DIVIDE ET IMPERA

1. Prezentare generalã

Divide Et Impera se bazeaza pe urmatorul principiu :

1. descompunem problema in doua sau mai multe subprobleme (mai usoare),care se rezolva

P -> SP1 -> S1, SP2 -> S2 ; S1 + S2 -> S

In general se executa impartirea in doua subprobleme de dimensiuni aproximativ egale

2. solutia pentru problema initiala se obtine combinand solutiile problemelor in care a fost descompusa.
3. Se presupune ca fiecare din probleme in care a fost descompusa problema initiala, se poate descompune in alte subprobleme, la fel cum a fost descompusa problema initiala.
4. Procedeul se reia pana cand in urma descompunerilor repetate se ajunge la probleme care admit rezolvare imediata.

***Divide et impera* este o tehnica ce admite o implementare recursiva.**

La un anumit nivel avem doua posibilitati:

1) am ajuns la o problema care admite o rezolvare imediata, caz in care se rezolva si se revine din apel(conditia de terminare);
2) nu am ajuns in situatia de la punctul 1, caz in care se descompune problema in doua sau mai multe subprobleme, pentru fiecare din ele se reapeleaza functia,urmind apoi combinarea rezultatelor si se revine din apel.

2. Conditii pentru a putea utiliza Metoda DIVIDE ET IMPERA

Problema de rezolvat :

a) se poate descompune in (doua sau mai multe) suprobleme ;

b) aceste suprobleme sunt independente una fata de alta (o subproblema nu se rezolva pe baza alteia si nu se foloseste rezultate celeilalte);

c) aceste subprobleme sunt similare cu problema initiala;

d) la randul lor subproblemele se pot descompune (daca este necesar) in alte subprobleme mai simple;

e) aceste subprobleme simple se pot solutiona imediat prin algoritmul simplificat.

3. Etapele rezolvarii unei probleme (numita problema initiala) :

i. descompunerea problemei initiale in subprobleme independente ,similare problemei de baza ,de dimensiuni mai mici ;

ii. descompunerea treptata a subproblemelor in alte subprobleme din ce in ce mai simple ,pana cand se pot rezolva imediata ,prin algoritmul simplificat ;

iii. rezolvarea subproblemelor simple ;

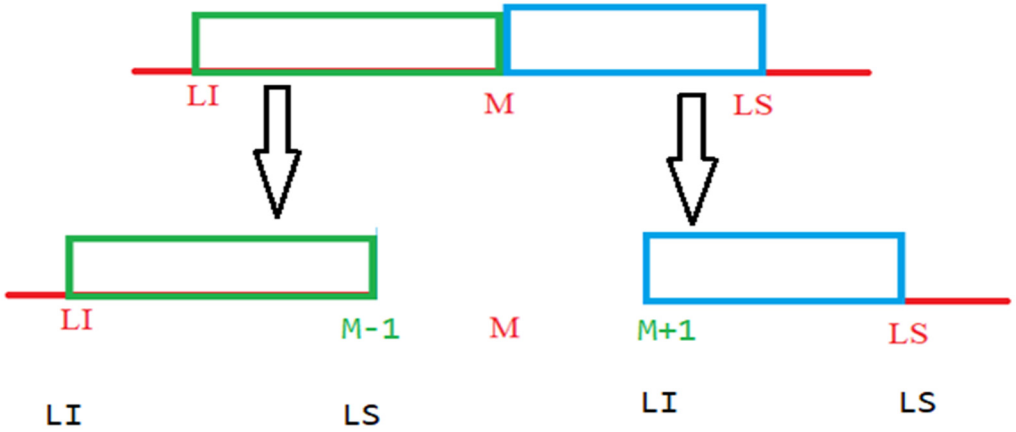
iv. combinarea solutiilor gasite pentru construirea solutiilor subproblemelor de dimensiuni din ce in ce mai mari ;

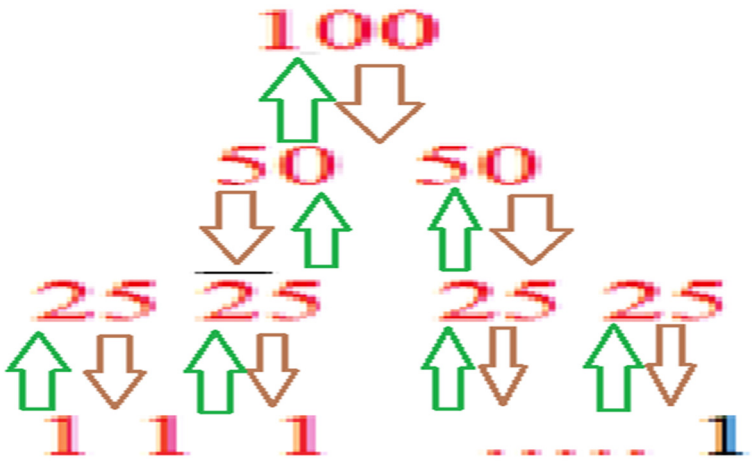
v. combinarea ultimelor solutii determina obtinerea solutiei problemei initiale

```
Subprogram DIVIMP (PROBLEMA);
        Daca PROBLEMA este simpla
                Atunci se rezolva si se obtine solutia SOL
        Altfel pentru i=1,k executa DIVIMP(PROBLEMA) si se obtine SOLi;
        Se combina solutiile SOL 1,... ,SOL k  si se obtine SOL;
Sfarsit _subprogram;
```

```
Procedura  DEI(li,ls,sol);
    daca ((ls-li)<=eps)              atunci REZOLVA (li,ls,sol);
    altfel  {
                DIVIDE (li,m,ls);
                 DEI(li,msol1);
                DEI(m,ls,sol2);
                COMBINA(sol1,sol2,sol);
            }
    Sfarsit_ procedura;
```

4. **Exemple :**





1. Pentru vectorul v cu n elemente numere intregi determinare max cu DEI

```
#include <iostream>
using namespace std;
 int n,v[29];
int maxdei(int li,int ls) // maxim din vector
{
if(li==ls) return v[li];
else {
          int m=(li+ls)/2;
          int a=maxdei(li,m);
          int b=maxdei(m+1,ls);
          if (a>b) return a;
          else return b;
          }
}


int  main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>v[i];
   int m=maxdei(1,n);
   cout<<m;
   return 0;
}
```

2. Pentru vectorul v cu n elemente numere intregi determinare cu DEI, cite elemente nu sint multiplu de 3

```
1  #include <iostream>
2  using namespace std;
3  //cite elem sint diferite de 3
4  int n,v[29];
5  int dei(int li,int ls)
6  {
7   if(li==ls) { if(v[li]%3==0) return 0;
8                else return 1;          }
9   else
10  { int m=(li+ls)/2;
11    int a=dei(li,m);
12    int b=dei(m+1,ls);
13    return a+b;}
14 }
15
16 int  main()
17 {
18   cin>>n;
19   for(int i=1;i<=n;i++) cin>>v[i];
20   int m=dei(1,n);
21   cout<<m;
22   return 0;
23 }
```

1. Minim din vector;
2. Maximul din vector;
3. Determinare simultan min si max din vector;

Cautare x numar real in vectorul v ordonat , cu n elemente numere reale:

4. afisare pozitia ;
5. afisare mesaj – DA (daca exista) sau NU (daca nu exista);

6. Citire elementele vectorului;
7. Suma elementelor dintr-un vector;
8. Produsul elementelor dintr-un vector;
9. Suma elementelor pare dintr-un vector;
10. Daca in vector exista elemente pozitive;
11. Suma elementelor prime dintr-un vector;
12. Produsul elementelor impare dintr-un vector;
13. Cite elemente divizibile cu 3 sint in vectorul v;
14. Suma elementelor din sir divizibile cu x;
15. Cite elemente cu inversul numar prim sint in vectorul v;
16. Cite elemente dintr-un sir au proprietatea de nr perfect (Un **numar** se numeste **perfect** daca este egal cu suma divizorilor sai inclusiv 1, mai putin el insusi);
17. Cite nr. prime sint in vectorul v cu n elemente numere intregi;
18. Produsul ultimilor cifre din elementele vectorului v cu n elemente numere intregi;
19. Pentru un sir ordonat crescator de n numere intregi determinare indicele m din [1,n] astfel incit a[m]=m, daca exista ;
20. Produsul a doi vectori;
21. Valoarea unui polinom intr-un punct;
22. Cmmdc pentru un vector v;
23. Produsul scalar a doi vectori ;
24. S=1*2+2*3+....+n(n+1);
25. s2=1/(1*2)+1/(2*3)+...+1/n*(n+1);