Divide et impera

Este o metodă specială și se bazează pe următorul principiu:

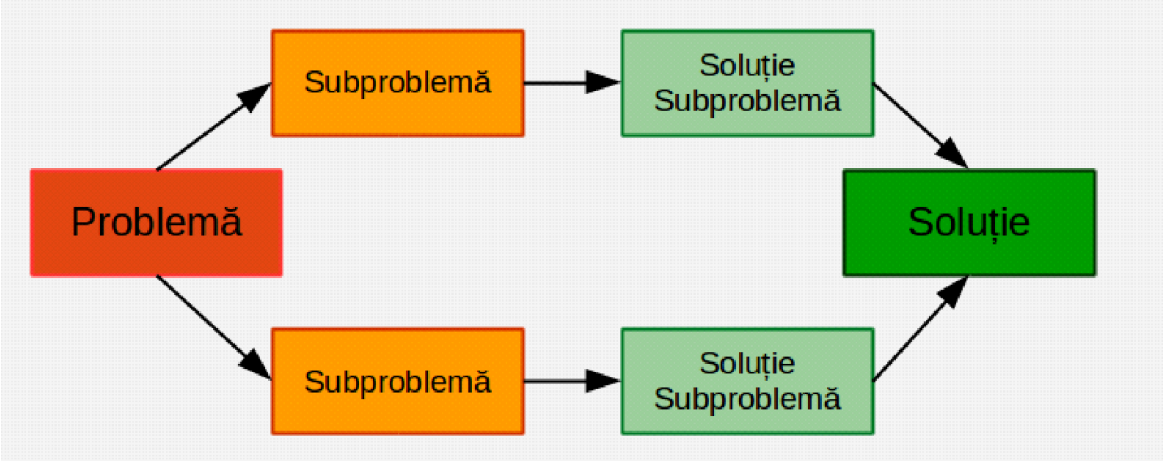
Descompunem problema în două sau mai multe subprobleme (mai ușoare) care se rezolvă pe rând iar soluția se obține prin combinarea rezultatelor.

Aceste probleme admit o reimplementare recursivă

Folosind această metodă există 2 cazuri:

1)Ajungem la o problemă care admite rezolvarea imediată caz în care se rezolvă și se revine din apel.

2)Nu se poate rezolva recursiv și descompunem problema în 2 sau mai multe subprobleme pentru fiecare dintre ele reapelăm funcția,combinăm rezultatele și revenim în apel.



Algoritmul de bază prin care funcționează divide et impera este următorul:

```
Algoritm DivImp(P)
    Dacă P este problemă elementară
        R <- RezolvăDirect(P)
    Altfel
        [P1,P2] <- Descompune(P)
        R1 <- DivImp(P1)
        R2 <- DivImp(P2)
        R <- Combină(R1,R2)
    SfârșitDacă
```

Suma elementelor dintr-un vector

Fie un vector V cu n elemente întregi,indexate de la 1 la n

Această problemă poate fi rezolvată astfel utilizând divide et impera

Inițial se ia st=1 și dr=n

Se verifică dacă acestea sunt egale iar dacă sunt rezultatul este egal cu st.

Dacă acestea nu sunt egale atunci incepe rezolvarea prin metoda divide et impera.

Se găsește mijlocul vectorului adică se împarte vectorul în două.

După ce am aflat cele două părți(“subprobleme”) acestea sunt rezolvate.

Se află S1 adică suma din prima secvență și S2 suma din a doua secvență.

În final combinăm rezultatele S1 și S2 astfel rezolvând problema principală.

Sortarea prin interclasare

Este o metodă eficientă de sortare care funcționează prin împarțirea în 2 vectori care odata sortați se interclasează.

Descompunerea unui vector în alți doi vectori are loc până când avem de sortat vectori cu 1 sau 2 componente.

QuickSort

Este o metodă de sortare care constă în:

-Alegerea unui element special al listei numit pivot.

- se ordonează elementele listei, astfel încât toate elementele din stânga pivotului să fie mai mici sau egale cu acesta, și toate elementele din dreapta pivotului să fie mai mari sau egale cu acesta

-se continuă recursiv cu secvența din stânga pivotului și cu cea din dreapta lui.

Aplicații:

1)SumVec #1015

Se consideră un șir cu n elemente, numere naturale. Folosind metoda Divide et Impera, determinați suma elementelor acestui șir.

Intrare

6

4 1 8 4 3 5

Ieșire

25

```cpp
#include<iostream>
using namespace std;
int S(int a[], int st, int dr){
if (st==dr)

 return a[st];
    else
    {

        int m=(st+dr)/2;
         return S(a, st,m)+S(a, m+1,dr);
    }
}
int main()

{
    int n,a[1001];
    cin>>n;
    for(int i=1; i<=n; i++)
        cin>>a[i];
    cout<<S(a,1,n);
    return 0;

}
//sumVec #1015

```

```
"C:\informatica c++\informatica\divide1.exe"
6
4 1 8 4 3 5
25
Process returned 0 (0x0)   execution time : 9.213 s
Press any key to continue.
```

2)ExistaImpareDivImp #1148

Se dă un şir cu n elemente, numere naturale. Folosind metoda Divide et Impera să se verifice dacă în şir există elemente impare.

Intrare

5

2 8 6 10 8

Ieşire

NU

```cpp
#include<iostream>
using namespace std;
bool imp(int a[],int st,int dr)
{

   if(st==dr)
    return a[st]%2==1;
   else{
   int m=(st+dr)/2;
   return imp(a,st,m)|| imp(a,m+1,dr);
}
}

int main()
{
    int n,a[1001];
    cin>>n;
    for(int i=1;i<=n;++i)
        cin>>a[i];
    if(imp(a,1,n))
        cout<<"DA";
    else
        cout<<"NU";
        return 0;
}
// ExistaImpareDivImp #1148

```

```
"C:\informatica c++\informatica\divide2.exe"
5
2 8 6 10 8
NU
Process returned 0 (0x0)   execution time : 8.933 s
Press any key to continue.
```

#1151 VerifEgaleDivImp

Se dă un vector cu n elemente numere naturale. Folosind metoda **Divide et Impera** să se verifice dacă toate elementele vectorului sunt egale.

Intrare

5

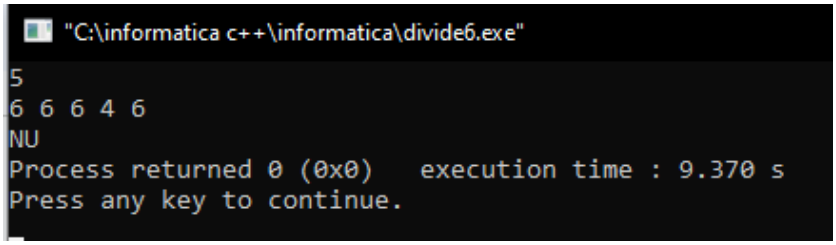
6 6 6 4 6

Ieșire

NU

```cpp
#include<iostream>
using namespace std;

bool egal(int a[],int st,int dr)
{
    if (st==dr)
        return a[st]==a[1];
    else
    {
        int m=(st+dr)/2;
        return egal(a,st,m) && egal(a, m+1,dr);
    }
}
int main()
{
int n,a[501];
cin>>n;
for(int i = 1;i <=n; ++i)
    cin>>a[i];
    if(egal(a, 1 ,n))
        cout<<"DA";
    else
        cout<<"NU";
    return 0;
    }
//#1151 VerifEgaleDivImp
```



#1150 VerifPareDivImp

Se dă un şir cu n elemente, numere naturale. Folosind metoda **Divide et Impera** să se verifice dacă toate elementele şirului sunt pare.

Intrare

5

2 8 6 10 8

Ieşire

DA

```cpp
#include<iostream>
using namespace std;

bool par(int a[],int st, int dr)
{
if (st==dr)
    return a[st]%2==0;
else{
    int m =(st+dr)/2;
      return par(a,st,m)&&(a,m+1 ,dr);


}
}


int main()
{
    int v,a[101];
    cin>>v;
    for(int i=1;i<=v;++i)
        cin>>a[i];
    if(par(a,1,v))
        cout<<"DA";
    else
        cout<<"NU";
    return 0;
}
//#1150 VerifPareDivImp

```

```
"C:\informatica c++\informatica\divide5.exe"
5
2 8 6 10 8
DA
Process returned 0 (0x0)   execution time : 9.488 s
Press any key to continue.
```

#1018 CntImpare

Se consideră un șir cu n elemente, numere naturale. Folosind metoda **Divide et Impera**, determinați câte elemente impare sunt în acest șir.

Intrare

6

4 1 8 4 3 5

Ieșire:3

```cpp
#include<iostream>
using namespace std;

int imp(int a[],int st, int dr){
if (st==dr)
return a[st]%2==1;
else
{
    int m=(st + dr)/2;
    return imp(a , st , m)+ imp(a , m+1 , dr);
}
}
int main()
{
    int v,a[1001];
    cin>>v;
    for(int i=1;i<=v;++i)
        cin>>a[i];

    cout<< imp (a ,1 ,v);
    return 0;
}
//#1018 CntImpare

```

```
"C:\informatica c++\informatica\divide4.exe"
6
4 1 8 4 3 5
3
Process returned 0 (0x0)   execution time : 15.512 s
Press any key to continue.
```