

# Tehnica Greedy

(Realizat de Cuciuc Loredana)

Această metodă presupune că problemele pe care trebuie să le rezolvăm au următoarea

structură:

se dă o mulțime  $A = \{a_1, a_2, \dots, a_n\}$  formată din  $n$  elemente;

se cere să determinăm o submulțime  $B$ ,  $B$  se include în  $A$ , care îndeplinește anumite condiții

pentru a fi acceptată ca soluție.



```
#include <iostream>
using namespace std;
#include <math>
#include <fcomplex>

fcomplex RMod(fcomplex z)
{
    float r = sqrt(z.r*z.r + z.i*z.i);
    return r;
}

fcomplex Conj(fcomplex z)
{
    fcomplex c;
    c.r = z.r;
    c.i = -z.i;
    return c;
}

fcomplex Cinv(fcomplex z)
{
    fcomplex c;
    float r = RMod(z);
    if (r == 0.0)
        return c;
    c.r = z.r / (r*r);
    c.i = -z.i / (r*r);
    return c;
}

fcomplex csqrt(fcomplex z)
{
    fcomplex c;
    float w;
    if ((z.r == 0.0) && (z.i == 0.0))
        c.r = 0.0;
        c.i = 0.0;
    } else {
        w = sqrt((sqrt(z.r*z.r + z.i*z.i) + z.r) / 2);
        if (z.r >= 0.0)
            c.r = w;
            c.i = z.i / (2*w);
        } else {
            c.r = -w;
            c.i = z.i / (2*w);
        }
    }
    return c;
}

int main()
{
    fcomplex z;
    z.r = 1.0;
    z.i = 1.0;
    fcomplex r = RMod(z);
    fcomplex c = Conj(z);
    fcomplex i = Cinv(z);
    fcomplex s = csqrt(z);
    cout << "RMod(" << z.r << " + j" << z.i << ") = " << r << endl;
    cout << "Conj(" << z.r << " + j" << z.i << ") = " << c.r << " + j" << c.i << endl;
    cout << "Cinv(" << z.r << " + j" << z.i << ") = " << i.r << " + j" << i.i << endl;
    cout << "csqrt(" << z.r << " + j" << z.i << ") = " << s.r << " + j" << s.i << endl;
    return 0;
}
```

# Utilizarea.

- ◆ în metoda *Greedy* se utilizează
- ◆ un **criteriu (o regulă)** care asigură alegerea directă a elementelor necesare
- ◆ din mulțimea  $A$ . De obicei, criteriile sau regulile de selecție nu sînt indicate explicit
- ◆ în enunțul problemei și formularea lor cade în sarcina programatorului. Evident, în
- ◆ absența unor astfel de criterii metoda *Greedy* nu poate fi aplicată.

## Schema generala

- Schema generala a unui algoritm bazat pe metoda *Greedy* poate fi redată cu ajutorul unui ciclu :

```
While ExistaElemente do  
Begin  
  AlegeUnElement(x);  
  IncludeElementul(x);  
End.
```

# Avantaje

## Diferența dintre metoda Greedy și metoda Trierii :

- 1. Metoda greedy conduce mai repede la aflarea soluției.
- 2. Problemele de tip *greedy* **pot** fi rezolvate cu ajutorul metodei *trierii* însă cele de tip *triere* **nu** pot fi rezolvate cu ajutorul tehnicii *greedy*.
- 3. Metoda greedy se aplică doar atunci când se deduce regula care asigură selecția directă a elementelor necesare din mulțimea  $A$

◆ După cum se vede, în metoda *Greedy* soluția problemei se caută prin testarea consecutivă a elementelor din mulțimea  $A$  și prin includerea unora din ele în submulțimea  $B$ . Într-un limbaj plastic, submulțimea  $B$  încearcă să „înghită” elementele „gustoase” din mulțimea  $A$ , de unde provine și denumirea metodei (*greedy* □ lăcom, hrăpăreț).



Elaborati un program care afiseaza elementele mai mari sau egale cu 0 dintr-un sir.

- ◇ **Program** P1;
- ◇ **var** A : **array** [1..100] **of** real;
- ◇ I,n,m : integer
- ◇ B : **array** [1..100] **of** real;
- ◇ x : real;
- ◇ **Function** ExistaElemente : boolean;
- ◇ **var** i : integer;
- ◇ **begin**
- ◇ ExistaElemente:=false;
- ◇ **for** i:=1 **to** n **do**
- ◇ **if** A[i]>0**then** ExistaElemente:=true;
- ◇ **end**;

- ◇ **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**;
- ◇ **procedure** IncludeElementul(x : real);
- ◇ **begin**
- ◇ m:=m+1;
- ◇ B[m]:=x;
- ◇ **end**;

# Programul principal

```
    begin  
        write('Dați n='); readln(n);  
        writeln('Dați elementele mulțimii A:');  
        for i:=1 to n do read(A[i]);  
        writeln;  
        m:=0;  
        while ExistaElemente do  
            begin  
                AlegeUnElement(x);  
                IncludeElementul(x);  
            end;  
        writeln('Elementele mulțimii B:');  
        for i:=1 to m do writeln(B[i]);  
        readln;  
    end.
```

# Elaborati un program care afiseaza elementele pare dintr-un sir

- ◇ **Program** P2;
- ◇ **var** A : **array** [1..100] **of** real;
- ◇ I,n,m : integer
- ◇ B : **array** [1..100] **of** real;
- ◇ x : real;
- ◇ **Function** ExistaElemente : boolean;
- ◇ **var** i : integer;
- ◇ **begin**
- ◇ ExistaElemente:=false;
- ◇ **for** i:=1 **to** n **do**
- ◇ **if** A[i]>0**then** ExistaElemente:=true;
- ◇ **end**;

- ◇ **procedure** AlegeUnElement(**var** x : real);
- ◇ **var** i: integer
- ◇ **begin**
- ◇ i:=1;
- ◇ **while** A[i] mod 2=0 **do** i:=i+1;
- ◇ x:=A[i];
- ◇ A[i]:=0;
- ◇ **end**;
- ◇ **procedure** IncludeElementul(x : real);
- ◇ **begin**
- ◇ m:=m+1;
- ◇ B[m]:=x;
- ◇ **end**;

# Programul principal

```
write('Dați n='); readln(n);
writeln('Dați elementele mulțimii A:');
  for i:=1 to n do read(A[i]);
    writeln;
    m:=0;
  while ExistaElemente do
    begin
      AlegeUnElement(x);
      IncludeElementul(x);
    end;
writeln('Elementele mulțimii B:');
  for i:=1 to m do writeln(B[i]);
    readln;
  end.
```



# Concluzii

(realizat de Cuciuc Loredana)

## Metoda Greedy



Metoda greedy poate folosi în problemele care dându-se o mulțime finită  $A$  trebuie determinată o mulțime care să îndeplinească anumite condiții.

Metoda dată furnizează o singură soluție, reprezentat prin elementul mulțimii  $S$ .

**Scop** - indentificarea problemelor în care soluțiile optimă este o submulțime inclusă într-o submulțime de dată, care trebuie să îndeplinească anumite condiții.