

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики
Кафедра програмування



ЛАБОРАТОРНА РОБОТА №5

Виконала:
студентка групи ПМОм-11
Кравець Ольга

Львів 2025

Візуалізація виконання алгоритму

Завдання 1

Задача: написати програму, яка аналізує послідовність чисел, що вводяться користувачем, і знаходить найдовшу зростаючу підпослідовність.

Програма використовує галуження для перевірки, чи поточне число більше за попереднє, і цикли для аналізу послідовності.

```
def longest_increasing_subsequence(sequence):
    longest = []
    current = []

    for i in range(len(sequence)):
        #контрольна точка - друк поточного числа
        print(f"Обробляється число {sequence[i]}")

        if i == 0 or sequence[i] > sequence[i - 1]:
            current.append(sequence[i]) #додаємо число до поточної
підпослідовності
        else:
            if len(current) > len(longest):
                longest = current #оновлення найдовшої підпослідовності
            current = [sequence[i]] #початок нової підпослідовності

        #контрольна точка - відображення поточної підпослідовності
        print(f"Поточна зростаюча підпослідовність: {current}")

    if len(current) > len(longest):
        longest = current #оновлення найдовшої підпослідовності, якщо
потрібно

    return longest

#зчитування послідовності чисел
numbers = list(map(int, input("Введіть послідовність чисел через пробіл:
").split()))

#контрольна точка - вхідні дані
print(f"Введена послідовність: {numbers}")

result = longest_increasing_subsequence(numbers)

print(f"Найдовша зростаюча підпослідовність: {result}")
```

```

Введіть послідовність чисел через пробіл: 2 4 7 3 2 5 8 4 2 8 0
Введена послідовність: [2, 4, 7, 3, 2, 5, 8, 4, 2, 8, 0]
Обробляється число 2
Поточна зростаюча підпослідовність: [2]
Обробляється число 4
Поточна зростаюча підпослідовність: [2, 4]
Обробляється число 7
Поточна зростаюча підпослідовність: [2, 4, 7]
Обробляється число 3
Поточна зростаюча підпослідовність: [3]
Обробляється число 2
Поточна зростаюча підпослідовність: [2]
Обробляється число 5
Поточна зростаюча підпослідовність: [2, 5]
Обробляється число 8
Поточна зростаюча підпослідовність: [2, 5, 8]
Обробляється число 4
Поточна зростаюча підпослідовність: [4]
Обробляється число 2
Поточна зростаюча підпослідовність: [2]
Обробляється число 8
Поточна зростаюча підпослідовність: [2, 8]
Обробляється число 0
Поточна зростаюча підпослідовність: [0]
Найдовша зростаюча підпослідовність: [2, 4, 7]

```

Завдання 2

```

def print_checkpoint(message, data):
    """Функція друку контрольної точки"""
    print(f"Контрольна точка - {message}: {data}")

def longest_increasing_subsequence(sequence):
    longest = []
    current = []

    for i in range(len(sequence)):
        #виклик функції друку контрольної точки
        print_checkpoint("Обробляється число", sequence[i])

        if i == 0 or sequence[i] > sequence[i - 1]:
            current.append(sequence[i]) #додаємо число до поточної
підпослідовності
        else:
            if len(current) > len(longest):
                longest = current #оновлення найдовшої підпослідовності
            current = [sequence[i]] #початок нової підпослідовності

        #виклик функції друку контрольної точки
        print_checkpoint("Поточна зростаюча підпослідовність", current)

```

```

    if len(current) > len(longest):
        longest = current #оновлення найдовшої підпослідовності, якщо
        потрібно

    return longest

```

```

#зчитування послідовності чисел
numbers = list(map(int, input("Введіть послідовність чисел через пробіл:
").split()))

```

```

#виклик функції друку контрольної точки
print_checkpoint("Введена послідовність", numbers)

```

```

result = longest_increasing_subsequence(numbers)

```

```

print(f"Найдовша зростаюча підпослідовність: {result}")

```

```

Введіть послідовність чисел через пробіл: 2 4 7 3 2 5 8 4 2 8 0
Контрольна точка - Введена послідовність: [2, 4, 7, 3, 2, 5, 8, 4, 2, 8, 0]
Контрольна точка - Обробляється число: 2
Контрольна точка - Поточна зростаюча підпослідовність: [2]
Контрольна точка - Обробляється число: 4
Контрольна точка - Поточна зростаюча підпослідовність: [2, 4]
Контрольна точка - Обробляється число: 7
Контрольна точка - Поточна зростаюча підпослідовність: [2, 4, 7]
Контрольна точка - Обробляється число: 3
Контрольна точка - Поточна зростаюча підпослідовність: [3]
Контрольна точка - Обробляється число: 2
Контрольна точка - Поточна зростаюча підпослідовність: [2]
Контрольна точка - Обробляється число: 5
Контрольна точка - Поточна зростаюча підпослідовність: [2, 5]
Контрольна точка - Обробляється число: 8
Контрольна точка - Поточна зростаюча підпослідовність: [2, 5, 8]
Контрольна точка - Обробляється число: 4
Контрольна точка - Поточна зростаюча підпослідовність: [4]
Контрольна точка - Обробляється число: 2
Контрольна точка - Поточна зростаюча підпослідовність: [2]
Контрольна точка - Обробляється число: 8
Контрольна точка - Поточна зростаюча підпослідовність: [2, 8]
Контрольна точка - Обробляється число: 0
Контрольна точка - Поточна зростаюча підпослідовність: [0]
Найдовша зростаюча підпослідовність: [2, 4, 7]

```

Завдання 3

Програма знаходить найдовшу зростаючу підпослідовність у введеному масиві чисел. Контрольні точки реалізовані через функцію `printCheckpoint`, яка виводить поточний стан програми.

```

#include <iostream>
#include <vector>
#include <windows.h>

using namespace std;

//функція друку контрольної точки
void printCheckpoint(const string& message, const vector<int>& data) {
    cout << "Контрольна точка - " << message << " [";
    for (size_t i = 0; i < data.size(); i++) {
        cout << data[i] << (i < data.size() - 1 ? ", " : "");
    }
    cout << "]" << endl;
}

vector<int> longestIncreasingSubsequence(const vector<int>& sequence) {
    vector<int> longest, current;

    for (size_t i = 0; i < sequence.size(); i++) {
        //контрольна точка - обробка поточного числа
        cout << "Контрольна точка- Обробляється число " << sequence[i] << endl;

        if (i == 0 || sequence[i] > sequence[i - 1]) {
            current.push_back(sequence[i]);
        }
        else {
            if (current.size() > longest.size()) {
                longest = current;
            }
            current = { sequence[i] };
        }

        //контрольна точка - поточна зростаюча підпоследовність
        printCheckpoint("Поточна зростаюча підпоследовність", current);
    }

    if (current.size() > longest.size()) {
        longest = current;
    }

    return longest;
}

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    vector<int> numbers;
    int n;
    cout << "Введіть кількість елементів: ";
    cin >> n;
    numbers.resize(n);

    cout << "Введіть числа: ";
    for (int i = 0; i < n; i++) {
        cin >> numbers[i];
    }

    printCheckpoint("Введена последовність", numbers);

    vector<int> result = longestIncreasingSubsequence(numbers);
}

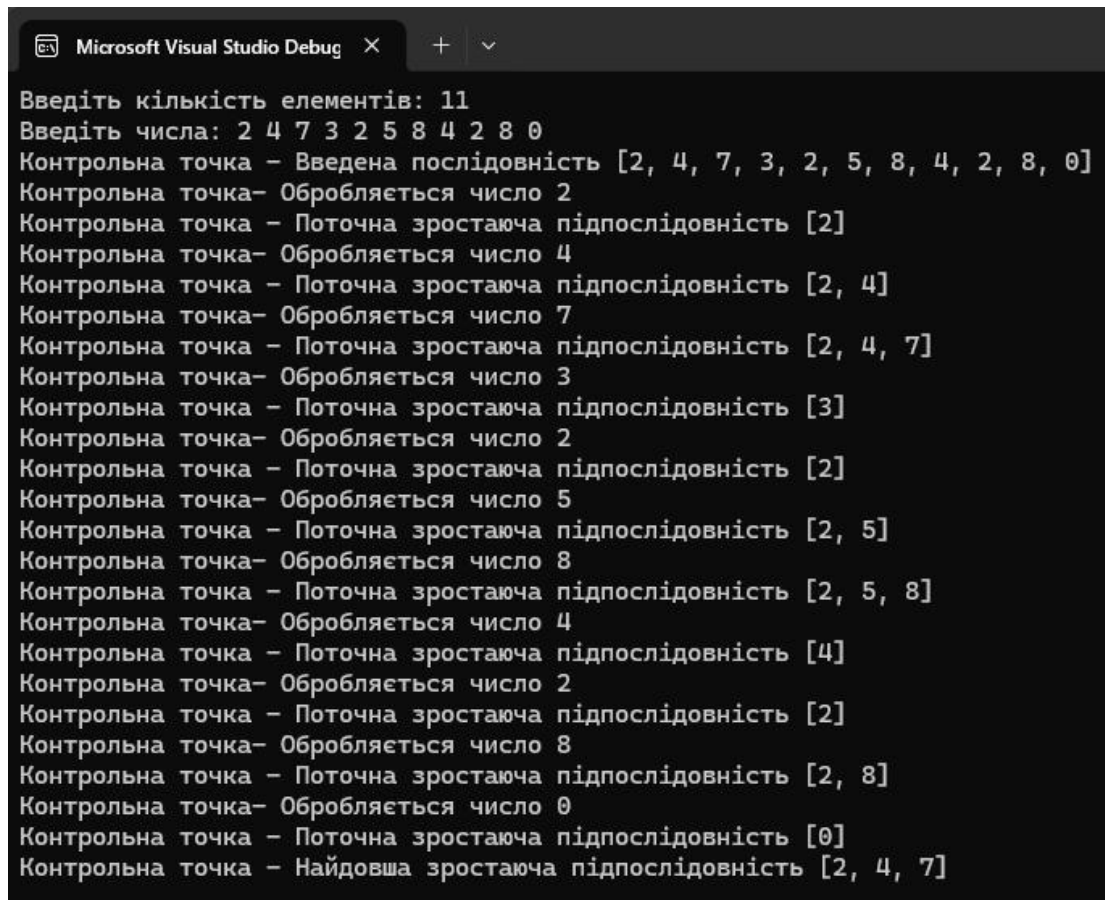
```

```

printCheckpoint("Найдовша зростаюча підпоследовність", result);

return 0;
}

```



```

Microsoft Visual Studio Debug X + v
Введіть кількість елементів: 11
Введіть числа: 2 4 7 3 2 5 8 4 2 8 0
Контрольна точка - Введена послідовність [2, 4, 7, 3, 2, 5, 8, 4, 2, 8, 0]
Контрольна точка- Обробляється число 2
Контрольна точка - Поточна зростаюча підпоследовність [2]
Контрольна точка- Обробляється число 4
Контрольна точка - Поточна зростаюча підпоследовність [2, 4]
Контрольна точка- Обробляється число 7
Контрольна точка - Поточна зростаюча підпоследовність [2, 4, 7]
Контрольна точка- Обробляється число 3
Контрольна точка - Поточна зростаюча підпоследовність [3]
Контрольна точка- Обробляється число 2
Контрольна точка - Поточна зростаюча підпоследовність [2]
Контрольна точка- Обробляється число 5
Контрольна точка - Поточна зростаюча підпоследовність [2, 5]
Контрольна точка- Обробляється число 8
Контрольна точка - Поточна зростаюча підпоследовність [2, 5, 8]
Контрольна точка- Обробляється число 4
Контрольна точка - Поточна зростаюча підпоследовність [4]
Контрольна точка- Обробляється число 2
Контрольна точка - Поточна зростаюча підпоследовність [2]
Контрольна точка- Обробляється число 8
Контрольна точка - Поточна зростаюча підпоследовність [2, 8]
Контрольна точка- Обробляється число 0
Контрольна точка - Поточна зростаюча підпоследовність [0]
Контрольна точка - Найдовша зростаюча підпоследовність [2, 4, 7]

```

Завдання 4

Посилання на деякі системи графічної візуалізації:

<https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>

Data Structure Visualizations by USFCA - цей сайт пропонує інтерактивні анімації для різних структур даних та алгоритмів, включаючи сортування, рекурсію, графові алгоритми та інші.

<https://csvistool.com>

CS 1332 Data Structures and Algorithms Visualizations - офіційний інструмент візуалізації структур даних та алгоритмів для курсу CS 1332 в Georgia Tech, який охоплює широкий спектр тем.

<https://tobinatore.github.io/algovis/index.html>

AlgoVis.io - інтерактивна платформа, яка надає короткі пояснення роботи алгоритмів, псевдокод з динамічним підсвічуванням рядків та якісні візуалізації на всіх пристроях.

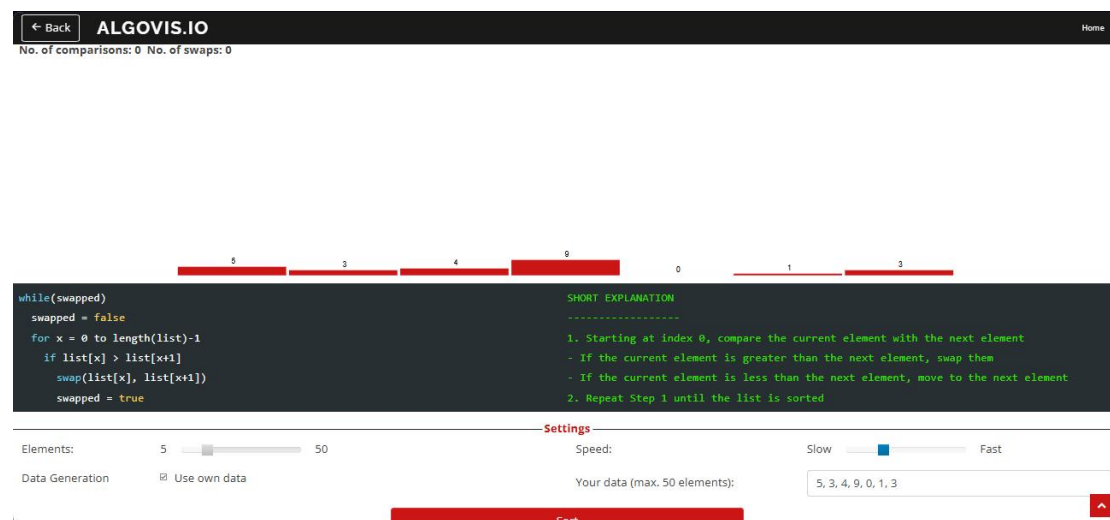
Експеримент з використанням [AlgoVis.io](https://tobinatore.github.io/algovis/index.html)

Я обрала алгоритм Bubble Sort (сортування бульбашкою) для візуалізації.

Вхідні дані: масив чисел [5, 3, 4, 9, 0, 1, 3].

Використовуючи AlgoVis.io, я ввела цей масив та запустила візуалізацію алгоритму сортування злиттям. На кожному кроці алгоритм рекурсивно ділив масив на підмасиви, сортував їх та об'єднував назад, що було наочно продемонстровано на екрані.

На знімках не кожен крок зображено, бо сортує швидко і не встигала вловити кожен крок.



No. of comparisons: 0 No. of swaps: 0

3549013

```
while(swapped)
  swapped = false
  for x = 0 to length(list)-1
    if list[x] > list[x+1]
      swap(list[x], list[x+1])
  swapped = true
```

SHORT EXPLANATION

1. Starting at index 0, compare the current element with the next element

- If the current element is greater than the next element, swap them
- If the current element is less than the next element, move to the next element

2. Repeat Step 1 until the list is sorted

Settings

Elements: ☐ Use own data

Speed: Slow Fast

Your data (max. 50 elements): 5, 3, 4, 9, 0, 1, 3

Sort

← Back

ALGOVIS.IO

Home

No. of comparisons: 8 No. of swaps: 5

3406139

```
while(swapped)
  swapped = false
  for x = 0 to length(list)-1
    if list[x] > list[x+1]
      swap(list[x], list[x+1])
  swapped = true
```

SHORT EXPLANATION

1. Starting at index 0, compare the current element with the next element

- If the current element is greater than the next element, swap them
- If the current element is less than the next element, move to the next element

2. Repeat Step 1 until the list is sorted

Settings

Elements: ☐ Use own data

Speed: Slow Fast

Your data (max. 50 elements): 5, 3, 4, 9, 0, 1, 3

Sort

No. of comparisons: 14 No. of swaps: 10

3014358

```
while(swapped)
  swapped = false
  for x = 0 to length(list)-1
    if list[x] > list[x+1]
      swap(list[x], list[x+1])
  swapped = true
```

SHORT EXPLANATION

1. Starting at index 0, compare the current element with the next element

- If the current element is greater than the next element, swap them
- If the current element is less than the next element, move to the next element

2. Repeat Step 1 until the list is sorted

Settings

Elements: ☐ Use own data

Speed: Slow Fast

Your data (max. 50 elements): 5, 3, 4, 9, 0, 1, 3

Sort

Back
ALGOVIS.IO
Home

No. of comparisons: 20
No. of swaps: 13

0
1
3
3
4
5
9

```

while(swapped)
  swapped = false
  for x = 0 to length(list)-1
    if list[x] > list[x+1]
      swap(list[x], list[x+1])
      swapped = true

```

SHORT EXPLANATION
1. Starting at index 0, compare the current element with the next element
- If the current element is greater than the next element, swap them
- If the current element is less than the next element, move to the next element
2. Repeat Step 1 until the list is sorted

Elements:
5
50

Speed:
Slow
Fast

Data Generation
☒ Use own data

Your data (max. 50 elements):
5, 3, 4, 9, 0, 1, 3

Sort

1-й прохід (переміщення найбільшого елемента в кінець)

$5 > 3 \rightarrow$ поміняли місцями $\rightarrow [3, 5, 4, 9, 0, 1, 3]$

$5 > 4 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 5, 9, 0, 1, 3]$

$5 < 9 \rightarrow$ без змін

$9 > 0 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 5, 0, 9, 1, 3]$

$9 > 1 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 5, 0, 1, 9, 3]$

$9 > 3 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 5, 0, 1, 3, 9]$

(9 зайняло своє місце в кінці)

2-й прохід

$3 < 4 \rightarrow$ без змін

$4 < 5 \rightarrow$ без змін

$5 > 0 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 0, 5, 1, 3, 9]$

$5 > 1 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 0, 1, 5, 3, 9]$

$5 > 3 \rightarrow$ поміняли місцями $\rightarrow [3, 4, 0, 1, 3, 5, 9]$

(5 зайняло своє місце перед 9)

3-й прохід

$3 < 4 \rightarrow$ без змін

$4 > 0 \rightarrow$ поміняли місцями $\rightarrow [3, 0, 4, 1, 3, 5, 9]$

$4 > 1 \rightarrow$ поміняли місцями $\rightarrow [3, 0, 1, 4, 3, 5, 9]$

$4 > 3 \rightarrow$ поміняли місцями $\rightarrow [3, 0, 1, 3, 4, 5, 9]$

(4 зайняло своє місце)

4-й прохід

$3 > 0 \rightarrow$ поміняли місцями $\rightarrow [0, 3, 1, 3, 4, 5, 9]$

$3 > 1 \rightarrow$ поміняли місцями $\rightarrow [0, 1, 3, 3, 4, 5, 9]$

(3 зайняли свої місця)

5-й прохід

Вже все відсортовано, обмінів немає \rightarrow алгоритм завершено

У AlgoVis.io добре видно, як найбільші елементи “виштовхуються” догори, а маленькі просідають вниз.