
# Tehnici de programare

# Divide et Impera

ovidiu.banias@aut.upt.ro

# Divide et Impera (Divide and Conquer)

- ***Maxim din vector (didactic)***
- ***Merge sort (sortarea prin interclasare)***
- ***Turnurile din Hanoi***
- ***Quick sort***

# Algoritmul Divide et Impera

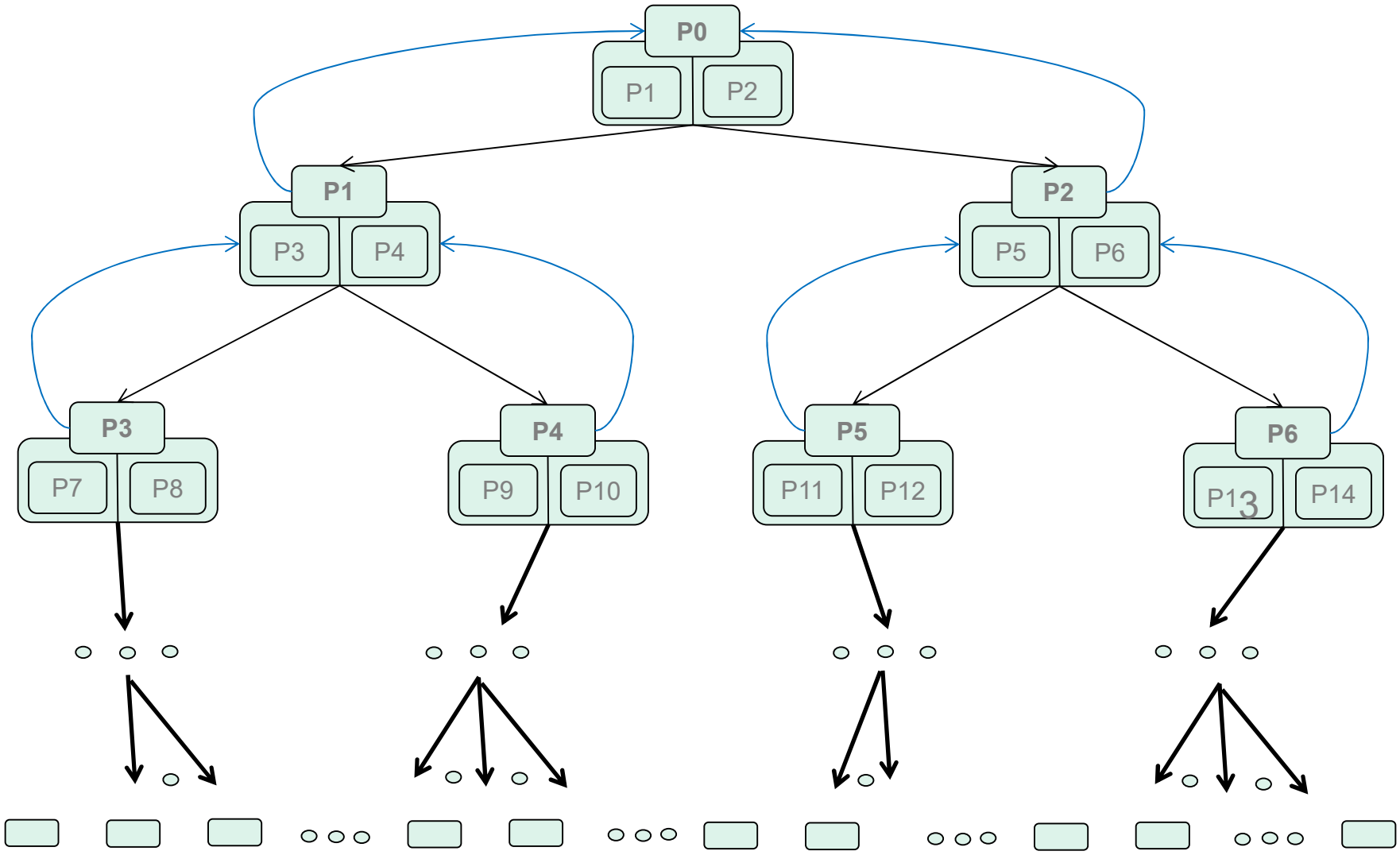
- Împarte problema în subprobleme de acelaşi tip
- Rezolvă fiecare subproblemă independent
- Combină soluțiile subproblemelor pentru a obține soluția problemei inițiale
- DI admite implementare recursivă

– problema permite

descompunere succesivă

în subprobleme

# DI. Schemă de principiu



# Ex1. Maxim în vector neordonat (pur didactic)

**Problemă**: Se dă un vector neordonat cu ***n*** elemente. Să se găsească maximul folosind metoda D et I.

**Rezolvare**:

- Se porneşte de la următoarea observație:

$$Max(v[0],...,v[n-1]) = \max(Max(v[0],...,v[k]), Max(v[k+1],...,v[n-1]))$$

$$\max(a,b) = \begin{cases} a, a > b \\ b, b >= a \end{cases}$$

- Problema admite descompunerea în subprobleme de **acelaşi** tip
- Problema/subproblemele se descompun succesiv până nu mai este necesară descompunerea şi rezolvarea este simplă – căutarea maximului între 1 sau 2 elemente

# Implementare

```
int DI(int li, int ls){

    int x, y;

    if (li<ls){
        x = DI(li,(li+ls)/2);
        y = DI((li+ls)/2+1,ls);
        if (x<y) return y;
        else return x;
    }else
        return a[li];
}

int main(){
{
  ...; printf("%d ",DI(0,n-1));...

}
```

# Ex2. Interclasarea a doi vectori ordonați

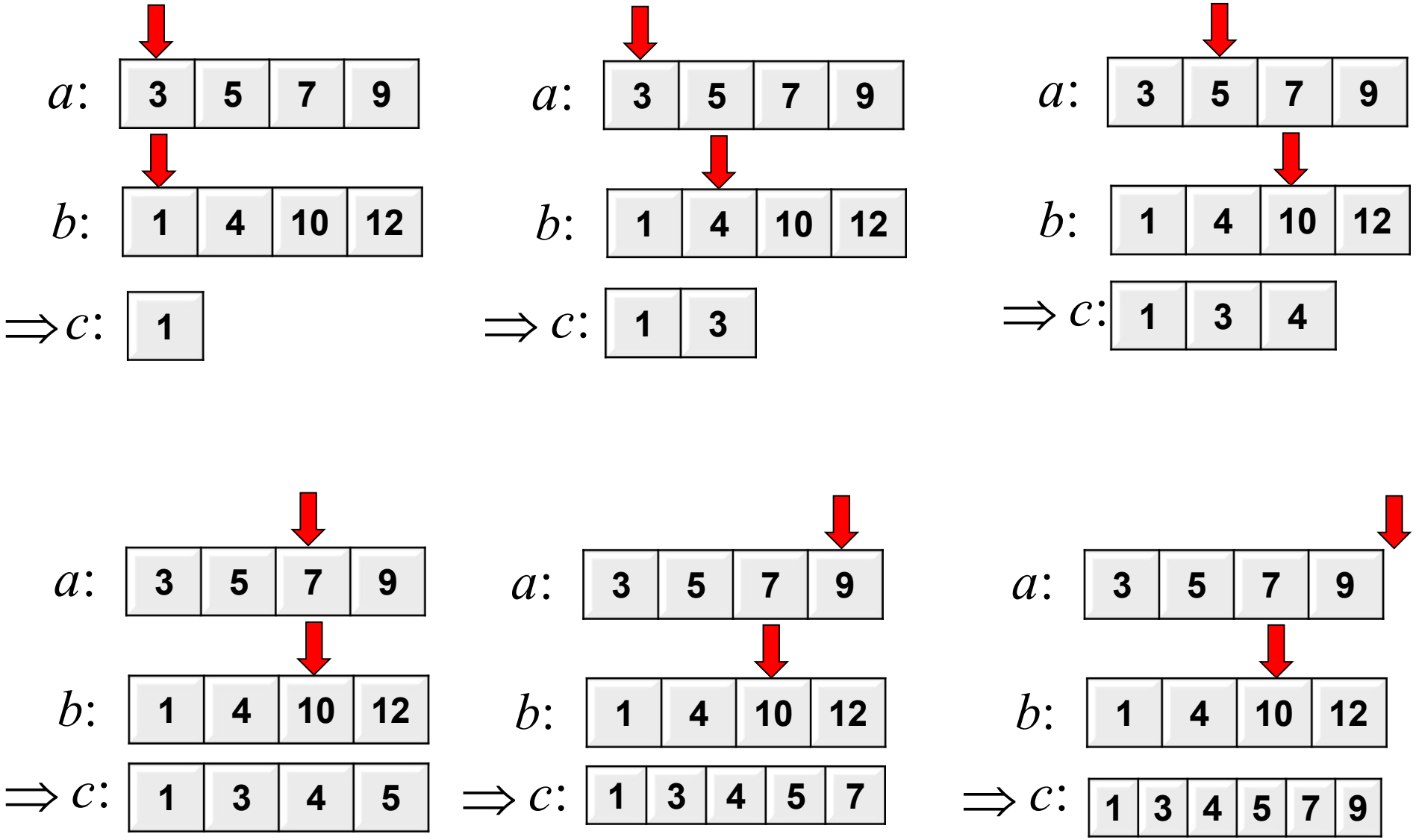
**Problemă (preambul MergeSort**): Se dau doi vectori **a, b**, ordonați crescător având **n**, respectiv **m** elemente. Se doreşte obținerea unui vector **c**, ordonat, de dimensiune **n+m**, care să conțină toate elementele din a şi din b.

# Ex2. Interclasarea a doi vectori ordonați


**Rezolvare**:

- Se selectează succesiv elemente din a sau din b şi se introduc în c
- Dacă elementul curent din a este mai mic decât elementul curent din b, atunci se selectează elementul din a, în caz contrar, cel din b
- Procesul continuă până sunt epuizate(selectate) toate elementele fie din a, fie din b
- Ultimul pas constă în copierea tuturor elementelor neselectate în vectorul c
- Complexitate?

# Ex2. Interclasarea a doi vectori ordonați



# Ex3. Sortarea prin interclasare (Merge sort)

**Problemă (1945, von Neumann**): Se dă un vector neordonat **v** cu **n** elemente. Să se ordoneze crescător vectorul **v** prin metoda Merge sort.

**Rezolvare**:

- Se divide vectorul de dimensiune n în doi vectori de dimensiune n/2
- Procesul de divizare(împărțire) a subvectorilor continuă până când se ajunge la vectori de dimensiune 1
- se interclasează elementele a doi subvectori aparținând aceleiaşi divizări şi se obține un vector sortat
- Procesul de interclasare continuă (bottom-up) până este sortat vectorul inițial
- Complexitate?

# Implementare

```
void DI(int li, int ls){

    if (li<ls){
        DI(li,(li+ls)/2);
        DI((li+ls)/2+1,ls);
        Merge(li,ls); //interclasare
    }
}


int main(){
{
  ...; DI(0,n-1) ;...
 }
```

# Implementare

```
void Merge(int li, int ls){ // merge = interclasare

  int i,j,k,m[100], c=0;

  i=li;k=(li+ls)/2;j=k+1;

  while (i<=k && j<=ls){ // selecteaza a[i] sau a[j]
     if (a[i]<a[j])
        m[c++]=a[i++];
     else
        m[c++]=a[j++];
  }

  while(i<=k) m[c++]=a[i++];
                // daca au ramas elemente in stanga

  while (j<=ls) m[c++]=a[j++];
                // daca au ramas elemente in dreapta

  for (i=li;i<=ls;i++)
     a[i]=m[i-li];
}
```

# Ex4. Turnurile din Hanoi

**Problemă**: Se dau trei tije, **a**, **b** şi **c**. Pe tija **a** se află **n** discuri de dimensiuni diferite, ordonate în ordinea diametrelor (discul cel mai mare la bază). Se doreşte mutarea tuturor discurilor de pe tija **a** pe tija **b**, utilizând tija intermediară **c**, cu condiția ca un disc cu diametrul mai mare să nu fie pus pe vreo tijă peste un disc cu dimensiune mai mică.

**Rezolvare**:

- Se caută o modalitate de divizare a problemei inițiale în subprobleme identice

$$H(n,a,b,c)=\begin{cases} a \to b, n=1 \\ H(n-1,a,c,b), a \to b, H(n-1,c,b,a), n>1 \end{cases}$$

- Complexitate?

# Probleme propuse

1. QuickSort

2. Să se găsească elementul majoritar dintr-un vector – numărul de apariţii să fie mai mare decât jumătate din numărul de elemente