Metoda trierii

CIOBANU BOGDAN
11C

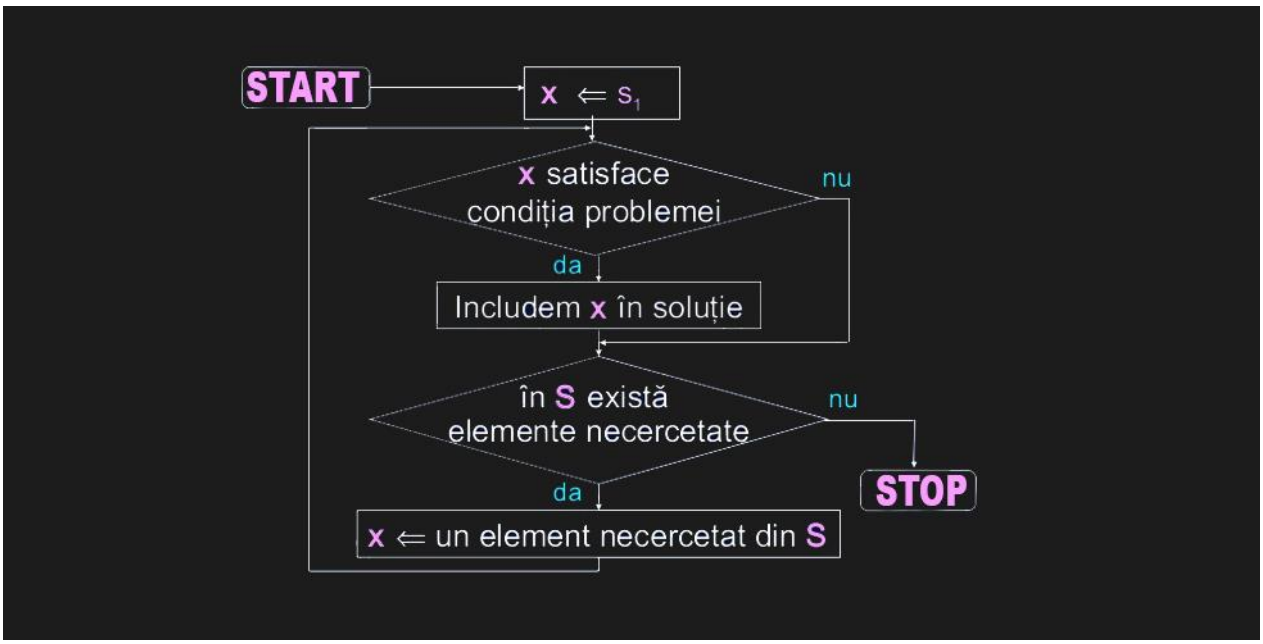
Cuprins

1. Aspecte teoretice

1.1 Definiție

Se numește metoda trierii metoda ce identifică toate soluțiile unei probleme in dependență de mulțimea soluțiilor posibile. Toate soluțiile se indentifică prin valori, ce aparțin tipurilor de date studiate: integer, boolean, enumerare sau subdomeniu. În probleme mai complicate este nevoie de a reprezenta aceste elemente prin tablouri, articole sau mulțimi.

Schema de aplicare a metodei trierii este reprezentată mai jos:



1.2 Schema generală:

for i:=1 to k do if SolutiePosibila (si) then PrelucrareaSolutiei (si)

SolutiePosibila este o funcţie booleana care returneaza valoarea true dacă elementul si satisface condiţiile problemei şi false în caz contrar, iar PrelucrareaSolutiei este o procedură care efectuează prelucrarea elementului selectat. De obicei, în această procedură soluţia si este afişată la ecran.

1.3 Operaţii legate de prelucrarea unor mulţimi:

- Reuniunea
- Intersecţia
- Diferenţa
- Generarea tuturor submulţimilor
- Generarea elementelor unui produs cartezian

• Generarea permutărilor, aranjamentelor sau combinărilor de obiecte etc.

2. Avantaje și dezavantaje

2.1 Avantaje

• Programele respective sînt relativ simple, iar depănarea lor nu necesită teste sofisticate și la verificare nu trebuie de introdus multe date;

• Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mjulțimea soluțiilor posibile S;

• Problemele relativ simple sunt efectuate rapid, incadrându-se în timpul minim de execuție.

2.2 Dezavantaje

• Întrucât algoritmii exponențiali sunt inacceptabili în cazul datelor de intrare foarte mari, metoda trierii este aplicată numai în scopuri didactice sau pentru elaborarea unor programe al căror timp de execuție nu este critic;

• Dezavantajul metodei trierii constă în faptul că timpul cerut de algoritmii respectivi este foarte mare.

3. Exemple

1. Program P1;
   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.
```

2. For j:=1 to n do

For m:=1 to n do

If SolutiePosibila(Pj, Pm) then PrelucrareaSolutiei(Pj, Pm)

Distanţa dintre punctele Pj, Pm se calculează cu ajutorul formulei:

$$Djm = \sqrt{(Xj-Xm)2+ (Yj- Ym)2}.$$

```
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.
```

3. Program P2; { Suma cifrelor unui număr natural }

```
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; { SumaCifrelor }
function SolutiePosibila(i:Natural):boolean;
begin
```

```
if SumaCifrelor(i)=m then SolutiePosibila:=true else SolutiePosibila:=false;
end; { SumaCifrelor }
procedure PrelucrareaSolutiei(i:Natural);
begin
writeln('i=', i);
K:=K+1;
end; { PrelucrareaSolutiei }
begin
write('Daţi n='); readln(n); write('Daţi m='); readln(m);
K:=0;
for i:=0 to n do
if SolutiePosibila(i) then PrelucrareaSolutiei(i);
writeln('K=', K);
readln;
end.
```

4. Program P3; { Puncte pe un plan euclidian }

```
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; { distanţa maxima } PA, PB : Punct;
function Distanta(A, B : Punct) : real;
begin
Distanta:=sqrt(sqr(A.x-B.x)+sqr(A.y-B.y));
end; { Distanta }
function SolutiePosibila(j,m:Indice):boolean;
begin
if j<>m then SolutiePosibila:=true else
SolutiePosibila:=false;
end; { SolutiePosibila }
procedure PrelucrareaSolutiei(A, B : Punct);
Begin
if Distanta(A, B)>dmax then
begin PA:=A; PB:=B; dmax:=Distanta(A, B);
end;
end; { PrelucrareaSolutiei }
begin
write(’Dati n=’); readln(n); writeln(’Daţi 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(’Soluţia: PA=(’, PA.x:5:2, ’,’, PA.y:5:2, ’)’);
writeln(’ PB=(’, PB.x:5:2, ’,’, PB.y:5:2, ’)’);
readln;
end.
```

5. Program P5 {Determinarea dacă nr. n este prim}

```
Var n,i: 1..MaxInt;
      T: boolean;
      r: real;
begin
writeln (‘Introduceți numărul n=’); readln(n);
T:=true;
r:=sqr(N);
i:=2;
while (i<=r) and t do
begin
```

```
if N mod i=0 then T:=false;
i:=i+1;
end;
write(`raspuns');
if T then writeln (`Numarul',n,'este prim');
else writeln (`Numarul',n,' nu este prim');
end.
```

4. Concluzii

Avantajul principal al algoritmilor bazați pe metoda trierii constă în faptul că programele respective sunt relativ simple, iar depanarea lor nu necesită teste sofisticate. În majoritatea problemelor de o reală importanță practică metoda trierii conduce la algoritmii exponențiali. Întrucît algoritmii exponențiali sunt inacceptabili în cazul datelor de intrare foarte mari, metoda trierii este aplicată numai în scopuri didactice sau pentru elaborarea unor programe al căror timp de execuție este critic. De obicei, algoritmii bazați pe metoda Greedy sunt algoritmi polinomiali.

5. Bibliografie

1. https://prezi.com/p/2fundh826js1/metoda-trierii/
2. https://www.slideshare.net/foegirl/metoda-trierii-33371122
3. http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html
4. https://ro.scribd.com/doc/60874739/Proiect-la-informatica
5. Manual de informatică, clasa 11