# Metoda Trierii

Gutu Dinu

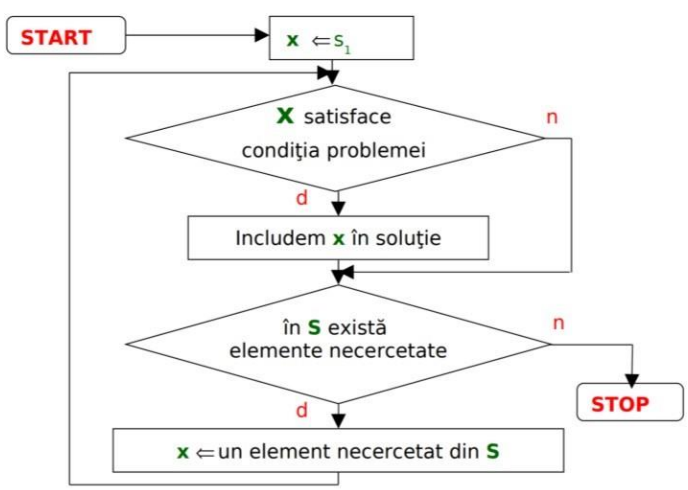
# Cuprins

# 1.Aspecte Teoretice

Se numeşte metoda trierii o metodă ce indentifică toate soluţiile unei probleme în dependenţă de mulţimea soluţiilor posibile.  Toate soluţiile se identifică prin valori, ce aparţin tipurilor de date studiate: integer, boolean, enumerare, char, subdomeniu, tablouri unidimensionale. Datorita acestei structuri de solutionare, problemele relativ simple sint efectuate rapid, astfel incadrandu-se in timpul minim de executare.

Schema generală a unui algoritm bazat pe metoda trierii poate fi redată cu ajutorul unui ciclu:
For `i:=1` to `k` do

If `SolutiePosibila(Si)` then `PrelucrareaSolutiei(Si)`, unde `SolutiePosibila` este o functie booleana care returneaza valoarea `true` daca elementul `Si` satisface conditiile problemei si `false` in caz contrar, iar `PrelucrareaSolutiei` este o procedura care efectueaza prelucrarea elementului selectat, de obicei, in aceasta procedura `Si` este afisata la ecran.



# 2. Tehnica Greedy

Metoda Greedy este o metoda care poate fi uneori folosita in rezolvarea problemelor de urmatorul tip: Se da o multime `A`. Sa se determine o submultime `B` a lui `A` astfel incat sa fie indeplinite anumite conditii, acestea depinzand de conditia propriu-zisa. In principiu problemele de acest tip pot fi rezolvate prin metoda trierii, generand cele `2^n` submultimi `A`, ale multimii `A`. Dezavantajul metodei trierii consta in faptul ca timpul cerut de algoritmii respectivi este foarte mare. Pentru a evita trierea tuturor submulţimilor `Ai`, `Ai ∈ A`, în metoda Greedy se utilizează un criteriu care asigură alegerea directă a elementelor necesare în mulţimea `A`. De obicei, criteriile sau regulile de selecţie nu sînt indicate explicit în enunţul problemei şi formularea lor cade în sarcina programatorului. Evident, în absenţa unor astfel de criterii metoda Greedy nu poate fi aplicată.

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. Funcţia `ExistaElemente` returnează valoarea true dacă în mulţimea `A` există elemente care satisfac criteriul de selecţie. Procedura `AlegeUnElement` extrage din mulţimea `A` un astfel de element x, iar procedura `IncludeElementul` înscrie elementul selectat în submulţimea `B`. Iniţial `B` este o mulţime vida. După cum se vede, în metoda Greedy soluţia problemei se caută prin testarea consecutivă a elementelor din mulţimea `A` şi prin includerea unora din ele în submulţimea `B`. Într-un limbaj plastic, submulţimea `B` încearcă să "înghită elementele "gustoase" din mulţimea `A`, de unde provine şi denumirea metodei.

# 2.1 Avantajele metodei Greedy:

1.medicii deseori se confruntă cu necesitatea aplicării metodei trierii cazurilor, când numărul răniților sau bolnavilor este foarte mare, medicul fiind suprasolicitat, în cazul unui război, sau când își periclitează propria viață în cazul unei epidemii periculoase.

2. generarea submulțimilor unei mulțimi (aflarea tuturor combinațiilor posibile), ceea ce ne poate fi foarte util în viața de zi cu zi.

3. afișarea coordonatelor a două puncte date ce au distanță minimă sau maximă, ceea ce va fi foarte folositor dacă plănuim o călătorie.

4. aflarea numărului minim de monede care pot fi date drept plată sau rest.

# 3. Probleme

1.Se consideră mulţimea P={ P1, P2, …, Pn} formată din n puncte ($2 \leq n \leq 30$) pe un plan Euclidian. Fiecare punct Pj este definit prin coordonatele sale Xj, Yj. Elaboraţi un program care afişează la ecran coordonatele punctelor Pa, Pb distanţa dintre care este maximă. Rezolvare. Mulţimea soluţiilor posibile S=P×P. Elementele (Pj, Pm) ale produsului cartezian P×P pot fi generate cu ajutorul a doua cicluri imbricate:

```
Program P152;
 Const nmax=30;
Type Punct = record
 X, y: real;
End;
Indice = 1..nmax;
```

```
Var P:array[Indice] of Punct;
J, m, n:Indice;
Dmax:real;
PA, PA: Punct;
Function Distanta(A, B: Punct): real;
Begin
Ditanta:=sqrt(sqr(A.x-B.x)+sqr(A.y-B.y));
End;
Function SolutiePosibila(j, m:Indice):Boolean;
Begin
If j<>m then SolutiePosibila:=true
 Else SolutiePosibila:=false;
End;
Procedure PrelucrareaSolutiei(A, B: Punct);
Begin
If Distanta(A,B)>dmax then
Begin
PA:=A; PB:=B;
Dmax:=Distanta(A,B);
End;
End;
Begin
Write('Dati n='); readln(n);
Writeln('Dati coordonatele x, y ale punctelor');
For j:=1 to n do
Begin
Write('P[', j,']: '); readln(P[j].x, P[j].y);
End;
Dmax:=0;
For j:=1 to n do
For m:=1 to n do
If SolutiePosibila(j, m) then
 PrelucrareaSolutiei(P[j], P[m]);
Writeln('Solutia: PA=(', PA.x:5:2, ',', PA.y:5:2, ')');
 Wtieln('Solutia: PB=(', PB.x:5:2, ',', PB.y:5:2, ')'); Readln; End.
```

2.Metoda Greedy:

Se consideră mulţimea A = {a1, a2, …, ai, …, an} elementele căreia sînt numere reale, iar cel puţin unul din ele satisface condiţia ai>0. Elaboraţi un program care determină o submulţime B, B∈A, astfel încît suma elementelor din B să fie maximă. De exemplu, pentru A = {21,5; -3,4; 0; -12,3; 83,6} avem B = {21,5; 83,6}.

```
Program P153;
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;
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 IncudeElementul (x:real);
Begin
M:=m+1;
B[m]:=x;
```

```
End;
Begin
Write('Dati n='); readln(n);
Writeln('Dati elementele multimii A:');
For i:=1 to n do read(a[i]);
Writeln;
M:=0;
While ExistaElemente do
Begin
AlegeUnElement (x);
IncludeElementul (x);
End;
Writeln('Elementele multimii B:');
For i:=1 to m do
writeln(B[i]); Readln;
End.
```

3.Se consideră numerele naturale din mulţimea {0, 1, 2, …, n}. Elaboraţi un program care determină pentru cîte numere K din această mulţime suma cifrelor fiecărui număr este egală cu
m. În particular, pentru n=100 si m=2, în mulţimea {0, 1, 2, …, 100} există 3 numere care satisfac condiţiile problemei: 2, 11 si 20. Prin urmare, K=3.

```
Program P151;
Type Natural=0..MaxInt;
Var I, k, m, n : Natural;
Function SumaCifrelor(i:Natural): Natural;
Var suma: Natural;
Begin
Suma:=0;
Repeat
 Suma:=suma+(I mod 10);
i:=i div 10;
until i=0;
SumaCifrelor:=suma;
```

```
End;
Function SolutiePosibila(i:Natural):Boolean;
Begin
If SumaCifrelor(i)=m then SolutiaPosibila:=true
 Else
SolutiePosibila:=false;
End;
Procedure PrelucrareaSolutiei(i:Natural);
Begin
Writeln('i=', i);
K:=k+1;
End;
Begin
Write('Dati n='); readln(n);
Write('Dati m='); readln(m);
K:=0;
For i:=0 to n do
If SolutiePosibila(i) then PrelucrareaSolutiei(i);
Writeln('K='
,K); Readln;
End.1.
```

# 4.Concluzie

Avantajul principal al algortmilor bazaţi pe metoda trierii constă în faptul că programele respective sînt relative simple, iar depanarea lor nu necesita teste sophisticate. Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mulţimea soluţiilor posibile S. În majoritatea problemelor de o reala importanţă practica metoda trierii conduce la algoritmii exponenţiali. Întrucit algoritmii exponenţiali sînt inacceptabili in cazul datelor de intrare foarte mari, metoda trierii este aplicată numai în scopuri didactice sau pentru elaborarea unor programe al căror timp de execuţie nu este critic. Aceasta este aplicată pe larg în soluţionarea problemelor avînd scopuri didactice. Nu toate problemele pot fi rezolvate prin aceasta metoda, dar anume cele care implica enumerarea dupa calcul, alegerea intre elemente.

# 5.Bibliografie


https://prezi.com/fgxeasy5v300/metoda-trierii/

https://www.pbinfo.ro/articole/16619/metoda-greedy

https://www.slideshare.net/BalanVeronica/metoda-trierii1?next_slideshow=1

https://www.scribd.com/doc/60874739/Proiect-la-informatica

https://www.mindmeister.com/689967199/metoda-trierii