<a[i].d<<" ";

}

void greedy(int n, spectacol a[])

{

spectacol s[100];

int i,k;

k=1;

s[1]=a[1];

for(i=2;i<=n;i++)

if(s[k].d<a[i].s) s[++k]=a[i];

afisare(k,s);

}

int main()

{

int n;

spectacol a[100];

citire(n,a);

ordonare(n,a);

greedy(n,a);

fin.close();

fout.close();

}