

Recapitulare close a 1x - o-

$$D_{12} = \{1, 2, 3, 4, 6, 12\} \rightarrow 12 = \begin{cases} 1 \\ 2 \\ 3 \end{cases} \cdot 12$$

```
for (int i=1; i * i <= n; ++i)  
if (n % i == 0)  
cout << i << endl;
```

$$D_{16} = \{1, 2, 4, 8, 16\} \rightarrow 16 = 1 \cdot 16
16 = 2 \cdot 8
16 = 4 \cdot 4$$

$$\text{Cmmmc}(a, b) = a * b / \text{cmmdc}(a, b)$$

Factorizare

$$\begin{array}{c|c}
 120 & 2 \\
 60 & 2 \\
 30 & 2 \\
 15 & 3 \\
 5 & 5 \\
 1 &
 \end{array} \Rightarrow 120 = 2^3 \cdot 3 \cdot 5$$

cond. de oprire

Determinarea numărului de divizori a unui număr din descompunerea în fact. prim:

$$120 = 2^3 \cdot 3 \cdot 5 ; \quad \text{Divizorii } 120 : 2^a \cdot 3^b = 120 = 2^3 \cdot 3 \cdot 5$$

$$a \cdot b = \boxed{2^3 \cdot 3 \cdot 5}$$

$$Q = 2^{p_1} \cdot 3^{p_2} \cdot 5^{p_3} \text{ cu } p_1 \in [0, 3], p_2 \in [0, 1], p_3 \in [0, 1]$$

$$b = 2^{3-p_1} \cdot 3^{1-p_2} \cdot 5^{1-p_3} \quad \exists \ 4 \cdot 2 \cdot 2 = 16 \text{ divizori} \\
 \text{20 valori distincte pt. } a$$

$$\text{Nr. div} = (p_1 + 1) \cdot (p_2 + 1) \cdot (p_3 + 1) \cdots$$

$$M = 2^{P_1} \cdot 3^{P_2} \cdot 5^{P_3} \cdots$$

$$\leftarrow 5_{(10)} = ?_{(2)}$$

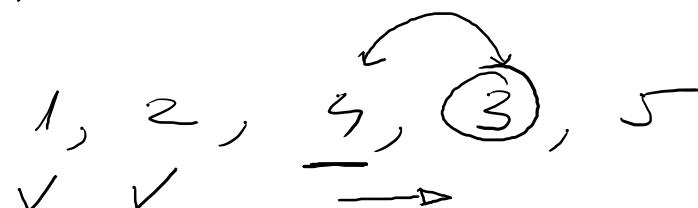
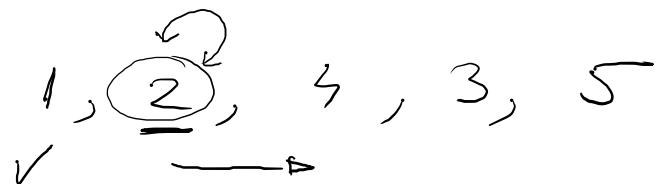
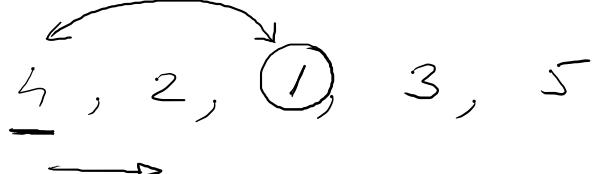
$$45_{(10)} = 101101_{(2)}$$

Stegene elementar vektor

$$\begin{array}{r}
 55 \overline{)222} \\
 -22 \\
 \hline
 = 0 \\
 \end{array}
 \quad
 \begin{array}{r}
 11 \overline{)222} \\
 -22 \\
 \hline
 = 0 \\
 \end{array}
 \quad
 \begin{array}{r}
 5 \overline{)222} \\
 -22 \\
 \hline
 = 0 \\
 \end{array}
 \quad
 \begin{array}{r}
 1 \overline{)222} \\
 -22 \\
 \hline
 = 0 \\
 \end{array}$$

Inserts new element:

Selection Sort



```

for (int i = 1; i <= n; ++i)
    for (int j = i+1; j <= n; ++j)
        if (e[j] < e[i])
            swap(e[i], e[j]);
    }
}

```

```

for (int i = 1; i <= n; ++i) {
    int imin = i;
    for (int j = i+1; j <= n; ++j)
        if (e[j] < e[imin])
            imin = j;
    swap(e[i], e[imin]);
}

```

Insertion sort :

3, 2, 5, 1, 3
↑ →

2, 4, 5, 1, 3

2, 4, 5, 1, 3

1, 2, 3, 4, 5
↓
abrupt

⇒ 1, 2, 3, 4, 5

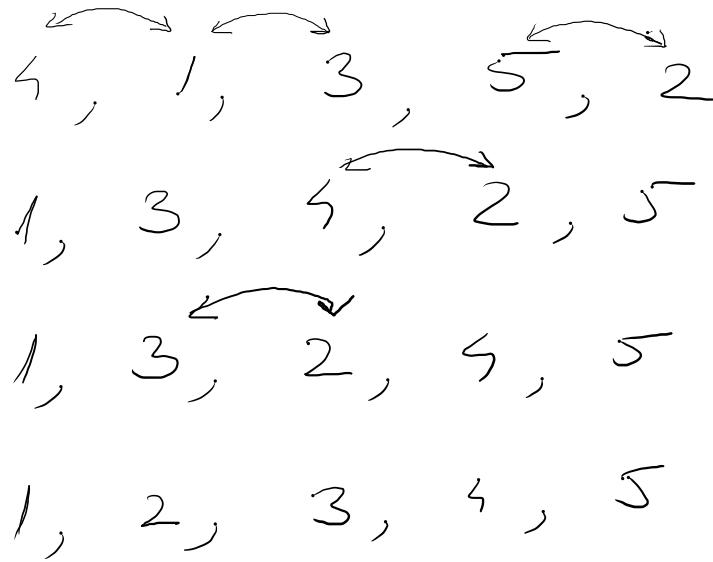
for (int i = 2; i <= n; ++i) {
 int j = i;
}

while ($a[j] < a[j-1]$ & $j \geq 2$)
 swap ($a[j]$, $a[j-1]$); $j--$

}

for (int i = 2; i <= n; ++i) {
 int pos = i;
 for (int j = i - 1; j >= 1; --j)
 if ($a[j] > a[i]$)
 pos = j;
 int copy = $a[i]$;
 for (int j = i - 1; j >= pos; --j)
 $a[j+1] = a[j]$;
 $a[pos] = copy$;

Bubble Sort



```

bool ordered = false;
while (!ordered) {
    ordered = true;
    for (int i = 1; i < n; ++i)
        if (e[i] > e[i+1]) {
            ordered = false;
            swap(e[i], e[i+1]);
        }
}

```

}

```
for (int k = 1; k < m; ++k)
```

```
    for (int i = 1; i < n; ++i)
```

```
        if (e[i] > e[i+1])
```

```
            swap(e[i], e[i+1]);
```

! Differente dină!

Bubble și Selection este

că la Bubble primul

fa nu are nici-o logică
a continuă să sortă.

Inclusiones

a[] = 1, 3, 5, 5, 7, 8, 10.

b[] = 2, 2, 3, 5, 8, 6, 5

c[] = 1, 2, 2, 3, 3,

while (ind_a <= n) {

 ind_c++;

 c[ind_c] = a[ind_a];

 ind_a++;

}

while (ind_b <= m) {

 ind_c++;

 c[ind_c] = b[ind_b];

 ind_b++;

c[+ind_c] = a[ind_a];

int ind_a = 1, ind_b = 1, ind_c = 0;
while (ind_a <= n && ind_b <= m)

 if (a[ind_a] < b[ind_b]) {

 ind_c++;

 c[ind_c] = a[ind_a];

 ind_a++;

}

 else {

 ind_c++;

 c[ind_c] = b[ind_b];

 ind_b++;

}

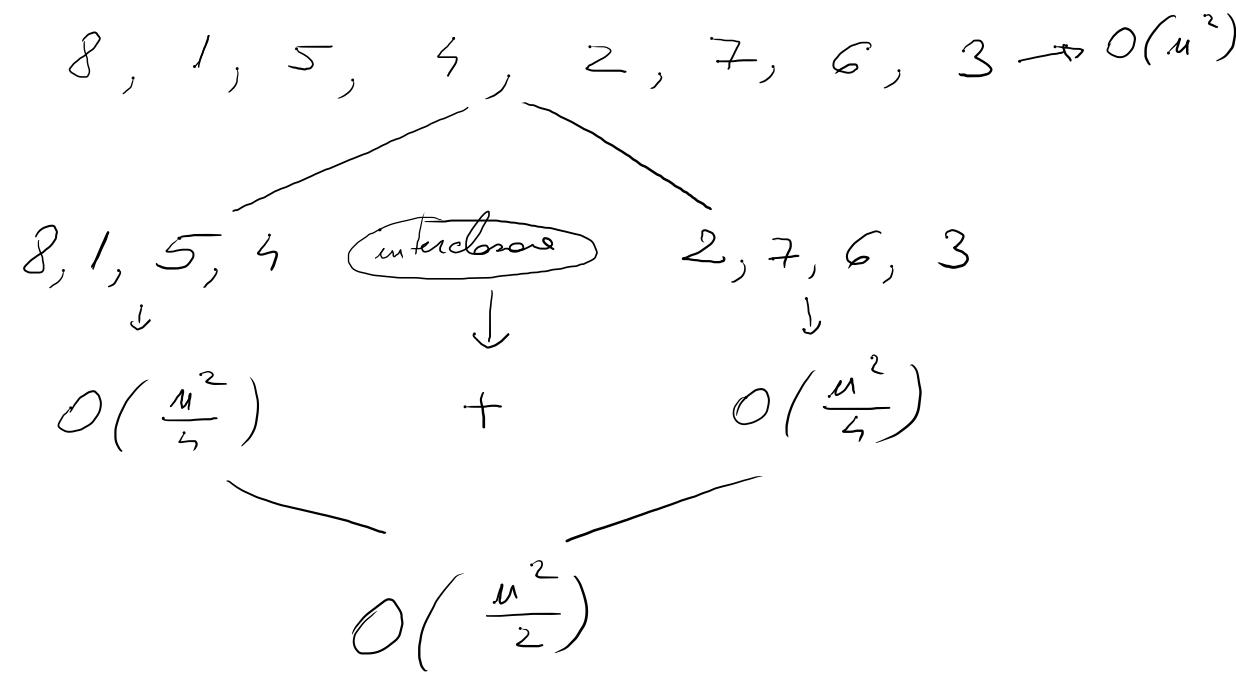
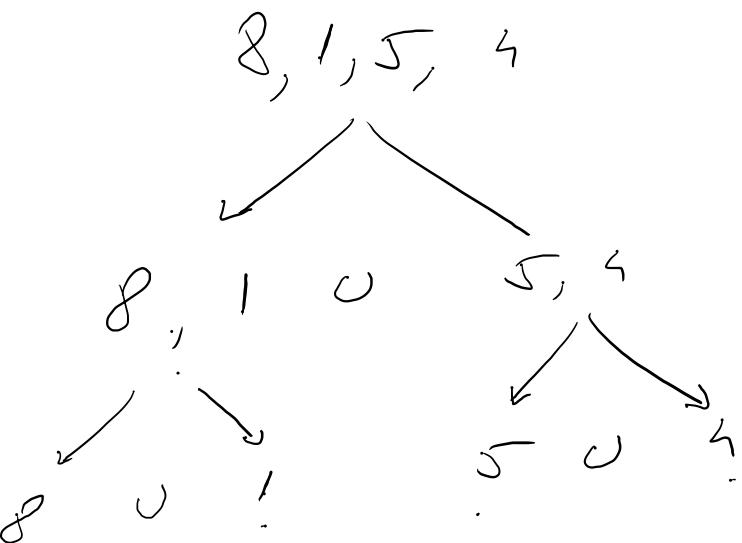
for (int i=1; i <= ind_c; ++i)
 cout << c[i] << ' ';

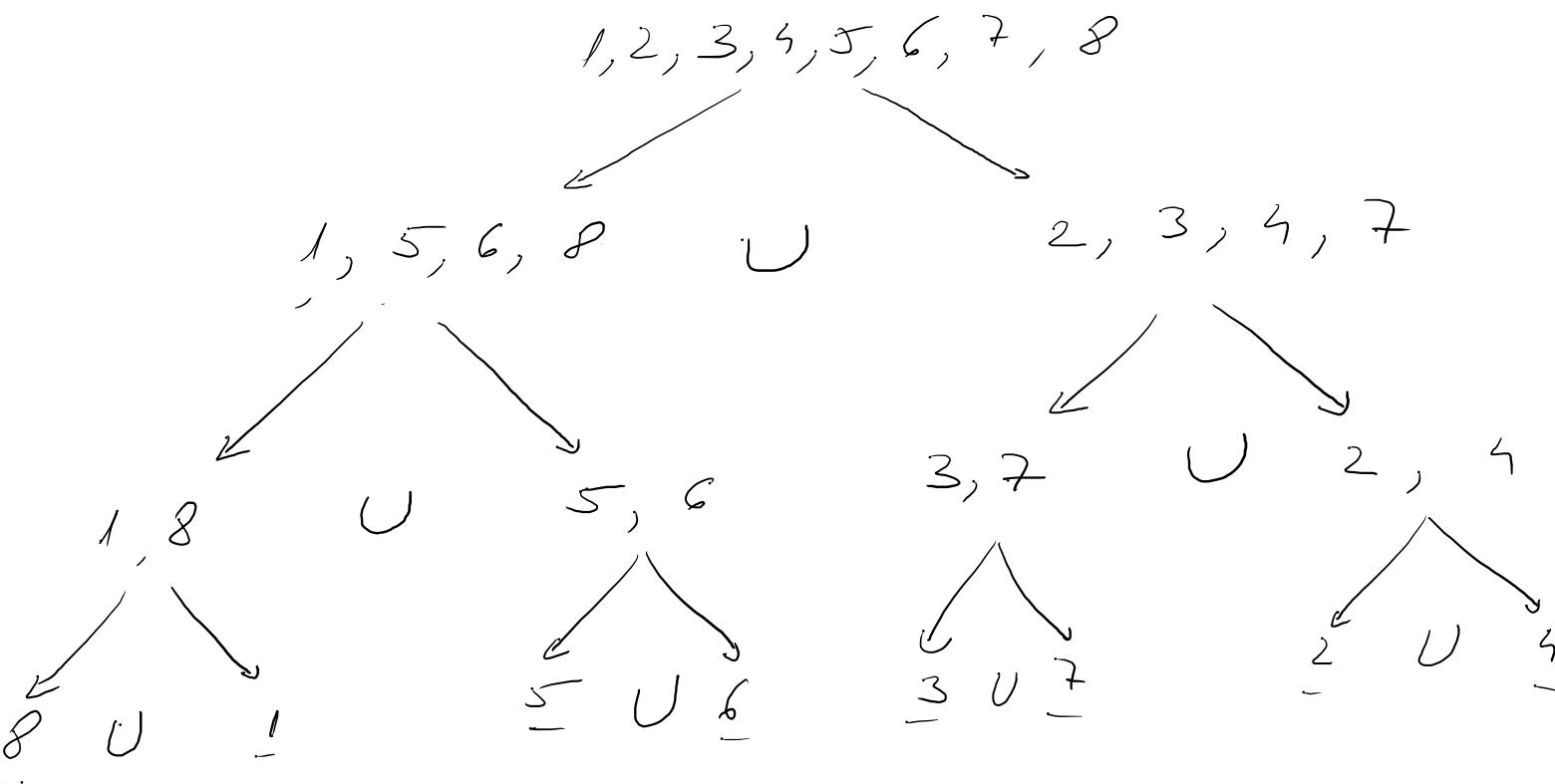
Diferente dintre ++i și i++

int i = 5;	int i = 5;
cout << i++; // 5	cout << ++i; // 6
cout << i; // 6	cout << i; // 6

Merge Sort :

→ Divide et Impera





```

void sortone (int st, int dr) {
    if (st < dr) {
        int mij = (st + dr) / 2;
        sortone (st, mij);
        sortone (mij + 1, dr);
    }
}

```

int closure [(st, mij) + (mij + 1, dr)];

Căutare Binară

• 1...100

↓
50

⇒ 51, 100

↓
75

68

⇒ 51, 75

↓
62

⇒ 62, 75

↓
68

→ Găsit !!!

Din că introducere am găsit numărul pe care îl căutăm.

• 1, 5, 12, 45, 52, 80, 95 , val = 24

bool exists (int e[], int u, int val) {

int st = 1, dr = n;

while (st <= dr) {

int mij = (st + dr) / 2;

if (e[mij] == val) return true;

else if (e[mij] > val) dr = mij - 1;

else st = mij + 1; } return false; }

- Se dă un număr n . Verificăți dacă n este p.p.
 - Dex. în fapt prim; să verificăm că toți factorii să opere la oprire poră.
 $\hookrightarrow O(\sqrt{n})$
 - Parcurgem de la $1 \dots \sqrt{n}$ toate numerele și verificăm dacă există o roădăinie.
 $\hookrightarrow O(\sqrt{n})$
 - Calculăm n -div de n ; și dacă e impar $\Rightarrow n = p \cdot p$
 $\hookrightarrow O(\sqrt{n})$
-
- $\sqrt{n} \in [1, n] \Rightarrow$ îl căutăm binar în $[1, n]$
 int st = 1, dr = n;
 while (st <= dr) {
 int mij = (st + dr) / 2;
 if (mij * mij == n)
 return true;
 else if (mij * mij > n)
 dr = mij - 1;
 else st = mij + 1;
 }
 return false;
- $O(\log_2 n)$