METODA TRIERII

Buzu Alexandru

CL.11”D” 20.05.2020

Cuprins

Aspecte teoretice

Definiția

Se numeşte metoda trierii metoda 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*.

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 , ... , s_n} . Soluţia se determină prin analiza fiecărui element s_i din mulţimea **S**[1].

Schema generală

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

SolutiePosibila este o funcţie booleană care returnează valoarea true dacă elementul s_i satisface condiţiile problemei şi false în caz contrar, iar PrelucrareaSolutiei este o procedură care efectuează prelucrarea elementului selectat. De obicei, în această procedură soluţia s_i este afişată la ecran[2].

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]

Probleme rezolvate

1. Se consideră numerele naturale din mulțimea {0,1,2,…,n}.Elaborați un program care determină câte numere prime sunt mai mari decât numărul natural dat m[3].

```
Program P1;
 Var i,k,n,m:integer;
Function NrPrim(a:integer):boolean;
 Var i:integer;
 t:boolean; r:real;
begin
 t:=true;
 r:=sqrt(a);
 i:=2;
 while (i<=r) and t do
   begin
    if(a mod i)=0 the nt:=false;
    i:=i+1;
   end ;
 NrPrim:=t;
end ;
function SolutiePosibila(i:integer):boolean;
begin
 SolutiePosibila:=NrPrim(n);
end ;
procedure PrelucrareaSolutiei(i:integer);
  begin
    writeln('i=',i);
    k:=k+1;
 end ;
begin
 writeln('Dati n=');readln(n);
 writeln('Dati m=');readln(m);
 k:=0;
 writeln('raspuns');
 if NrPrim(n) then writeln('numarul ',n,' este prim')
              else writeln('numarul',n,' nu este prim');
 for i:=n+1 to m do
  if NrPrim(i)=true and (i<m) then PrelucrareaSolutiei(i);
 writeln('k=',k);
end .
```

2. Un număr se numește perfect dacă este egal cu suma divizorilor lui, în afară de el însuși. Se afla numerele perfecte mai mici decat numarul natural dat[4].

```
Program P2;
var n,i,m,k,x,sum:integer;
function nrperfect(x:integer):boolean;
begin
   i:=1; sum:=0;
   repeat
     if (x mod i)=0 then sum:=sum+i;
     i:=i+1;
  until i>(x div 2);
  if sum=x then nrperfect:=true
           else nrperfect:=false;
end;
procedure prelucrareasolutiei(x:integer);
begin
  writeln(x);
  k:=k+1;
end;
begin
  k:=0;
  readln(m);
  for x:=1 to m-1 do
    if nrperfect(x) then prelucrareasolutiei(x);
  writeln('k=',k);
end.
```

3. Să se scrie un program care determină toate secvenţele binare de lungime n, fiecare din ele conţinînd nu mai puţin de k cifre de 1.
Intrare: numere naturale n, 1<n
Ieşire: fiecare linie a fişierului text OUT.TXT va conţine câte o secvenţă binară distinctă, ce corespunde condiţiilor din enunţul problemei[5].

```
Program Triere;
Const nmax=20;
Type secventa= array[1..nmax] of 0..1;
Var b:secventa;
     r,i,n,k:integer;
function numara:integer;
var s,j:integer;
begin
 s:=0;
 for j:=1 to n do s:=s+b[j];
 numara:=s;
end;
procedure scrie;
var j: integer;
begin
 for j:=1 to n do write(f,b[j]);
 writeln(f);
end;
procedure urmator (var x:secventa);
var j:integer
begin
 j:=n;
 while x[j]=1 do
   begin
     x[j]:=0; j:=j-1;
   end;
 x[j]:=1;
end;
begin
 readln(n,k);
 assign(f,'OUT.TXT'); rewrite(f);
 for i:=1 to n do b[i]:=0;
 repeat
  r:= numara;
  if r >= k then scrie;
  if r < n then urmator(b);
 until r=n;
 close(f);
end.
```

4. 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[2].

```
Program P4;
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.
```

5. Se consideră mulţimea P = {P1, P2, …, Pn} formată din n puncte (2≤ n≤30) pe un plan euclidian. Fiecare punct Pj este defi nit prin coordonatele sale xj, yj. Elaboraţi un program care afi şează la ecran coordonatele punctelor Pa, Pb distanţa dintre care este maximă[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, PB : Punct;
function Distanta(A, B : Punct) : real;
begin
  Distanta:=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('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.
```

Avantaje, dezavantaje și concluzii

Avantajul principal al algoritmilor bazaţi pe metoda trierii constă în faptul că programele respective sînt relativ simple, iar depanarea lor nu necesită teste sofisticate. Complexitatea temporală a acestor algoritmi este determinată de numărul de elemente k din mulţimea soluţiilor posibile S. În majoritatea problemelor de o reală importanţă practică metoda trierii conduce la algoritmii exponenţiali. Întrucît algoritmii exponenţiali sînt 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[2].

Bibliografie

1. http://tehniciprogramare2017.blogspot.com/2017/05/metoda-trierii.html
2. „Informatica” de Anatol Gremalschi, Editura Știința, 2014
3. https://www.scribd.com/document/439549096/Probleme-rezolvate-prin-metoda-trierii
4. https://padlet.com/alionu6ka13/w8ua77gryqlz
5. https://de.slideshare.net/BalanVeronica/metoda-trierii1