METODA TRIERII

CARLASUC SABINA CL.XI,,C”
PROF:MARIA GUTU
IPTL ,,SPIRU HARET”

CUPRINS:

- DEFINITIE
- DESCRIERE
- SCHEMA GENERALA
- OPERATII LEGATE DE PRELUAREA UNOR MULTIMI
- EXEMPLE SI CONCLUZII
- AVANTAJE/DEZAVANTAJE
- BIBLIOGRAFIE

DEFINITIE!

- Se numeste **metoda trierii** o metoda ce identifica toate solutiile unei probleme in dependenta de multimea solutiilor posibile.
- În cele mai simple cazuri **multimea solutiilor posibile** pot fi reprezentate prin valori aparţinînd unor tipuri ordinale de date: integer, boolean, char, enumerare sau subdomeniu.

DESCRIERE

- Fie P o problema, solutia careia se afla printre elementele multimii S cu un numar finit de elemente.

S = {s1, s2, …, si , …, sk}

- Solutia se determina prin analiza fiecarui element din multimea S.

Schema Generala!

- Schema generală a unui algoritm bazat pe metoda trierii poate fi redată cu ajutorul unui ciclu:

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

- *SolutiePosibila* este o funcţie booleană care returnează valoarea true dacă elementul satisface condiţiile problemei şi false în caz contrar, iar *PrelucrareaSolutiei* este o procedură care efectuează prelucrarea elementului selectat.

OPERATII LEGATE DE PRELUAREA UNOR MULTIMI...

In 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.

EXEMPLE

- **Program P151; { Suma cifrelor unui număr natural }**

```
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; { SumaCifrelor }
function SolutiePosibila(i:Natural):boolean;
begin
if SumaCifrelor(i)=m then SolutiePosibila:=true else SolutiePosibila:=false;
end; { SumaCifrelor }
procedure PrelucrareaSolutiei(i:Natural);
begin
writeln('i=', i);
K:=K+1;
end; { PrelucrareaSolutiei }
begin
write('Daţi n='); readln(n); write('Daţi m='); readln(m);
K:=0;
for i:=0 to n do
if SolutiePosibila(i) then PrelucrareaSolutiei(i);
writeln('K=', K);
readln;
end.
```

- **Program P152; { Puncte pe un plan euclidian }**

```
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; { distanţa maxima } PA, PB : Punct;
function Distanta(A, B : Punct) : real;
begin
Distanta:=sqrt(sqr(A.x-B.x)+sqr(A.y-B.y));
end; { Distanta }
function SolutiePosibila(j,m:Indice):boolean;
begin
if j<>m then SolutiePosibila:=true else
SolutiePosibila:=false;
end; { SolutiePosibila }
procedure PrelucrareaSolutiei(A, B : Punct);
Begin
 if Distanta(A, B)>dmax then
begin PA:=A; PB:=B; dmax:=Distanta(A, B);
end;
end; { PrelucrareaSolutiei }
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.
```

Concluzii bazate pe exemple:

- Din analiza programului P151 rezultă că complexitatea temporară a algoritmului respectiv este O(n). Iar complexitatea temporală a algoritmului descris cu ajutorul programului P152 este O(n2(la patrat)). Din exemplele prezentate mai sus se observă că în algoritmii bazaţi pe metoda trierii se calculează, implicit sau explicit, mulţimea soluţiilor posibile S. În problemele relativ simple (exemplul 1) elementele din mulţimea soluţiilor posibile pot fi enumerate direct. În problemele mai complicate (exemplul 2) generarea soluţiilor posibile necesită elaborarea unor algoritmi speciali.

Avantaje/Dezavantaje

- 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**.
- Dezavantajul ar fi ca 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.**

Bibliografie:

- https://www.slideshare.net/foegirl/metoda-trierii-33371122
- file:///C:/Users/1/Downloads/XI_Informatica%20(in%20limba%20romana).pdf