Metoda trierii

Cuprins

Introducere

Un algoritm, atât în matematică cât şi informatică este o metodă sau o procedură de calcul, alcătuită din paşii elementari necesari pentru rezolvarea unei probleme sau categorii de probleme. De obicei algoritmii se implementează în mod concret prin programarea adecvată a unui calculator, sau a mai multora.

Multe probleme de o importanță practică pot fi rezolvare cu ajutorul unor metode standard denumite thenici de programare: Recursia; Trierea; Metoda reluării; Metode euristice.

Definiție

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.

Funcțiuni

Metoda trierii presupune că soluția unei probleme poate fi găsită analizînd consecutiv elementele si ale unei mulțimi fi nite S = {s1, s2, …, si , …, sk}, denumită mulțimea soluțiilor posibile. În cele mai simple cazuri elementele pot fi reprezentate prin valori aparținînd unor tipuri ordinale de date: integer, boolean, char, enumerare sau subdomeniu.

Trierea este sortarea in orice proces de aranjare a articolelor în funcție de o anumită secvență sau în seturi diferite și, prin urmare, are două sensuri

comune, dar distincte: 1. ordonarea: aranjarea de articole de același fel, clasă sau natura, într-o secvență ordonată, 2. clasificarea: gruparea și etichetarea articolelor cu proprietăți similare împreună.

Aplicații

Metoda Trierii cercetează toate cazurile posibile introduse astfel selectînd soluțiile care ar îndeplini condiția problemei. Datorită acestei structuri de soluționare, probleme relativ simple sînt efectuate rapid, încadrîndu-se în timpul minim de execuție. 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)* unde SolutiePosibila este o funcţie booleană care returnează valoarea true dacă elementul si satisface condiţiile problemei și false în caz contrar, iar PrelucrareaSolutiei este o procedură care efectuează prelucrarea elementului selectat.

Exemple de probleme

1.
```
Procedure P1(numbers : Array of Integer; size : Integer);
Var
     i, j, temp : Integer;

Begin
     For i := size-1 DownTo 1 do
          For j := 2 to i do
               If (numbers[j-1] > numbers[j]) Then
               Begin
                    temp := numbers[j-1];
                    numbers[j-1] := numbers[j];
                    numbers[j] := temp;
               End;

End.
```
2.
```
Procedure P2(numbers : Array of Integer; size : Integer);
Var
     i, j, index : Integer;


Begin
     For i := 2 to size-1 do
     Begin
          index := numbers[i];
```

```
          j := i;
          While ((j > 1) AND (numbers[j-1] > index)) do
          Begin
               numbers[j] := numbers[j-1];
               j := j - 1;
          End;
          numbers[j] := index;
     End;
End.
```

3.
```
Procedure QSort(numbers : Array of Integer; left : Integer;
right : Integer);
Var
     pivot, l_ptr, r_ptr : Integer;

Begin
     l_ptr := left;
     r_ptr := right;
     pivot := numbers[left];

     While (left < right) do
     Begin
          While ((numbers[right] >= pivot) AND (left < right))
do
               right := right - 1;

          If (left <> right) Then
          Begin
               numbers[left] := numbers[right];
               left := left + 1;
          End;

          While ((numbers[left] <= pivot) AND (left < right))
do
               left := left + 1;

          If (left <> right) Then
          Begin
               numbers[right] := numbers[left];
               right := right - 1;
          End;
     End;

     numbers[left] := pivot;
     pivot := left;
     left := l_ptr;
     right := r_ptr;

     If (left < pivot) Then
```

```
          QSort(numbers, left, pivot-1);

     If (right > pivot) Then
          QSort(numbers, pivot+1, right);
End;

Procedure QuickSort(numbers : Array of Integer; size :
Integer);
Begin
     QSort(numbers, 0, size-1);
End;
```

4.
```
var d : array of integer;
 a : array of Integer;
 i: integer;
 begin
 SetLength(d, 10);
 for i:=0 to length(d)-1 do d[i]:=random(200);
 SetLength(a, length(d));
 for i:=0 to length(a)-1 do a[i]:=d[i];
 for i:=0 to length(d)-1 do write(d[i],' ');
 writeln;
 SortArrayInteger(d,length(d));
 for i:=0 to length(d)-1 do write(d[i],' ');
 end.
```

5.
```
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('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.
```

Concluzie

Trierea este o tehnică utilă care este folosită de la program la program. Există diferite metode de triere care sunt utilizate în funcție de o anumită situație. Avantajul principal al algoritmilor bazați 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 mulțimea soluțiilor posibile S.

Dezavantajul ar fi ca algoritmii exponențiali 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 execuție nu este critic.

Bibliografie

1.https://www.studocu.com/en-au/document/monash-university/computer-programming/lecture-notes/sorting-techniques-in-pascal/1304173/view
2. http://www.leisy.zavodne.cz/leisy/manuals/sorting.html
3. https://www.pascal-programming.info/articles/sorting.php
4.Manual