Metoda Divide et Impera

desparte şi stăpâneşte

Algoritmi clasici

Recapitulare

Căutarea binară

Căutarea binară

- **Se consideră vectorul $a=(a_1,\ldots,a_n)$ ordonat crescător şi o valoare x. Se cere să se determine dacă x apare printre componentele vectorului.**

- Mai exact căutăm perechea (b,i) dată de:
  - `(true,i)` dacă $a_i = x$;
  - `(false,i)` dacă $\mathbf{a_{i-1} < x < a_i}$,

unde, prin convenţie,

$$a_0 = -\infty,\ a_{n+1} = +\infty.$$

Căutarea binară

```
void cautBin(int a[], int n, int x){
    int p = 1, u = n;
    while (p<=u){
        int m = (p+u)/2; //compar cu elem. din mijl. secv.
        if (a[m] > x )
            u = m-1; //caut in subsecv stanga
        else
            if (a[m] == x) {
                System.out.println("true "+m); return;
            }
            else
                p = m+1; //caut in subsecv dreapta
    }
    System.out.println("false "+p);
}
```

Căutarea binară

- **Complexitate:** $O(\log n)$

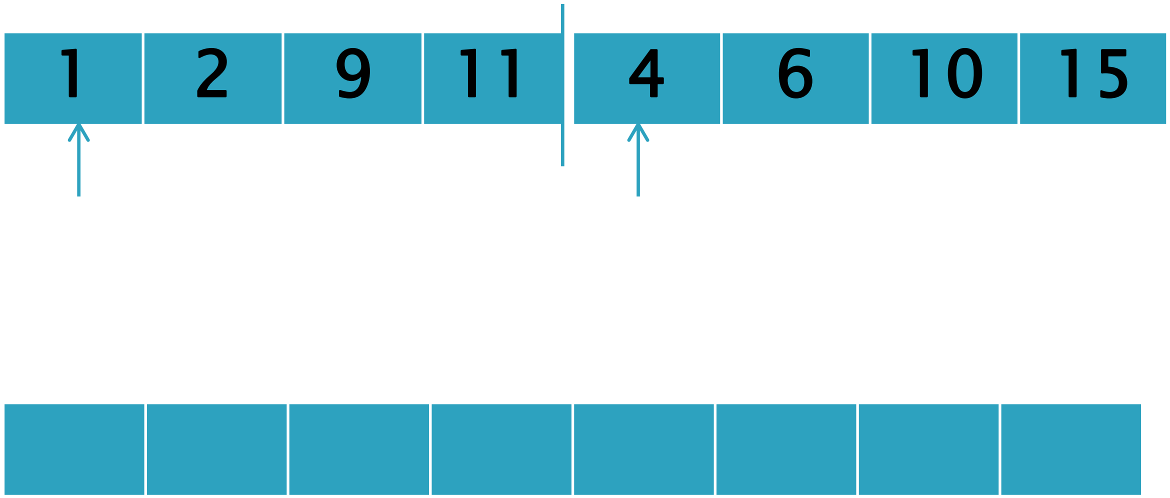
  - $T(n) = T(n/2)+c$
    $= \ldots = T(n/2^k)+kc$

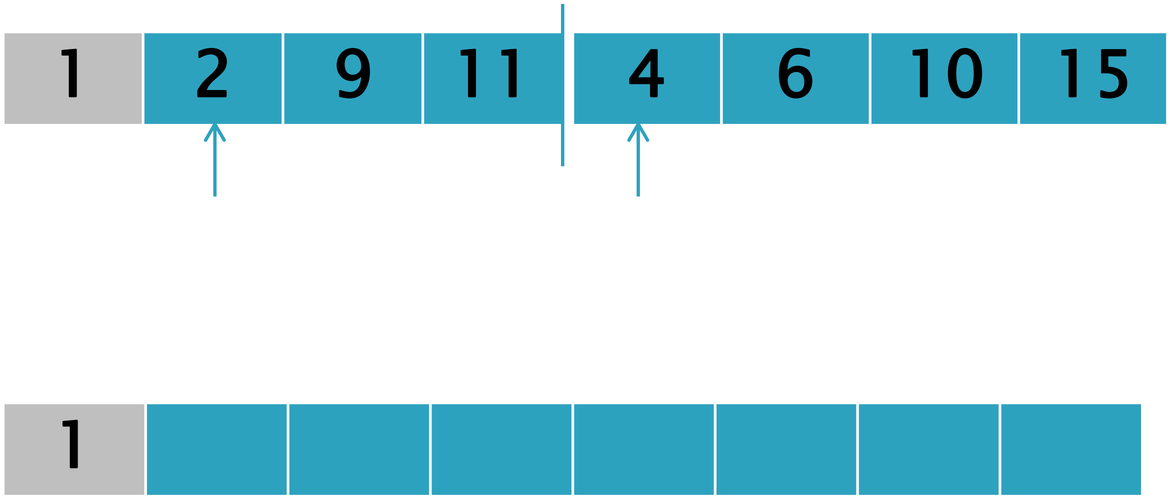
  - $k = \log_2 n$

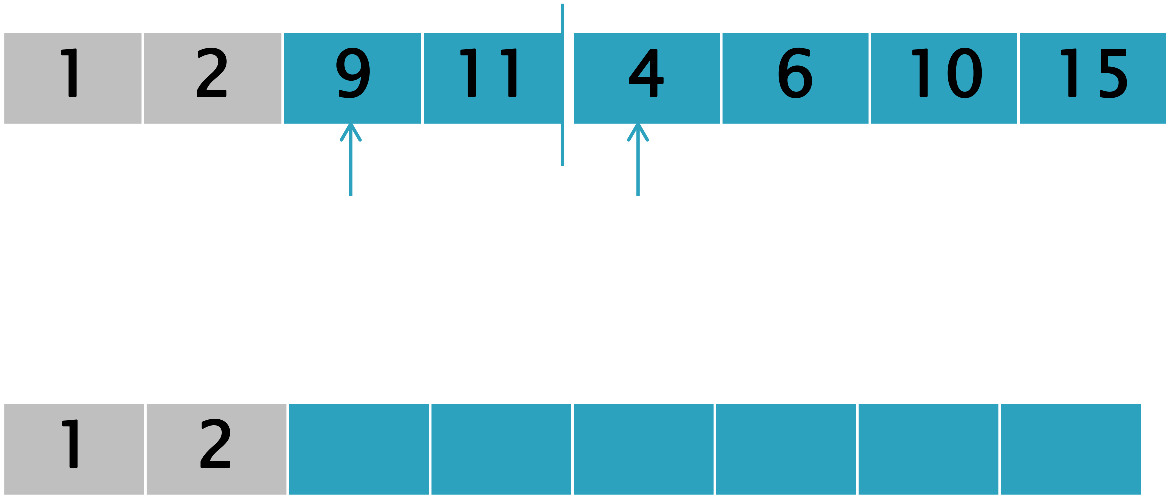
Sortarea prin interclasare

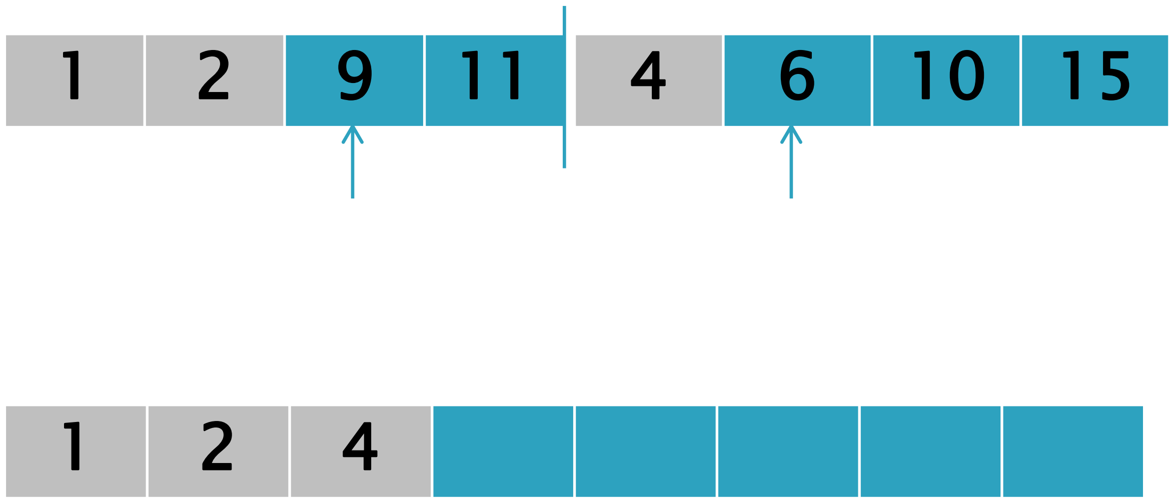
Sortare prin interclasare

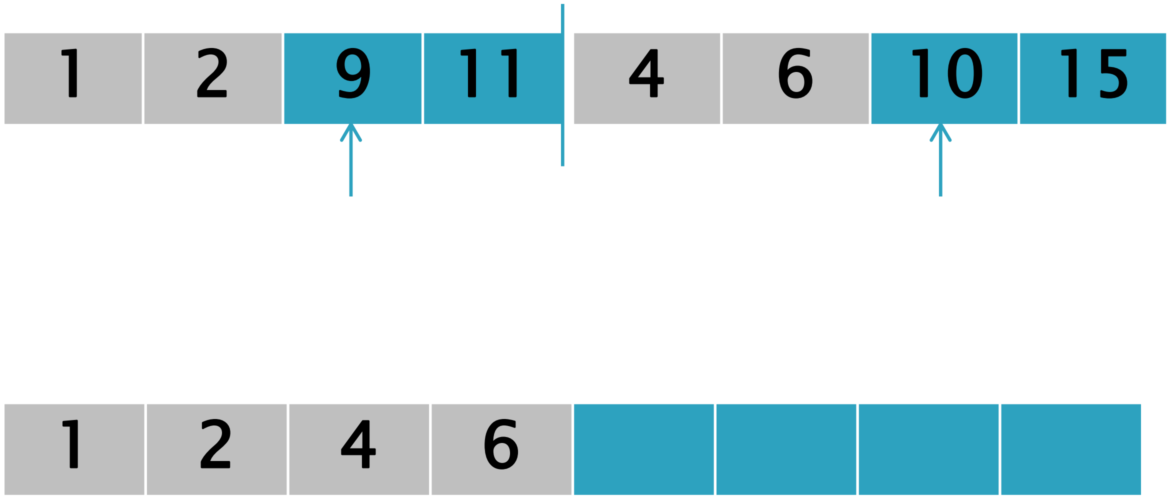
- **Idee:**
  - împărţim vectorul în doi subvectori
  - ordonăm crescător fiecare subvector
  - asamblăm rezultatele prin ***interclasare***

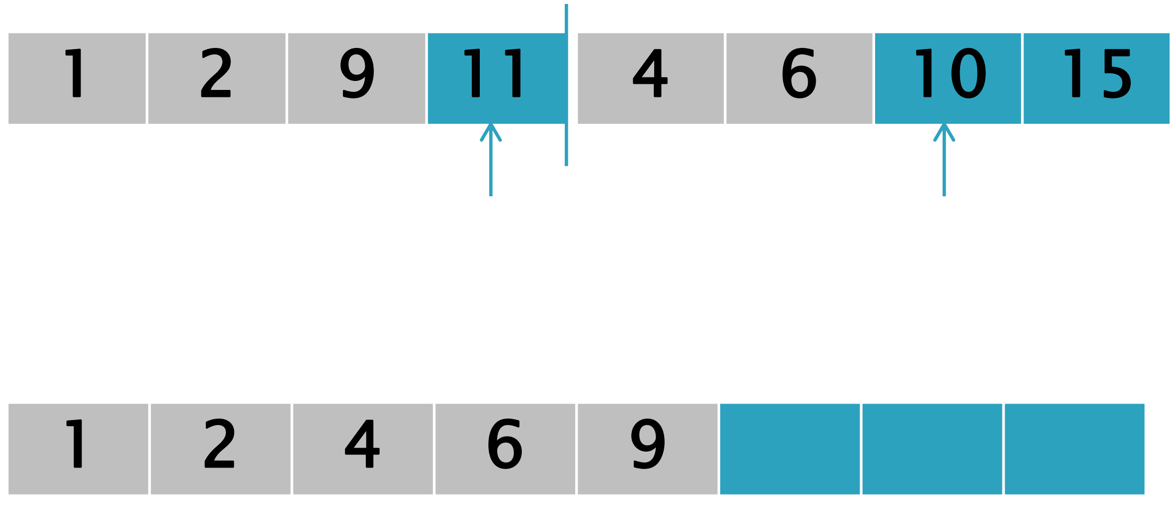

1
2
9
11
4
6
10
15

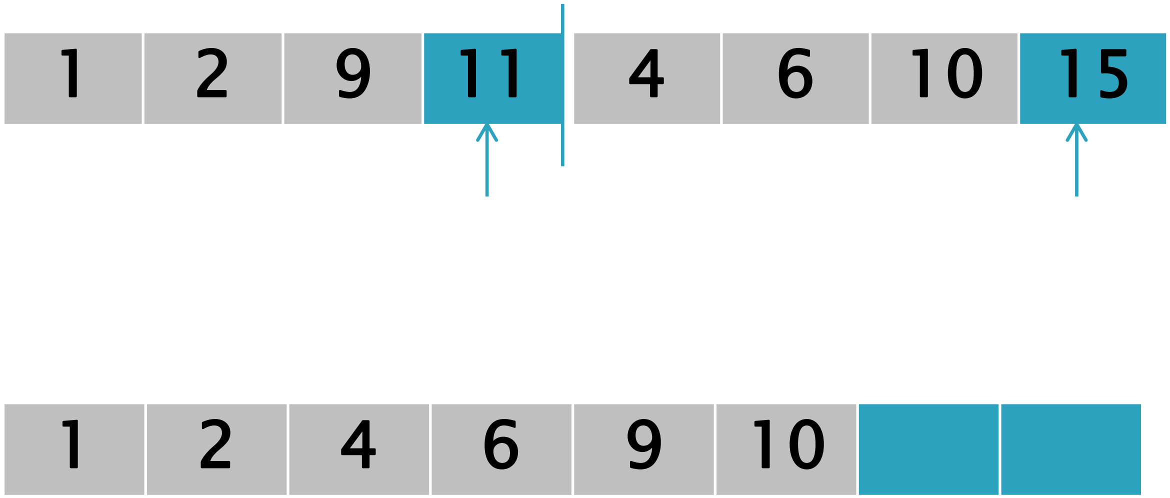

1
2
9
11
4
6
10
15
1


1 2 9 11 4 6 10 15
1 2


1 2 9 11 4 6 10 15
1 2 4


1 2 9 11 4 6 10 15
1 2 4 6


1 2 9 11 4 6 10 15
1 2 4 6 9


1 2 9 11 4 6 10 15
1 2 4 6 9 10

1	2	9	11	4	6	10	15

1	2	4	6	9	10	11	

1	2	9	11	4	6	10	15

1	2	4	6	9	10	11	15

Sortare prin interclasare

```
void sortInter(int p,int u){
    if (p==u) {}
    else {
         int m = (p+u)/2;
         sortInter(p,m);
         sortInter(m+1,u);
         interclaseaza(p,m,u);
    }
}
```

```
void interclaseaza(int p,int m,int u){
      int b[]=new int[u-p+1], k1=p, k2=m+1, kb=0;
      while ((k1<=m)&&(k2<=u)){
            if (a[k1]<=a[k2]){ b[kb]=a[k1]; k1++; }
            else{ b[kb]=a[k2]; k2++;}
            kb++;
      }
      while(k1<=m){ b[kb]=a[k1]; kb++; k1++; }
      while(k2<=u){ b[kb]=a[k2]; kb++; k2++; }
      for (int i=p;i<=u;i++)
           a[i]=b[i-p];
}
```

Sortare prin interclasare

- **Complexitate:** O(n log n)

Sortare prin interclasare

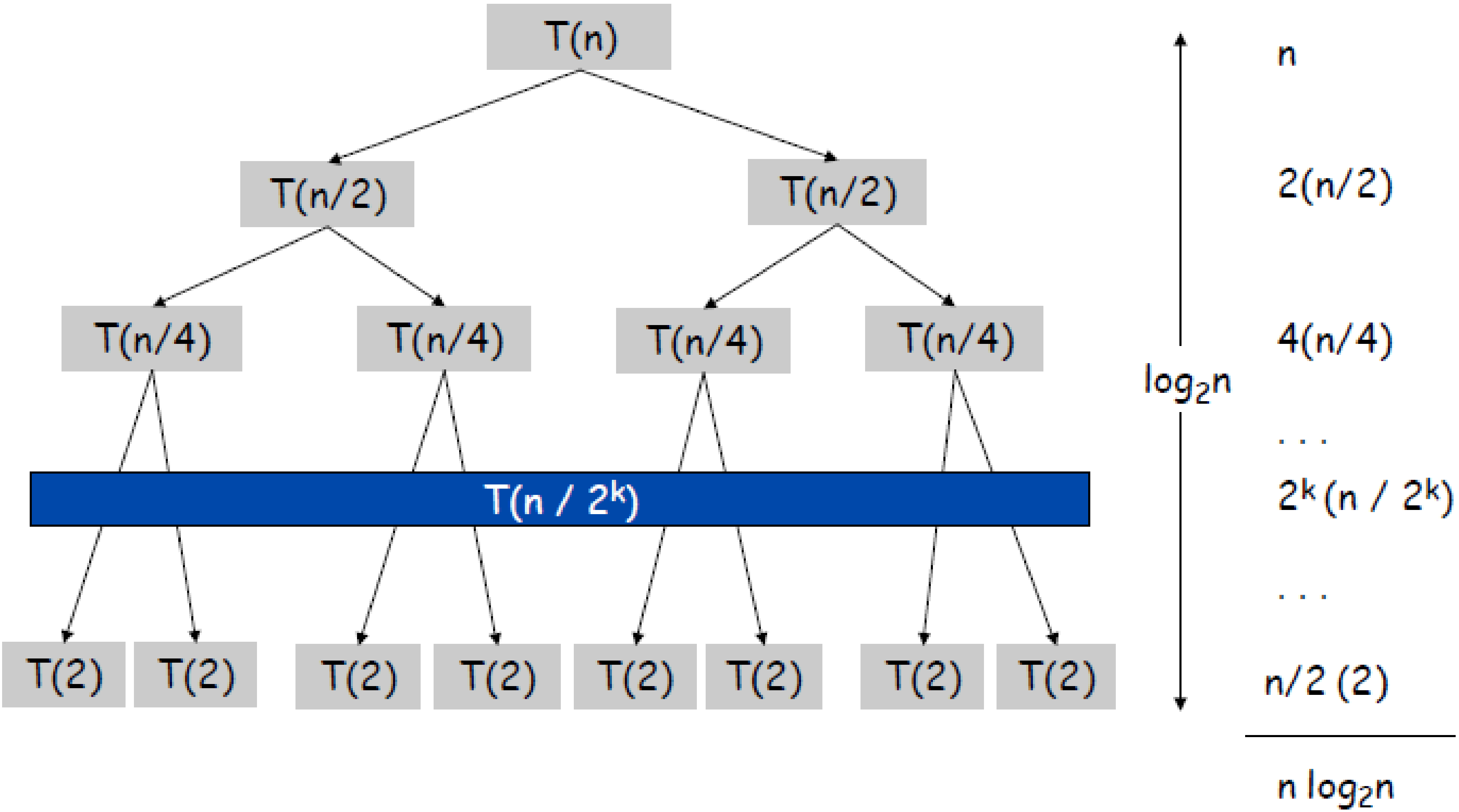
- **Complexitate:** $O(n \log n)$

- $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn$, pentru $n>1$

- Pentru $n=2^k$

  $T(n) = 2T(n/2)+cn$, pentru $n>1$

Sortare prin interclasare



http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

Sortare prin interclasare

$$T(n) = T(2^k) = 2\ T(2^{k-1}) + c \cdot 2^k =$$

$$= 2\ [2T(2^{k-2}) + c \cdot 2^{k-1}] + c \cdot 2^k = 2^2 T(2^{k-2}) + 2 \cdot c \cdot 2^k$$

$$= 2^2 [T(2^{k-3}) + c \cdot 2^{k-2}] + 2 \cdot c \cdot 2^k = 2^3 T(2^{k-3}) + 3 \cdot c \cdot 2^k$$

$$= \ldots = 2^i T(2^{k-i}) + i \cdot c \cdot 2^k =$$

$$= 2^k T(1) + k \cdot c \cdot 2^k = n t_0 + c \cdot n \cdot \log_2 n.$$

Quicksort
Sortarea rapidă

Quicksort

- **Idee:**
  - poziționăm primul element al secvenței (**pivotul**) pe poziția sa finală = astfel încât elementele din stânga sa sunt mai mici, iar cele din dreapta mai mari
  - ordonăm crescător elementele din stânga
  - ordonăm crescător elementele din dreapta

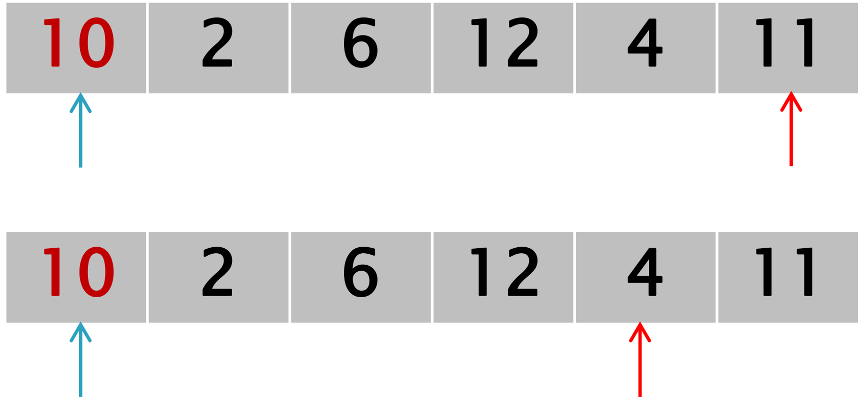
Quicksort

```
void quicksort(int p,int u)
    if (p==u) {}
    else {
        int m = poz(p,u);
        quicksort(p,m-1);
        quicksort(m+1,u);
    }
}
```

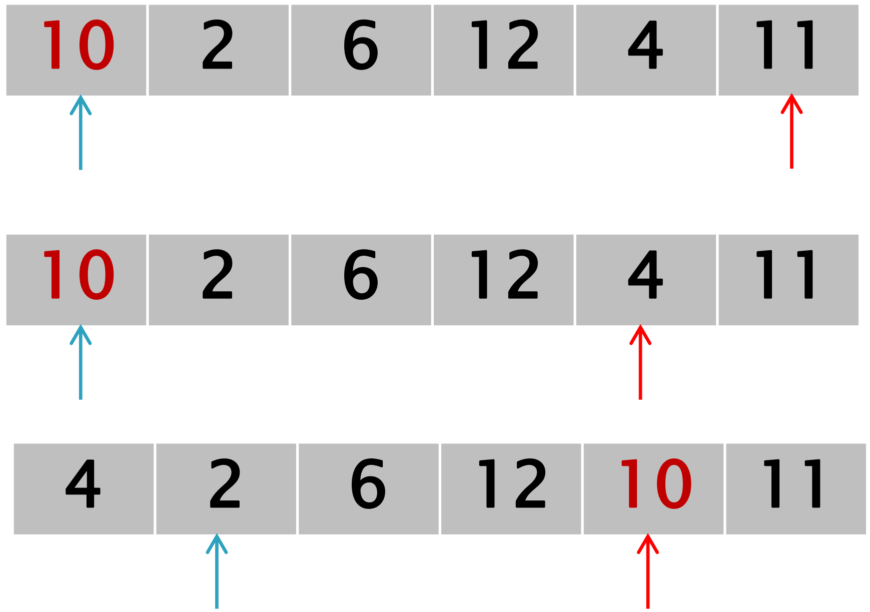
```
int poz(int p,int u){
   int i=p,j=u, depli=0,deplj=-1;
   while(i<j){
     if(a[i]>a[j]){
        int aux=a[i];a[i]=a[j];a[j]=aux;
        aux=depli;
        depli=-deplj;
        deplj=-aux;
     }
     i+=depli;j+=deplj;
   }
   return i;
 }
```

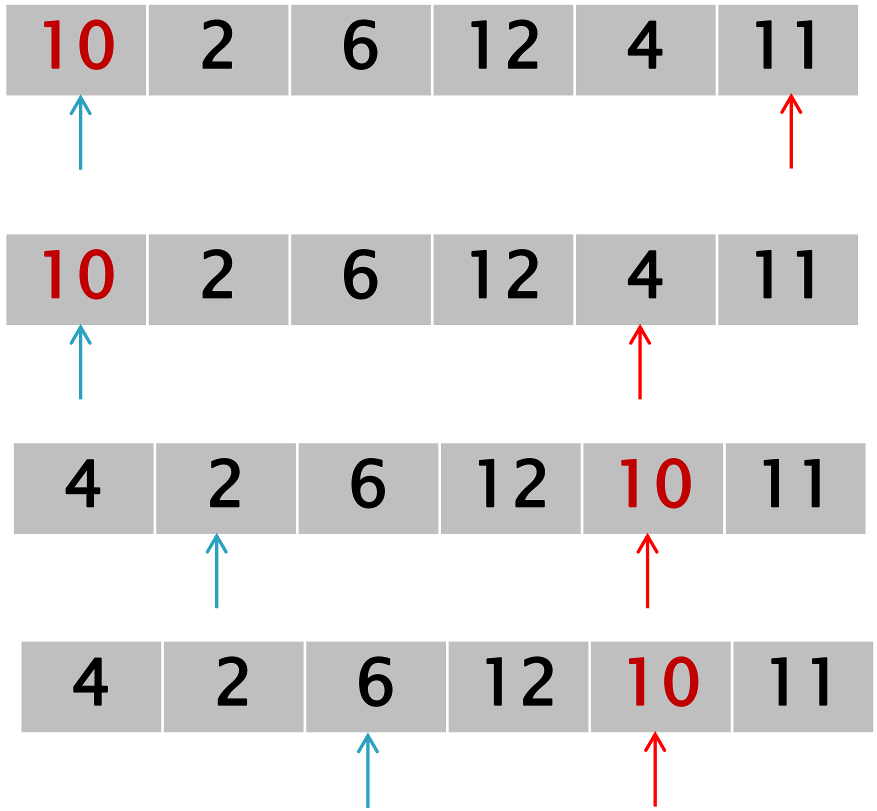
Exemplu – poziţionare pivot

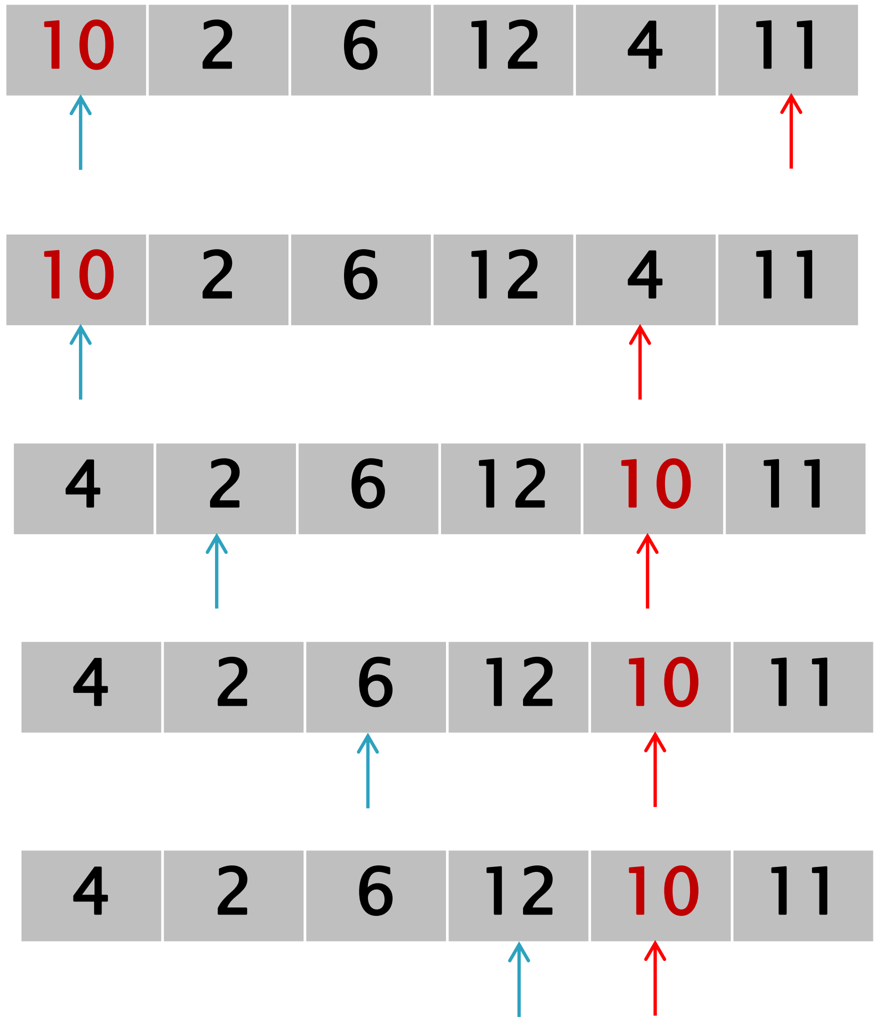
10	2	6	12	4	11

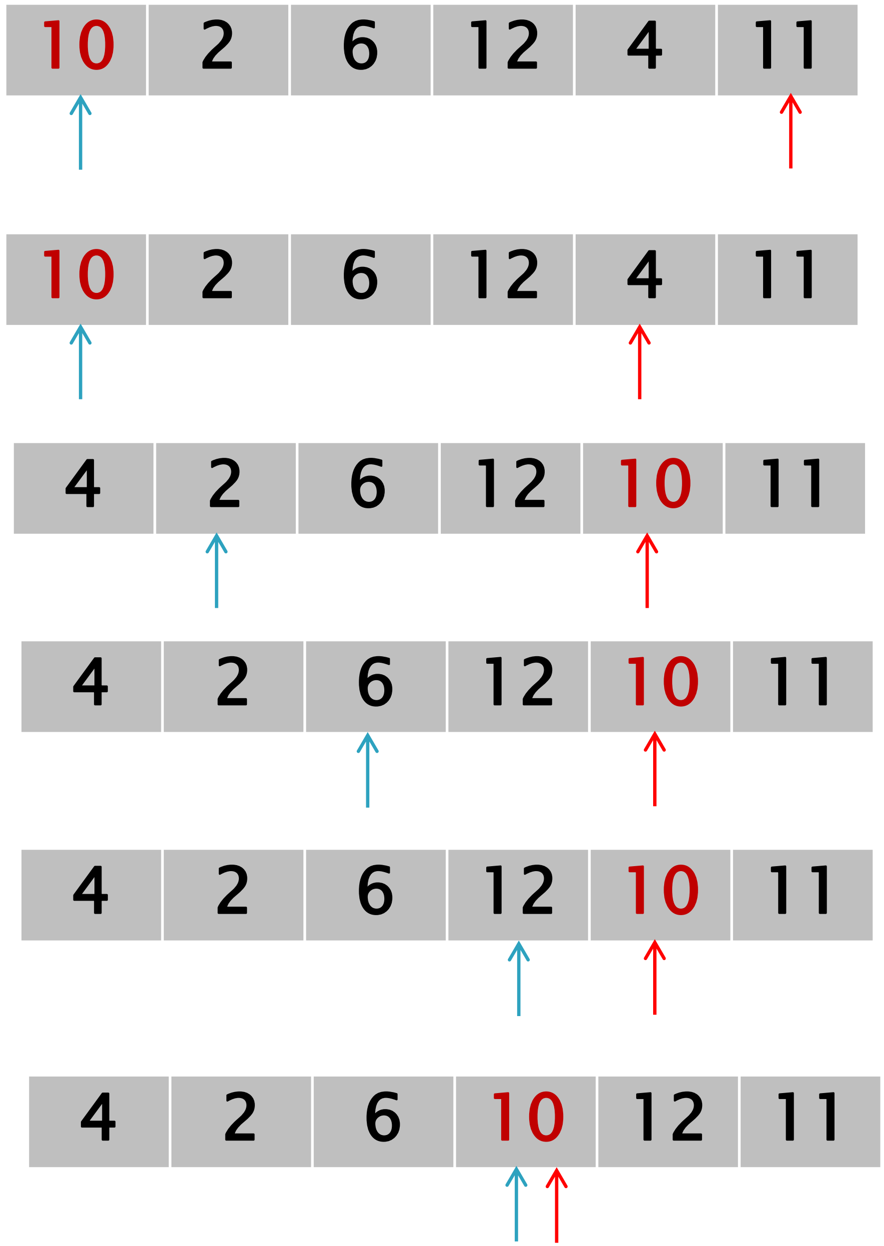


10 2 6 12 4 11
10 2 6 12 4 11


10 2 6 12 4 11
10 2 6 12 4 11
4 2 6 12 10 11


10 2 6 12 4 11
10 2 6 12 4 11
4 2 6 12 10 11
4 2 6 12 10 11


10 2 6 12 4 11
10 2 6 12 4 11
4 2 6 12 10 11
4 2 6 12 10 11
4 2 6 12 10 11


10 2 6 12 4 11
10 2 6 12 4 11
4 2 6 12 10 11
4 2 6 12 10 11
4 2 6 12 10 11
4 2 6 10 12 11

Quicksort

- **Complexitate:**
  - **Defavorabil**: $O(n^2)$
  - **Mediu**: $O(n \log n)$ (demonstraţie în pdf)

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

$$nT(n) = n(n-1)+2[T(0)+T(1)+...+T(n-1)]$$

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

$$nT(n) = n(n-1)+2[T(0)+T(1)+...+T(n-1)]$$

$$(n-1)T(n-1) = (n-1)(n-2)+2[T(0)+...+T(n-2)]$$

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

$$nT(n) = n(n-1)+2[T(0)+T(1)+...+T(n-1)]$$

$$(n-1)T(n-1) = (n-1)(n-2)+2[T(0)+...+T(n-2)]$$

Scădem cele două relații:

$$nT(n)-(n-1)T(n-1) = 2(n-1)+ 2T(n-1)$$

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

$$nT(n) = n(n-1)+2[T(0)+T(1)+...+T(n-1)]$$

$$(n-1)T(n-1) = (n-1)(n-2)+2[T(0)+...+T(n-2)]$$

$$nT(n)-(n-1)T(n-1) = 2(n-1)+ 2T(n-1)$$

$$nT(n) = (n+1)T(n-1)+2(n-1)$$

Quicksort

$$\begin{cases} T(n) = n - 1 + \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}[T(k-1) + T(n-k)] \\ T(1) = T(0) = 0 \end{cases}$$

$$nT(n) = n(n-1)+2[T(0)+T(1)+...+T(n-1)]$$

$$(n-1)T(n-1) = (n-1)(n-2)+2[T(0)+...+T(n-2)]$$

$$nT(n)-(n-1)T(n-1) = 2(n-1)+ 2T(n-1)$$

$$nT(n) = (n+1)T(n-1)+2(n-1) \quad \mathbf{| \; :n(n+1)}$$

Quicksort

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

Quicksort

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + 2\left(\frac{2}{n+1} - \frac{1}{n}\right)$$

Quicksort

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + 2\left(\frac{2}{n+1} - \frac{1}{n}\right)$$

$$\frac{T(n-1)}{n} = \frac{T(n-2)}{n-1} + 2\left(\frac{2}{n} - \frac{1}{n-1}\right)$$

. . . .

$$\frac{T(2)}{3} = \frac{T(1)}{2} + 2\left(\frac{2}{3} - \frac{1}{2}\right)$$

Quicksort

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + 2\left(\frac{2}{n+1} - \frac{1}{n}\right)$$

$$\frac{T(n-1)}{n} = \frac{T(n-2)}{n-1} + 2\left(\frac{2}{n} - \frac{1}{n-1}\right)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\frac{T(2)}{3} = \frac{T(1)}{2} + 2\left(\frac{2}{3} - \frac{1}{2}\right)$$

$$+$$

$$\frac{T(n)}{n+1} = 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right) + \frac{2}{n+1} - 1$$

Quicksort

$$\frac{T(n)}{n+1} = 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right) + \frac{2}{n+1} - 1$$

$$\leq 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right)$$

sume Riemann inferioare pentru funcţia f(x)=1/x

$\Delta = \{2,3,\ldots, n+1\}$

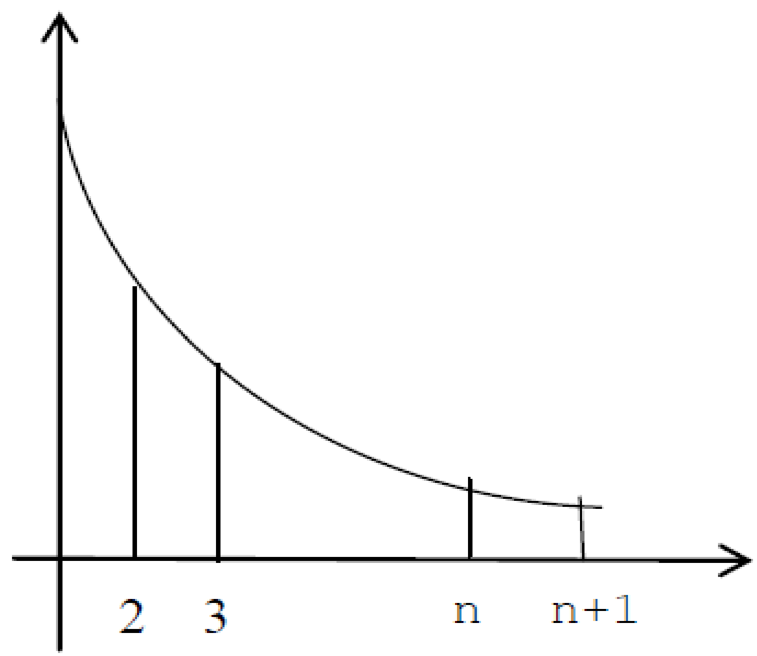
Quicksort

$$\frac{T(n)}{n+1} = 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right) + \frac{2}{n+1} - 1$$

$$\leq 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right)$$

sume Riemann inferioare pentru funcţia f(x)=1/x

$\Delta = \{2, 3, \ldots, n+1\}$



Quicksort

$$\frac{T(n)}{n+1} = 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right) + \frac{2}{n+1} - 1$$

$$\leq 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots\ldots + \frac{1}{3}\right)$$

sume Riemann inferioare pentru funcţia $f(x)=1/x$

$\Delta = \{2,3,\ldots, n+1\}$

$$\frac{T(n)}{n+1} \leq 2\int_2^{n+1} \frac{1}{x}dx = 2\ln x\,\Big|_2^{n+1} \leq 2\cdot\ln(n+1)$$

Quicksort

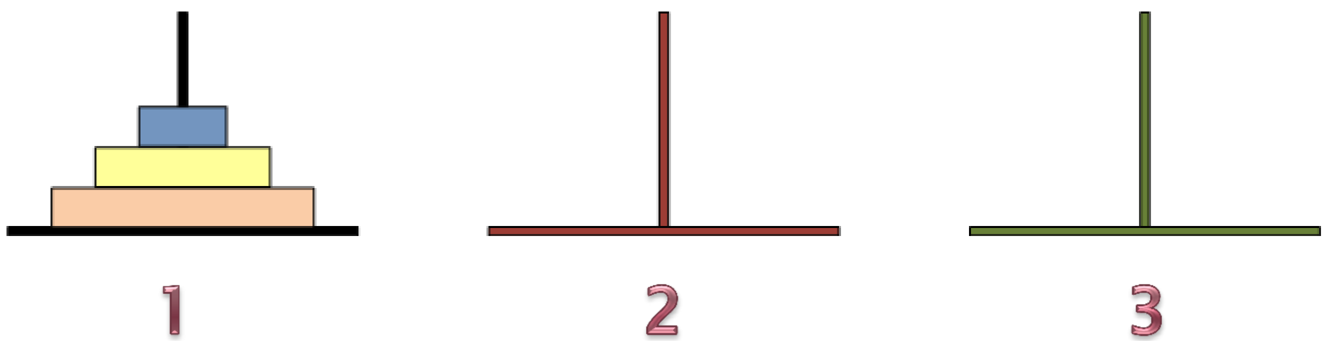
- **Variantă** – pivot aleator

```
int pozRand(int p,int u){
        r ← random(p,u)
        a[r] ↔ a[p]
        return poz(p,u)
}
```

Problema turnurilor din Hanoi

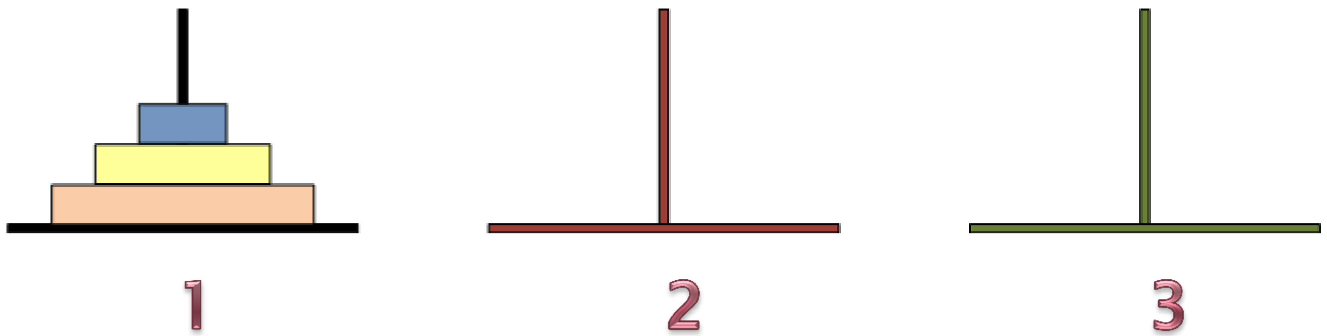
Problema turnurilor din Hanoi

- Se consideră 3 tije. Iniţial pe tija 1 se află n discuri cu diametrele descrescătoare privind de la bază către vârf, iar pe tijele 2 şi 3 nu se află nici un disc.



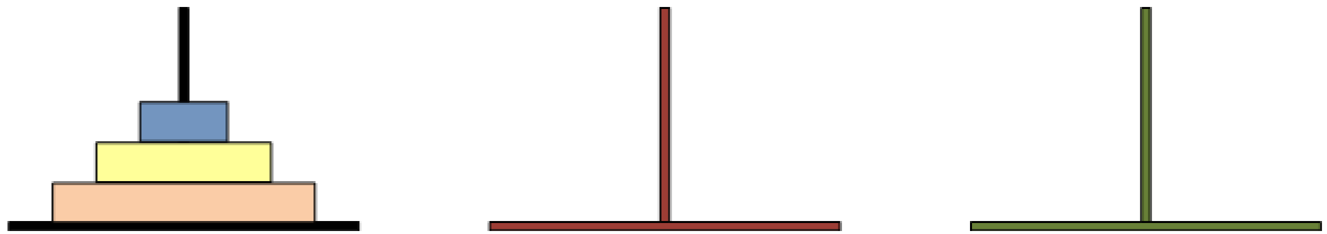
Problema turnurilor din Hanoi

- Se cere să se mute aceste discuri pe tija 2, ajutându-ne şi de tija 3.
- O **mutare** (i,j) constă în deplasarea discului din vârful tijei i în vârful tijei j. Mutarea este **corectă** dacă stiva j este goală sau are în vârf un disc de diametru mai mare decât discul deplasat



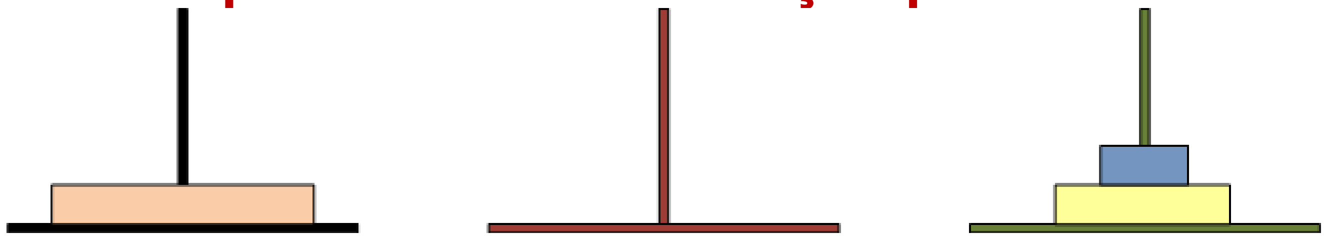
Problema turnurilor din Hanoi

- Pentru n = 3



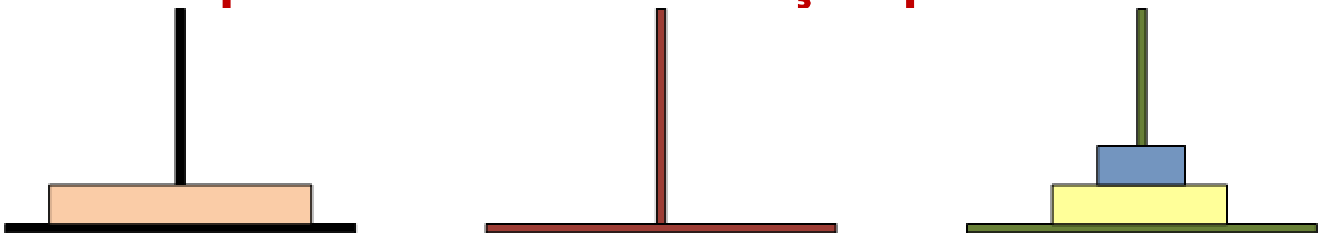
Problema turnurilor din Hanoi

- Mutăm două discuri de pe stiva 1 pe stiva 3, folosind stiva 2 – **subproblemă de acelaşi tip**

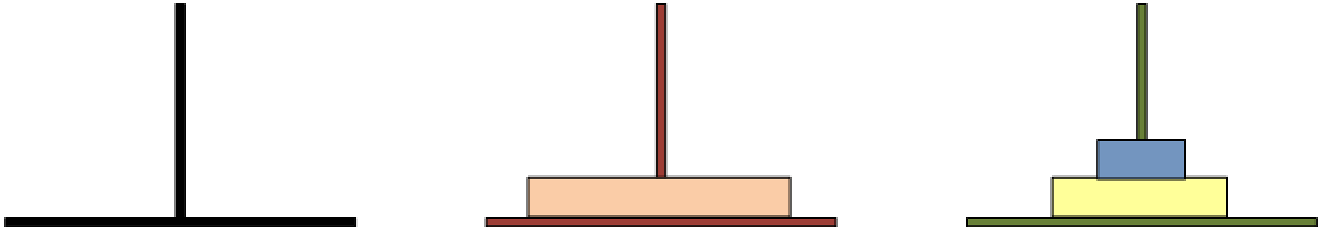


Problema turnurilor din Hanoi

- Mutăm două discuri de pe stiva 1 pe stiva 3, folosind stiva 2 – **subproblemă de acelaşi tip**

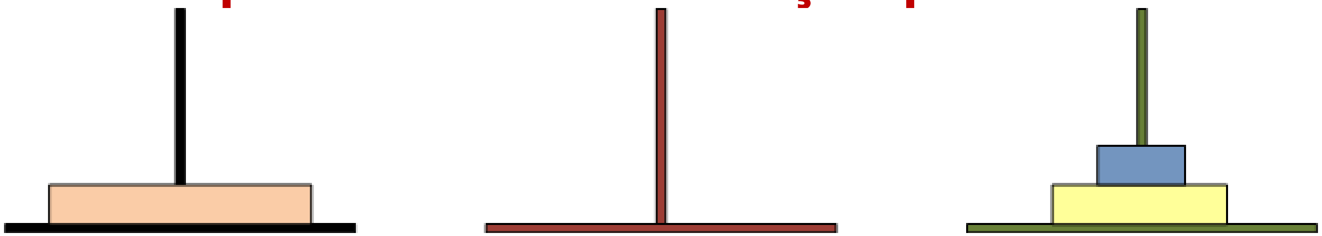


- Mutăm unicul disc rămas pe stiva 1 (cu diametrul maxim) pe stiva 2

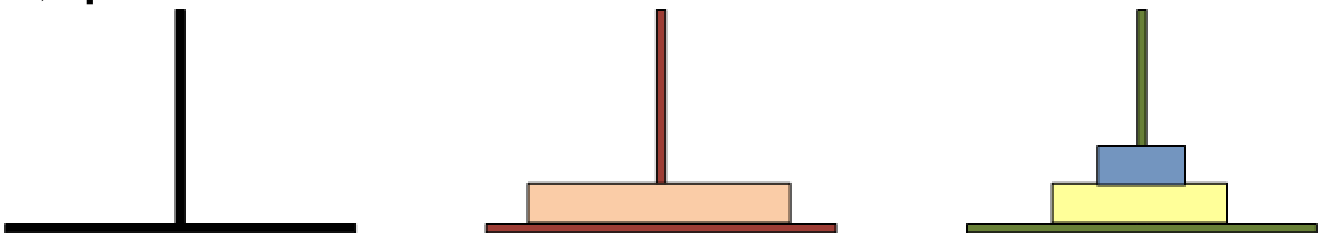


Problema turnurilor din Hanoi

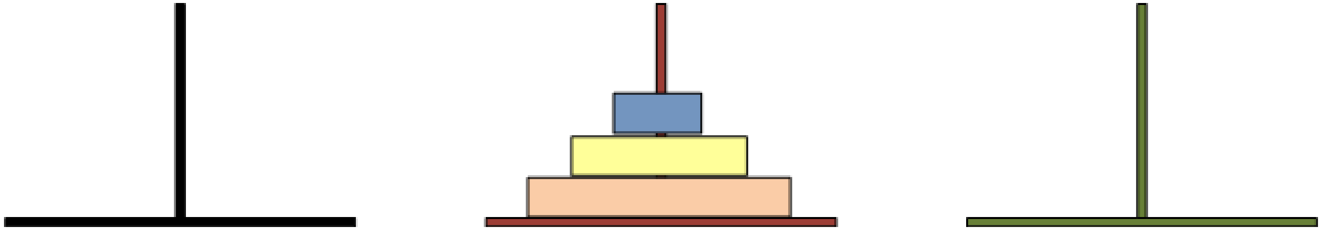
- Mutăm două discuri de pe stiva 1 pe stiva 3, folosind stiva 2 – **subproblemă de acelaşi tip**



- Mutăm unicul disc rămas pe stiva 1 (cu diametrul maxim) pe stiva 2



- Mutăm cele două discuri de pe stiva auxiliară 3 pe stiva 2 folosind stiva 1 – **subproblemă de acelaşi tip**

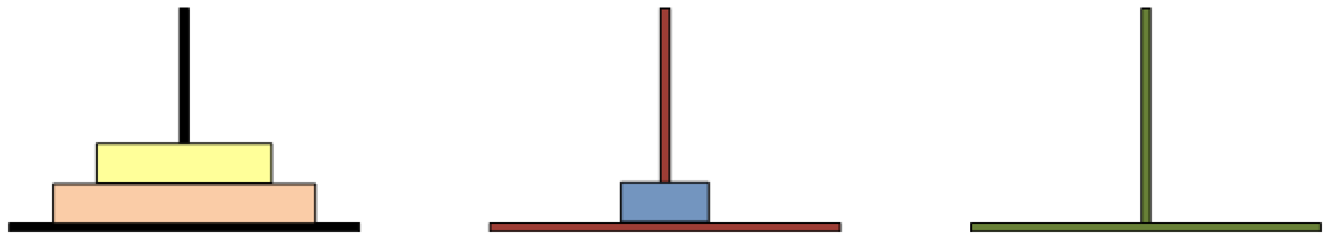


Problema turnurilor din Hanoi

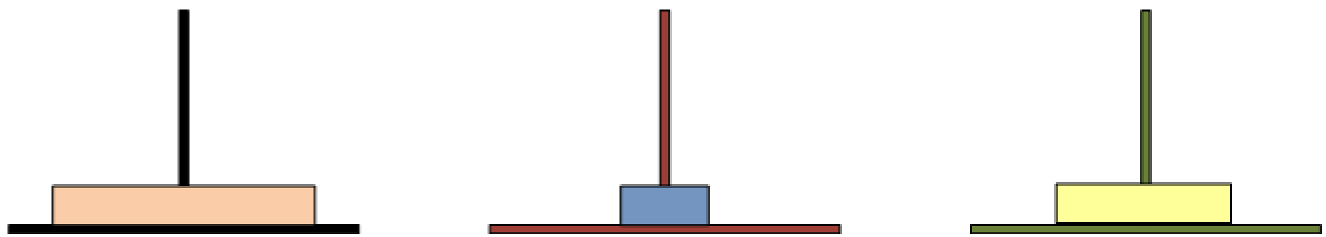
- Rezolvăm, recursiv, **subproblemele de acelaşi tip**

Mutăm două discuri de pe stiva 1 pe stiva 3, folosind stiva 2

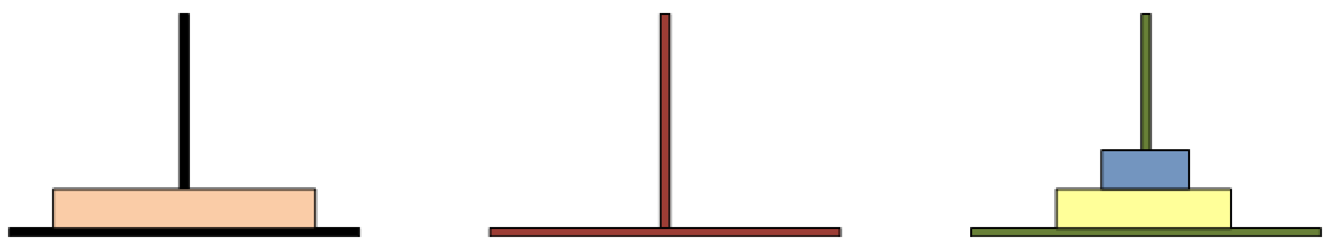
(1,2)



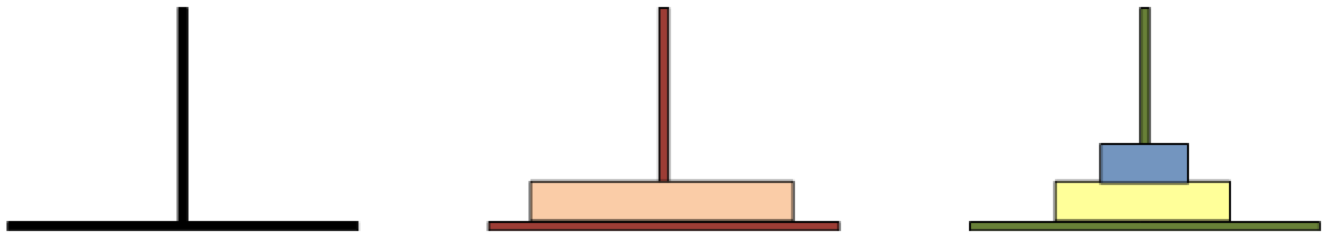
(1,3)



(2,3)



Mutăm unicul disc rămas pe stiva 1 (cu diametrul maxim) pe stiva 2

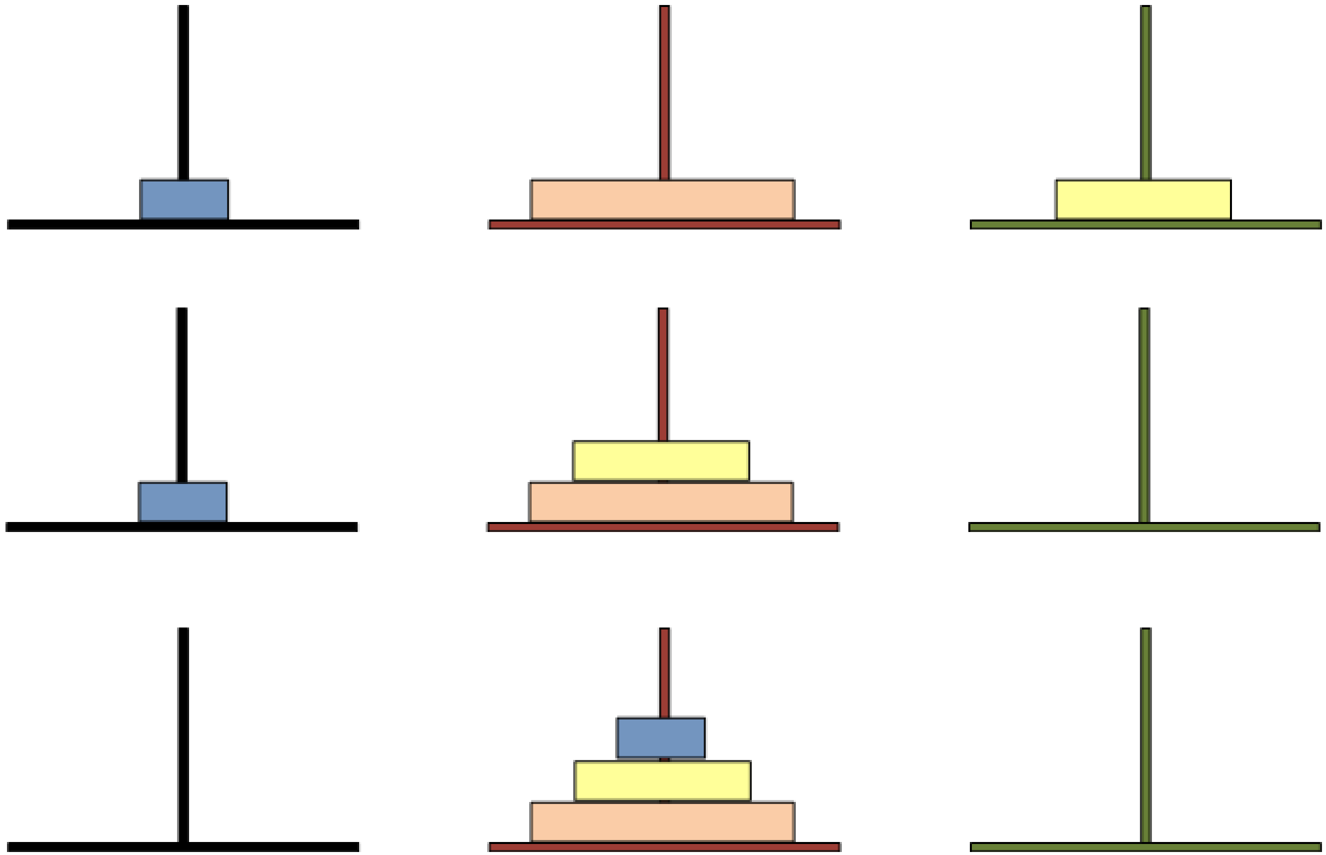


Mutăm două discuri de pe stiva 3 pe stiva 2, folosind stiva 1

(3,1)

(3,2)

(1,2)



Problema turnurilor din Hanoi

- Pentru $n>1$ are loc recurenţa

$$\mathbf{Han(n;i,j) = Han(n-1;i,k) + (i,j) + Han(n-1;k,j)}$$

unde $k = 6 - i - j$

Problema turnurilor din Hanoi

```
procedure Hanoi(n,i,j)
    if n = 1
         scrie(i,j)
    else k←6-i-j;
         Hanoi(n-1,i,k);
         Hanoi(1,i,j);    // scrie(i,j)
         Hanoi(n-1,k,j)
end
```

Apel: **`Hanoi(n,1,2)`**

Problema turnurilor din Hanoi

- $2^n - 1$ mutări

Problema turnurilor din Hanoi

- $2^n - 1$ mutări
- $T(n) = 2T(n-1)+1$, pentru $n>1$