

Instutuția Publică Liceul Teoretic Spiru Haret.

Analiză:

Metoda

Desparte și stapânește.

- 1.Date Generale.
- 2.Avantaje.
- 3.Dezavantaje.
- 4.Concluzii.
- 5.Aplicații(Exemple de probleme)
- 6.Bibliografie

Realizat de: Colomiicenco Sanda,
clasa a XI-a „D”

Date Generale:

Metoda desparte și stăpânește (divide et impera) e o metodă de elaborare a algoritmilor ce presupune:

- Împărțirea repetată a unei probleme de dimensiuni mari în $[n]$ subprobleme de acelaș tip, de dimensiuni mai mici.
- Rezolvarea subproblemelor în mod direct, dacă dimensiunea lor permite, sau redivizarea acestora.
- Combinarea soluțiilor găsite pentru construirea soluțiilor subproblemelor de dimensiuni din ce în ce mai mari
- Combinarea ultimelor soluții ale subprogramelor pentru a obține soluția finală.

Problemele rezolvate prin această metoda trebuie să îndeplinească anumite condiții:

- Se poate descompune în (doua sau mai multe) subprobleme.
- Aceste subprobleme sunt independente una față de alta (o subproblema nu se rezolvă pe baza alteia și nu se folosește rezultate celeilalte).
- Aceste subprobleme sunt similare cu problema inițială.
- La rândul lor subproblemele se pot descompune (dacă este necesar) în alte subprobleme mai simple.
- Aceste subprobleme simple se pot soluționa imediat prin algoritmul simplificat.

Totuși, deoarece nu toate problemele existente se supun acestor condiții, metoda este rar aplicată și la anumite tipuri de probleme.

Algoritmul general al metodei:

Metoda DIVIDE ET IMPERA admite o implementare recursivă, deoarece subproblemele sunt similare problemei inițiale, dar de dimensiuni mai mici.

Principiul fundamental al recursivității este autoapelarea unui subprogram când acesta este activ, ceea ce se întâmplă la un nivel, se întâmplă la orice nivel, având grijă să asigurăm condiția de terminare ale apelurilor repetate. Asemănător se întâmplă și în cazul metodei DIVITE ET IMPERA.

```
procedure DesparteSiStapineste(i, j : integer; var x : tip);
var m : integer;
x1, x2 : tip;
begin
if SolutieDirecta(i, j) then Prelucrare(i, j, x)
else
begin
m:=(j-i) div 2;
DesparteSiStapineste(i, i+m, x1);
DesparteSiStapineste(i+m+1, j, x2);
Combina(x1, x2, x);
```

end;
end;

Parametrii i și j din antetul procedurii definesc submulțimea curentă supusă prelucrării.

Funcția SolutieDirecta returnează valoarea true dacă subproblema curentă admite o rezolvare directă și false în caz contrar.

Procedura Prelucrare returnează prin parametrul x soluția subproblemei curente, calculată în mod direct.

Dacă calculul direct este imposibil, se execută două apeluri recursive

– unul pentru submulțimea $(a_i, a_{i+1}, \dots, a_m)$ și altul pentru submulțimea $(a_{i+m+1}, a_{i+m+2}, \dots, a_j)$.

Procedura Combina prelucrează soluțiile x_1 și x_2 ale subproblemelor respective și returnează soluția x a problemei curente.

Avantaje vs Dezavantaje:

+ Poate fi aplicată problemelor precum: căutarea binară, sortare rapidă, gasirea unui element în poziție, partiționarea unui șir etc.

+ Programele elaborate în baza acestei metode sunt de obicei simple, și datorită metodei sunt efectuate într-un timp relativ scurt.

-Nu toate problemele îndeplinesc criteriile necesare pentru a fi rezolvate prin această metodă, fapt ce o face limitată.

-Este aplicată doar atunci când programul poate fi divizat în subprogram mai mici cu un șablon anumit.

Concluzii:

Folosirea acestei metode ca și metodă principală, deși are caracteristici pozitive în condițiile abordate, nu este universală, fapt ce îi face întrebuințare fiind foarte rară. Totuși această metodă bazându-se pe un proces recursive și divizarea unui program masiv în subprogram de dimensiuni mai mici se dovedește a fi utilă dacă problemele îndeplinesc condițiile necesare.

Probleme rezolvate:

Maximul unui vector:

Se citeste un vector cu n componente, numere naturale. Se cere sa se tipareasca valoarea maxima.

daca $i=j$, valoarea maxima va fi $v[i]$;

contrar vom imparti vectorul in doi vectori: primul vector va contine componentele de la i la $(i+j) \div 2$, al doilea vector va contine componentele de la $(i+j) \div 2 + 1$ la j aflam maximul pentru fiecare din vectori iar solutia problemei va fi data de valoarea maxima dintre rezultatele celor doua subprobleme.

```
program maxim;
var v:array[1..10] of integer;
    n,i:integer;
function max(i,j:integer):integer;
var a,b:integer;
begin
  if i=j then max:=v[i]
  else begin
    a:=max(i, (i+j) div 2);
    b:=max((i+j) div 2+1,j);
    if a>b then max:=a
    else max:=b;
  end;
end;
begin
  write('n=');
  readln(n);
  for i:=1 to n do read(v[i]);
  writeln(maximul este ',max(1,n));
end.
```

Cel mai mare divizor comun:

Fie n valori numere naturale $a_1, a_2, a_3, \dots, a_n$. Determinati cel mai mare divizor comun al lor prin metoda Divide Et Impera.

Se imparte sirul $a_1, a_2, a_3, \dots, a_n$ in doua subsiruri $(a_1, a_2, a_3, \dots, a_m)$, respectiv $(a_{m+1}, a_{m+2}, \dots, a_n)$.

m reprezintă poziția de mijloc, $m = (1+n) \text{ div } 2$.

$\text{Cmmdc}(a_1, a_2, \dots, a_n) = \text{Cmmdc}(a_1, a_2, \dots, a_m), \text{Cmmdc}(a_{m+1}, a_{m+2}, \dots, a_n)$

```
program cmmdc_sir;
const nmax=20;
type indice=1..nmax;
var a:array[indice] of word;
n:indice;
procedure citire;
var i:indice;
begin
  readln(n);
  for i:=1 to n do read(a[i]);
end;
function euclid(x,y:word):word;
var r:word;
begin
  while y<>0 do
  begin
    r:=x mod y;
    x:=y;
    y:=r;
  end;
  euclid:=x;
end;
```

```
function cmmdc(p,q:indice):word;
var m:indice;
begin
  if q-p<=1 then
    cmmdc:=euclid(a[p],a[q])
  else
    begin
      m:=(p+q) div 2;
      cmmdc:=euclid(cmmdc(p,m),cmmdc(m+1,q));
    end;
end;
begin
  citire;
  writeln('cmmdc=',cmmdc(1,n));
  readln;
end.
```

Cautarea binara intr-un sir:

Sa se verifice daca o valoare data x exista intr-un sir de numere intregi ordonate crescator. Se va folosi un algoritm de cautare bazat pe metoda Divide Et Impera.

Se descompune problema in doua subprobleme de acelasi tip. Se imparte vectorul in doi subvectori: primul subvector va contine elementele pana la pozitia de mijloc, iar al doilea va fi alcatuit din elementele de dupa pozitia din mijloc.

Verificăm daca valoarea cautată se găsește chiar pe poziția din mijloc si in caz afirmativ cautarea se opreste.

Daca valoarea cautata este mai mica decat elementul situat pe pozitia din mijloc, cautarea trebuie continuata in subvectorul stang, iar daca este mai mare, cautarea continua in subvectorul drept.

```
program cautare;
type vector=array[1..20] of integer;
var v:vector;n,x,i:integer;
function caut(p,q:integer):integer;
var mij:integer;
begin
  if q<p then caut:=-1
  else
    begin
      mij:=(p+q) div 2;
      if v[mij]=x then caut:=mij
      else if x<v[mij] then caut:=(p,mij-1)
      else caut:=caut(mij+1,q);
    end;
  end;
begin
  write('n=');
  readln(n);
  write('v[1]=');
  readln(v[1]);
  for i:=2 to n do
    repeat
      write('v[',i,']=');
      readln(v[i]);
    until v[i]>v[i-1];
  write('x=');
  readln(x);
  writeln(caut(1,n));
end.
```

Problema plierilor:

Se considera un vector de lungime n . Definim plierea vectorului prin suprapunerea unei jumătăți, numită donoare, peste cealaltă jumătate, numită receptoare, cu precizarea că dacă numărul de elemente este impar, elementul din mijloc este eliminat. Prin pliere, elementele subvectorului obținut vor avea număratoarea jumătății receptoare. Plierea se poate aplica în mod repetat, până când se ajunge la un subvector format dintr-un singur element, numit element final. Scrieți un program care să afișeze toate elementele finale posibile și să figureze pe ecran pentru fiecare element final o succesiune de plieri.

```
program plieri;
const nmax=50;
type element=1..nmax;
var n,i:element;
efinal:array[element] of boolean;
m:array[element] of string;
procedure pliaza(p,q:element);
begin
  if p=q then efinal [p]:=true
  else
    begin
      if (q-p+1) mod 2=1 then
        begin
          efinal[(p+q) div 2]:=false;
          ls:=(p+q) div 2-1;
        end
      else
        ls:=(p+q) div 2;
        ld:=(p+q) div 2+1;
        pliaza(p,ls);
        str(ls,ss);
        str(ld,sd);
        for i:=p to ls do
          m[i]:='d'+sd+' '+m[i];
        end;
      end;
    begin
      write(n=');
      readln(n);
      for i:=1 to n do m[i]:=' ';
      pliaza(1,n);
      writeln('elementele finale sunt:');
      for i:=1 to n do if efinal[i] then begin write(' ');
        writeln(m[i]);
      end;
      writeln;
    end.
end.
```

Bibliografie:

<http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php>

<http://www.scribub.com/stiinta/informatica/METODA-DIVIDE-ET-IMPERA25186243.php>

<https://www.slideshare.net/LuminiaMihailov/metoda-divide-et-impera-47939776>

<https://tehnicideprogramare12.weebly.com/metoda-divide-et-impera.html>