2020

Metoda Trierii

GORGAN BOGDAN

IPLT ,,SPIRU HARET''

Cuprins

Introducere

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanţă practică pot fi rezolvate cu ajutorul unor metode standard, denumite **tehnici de programare**: recursia, **trierea**, metoda reluării, metodele euristice ş.a. Una din cele mai răspîndite tehnici de programare este recursia. Amintim că recursia se defi neşte ca o situaţie în care un subprogram se autoapelează fi e direct, fi e prin intermediul altui subprogram. Tehnica de studiu ce va fi studiata in acest referat este **metoda trierii**.

1. Aspecte teoretice

Trierea este 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 sau tablouri unidimensionale.*

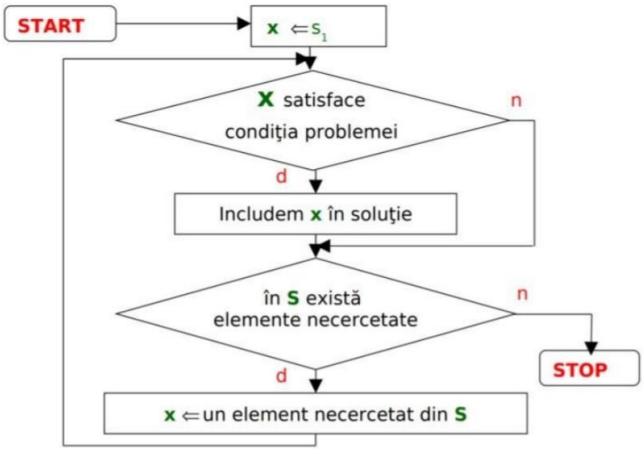
Pentru exemplificare, avem:

Fie P o problemă, soluţia căreia se află printre elementele mulţimii S cu un număr finit de elemente. S={s1, s2 , s3 , ... , sn} . 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.*

2. Exemple de programe

a. Comparatea diferentei a 2 numere cu 0:

```
   var i,rsp,x,k:integer;
 function diferenta(x:integer):integer;
   var i, d ,er :integer;
  s:string;
   begin
  str(x,s);
 val(s[1],d,er);
    for i:=2 to length(s) do begin
 val(s[i],x,er);
  d:=d-x;
   end;
  diferenta:=d;
 end;
    function solutieposibila(x:integer):boolean;
  begin
   if diferenta(x) > 0 then solutieposibila:=true;
  end;

    procedure prelucrares(x:integer);
  begin
   if solutieposibila(x) = true then
 writeln('diferenta cifrelor este pozitiva')
  else
 writeln('diferenta cifrelor este negativa');
  end;
begin
writeln('Dati numarul'); readln(x);
prelucrares(x); end.
```

b.Suma a n cifre este egala cu m:

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.

```
 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.
```

c. Numarul de patrate perfecte dintr-un sir:

```
 var
p, n, k: integer;
  function patrat(p: integer): real;
var
x, y: real;
  begin
 x := sqrt(p);
 x := round(x);rotungeste radacina pana la cel mai apropiat nr
intreg
 y := x * x;
 patrat := y;
  end;
  function solutieposibila(p: integer): boolean;
begin
 if patrat(p) = p then solutieposibila := true else
solutieposibila := false; verifica daca radacina ridicata la
patrat este egala cu nr de la 1 la n
 end;
  procedure prelucrareasolutiei(p: integer; var k: integer);
```

```
   begin
 if solutieposibila(p) = true then k := k + 1; daca este
adevarat atunci numara cate numere sunt patrate perfecte
   end;
 begin
  write('n='); readln(n);
  for p := 1 to n do prelucrareasolutiei(p, k);
  writeln(k);
 end.
```

d. Determina numarul de vocale dintr-un string si afisarea lor:

```
   var     s: string;
  I,k: integer;
   C: set of   char;
function  SolutiePosibila(s1: string;   i1: integer): boolean;
  begin
 if  s1[i1]    in   C    then SolutiePosibila:=true
   else   SolutiePosibila:=false;  <<verifică    dacă litera
     se   include in mulțimea C,   din  care fac  parte  vocalele,
     iar  dacă se   include, funcția   ia   valoarea  true.   In
     caz  contrar,  aceasta   ia   valoarea  false >>
if   s1[i1]    in   C    then C:=C-[s1[i1]];
    end;
  procedure PrelucrareaSolutiei(s1: string; i1: integer);
 begin
write(s1[i1],  '');
 end;
  begin
      k:=0;
      C:=['a', 'e', 'i', 'o', 'u']; writeln('Introduceti fraza');
     readln(s);
writeln('Vocalele din string sunt: ');
for  i:=1 to   length(s) do
if   SolutiePosibila(s,i)     then PrelucrareaSolutiei(s,i);
k:=k+1;
<<pentru   fiecare  literă    se   verifică  indeplinirea
condiției problemei, adică dacă   litera    este o    vocală, iar
dacă aceasta   se   indeplinește,  atunci se apelează  procedura
PrelucrareaSolutiei(s,i)>>
end.
```

e.Numarul numerelor prime dintr-un sir:

```
 type     Type1=2..MaxInt;
var  n,   m:Type1;
  function     SolutiePosibila(n1: Type1):boolean;
 var i: Type1;
  begin
  SolutiePosibila:=true; <<funcția va   avea valoarea  true,
     dacă această   valoare   nu   este schimbată>>
```

```
 for i:=2 to   n-1  do
if   n1   mod  i=0  then SolutiePosibila:=false;
 end;
   procedure   PrelucrareaSolutiei(n1: Type1);
 begin
 writeln('n=', n1);
end;
begin
writeln('Dati  m'); readln(m);
writeln('Numerele   prime     de   la   2    la   m    sunt:
     ');
for  n:=2 to   m    do
if   SolutiePosibila(n)  then PrelucrareaSolutiei(n); <<pentru
 fiecare  număr de  la 2 la   m se verifică  indeplinirea
condițiilor problemei, iar   dacă acestea se indeplinesc, atunci
se   apelează procedura  PrelucrareaSolutiei(i)>>
end.
```

3. Concluzii

3.1 AVANTAJE SI DEZAVANTAJE:

AVANTAJE:

- -Programele respective sînt relativ simple, iar depanarea lor nu necesită teste sofisticate si la verificare nu trebuie de introdus multe date
- -Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mulţimea soluţiilor posibile S.
- -Problemele relativ simple sunt efectuate rapid, incadrindu-se in timpul minim de executie

DEZAVANTAJE:

- -Întrucît algoritmii exponenţiali 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 execuţie nu este critic.
- -Dezavantajul metodei trierii constă în faptul că timpul cerut de algoritmii respectivi este foarte mare.

În concluzie putem spune că fiecare program necesita o abordare diferita în corespondență cu cerințele și soluțiile lui, iar metoda trierii constă în determinarea soluțiilor unei probleme prin analizarea elementelor unei mulțimi. Această metodă

include algoritmi simpli, a căror depanare este ușoară. Cu toate acestea, utilizarea metodei trierii duce deseori la obținerea unor algoritmi exponențiali, al căror timp de execuție este mare. De asemenea, complexitatea soluției scrise prin metoda trierii depinde de evaluarea funcției SolutiePosibila și nu poate fi mai eficientă decât aceasta. Astfel, metoda dată este aplicată numai în scopuri didactice sau în elaborarea unor programe ce nu au un timp de execuție critic.

4. Bibliografie

- https://prezi.com/fgxeasy5v300/metoda-trierii/
- http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html
- http://blogoinform.blogspot.com/p/metoda-trierii.html
- https://www.pbinfo.ro/articole/16619/metoda-greedy
- https://ru.scribd.com/doc/60874739/Proiect-la-informatica