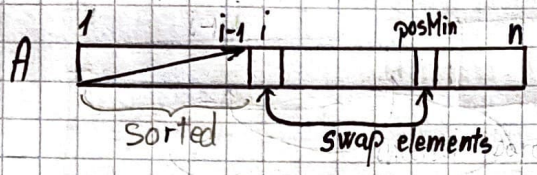
# SEMINARUL 1

# Metode de sortare

## 1. SORTAREA PRIN SELECȚIE → de mod




```
selection(A, n)
index ← 0          // i
while (index < n)
    EXTRACT_MIN_POS(A, index, n, adr. lui required_position)
    SWAP(A, index, required_position)   // swap A[index], A[req_pos]
                                        → only if index != required_position
    index ← index + 1
```

unde `EXTRACT_MIN_POS(A, index, n, required_position)`

```
foreach j in range index, n
    if (A[j] < min)
        min ← A[j]
        required_position ← j   // by reference
```

OBS! operațiile de comparație sunt mai puțin costisitoare ca timp decât operațiile pe structuri (e.g. swap) ; flăgari

OBS! în studiul nostru, noi vom număra atribuiri și comparații (împreună ⇒ operații). Nu numărăm operații pe indecși și flăgari!

Variantă de pseudocod propusă, analizată și eficientizată:

```
selection(A, n)
    for i = 1, n-1
```

$\Sigma_{comp}$ (nr. de comparații)

$i=1$ : $j=2 \to n$ : $n-1$ comp.

$i=2$ : $j=3 \to n$ : $n-2$ comp.

⋮

$i=n-1$ : $j=n \to n$ : 1 comp.

$\frac{n(n-1)}{2}$

$\Sigma_{atribuiri\_best} = 0$ (șir crescător)

$\Sigma_{atribuiri\_worst} = 3(n-1)$ (cel mai mic elem. să fie la final, iar restul șirului să fie ordonat) (orice rotire a șirului ne dă așa ceva)

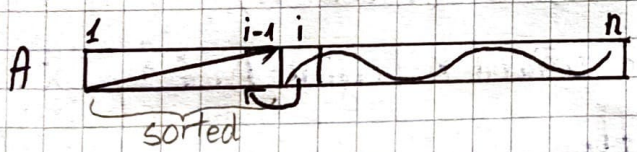
$\Sigma_{atribuiri\_avg} = const(n-1)$

↳ mai mică decât 3, but close enough

Este alg. stabil? Nu! Contraexemplu: $2_1, 2_2, 1$ → $1, 2_2, 2_1$

↳ 2 el. egale își păstrează pozițiile sau se apropie, dar nu se interschimbă

## 2. SORTAREA PRIN INSERARE (insertion) → de implementat cu binary search → BONUS




```
insertion(A, n)
    high_index ← 1
    while (high_index ≤ n-1)
        new_pos ← -1
        BINARY_INSERT_POS(A, high_index, &new_pos)
        aux ← A[high_index]
        foreach i ↓ downto (high_index, new_pos+1)
            A[i] ← A[i-1]
        A[new_pos] ← aux
        high_index ← high_index + 1
```

unde BINARY_INSERT_POS (A, high-edge, mare-pe) dă cumva.

Variantă de pseudocod propusă, analizată și eficientizată:

```
for i = 2 to n
    key = A[i]                          ← nu este inf. de date de intrare
    j = i-1
    while j >= 1 and A[j] > key         ← dar așa este
        A[j+1] ← A[j]
        j = j-1
    A[j+1] ← key
```

profiler.countOperation("atribIns", n, 2)

$\sum_{atrib\_best} = 2(n-1)$ (sortat)

$\sum_{comparații\_best} = n-1$

$\sum_{atrib\_worst} = \frac{n(n-1)}{2}$

$\sum_{comp\_worst} = 2(n-1) + \frac{n(n-1)}{2}$

Notă: Selection face swap pe bucată de șir (vector) NESORTATĂ → instabilitate

3. SORTAREA PRIN INTERSCHIMBARE (bubblesort)

de implementat în varianta cu nr. variabil x lungime. Dacă după x parcurgeri șirul e sortat, get out ⇒ x = flag.