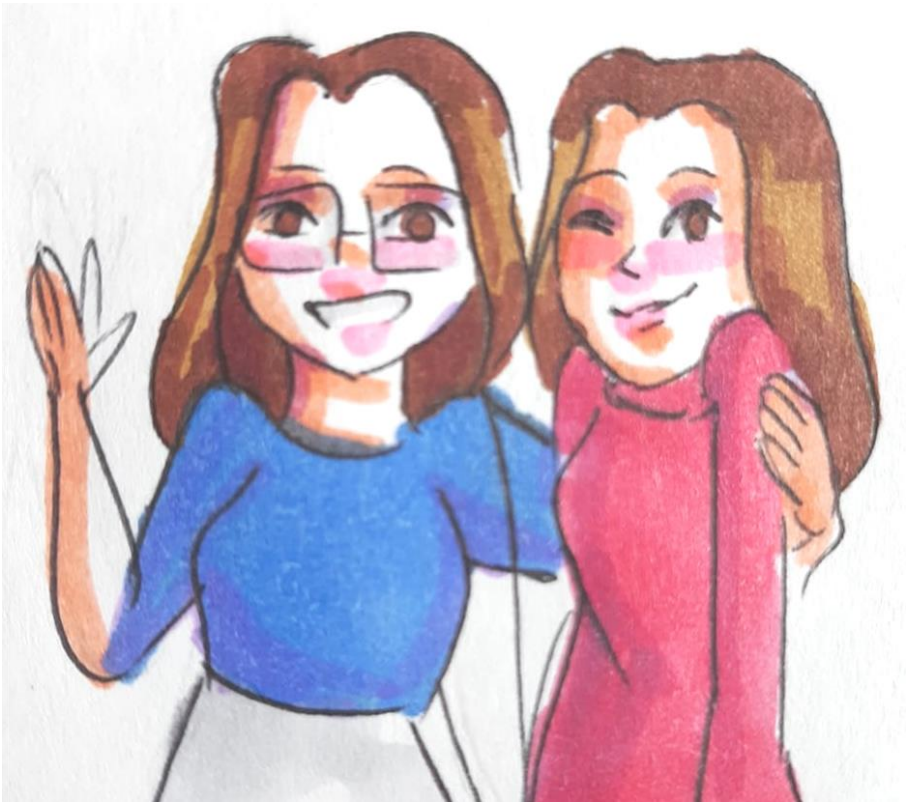
Metoda trierii

Richicinschi Mirela

Cuprins

Salutare, eu sunt Mirela si ea este Mirela si impreuna va vom prezenta Metoda treierii



1. Teorie

Metoda trierii este un algoritm ce gaseste solutia corecta dintr-un numar finit de posibile solutii care fac parte din tipul de date: integer, boolean, char, enumerare sau subdomeniu. In cazuri mai complexe solutiile sunt elemente de tablouri, articole sau multimi.

$$S=\{s1 , s2 , s3 , ... , si , ... ,sk\}$$

Trebuie de mentionat ca sarcinile problemelor nu dau niciodata mura-n gura si astfel trebuie, deobicei, programatorul sa elaboreze algoritmii pentru selectarea solutiilor din multime. Executia in sine are loc prin analinazarea consecutiva a fiecarui element.

Adevarat, dar programele executate prin metoda trierii sunt relativ simple. Cum ai observat si tu, Mirela, singura complexitate este numarul de solutii ce trebuie analizate dupa cerintele problemei. Deacea majoritatea sarcinilor ce necesita metoda data sunt cu un numar mic de date si deobicei folosite doar in scopuri didactice sau care nu necesita un timp de executie rapid

Wow..fiecare element este analizat, nu pare prea rapid sau eficient.

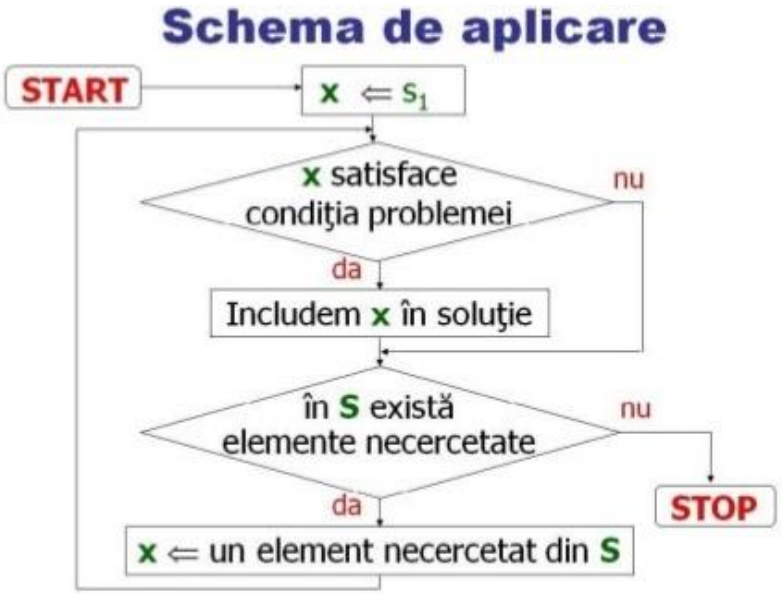
Desigur, mai jos vei gasi materialul necesar pentru a intelege mai multe.

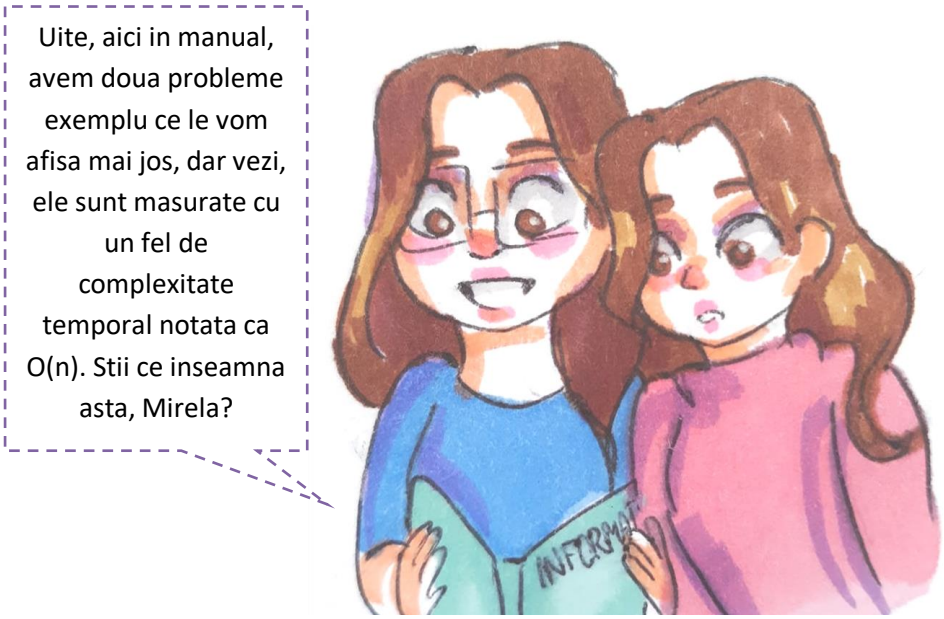
Mersi , Mirela. Ai putea sa-mi arati si schema generala a metodei trierii, te rog.

2.Schema generala a programului poate fi reprezenat prin:

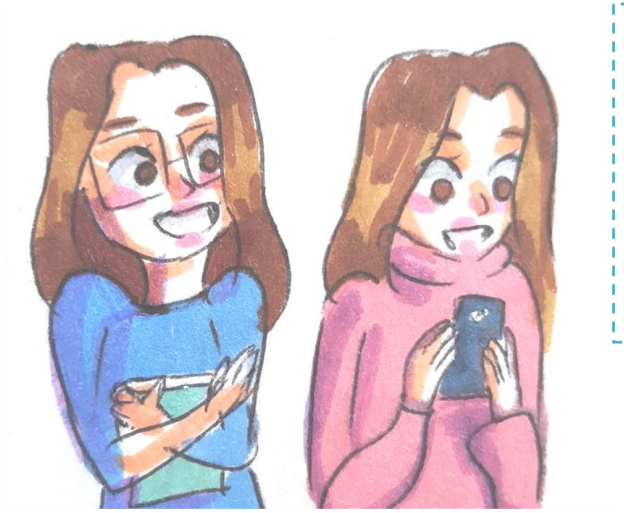
```
for i:= 1 to k do
if SolutiePosibila(si) then PrelucrareaSolutiei(si)
```

unde SolutiePosibila este o funcţie booleană care returnează valoarea true dacă elementul *si* 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.





Aha, am inteles, ei bine, atunci hai sa prezentam exemple mai jos



Problemele sunt comparate cu ajutorul unitatii relationate cu timp “O”. Notatia data este folosita de a masura eficienta unui algoritm de a executa o functie cu o multime de elemente n.

3. Exemple de probleme

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 fi ecărui număr este egală cu *m*. În particular, pentru *n* = 100 şi *m* = 2, în mulţimea {0, 1, 2, …, 100} există 3 numere care satisfac condiţiile problemei: 2, 11 şi 20. Prin urmare, *K* = 3.

```
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

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

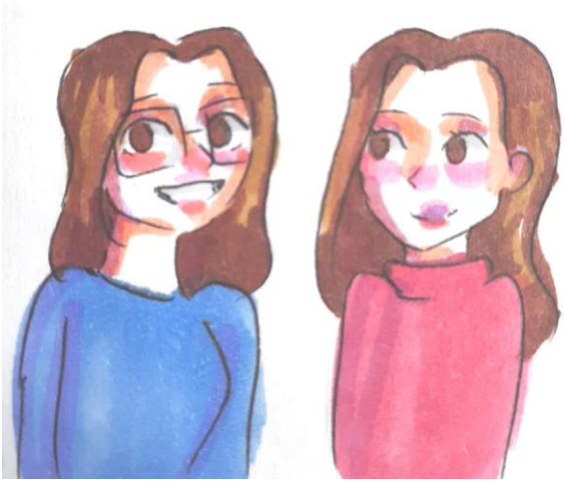
Aceasta problema are complexitatea temporara O(n)

2. Se consideră mulţimea *P* = {*P*1, *P*2, …, *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.

```
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.
```

Exemplul 2 este de complexitate O(n^2)

În problemele mai complicate (*exemplul 2*) 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.



CONCLUZIE

Folositi ce metoda va este mai comoda pana la urma

Bibliografie

1. Manualul de informatica clasa a XI-a de Anatol Gremalschi
2. https://www.pascal-programming.info/articles/sorting.php