Divide et impera

https://www.pbinfo.ro/?pagina=articole&subpagina=afisare&id=7651

Metoda Divide et Impera este o metoda generala de elaborare a algoritmilor, care se foloseste pentru rezolvarea problemele care pot fi descompuse in subprobleme de acelasi tip, independente unele de altele (care folosesc multimi de date de intrare disjuncte).

Metoda Divide et Impera se bazeaza pe descompunerea unei probleme in subprobleme similare, prin intermediul unu proces recursiv. Procesul recursiv de descompunere a unei subprobleme in alte subprobleme continua pana se obtine o subproblema cu rezolvare imediata (cazul de baza), dupa care se combina solutiile subproblemelor – pana se obtine solutia problemei initiale.

Divide et impera este o tehnică ce admite o implementare recursivă. Așadar, un algoritm prin divide et impera se elaborează astfel:

1. Se testeaza daca s-a ajuns la o problemă care admite o rezolvare imediată (condiția de terminare), caz în care se rezolvă și se revine din apel;
2. Altfel, problema curentă este descompusă în (două sau mai multe) subprobleme similare care se rezolvă în mod recursiv prin acelaşi procedeu
3. se combină soluţiile subproblemelor pentru a obţine soluţia problemei iniţiale.

Maximul dintr-un vector

Se citește un vector cu n componente, numere naturale. Se cere să se afişeze valoarea maximă din vector.

Funcția căutată va determina valoarea maximă dintre v[i] şi v[j], cu i≤j (la apel, i=1şi j=n). Pentru aceasta, se procedează astfel:

- dacă i=j, valoarea maximă va fi v[i]
- în caz contrar, se împarte vectorul în doi subvectori. Se calculează mijlocul m al intervalului [i, j] prin m = (i+j) / 2. Primul subvector va conține componentele cu indice de la i la m, al doilea va conține componentele cu indici de la (m+1) la j; se rezolvă subproblemele (aflându-se astfel maximul pentru fiecare din subvectori), iar soluția curentă va fi dată de valoarea maximă dintre rezultatele celor două subprobleme.

```
#include <iostream>
 using namespace std;
 int v[10],n;

 int max(int i, int j)
 {
   int a, b, m;
   if (i==j) return v[i];
   else
   {
     m = (i+j)/2;
     a = max(i, m);
     b = max(m+1, j);
     if  (a>b) return a;
     else return b;
   }
 }

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

Căutare binară

Se citește un vector cu n componente numere întregi distincte (numerele se presupun ordonate crescător) și un număr întreg x. Să se determine dacă x se găsește sau nu printre numerele citite, iar în caz afirmativ să se afişeze poziţia pe care se află în vector. Altfel, se va afişa mesajul “Nu există”.

Funcția care va fi implementată va determina dacă valoarea căutată se găsește printre numerele aflate pe poziții de indici cuprinşi între i și j (inițial, i=1, j=n). Pentru aceasta, se va proceda astfel:

- dacă nu mai există numere între i şi j (pentru că i>j) înseamnă că x nu se află în vector;
- altfel, dacă x coincide cu valoarea de la mijloc, aflată pe poziția de indice m=(i+j)/2, se afişează indicele și se revine din apel (problema a fost rezolvată);
- în caz contrar, dacă mai sunt și alte elemente de analizat (adică i≤j), problema se descompune astfel:
  - dacă x este mai mic decât v[m] (elementul din mijloc), functia se autoapelează pe secvenţa (i,m-1)
  - dacă x este mai mare decât v[m] (elementul din mijloc), functia se autoapelează pe secvenţa (m+1,j)

În această problemă **dispare partea de combinare a soluțiilor subproblemelor rezolvate**.

```
#include <iostream>
 using namespace std;
 int v[100], n, x;

 void caut(int i, int j)
 {
   int m;
   if (i>j) cout<<"Nu exista";
   else
     { m = (i+j)/2;
       if (x == v[m]) cout<<m;
       else if  (x < v[m]) caut(i, m-1);
            else caut(m+1, j);
     }
   }

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

Cu functie care returneaza pozitia lui x in vector, sau -1 daca nu se gaseste.

```
int caut(int i, int j)
 {
   int m;
   if (i>j) return -1;
   else
     { m = (i+j)/2;
       if (x == v[m]) return m;
       else if  (x < v[m]) return caut(i, m-1);
            else return caut(m+1, j);
     }
   }
```

Se citeste un vector cu n elemente numere naturale. Sa se calculeze CMMDC al elementelor vectorului folosind metoda divide et impera.

```
#include<iostream>
using namespace std;
int a[100],n,i;

int cmmdc(int i, int j)
{ int m,x,y,r;
  if(i==j) return a[i];
  else
  { m=(i+j)/2;
    x=cmmdc(i,m);
    y=cmmdc(m+1,j);

    while(y>0) //combinarea rezultatelor
       {r=x%y;
        x=y;
        y=r;
        }
    return x;
  }
}

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

Calculul puterii X^n in timp logaritmic

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

double x;
int n;

double putere(double x, int n)
{
    double a;
    if (n==0) return 1;
    else if (n==1) return x;
         else { a=putere(x,n/2);
                if (n%2==0) return a*a;
                else return x*a*a;
              }
}

int main()
{
    cin>>x>>n;
    cout<<putere(x,n)<<endl;
    return 0;
}
```

Turnurile din Hanoi

https://www.pbinfo.ro/?pagina=probleme&id=2527

Fie trei tije verticale notate A,B,C. Pe tija A se gasesc asezate n discuri de diametre diferite, in ordinea crescatoare a diametrelor, privind de sus in jos. Initial, tijele B si C sunt goale. Sa se afiseze toate mutarile prin care discurile de pe tija A se muta pe tija B, in aceeasi ordine, folosind ca tija de manevra C si respectand urmatoarele reguli:

- la fiecare pas se muta un singur disc;
- un disc se poate aseza numai peste un disc cu diametrul mai mare.

Rezolvarea acestei probleme se bazeaza pe urmatoarele considerente logice:

- daca n=1 ,atunci mutarea este immediata AB (mut discul de pe A pe B);
- daca n=2,atunci sirul mutarilor este : AC, AB, CB;
- daca n>2 procedam astfel :
  - mut (n-1) discuri A->C;
  - mut un disc A->B ;
  - mut cele (n-1) discuri C->B.

Aceste subprobleme sunt independente, deoarece tijele initial (pe care sunt dispuse discurile), tijele finale si tijele intermediare sunt diferite. Notam H(n,A,B,C)=sirul mutarilor a n discuri de pe A pe B, folosind C.

- n=1 => AB
- n>1 => H(n,A,B,C)= H(n-1,A,C,B), AB, H(n-1,C,B,A)

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

char a,b,c;
int n;

void Hanoi (int n, char a, char b, int c)
{
    if (n==1) cout<<a<<b<<" ";
    else { Hanoi(n-1,a,c,b);
           cout<<a<<b<<" ";
           Hanoi(n-1,c,b,a);
         }
}

int main()
{
    cin>>n;
    a='a'; b='b'; c='c';
    Hanoi(n,a,b,c);
    return 0;
}

Tema: H(3,a,b,c)=….
```