

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»  
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ**

Лабораторна робота № 2  
з дисципліни «Алгоритми та структури даних»  
На тему: «Методи сортування масивів»

Виконав:  
студент групи ФІ-12  
Завалій Олександр

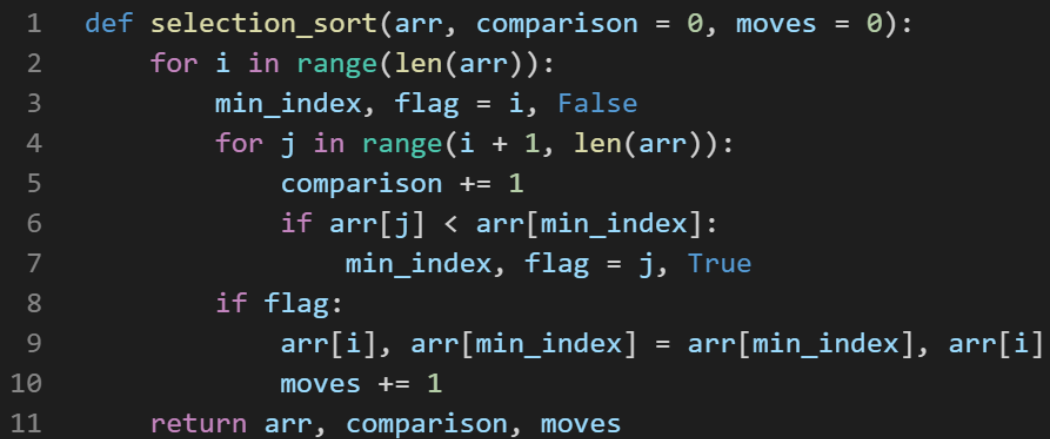
# Реалізація завдання

## Task I

### Варіант №5

Написати програму, що реалізує один з простих методів сортування. Сортування методом вибору.

$$C = \frac{(n-1)n}{2}$$
$$M = n - 1$$

A screenshot of a terminal window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The code is a Python function named 'selection\_sort' that takes an array 'arr' and two optional parameters 'comparison' and 'moves', both defaulting to 0. It uses nested loops to find the minimum element in the unsorted part of the array and swap it with the first element. The function returns the sorted array, the total number of comparisons, and the total number of moves.

```
1 def selection_sort(arr, comparison = 0, moves = 0):
2     for i in range(len(arr)):
3         min_index, flag = i, False
4         for j in range(i + 1, len(arr)):
5             comparison += 1
6             if arr[j] < arr[min_index]:
7                 min_index, flag = j, True
8         if flag:
9             arr[i], arr[min_index] = arr[min_index], arr[i]
10            moves += 1
11    return arr, comparison, moves
```

Результати виконання алгоритму.

```
Selection sort
Comparison: 4950. Moves: 94
Comparison: 499500. Moves: 994
Comparison: 49995000. Moves: 9988

Execution time
Array: 100. Algorithm: selection_sort. Minimum execution time: 0.004801600007340312
Array: 1000. Algorithm: selection_sort. Minimum execution time: 0.4833672000095248
Array: 10000. Algorithm: selection_sort. Minimum execution time: 49.92777469998691
```

## Task II

Написати програму, що реалізує метод швидкого сортування.

$\sim n \log_2 n$

```
1 moves1 = 0
2 comparison1 = 0
3 def merge(arr, l, h):
4     global moves1
5     global comparison1
6
7     pivot = arr[h]
8     item = l - 1
9
10    for i in range(l, h):
11        comparison1 += 1
12        if arr[i] <= pivot:
13            item = item + 1
14            arr[item], arr[i] = arr[i], arr[item]
15            moves1 += 1
16    moves1 += 1
17    arr[item + 1], arr[h] = arr[h], arr[item + 1]
18    return item + 1
19
20 def quick_sort(arr, l, h):
21     if l < h:
22         pivot = merge(arr, l, h)
23         quick_sort(arr, l, pivot - 1)
24         quick_sort(arr, pivot + 1, h)
25     return arr
```

Результати виконання алгоритму.

```
Quick sort
Comparison, moves: 1162
Comparison, moves: 10955
Comparison, moves: 159857

Execution time
Array: 100. Algorithm: quick_sort. Minimum execution time: 0.0010750999790616333
Array: 1000. Algorithm: quick_sort. Minimum execution time: 0.018154399993363768
Array: 10000. Algorithm: quick_sort. Minimum execution time: 0.25130500001250766
```

Функція, що використовується для підрахунку часу виконання іншого блоку кода.

```
1 def run_time(algorithm, array, high):
2     setup_code = f"from __main__ import {algorithm}"
3     if algorithm != 'selection_sort':
4         stmt = f"{algorithm}({array}, {0}, {high})"
5     else:
6         stmt = f"{algorithm}({array})"
7     times = repeat(setup=setup_code, stmt=stmt, repeat=3, number=10)
8     print(f"Array: {len(array)}. Algorithm: {algorithm}. Minimum execution time: {min(times)}")
```

Початковий та відсортовані масиви.

```
Initial array:
[-569, 759, 229, -767, -131, 401, 287, -656, -593, 667, 194, 936, 594, -377, -530, 511, -225, 614, 674, 612, 533, 416, -417,
 556, 956, 836, 250, -745, -233, 435, 717, -276, 24, -358, -542, -588, 146, 889, -700, -859, -528, 115, -448, -725, -359,
 816, -533, 450, -158, 361, 825, -595, 661, 421, -752, -885, 919, -533, 543, -177, -455, 257, 776, 210, 758, -35, -973, -765,
 -542, -754, -553, -57, -422, 116, 825, -661, -622, -410, 105, -282, 173, 647, 648, -100, 725, 740, 60, 213, -289, 580, -4
 90, -581, -672, -359, -804, 667, 590, 261, -580, -562]

Selection sort:
[-973, -885, -859, -804, -767, -765, -754, -752, -745, -725, -700, -672, -661, -656, -622, -595, -593, -588, -581, -580, -5
 69, -562, -553, -542, -542, -533, -533, -530, -528, -490, -455, -448, -422, -417, -410, -377, -359, -359, -358, -289, -282,
 -276, -233, -225, -177, -158, -131, -100, -57, -35, 24, 60, 105, 115, 116, 146, 173, 194, 210, 213, 229, 250, 257, 261, 28
 7, 361, 401, 416, 421, 435, 450, 511, 533, 543, 556, 580, 590, 594, 612, 614, 647, 648, 661, 667, 667, 674, 717, 725, 740,
 758, 759, 776, 816, 825, 825, 836, 889, 919, 936, 956]

Quick sort:
[-973, -885, -859, -804, -767, -765, -754, -752, -745, -725, -700, -672, -661, -656, -622, -595, -593, -588, -581, -580, -5
 69, -562, -553, -542, -542, -533, -533, -530, -528, -490, -455, -448, -422, -417, -410, -377, -359, -359, -358, -289, -282,
 -276, -233, -225, -177, -158, -131, -100, -57, -35, 24, 60, 105, 115, 116, 146, 173, 194, 210, 213, 229, 250, 257, 261, 28
 7, 361, 401, 416, 421, 435, 450, 511, 533, 543, 556, 580, 590, 594, 612, 614, 647, 648, 661, 667, 667, 674, 717, 725, 740,
 758, 759, 776, 816, 825, 825, 836, 889, 919, 936, 956]
```

## Результати порівняння методів сортування

	Сортування методом вибору					Швидке сортування				
N	К-ть копіювань		К-ть порівнянь		Час	К-ть копіювань		К-ть порівнянь		Час
	(М)		(С)			(М)		(С)		
	Теорет.	Експерим.	Теорет.	Експерим.	(Т)	Теорет.	Експерим.	Теорет.	Експерим.	(Т)
100	99	93	4950	4950	0.0047	664	688	664	1162	0.001176
1000	999	996	499500	499500	0.4166	9966	6285	9966	10955	0.000992
10000	9999	9992	49995000	49995000	43.4776	132877	81816	132877	159857	0.2304