Metoda trierii

Elev: Bujor Claudiu clasa a XI-a ”C”

Prof: Guțu Maria

IPLT ”Spiru Haret”

4/21/2020

Cuprins

1.Aspecte Teoretice:

Tehnici de programare (o mica introducere):

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanţă practică pot fi rezolvate cu ajutorul unor metode standart, denumite tehnici de programare: recursia, trierea, metoda reluării, metodele euristice.

Una din cele mai răspândite tehnici de programare este recursia. Recursivitatea este procesul iterativ prin care valoarea unei variabile se determină pe baza uneia sau a mai multora dintre propriile ei valori anterioare. Structurile recursive reprezintă o alternativă de realizare a proceselor repetitive fără a utiliza cicluri. Admitem că recursia se defineşte ca o situaţie în care un subprogram se autoapeleaza fie direct, fie prin intermediul altui subprogram. Tehnicile în studiu se numesc respectiv recursia directă şi recursia indirectă. [1]

Metoda Trierii:

1) Metoda trierii presupune că soluţia unei probleme poate fi găsită analizînd consecutiv elementele s_i ale unei mulţimi finite. [1]
2) 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. [2]

Fie P o problemă, soluţia căreia se află printre elementele mulţimii S cu un număr finit de elemente.

$$S=\{s_1, s_2 , s_3 , \dots , s_n\}.$$

Soluţia se determină prin analiza fiecărui element si din mulţimea S.

SCHEMA GENERALĂ:

```
for i:=1 to k do

if SolutiePosibila (si) then PrelucrareaSolutiei (si)
```

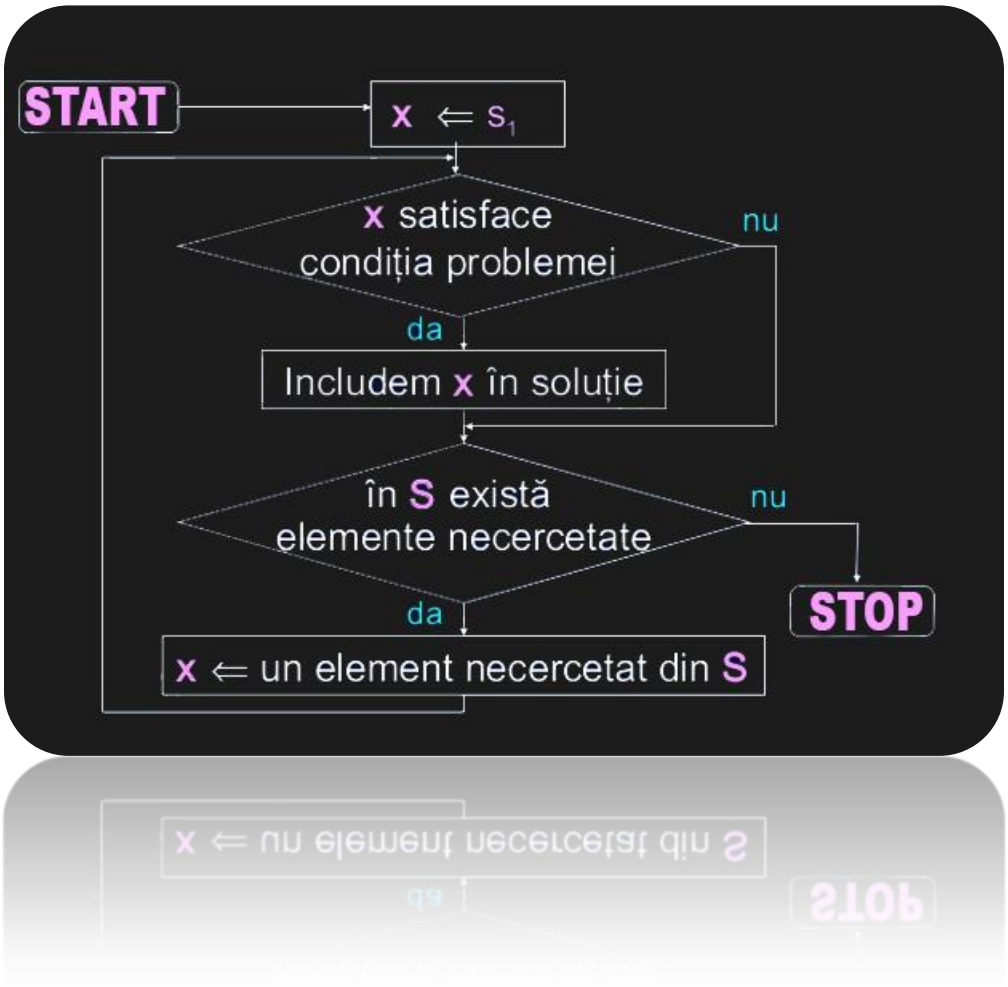
(**SolutiePosibila** este o funcţie booleana care returneaza valoarea true dacă elementul si satisface condiţiile problemei şi false în caz contrar, iar **PrelucrareaSolutiei** este o procedură care efectuează prelucrarea elementului selectat. De obicei, în această procedură soluţia si este afişată la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

Generarea soluţiilor posibile necesită elaborarea unor algoritmi speciali. În general,aceşti algoritmi realizează operaţiile legate de prelucrarea unor mulţimi:

- - reuniunea;
- - intersecţia;
- - diferenţa;
- - generarea tuturor submulţimilor;
- - generarea elementelor unui produs cartezian;
- - generarea permutărilor, aranjamentelor sau combinărilor de obiecte etc.

Avantajul principal al algoritmilor bazaţi pe metoda trierii constă în faptul că programele respective sunt relativ simple, iar depanarea lor nu necesită teste sofisticate. Complexitatea temporală a acestor algorimi este determinată de numărul de elemente k din mulţimea soluţiilor posibile S. În majoritatea problemelor de o reală importanţă practică metoda trierii conduce la algoritmiii exponenţiali. Întrucît algoritmii exponenţiali sunt inacceptabili în cazul datelor de intrare foarte mari,metoda trierii este aplicată numai în scopuri didactice sau pentur elaborarea unor programe al căror timp de execuţie nu este critic. [2]

Schema de aplicare a metodei trierii este reprezentată mai jos:
[3]



[3]

2.Avantaje Și Dezavantaje:

Avantajul principal al algoritmilor bazați pe metoda trierii constă în faptul că programele respective sunt relativ simple, iar depănarea lor nu necesită teste sofisticate. Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mulțimea soluțiilor posibile S. În majoritatea problemelor de o reală importanță practicarea metodei trierii conduce la algoritmi exponențiali. 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. [4]

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. [5]

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. [5]

3.Exemple:

Exemplul 1. [6]

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.

Rezolvare.

Evident, mulţimea soluţiilor posibile S = {0, 1, 2, …, n}. În programul ce urmează suma cifrelor oricărui număr natural i, i∈ S, se calculează cu ajutorul funcţiei SumaCifrelor. Separarea cifrelor zecimale din scrierea numărului natural “i” se efectuează de la dreapta la stinga prin împărţirea numărului “i” si a cîturilor respective la baza 10.

```
        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.
```

Exemplul 2. [6]

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:

```
            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:

$$D_{jm} = \sqrt{(X_j - X_m)^2 + (Y_j - Y_m)^2}.$$

```
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.
```

Exemplul 3. [1]

```
Program P151; { 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.
```

Exemplul 4. [1]

```
Program P152; { 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.
```

Exemplul 5. [7]

```
Program P5 {Determinarea dacă nr. n este prim}
Var n,i: 1..MaxInt;
       T: boolean;
       r: real;
begin
writeln ('Introduceţi 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.Concluzie:

Avantajul principal al algoritmilor bazați pe metoda trierii constă în faptul că programele respective sunt relativ simple, iar depănarea lor nu necesită teste sofisticate. Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mulțimea soluțiilor posibile S. În majoritatea problemelor de o reală importanță practicarea metodei trierii conduce la algoritmi exponențiali. 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.

5.Bibliografie:

1) file:///C:/Users/claud/Downloads/XI_Informatica%20(in%20limba%20romana).pdf
2) http://blogoinform.blogspot.com/p/metoda-trierii.html
3) http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html
4) https://prezi.com/fgxeasy5v300/metoda-trierii/
5) https://prezi.com/p/2fundh826js1/metoda-trierii/
6) https://www.scribd.com/doc/60874739/Proiect-la-informatica
7) https://padlet.com/alionu6ka13/w8ua77gryqlz