IPLT "Spiru Haret"

Metoda Trierii

Referat elaborat de eleva clasei a 11 "D"
Guzun Laura

Mai,2020

Cuprins

1. Aspecte teoretice

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanță practică pot fi rezolvate cu ajutorul unor metode standart, denumite tehnici de programare: recursia, trierea, metoda reluării, metodele euristice[1].

Se numeşte *metoda trierii* o metodă ce indentifică toate soluțiile unei probleme în dependență de mulțimea soluțiilor posibile. Toate soluțiile se identifică prin valori, ce aparțin tipurilor de date studiate: integer, boolean, enumerare, char, subdomeniu, tablouri unidimensionale [4].

Metoda trierii presupune că soluția unei probleme poate fi găsită analizînd consecutiv elementele s_i ale unei mulțimi finite

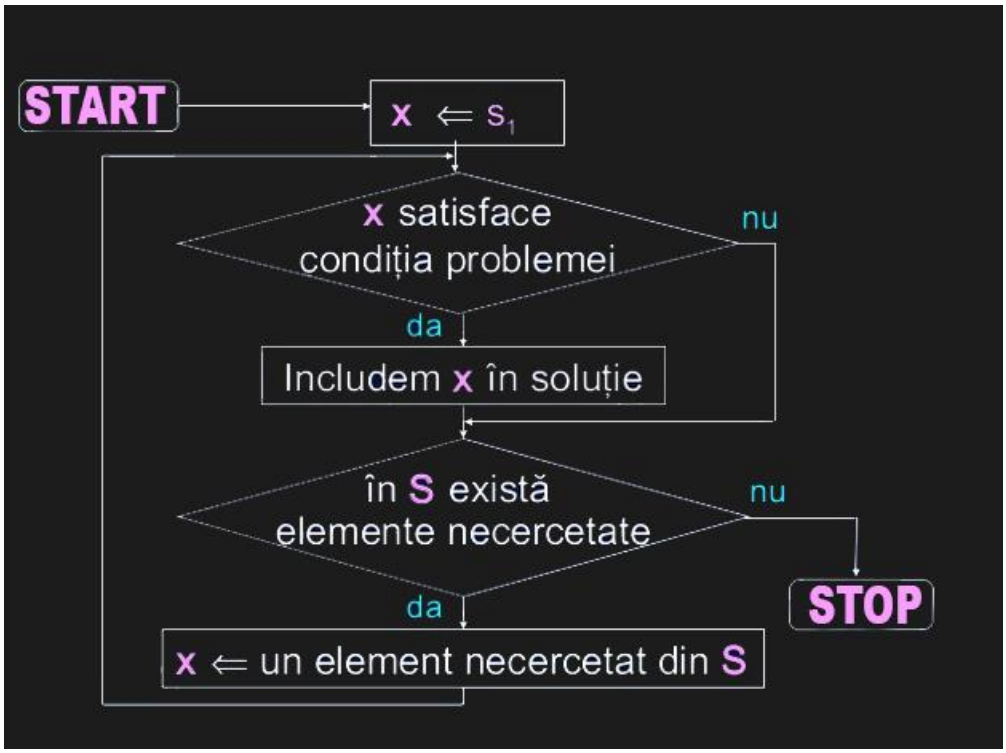
$$S = \{s_1, s_2, \ldots, s_i, \ldots, s_k\},$$

denumită **mulțimea soluțiilor posibile**. În cele mai simple cazuri elementele s, $s_i \in S$, pot fi reprezentate prin valori aparținînd unor tipuri ordinale de date: integer, boolean, char, enumerare sau subdomeniu. Menționăm că în majoritatea problemelor soluțiile posibile $s_1, s_2 \ldots, s_k$ nu sînt indicate explicit în enunțul problemei şi elaborarea algoritmilor pentru calcularea lor cade în sarcina programatorului[1]. Datorită acestei structuri de soluționare, probleme relativ de simple sînt efectuate rapid, încadrîndu-se în timpul minim de execuție[5].

Metoda trierii nu poate fi aplicată problemelor complexe ce necesită date de intrare a căror valori sunt foarte mari. Astfel de date ce sunt supuse prelucrării conduc spre algoritmi exponențiali. Pentru verificarea tuturor cazurilor va fi necesar un timp mare de execuție care depinde de numarul de elemente ce trebuie găsite in mulțimea soluțiilor posibile S. Însa timpul pentru majoritatea execuțiilor în ziua de azi este limitat de diferite circumstanțe de aceea metoda trierii se foloseste numai în cazul cînd timpul de execuție nu este critic[5].

✓*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* [4].

Schema de aplicare a metodei trierii este reprezentată mai jos[4]:



1.1 Probleme din viață rezolvabile prin metoda trierii [4]

Metoda trierii poate fi folosită pentru rezolvarea următoarelor probleme din viață:

- 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.

2. Probleme rezolvate

1. **Un numar natural se numeste palindrom daca citit de la stânga la dreapta sau de la dreapta la stânga, rămâne neschimbat. Sa se afle toate numerele palindroame mai mici decat numarul n, introdus de la tastatura [3].**

```
Program palindrom;
Var k, i, n, numar, nr:integer;

function Palindrom(nr:logint):boolean;
Var lungime, i:byte;
Temp:string;
begin
Palindrom:=true;
Str (nr, temp);
Lungime:=length (temp);
for i:=1 to lungime div 2 to begin
if temp[i] <> temp[lungime-i+1] then Palindrom:=false; end;
end;

function SolutiePosibila (nr:longint):boolean;
begin
if Palindrom(nr) then SolutiePosibila:=true else SolutiePosibila:=false;
end;

function PrelucrareSolutie (nr:longint):Boolean;
begin
writeln ('nr=', nr); k:=k+1;
end;

begin
write ( 'n=', n); readln(n);
for  nr:=0 to n do
if SolutiePosibila(nr) then PrelucrareSolutie(nr);
writeln ('Numarul total de palindroame:', k);
end.
```

2. **Sa se determine daca numarul n introdus de la tastatura este prim[3].**

```
Program Prim;
var n, i: 1..MaxInt;
t:boolean;
r:real;

begin
writeln ('n=', n); readln(n);
t:= true;
r:=sqr(n);
i:=2;
while (i<=r) and t do begin
if n mod i=o 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.
```

3. **Problema colorarii tarilor [2]:** n ţari sunt date precizându-se relaţiile de vecinătate. Se cere să se determine o posibilitate de colorare a hărţii (cu cele n ţari), astfel încât să nu existe ţări vecine colorate la fel. Se ştie faptul ca pentru colorarea unei hărţi sunt necesare cel mult 4 culori. Problema ar fi putut cere colorarea hărţii utilizând un număr minim de culori. Din păcate nu se cunosc algoritmi polinomiali care să poată colora o hartă prin utilizarea a numai 4 culori. Totuşi, colorarea hărţilor este o problemă reală. Presupunând un număr mare de ţări suntem obligaţi să folosim o metodă euristică, în care sa obţinem o colorare admisibilă, chiar dacă nu utilizează un număr minim de culori (poate chiar mai mare decât 4).

Algoritmul propus este următorul:

- ţara 1 va avea culoarea 1
- presupunem colorate primele i-1 tari
- ţara i va fi colorată cu cea mai mică culoare, cu un număr ataşat mai mare sau egal cu 1, astfel încât nici una din ţările vecine să nu fie colorată la fel.

```
Program colorare;
var a:array[1..50,1..50] of integer;
c:array[1..50] of integer;
n,i,j,cl:integer;
gasit:boolean;

begin
write ( 'numar tari: ');
readln (n);
for i:=1 to n do
for j:=1 to i-1 do begin
write('a[',i,',',j,']=');
readln(a[i,j]);
a[j,i]:=a[i,j];
end;
c[1]:=1;
for i:=2 to n do begin
cl:=1;
repeat gasit:=false;
for j:=1 to i-1 do
if (a[i,j]=1) and (cl=c[j]) then gasit:=true;
if not gasit then c[i]:=cl else cl:=cl+1;
until not gasit;
end;
for i:=1 to n do writeln ('tara ',i,' culoarea ',c[i]);
end.
```

4. **Se considera numerele naturale din multimea {0,1,2,3,…,n}. Elaborati un program care determina cate numere prime sunt mai mari decat numarul natural dat n [3].**

```
Program PP;
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) amd 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 PrelucrareSolutie (n:longint);
begin
writeln ('n=');
readln(n);
k:=k+1;
end;

begin
write ('t='); readln (t);
for n:=0 to t do
if SolutiePosibila(n) then PrelucrareSolutie(n);
writeln ('k=',k);
end.
```

5. Se consideră numerele naturale din mulțimea {0, 1, 2, …, *n*}. Elaborați un program care determină pentru cîte numere *K* din această mulțime suma cifrelor fiecărui număr este egală cu *m*. De exemplu, pentru *n* = 100 şi *m* = 2, în mulțimea {0, 1, 2, …, 100} există 3 numere care satisfac condițiile problemei: 2, 11 şi 20. Prin urmare, *K* = 3[1].

```
Program Numere;
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 SolutiePosibila:=true
else SolutiePosibila:=false;
end;
procedure PrelucrareaSolutiei(i:Natural);
begin
writeln('i=', i);
K:=K+1;
end;
begin
write('Dați n='); readln(n);
write('Dați m='); readln(m);
K:=0;
for i:=0 to n do
if SolutiePosibila(i) then PrelucrareaSolutiei(i);
writeln('K=', K);
readln;
end.
```

3. Concluzie

Avantajul principal al algoritmurilor bazati pe metoda trierii consta in faptul ca programele respective sunt relativ simple ,iar depanarea lor nu necesita teste sofisticate. Intrucat algoritmiii exponentiali sunt inacceptabili in cazul datelor de intrare foarte mari,metoda trierii este aplicata numai in scopuri didactice sau pentru elaborarea programelor al caror timp de executie nu este critic[4].

4. Bibliografie

1. Anatol Gremalschi, Manual de clasa a 11-a, editura Stiinta, 2014
2. https://www.slideshare.net/BalanVeronica/metoda-trieiialina?next_slideshow=1
3. https://padlet.com/alionu6ka13/w8ua77gryqlz
4. http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html
5. https://www.slideshare.net/BalanVeronica/metoda-trierii1