METODA TRIERII

Bulhac Bogdan-Grigorie

1. Aspecte teoretice

1.1. Definitia

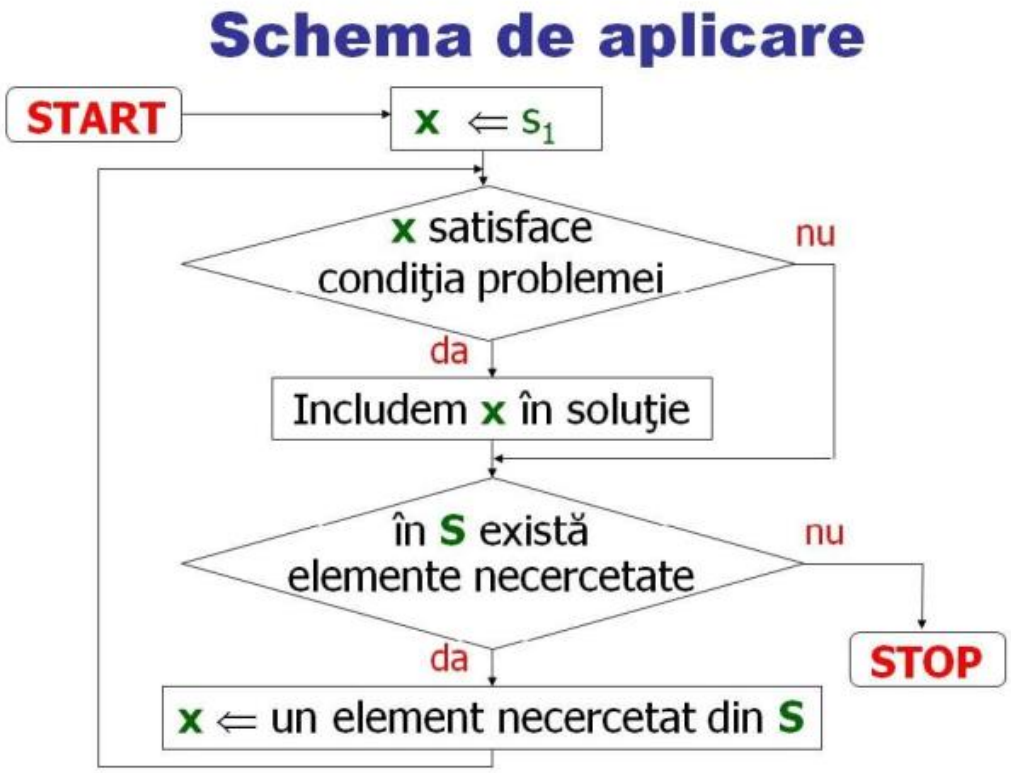
Se numește metoda trierii metoda ce identifică toate soluțiile unei probleme in dependență de mulțimea soluțiilor posibile. Toate soluțiile se indentifică prin valori, ce aparțin tipurilor de date studiate: integer, boolean, enumerare sau subdomeniu. În probleme mai complicate este nevoie de a reprezenta aceste elemente prin tablouri, articole sau mulțimi.[3]

1.2. Schema generala

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

unde 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 *si* este afişată la ecran. [1]

1.3. Schema de aplicare a metodei trierii



[2]

1.4. Operatii legate de prelucrarea unor multimi

– reuniunea;
– intersecţia;
– diferenţa;
– generarea tuturor submulţimilor;
– generarea elementelor unui produs cartezian;
– generarea permutărilor, aranjamentelor sau combinărilor de obiecte etc. [1]

1.5. Traatarea metodei in alte tari

Pentru a aplica metoda trierii, în alte țări se folosește cel mai des tipul de algoritm Greedy, care are rolul de a construi soluția optimă pas cu pas, la fiecare pas fiind selectat în soluție elementul care pare optim la momentul respectiv. [3]

2. Aplicatii si probleme

2.1. Probleme din viata rezolvabile prin metoda trierii

- 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 răniților sau bolnavilor este foarte mare, medicul fiind suprasolicitat, în cazul unui război, sau când își periclitează propria viață în cazul unei epidemii periculoase;
- aflarea ariei maxime a unui lot de teren, avînd la dispoziție o anumită lungime de sîrmă ghimpată, spre exemplu (ca perimetru dat);
- generarea submulțimilor unei mulțimi (aflarea tuturor combinațiilor posibile), ceea ce ne poate fi foarte util în viața 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 șanselor de a lua premiul mare la loterie etc. [3]

2.2. Probleme rezolvate

1.

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. [3]

Rezolvare:

```
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.

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. [2]

```
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.
```

3.

Se consideră mulţimea P = {P1, P2, ..., Pn} formată din n puncte (2 ≤n ≤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. [1]

Rezolvare: Mulţimea soluţiilor posibile S = P×P. Elementele (Pj, Pm) ale produsului cartezian P×P pot fi generate cu ajutorul a două cicluri imbricate:

```
for j:=1 to n do
for m:=1 to n do
if SolutiePosibilă(Pj, Pm) then PrelucrareaSolutiei(Pj, Pm)
```

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

d_{jm} Type equation here. $= \sqrt{(x_j - x_m)^2 + (y_j - y_m)^2}$

```
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, 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.
```

3. 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.[3]

Bibliografie

1. Anatol Gremalschi ‘’Manual pentru clasa a 11,, editura Stiinta 2014
2. Balan Veronica https://de.slideshare.net/BalanVeronica/metoda-trierii1
3. Caterina Macovenco http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html