1.

METODA TRIERII

PETRAȘCU BEATRICIA

CUPRINS

1. Aspecte Teoretice

1.1.Definiţie

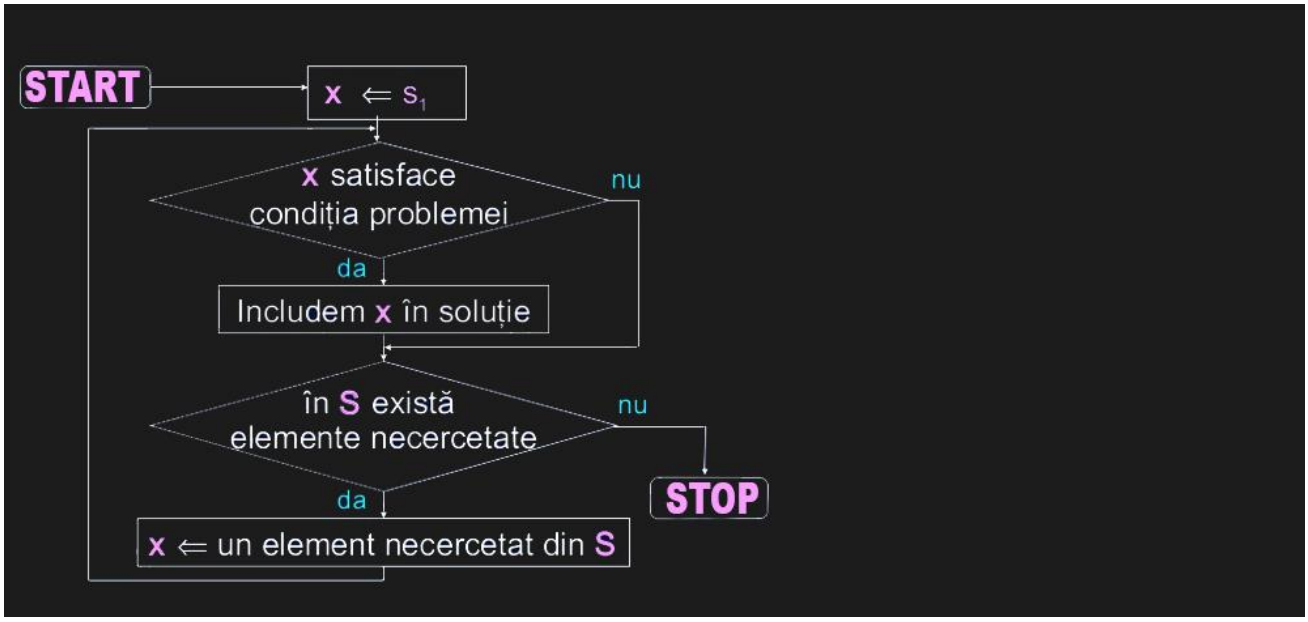
Se numeşte **metoda trierii** o metodă ce indentifică toate soluţiile unei probleme în dependenţă de mulţimea soluţiilor posibile.

Această metodă presupune că soluția unei probleme poate fi găsită analizând consecutiv elementele unei mulţimi finite numită mulţimea **soluțiilor posibile**.[1]

- Toate soluţiile se identifică prin valori, ce aparţin tipurilor de date studiate: integer, boolean, enumerare, char, subdomeniu, tablouri unidimensionale.[3]

1.2. Schema Aplicării

Schema de aplicare a metodei trierii este reprezentată mai jos:



1.3. Exemple

Metoda Trierii poate fi folosită pentru următoarele probleme din viaţă:

- Aflarea numărului minim de monede care pot fi date drept plată sau rest;
- Medicii deseori se confruntă cu necesitatea aplicării metodei trierii cazurilor, când numărul răniţilor sau bolnavilor este foarte mare, medicul fiind suprasolicitat, în cazul unui război, sau când îşi periclitează propria viaţă în cazul unei epidemii periculoase;
- Aflarea ariei maxime a unui lot de teren, avînd la dispoziţie o anumită lungime de sîrmă ghimpată, spre exemplu (ca perimetru dat);
- Generarea submulţimilor unei mulţimi (aflarea tuturor combinaţiilor posibile), ceea ce ne poate fi foarte util în viaţa de zi cu zi;
- Afișarea coordonatelor a două puncte date ce au distanță minimă sau maximă, ceea ce va fi foarte folositor dacă plănuim o călătorie;
- Calcularea şanselor de a lua premiul mare la loterie etc. [3]

2.Probleme rezolvate

2.1. Determină pentru câte numere K suma este 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.

```
Program Pex;
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.
```

2.2. Determinarea daca numarul n este prim

```
Program P12;
Var N,i:1..MaxInt;
    T:Boolean;
    R:real;
Begin
    Writeln ('Introduceto numarul 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.
```

2.3. Numerele perfecte mai mici decat numarul natural dat.

```
Program Palindrom;
Var numar:longint;
Function ePalindrom(nr:longint):Boolean;
Var lungime,i :byte;
     Temp:string;
Begin
   EPalindrom:=ture;
    Str(nr, temp);
    Lungime:=length (temp);
    For i:=1 to lungime div 2 do begin
        If temp [i] <>temp[lungime-i+i] then ePalindrom:=false;
      End;
End;
```

```
BEGIN
    Write('introduceti numarul:'); readln(numar);
    If ePalindrom(numar) then writeln ('numarul este palindrom')
                         Else writeln ('acest numar nu este un palidrom')
End.
```

2.4. Numerele polindroame mai mici decat numarul dat n.

```
Program Palindrom;
   Var k,i,n, numar,nr: integer;
Function ePalindrom(nr:longint):boolean;
   Var lungime,i :byte;
        Temp:string;
Begin
    EPalindrom:=ture;
    Str(nr, temp);
    Lungime:=length (temp);
    For i:=1 to lungime div 2 do begin
          If temp [i] <>temp[lungime-i+i] then ePalindrom:=false;
       End;
End;
Function SolutiePosibila (nr:longint):Boolean;
Begin
```

```
   If ePalindrom(nr) then SolutiaPosibila:=true
      Else SolutiaPosibila:=false;
Procedure SolutiaPosibila(nr : longint);
Begin
     Writeln('nr=' ,nr);
     K:=k+1;
     End;
Begin
   Write ('dati n='); readln(n);
    For nr:=0 to n do
     If SolutiePosibila(nr) then PrelucrareaSolutiei (nr);
     Writeln ('numarul total de palindrome:' ,k);
End.
```

2.5. Câte nr prime sunt mai mari decât nr natural dat.

```
Program P1;
Var N, t, k:integer;
Function prim (N:1…MaxInt):Boolean;
        Var i:1..MaxInt;
```

```
    T:Boolean; r:real;
Begin
   T:=True;
   R:=sqrt(N);
   i:=2;
  while (i<=r) and t do
begin
    if N mod i =0 then T:=false;
     i:=i+1;
    prim:=T;
end;
end;
Function SolutiePosibila(nr:longint):Boolean;
Begin
     If prim(N) then SolutiePosibila:=True else SolutiePosibila:=False;
End;
Procedure PrelucrareaSolutiei(N:longint);
Begin
    Writeln('N=');
     K:=k+1;
End;
BEGIN
   Write ('Dati t='); readln(t);
   For n:=0 to t do
```

If SolutiePosibila(n) **then** PrelucrareaSolutiei(n);

Writeln(‘k=’k);

END.

3. Avantaje și Dezavantaje

- In problemele mai complicate *(exemplul 5)* 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. [4]

- **Avantajele metodei:**

- 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

- **Dezavantajele metodei:**

- Este aplicată numai în scopuri didactice sau pentru elaborarea unor programe al căror timp de execuţie nu este critic.

- Timpul cerut de algoritmii respectivi este foarte mare.

- **Concluzie:**

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.

Bibliografie

1. Anatol Gremalshi, Manual de Informatică cl. XI-a, 2014
2. Daniela Oprescu, Manual de Informatică cl. XI-a, 2006
3. http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html
4. http://blogoinform.blogspot.com/p/metoda-trierii.html