Metoda trierii

Referat elaborat de: Furculiță Andreea, clasa a XI-a "C", IPLT "Spiru Haret"

Profesor: Guțu Maria

2020

Cuprins

Aspecte teoretice

Metoda trierii reprezintă o metodă ce identifică toate soluţiile unei probleme în dependenţă de mulţimea soluţiilor posibile.

Fie o problemă, soluţiile căreia se află printre elementele mulţimii S cu un număr finit de elemente, $S=\{s_1, s_2, s_3, \ldots, s_i, \ldots, s_k\}$. Aceste soluţii pot fi determinate analizându-se fiecare element s_i din mulţimea S. (1)

În cele mai simple cazuri, elementele $s_i \in S$ pot fi reprezentate prin valori aparţinând unor tipuri ordinale de date: integer, boolean, char, enumerare sau subdomeniu. În problemele mai complicate suntem nevoiţi să reprezentăm aceste elemente prin tablouri, articole sau mulţimi.

În cele mai multe probleme, soluţiile posibile nu sunt indicate explicit în enunţ, astfel încât programatorul trebuie să elaboreze algoritmii pentru calcularea lor. (2)

Metoda trierii realizează operaţiile legate de prelucrarea datelor unor mulţimi: reuniunea, intersecţia, diferenţa, generarea tuturor submulţimilor, generarea elementelor unui produs cartezian, generarea permutărilor, aranjamentelor sau combinărilor de obiecte, etc. (1)

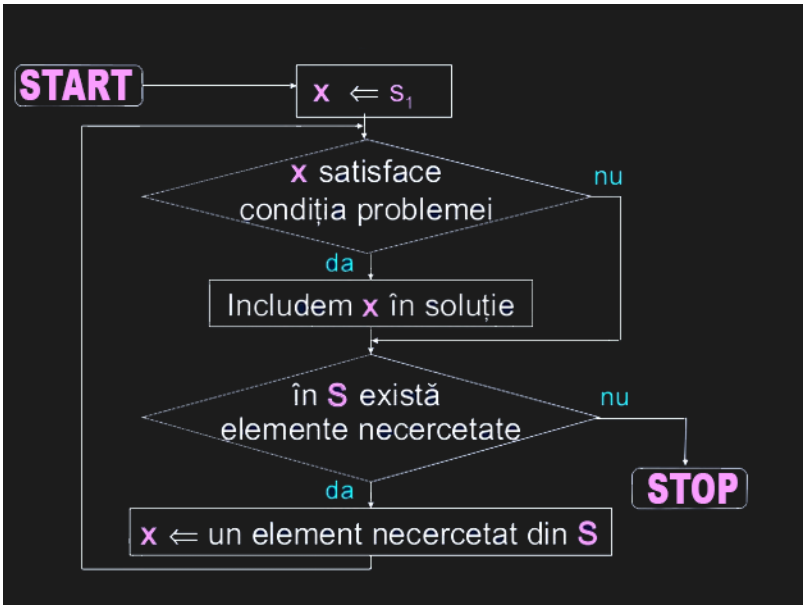
Schema generală a unui algoritm bazat pe metoda trierii poate fi redată cu ajutorul unui ciclu:

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

• SolutiePosibila reprezintă o funcţie booleană care returnează valoarea *true* dacă elementul s_i satisface condiţiile problemei şi *false* în caz contrar.

• PrelucrareaSolutiei reprezintă o procedură care efectuează prelucrarea elementului selectat. De obicei, în această procedură soluţia s_i este afişată la ecran. (2)

Schema de aplicare a metodei: (3)



Complexitatea temporală a algoritmilor bazaţi pe metoda trierii este determinată de numărul de elemente k din mulţimea soluţiilor posibile S, iar în cazul majorităţii problemelor de o reală importanţă practică, această metodă conduce la algoritmi exponenţiali. (2)

Avantaje și dezavantaje

Avantaje:

• Programele ce utilizează metoda trierii sunt relativ simple şi nu necesită foarte mult efort.

• Depanarea acestor programe nu necesită teste sofisticate.

• Pentru problemele cu un set mic de soluţii posibile, metoda trierii este calea cea mai uşoară metodă de rezolvare.

Dezavantaje:

• După cum am menţionat anterior, metoda trierii conduce la algoritmi exponenţiali atunci când este utilizată în rezolvarea problemelor de o reală importanţă practică, iar aceşti algoritmi sunt

inacceptabili în cazul datelor de intrare foarte mari, având în vedere timpul necesar de execuţie. (2)

• Pentru unele probleme, definirea funcţiei SolutiePosibila este foarte dificilă sau chiar imposibilă, ceea ce face ca metoda trierii să fie inaplicabilă.

Exemple de programe

Problema 1: Se consideră numerele naturale din mulţimea {0, 1, 2…, n}. Elaboraţi un program care determină numerele din această mulţime pentru care produsul cifrelor numărului este egal cu m şi câte numere de acest fel sunt.

```
Program Triere_1;
type Natural=0..MaxInt;
var i, k, m, n: Natural;
function ProdusulCifrelor(i: Natural): Natural; {calculează produsul cifrelor numărului}
var produsul:Natural;
begin
produsul:=1; {variabilei locale produsul i se atribuie valoarea 1, pentru ca la efectuarea primei înmulţiri, valoarea acestei variabile să devină egală cu primul factor din produs, ultima cifră din numărul i}
repeat
produsul:=produsul*(i mod 10); {produsul actual se înmulţeşte cu ultima cifră din număr, reprezentată de restul împărţirii numărului la 10}
i:=i div 10; {ultima cifră a numărului la momentul dat este eliminată, variabilei i atribuindu-se câtul împărţirii numărului actual la 10}
until i=0; {instrucţiunile din cadrul ciclului se repetă până când numărul devine egal cu 0}
ProdusulCifrelor:=produsul; {funcţiei i se atribuie valoarea variabilei produsul}
end;
```

```
function SolutiePosibila(i: Natural):boolean;
begin
if ProdusulCifrelor(i)=m then SolutiePosibila:=true
else SolutiePosibila:=false; {dacă produsul cifrelor numărului este egal cu m, atunci numărul îndeplineşte condiţiile problemei, iar funcţia ia valoarea true, iar în caz contrar, aceasta ia valoarea false}
end;
procedure PrelucrareaSolutiei(i: Natural);
begin
writeln('i=', i); {se afişează numărul}
inc(k); {k se măreşte cu 1}
end;
begin
writeln('Dati n'); readln(n);
writeln('Dati m'); readln(m); {se citesc datele de la tastatură}
k:=0; {variabilei k i se atribuie valoarea de pornire 0}
writeln('Solutiile problemei sunt: ');
for i:=0 to n do
if SolutiePosibila(i) then
PrelucrareaSolutiei(i); {pentru fiecare număr de la 0 la n se verifică îndeplinirea condiţiilor problemei, iar dacă acestea se îndeplinesc, atunci se apelează procedura PrelucrareaSolutiei(i)}
writeln('k=', k); {se afişează valoarea variabilei k, adică numărul de soluţii ale problemei}
end.
```

Problema 2: Elaboraţi un program ce afişează toate numerele pare din mulţimea {0, 1, 2,…, n}.

```
Program Triere_2;
type Natural=0..MaxInt;
var i,n: Natural;
function SolutiePosibila(i1:Natural):boolean;
begin
if i1 mod 2=0 then SolutiePosibila:=true
```

```
else solutiePosibila:=false; {dacă restul împărţirii la 2 este egal cu 0, atunci numărul este par şi îndeplineşte condiţiile problemei, iar funcţia ia valoarea true, iar în caz contrar, aceasta ia valoarea false}
end;
procedure PrelucrareaSolutiei(i1: Natural);
begin
writeln('i=', i1); {se afişează numărul}
end;
begin
writeln('Dati n'); readln(n); {se citesc datele de la tastatură}
writeln('Numerele pare sunt: ');
for i:=0 to n do
if SolutiePosibila(i) then
PrelucrareaSolutiei(i); {pentru fiecare număr de la 0 la n se verifică îndeplinirea condiţiilor problemei, iar dacă acestea se îndeplinesc, atunci se apelează procedura PrelucrareaSolutiei(i)}
end.
```

Problema 3: Elaboraţi un program ce afişează toate numerele prime de la 2 până la m, unde m este un număr natural mai mare sau egal ca 2.

```
Program Triere_3;
type Type1=2..MaxInt;
var n, m:Type1;
function SolutiePosibila(n1: Type1):boolean;
var i: Type1;
begin
SolutiePosibila:=true; {funcţia va avea valoarea true, dacă această valoare nu este schimbată}
for i:=2 to n-1 do
if n1 mod i=0 then SolutiePosibila:=false; {dacă numărul se împarte la alt număr decât 1 şi el însuşi, funcţia ia valoarea false}
end;
procedure PrelucrareaSolutiei(n1: Type1);
```

```
begin
writeln('n=', n1); {se scrie numărul}
end;
begin
writeln('Dati m'); readln(m); {citirea datelor de la tastatură}
writeln('Numerele prime de la 2 la m sunt: ');
for n:=2 to m do
if  SolutiePosibila(n) then PrelucrareaSolutiei(n); {pentru
fiecare număr de la 2 la m se verifică îndeplinirea condiţiilor
problemei, iar dacă acestea se îndeplinesc, atunci se apelează
procedura PrelucrareaSolutiei(i)}
end.
```

Problema 4: Se consideră numerele naturale din mulţimea {0, 1, 2, …, n}. Elaboraţi un program care determină numerele din această mulţime la care suma cifrelor se divide la 5 şi câte numere de acest fel sunt.

```
Program Triere_4;
type Natural=0..MaxInt;
var i, n, k: Natural;
function SumaCifrelor(i1: Natural): Natural;
var suma:Natural;
begin
suma:=0; {sumei i se atribuie valoarea 0, pentru ca la efectuarea
primei adunări, valoarea acestei variabile să devină egală cu
primul termen din produs, ultima cifră din număr}
repeat
suma:=suma+(i1 mod 10); {la suma actuală se adună ultima
cifră din număr, reprezentată de restul împărţirii numărului la
10}
i1:=i1 div 10; {ultima cifră a numărului la momentul dat este
eliminată, variabilei i atribuindu-se câtul împărţirii numărului
actual la 10}
until i1=0; {instrucţiunile din cadrul ciclului se repetă până când
numărul devine egal cu 0}
```

```
SumaCifrelor:=suma; {funcţiei i se atribuie valoarea variabilei suma}
end;
function SolutiePosibila(i1: Natural): boolean;
begin
if SumaCifrelor(i1) mod 5=0 then SolutiePosibila:=true else SolutiePosibila:=false; {dacă restul împărţirii sumei  cifrelor la 5 este 0, atunci aceasta se divide la 5 şi numărul îndeplineşte condiţiile problemei, iar funcţia ia valoarea true. În caz contrar, aceasta ia valoarea false}
end;
procedure PrelucrareaSolutiei(i1:Natural);
begin
writeln('i=',i1); {se scrie numărul}
inc(k); {Valoarea variabilei k se măreşte cu 1}
end;
begin
writeln('Dati n'); readln(n); {se citeşte n de la tastatură}
k:=0; {variabilei k i se atribuie valoarea de pornire 0}
writeln('Numerele la care suma cifrelor se divide la 5 sunt: ');
for i:=0 to n do
if SolutiePosibila(i) then PrelucrareaSolutiei(i); {pentru fiecare număr de la 0 la n se verifică îndeplinirea condiţiilor problemei, iar dacă acestea se îndeplinesc, atunci se apelează procedura PrelucrareaSolutiei(i)}
writeln('k=', k); {se afişează valoarea variabilei k, adică numărul de soluţii ale problemei}
end.
```

Problema 5: Elaboraţi un program care va afişa vocalele ce se includ într-o frază citită de la tastatură, a cărei cuvinte sunt formate din literele alfabetului român, însă nu conţin diacritice.

```
Program Triere_5;
var s: string;
i: integer;
C: set of char;
```

```
function SolutiePosibila(s1: string; i1: integer): boolean;
begin
if s1[i1] in C then SolutiePosibila:=true
else SolutiePosibila:=false; {se verifică dacă litera se include în mulţimea C, din care vor face parte vocalele, iar dacă se include, funcţia ia valoarea true. În caz contrar, aceasta ia valoarea false.}
if s1[i1] in C then C:=C-[s1[i1]]; {dacă litera data este o vocală, atunci aceasta se elimină din mulţime pentru a nu fi afişată o singură dată}
end;
procedure PrelucrareaSolutiei(s1: string; i1: integer);
begin
write(s1[i1], ' '); {se scrie litera}
end;
begin
C:=['a', 'e', 'i', 'o', 'u']; {se defineşte mulţimea vocalelor}
writeln('Introduceti o fraza');
readln(s); {se citeşte de la tastatură fraza}
s:=LowerCase(s); {toate literele sunt transformate ]n minuscule pentru a se determina toate vocalele, fără ca acele vocale ce sunt prezente atât ca majuscule, cât ca şi minuscule să fie afişate de două ori}
writeln('Vocalele din fraza sunt: ');
for i:=1 to length(s) do
if SolutiePosibila(s,i) then PrelucrareaSolutiei(s,i); {pentru fiecare literă se verifică îndeplinirea condiţiei problemei, adică dacă litera este o vocală, iar dacă aceasta se îndeplineşte, atunci se apelează procedura PrelucrareaSolutiei(s,i)}
end.
```

Concluzie

Metoda trierii constă în determinarea soluţiilor unei probleme prin analizarea elementelor unei mulţimi. Această metodă include algoritmi simpli, a căror depanare este uşoară. Cu toate acestea,

utilizarea metodei trierii duce deseori la obținerea unor algoritmi exponențiali, al căror timp de execuție este mare. De asemenea, complexitatea soluției scrise prin metoda trierii depinde de evaluarea funcției SolutiePosibila și nu poate fi mai eficientă decât aceasta. Astfel, metoda dată este aplicată numai în scopuri didactice sau în elaborarea unor programe ce nu au un timp de execuție critic.

Bibliografie

1. https://en.calameo.com/read/0053356148a6555e0d03a?fbclid=IwAR1lK8_chtLnE-ykW96VKjDR7b5Wfabqc_6Og7EdN0W8TgiYab-_gzCVAM4
2. Gremalschi Anatol, "Informatică. Manual pentru clasa a 11-a"
3. Macovenco Caterina http://caterinamacovenco.blogspot.com/p/tehnici-de-programare.html