МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ   
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНА ДИСЦИПЛІНА  
Алгоритми та методи обчислень

ЗВІТ

З ПРАКТИЧНИХ РОБІТ

Виконав:

студент групи КН-23-1

Батраков Є.Є.

Кременчук 2024

# Практична робота № 3

**Тема**. Алгоритми сортування та їх складність. Порівняння алгоритмів

сортування

**Мета**: опанувати основні алгоритми сортування та навчитись методам

аналізу їх асимптотичної складності.

## Хід роботи

### Завдання:



def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swapped = True

if not swapped:

break

return arr

arr = [64, 34, 25, 12, 22, 11, 90]

sorted\_arr = bubble\_sort(arr)

print("Відсортований масив:", sorted\_arr)

Найгірший випадок:

У найгіршому випадку (коли масив відсортований в зворотному порядку) кожен елемент потрібно порівняти з усіма іншими елементами, що дає складність: O(n2)

Найкращий випадок:

У найкращому випадку (коли масив вже відсортований), алгоритм проходить один раз через масив і не виконує жодного обміну, що дає складність: O(n)

**Сортування вставками**:

Найгірший випадок: O(n2) (коли масив відсортований у зворотному порядку)

Найкращий випадок: O(n) (коли масив вже відсортований)

**Чому бульбашковий алгоритм менш ефективний на практиці?**

1 Кількість порівнянь і обмінів: У порівнянні з сортуванням вставками, бульбашкове сортування виконує більше обмінів, що призводить до більшого часу виконання.

2 Неоптимальність: Навіть якщо масив частково відсортований, бульбашковий алгоритм все одно виконує багато зайвих порівнянь і обмінів.

**Сортування злиттям**:

Найгірший випадок: O(n logn)

Найкращий випадок: O(n logn)

**Чому сортування злиттям ефективніше**?

1 Часова складність: Сортування злиттям має кращу часову складність в усіх випадках O(n logn), що робить його набагато швидшим для великих масивів.

2 Рекурсивний підхід: Сортування злиттям ділить масив на менші підмасиви і сортує їх незалежно, що зменшує кількість необхідних порівнянь.

3 Стабільність: Сортування злиттям є стабільним алгоритмом, що зберігає порядок однакових елементів, що є важливим для певних застосувань

3)

def quicksort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quicksort(left) + middle + quicksort(right)

arr = [3, 6, 8, 10, 1, 2, 1]

print(quicksort(arr))