МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА
ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНОЇ РОБОТИ № 3

з навчальної дисципліни

«Алгоритми та структури даних**»**

Тема «Алгоритми сортування та їх складність.
Порівняння алгоритмів сортування»

Студентка гр. КН-24-1 Бояринцова П. С.
Викладач Сидоренко В. М.

Кременчук 2025

Тема роботи: Алгоритми сортування та їх складність. Порівняння алгоритмів сортування

1.1 Постановка завдання

Мета роботи: опанувати основні алгоритми сортування та навчитись методам аналізу їх асимптотичної складності.

Завдання: реалізувати алгоритми сортування

1.2 Розв'язання завдань

Завдання 1.

1. Записати алгоритм бульбашкового сортування. Оцінити асимптотику алгоритму. Порівняти бульбашковий алгоритм з алгоритмом сортування вставлянням.
2. Чому на практиці бульбашковий алгоритм виявляється менш ефективним у порівнянні з сортуванням методом зливанням?

Розв'язання:

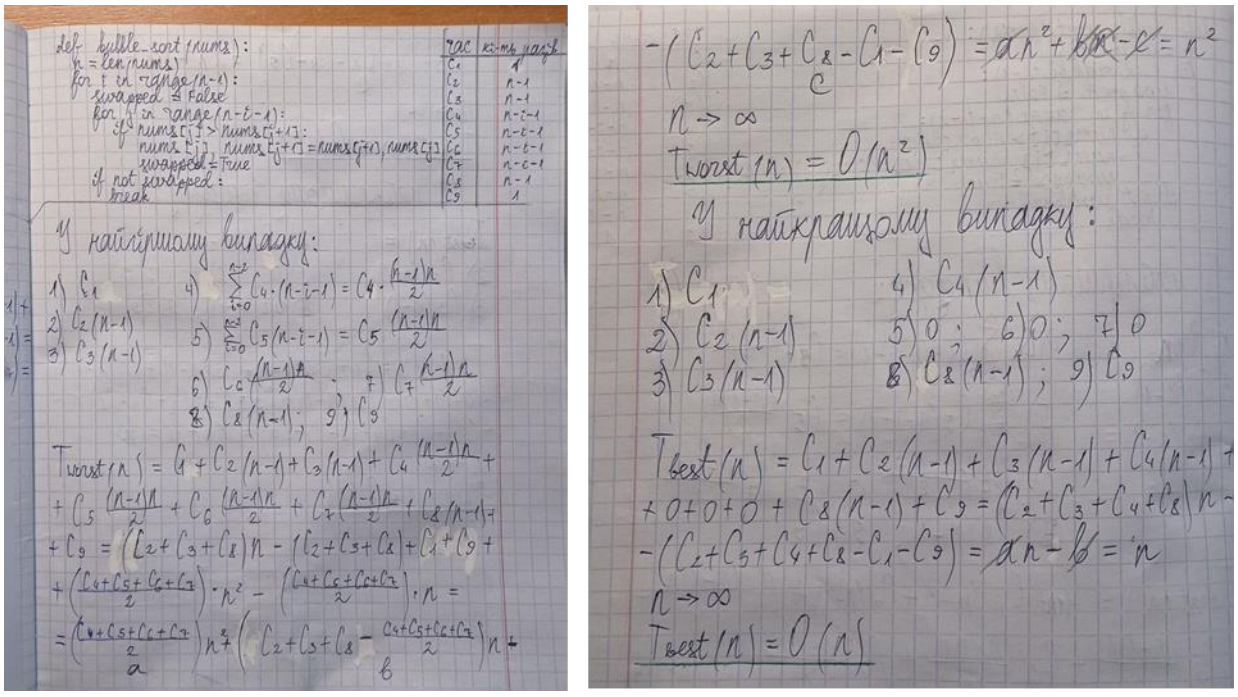


Рисунок 1 – Бульбашкове сортування

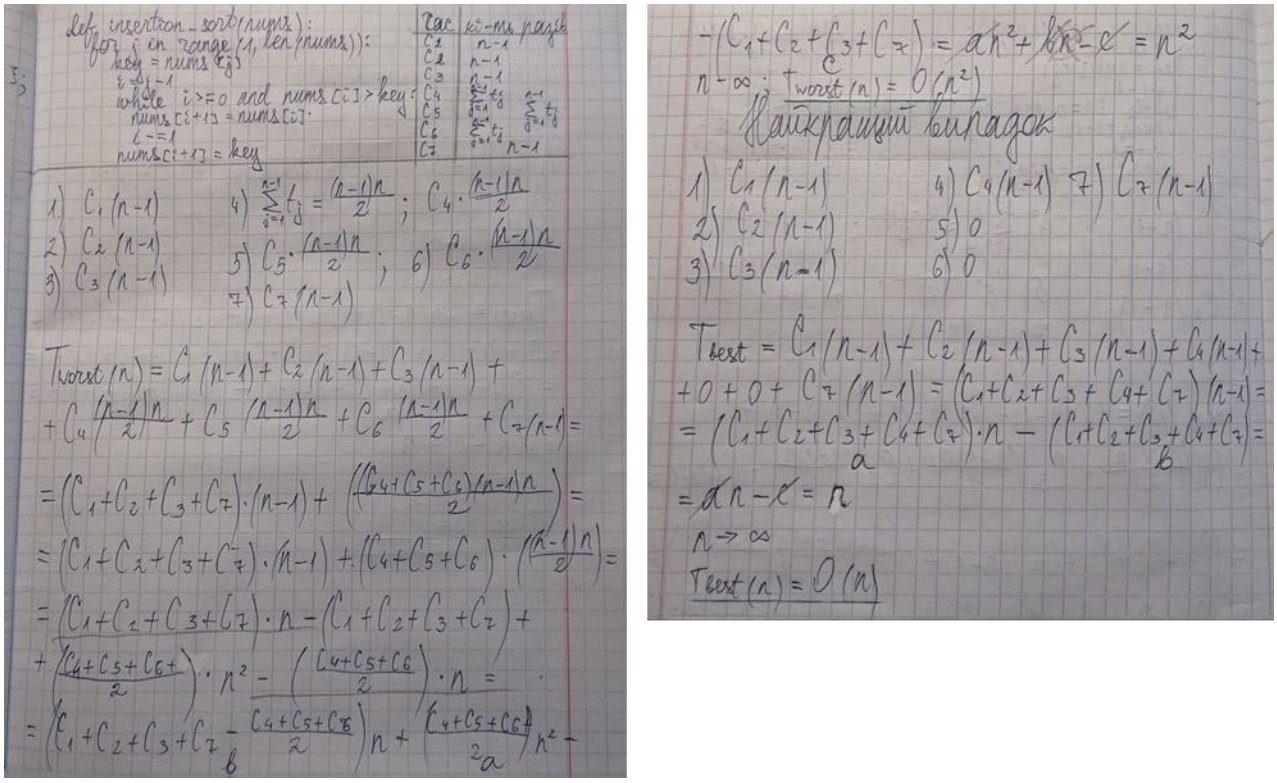


Рисунок 2 – Алгоритм вставлянням

Чому Bubble Sort менш ефективний, ніж Merge Sort?

- Bubble Sort завжди проходить по масиву кілька разів, роблячи велику кількість перестановок.
- Merge Sort працює по принципу "розділяй і володарюй" та виконує лише мінімальну кількість копіювань.
- Merge Sort має гарантовану складність $O(n \log n)$, яка набагато краща за $O(n^2)$ у Bubble Sort.

Завдання 2.

Оцінити асимптотичну складність алгоритму сортування зливанням, скориставшись основною теоремою рекурсії.

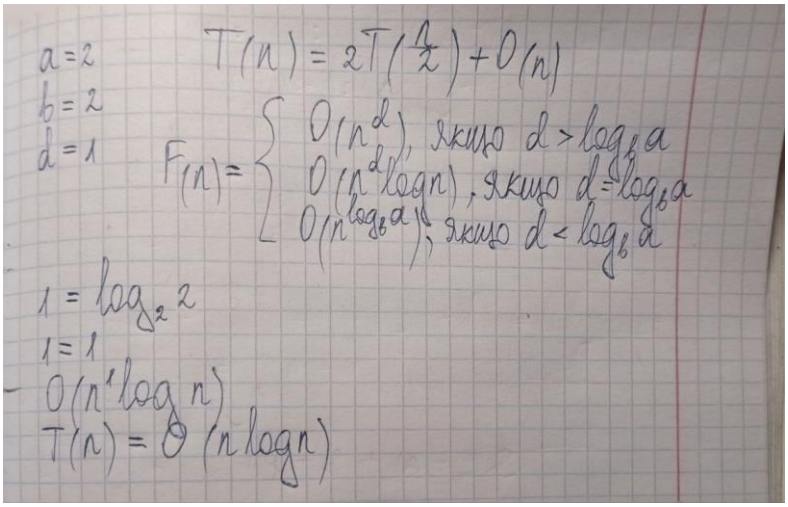
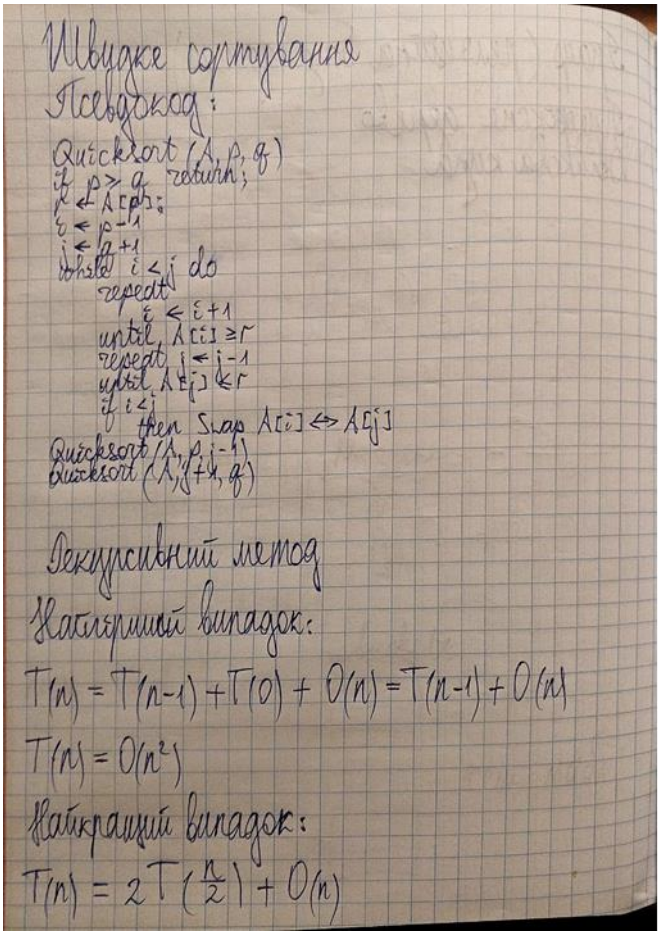


Рисунок 3 – Сортування зливанням

Завдання 3.

Записати алгоритм швидкого сортування. Оцінити асимптотичну складність алгоритму.



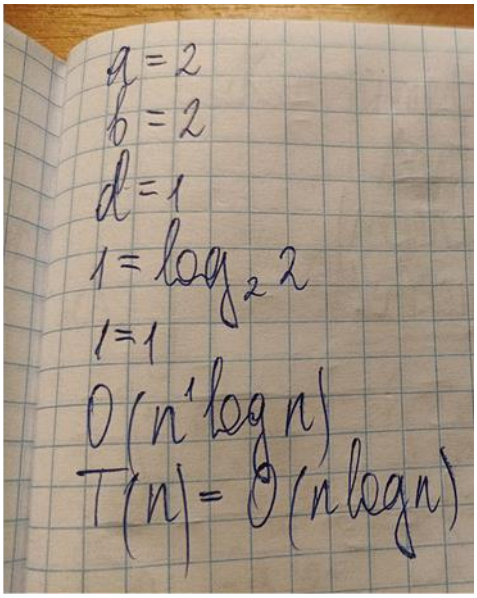


Рисунок 4 – Швидке сортування

1.3 Відповіді на контрольні питання

1. Що таке асимптотична складність алгоритму сортування і чому вона важлива для порівняння алгоритмів?

Асимптотична складність показує, скільки часу або пам'яті потребує алгоритм, коли розмір даних (n) стає дуже великим.

Чому це важливо:

Дає змогу порівнювати алгоритми між собою. Наприклад, один алгоритм працює швидше за інший при великих масивах, і це видно саме по складності.

2. Які алгоритми сортування мають квадратичну складність у найгіршому випадку? Поясніть, чому це може бути проблемою для великих обсягів даних.

Алгоритми з $O(n^2)$:

Бульбашкове сортування (Bubble Sort)

Сортування вибором (Selection Sort)

Сортування вставками (Insertion Sort)

Чому погано:

При великих масивах (наприклад, з мільйоном елементів) — ці алгоритми працюють дуже повільно. Це може займати години замість секунд.

3. В чому полягає перевага сортування злиттям над сортуванням вставками для великих наборів даних?

Сортування злиттям (Merge Sort) завжди має складність $O(n \log n)$ — швидко.

Сортування вставками (Insertion Sort) може працювати як $O(n^2)$ — повільно.

Висновок:

Сортування злиттям краще підходить для великих масивів, бо працює стабільно і швидше.

4. Які алгоритми сортування використовуються для сортування списків у стандартних бібліотеках мов програмування, таких як Python, Java або C++?

- Python: Timsort (поєднання злиття та вставок)
- Java: Timsort для об’єктів, QuickSort для чисел
- C++: Introsort (поєднання QuickSort, HeapSort і вставок)

Ці алгоритми підібрані так, щоб працювати швидко в реальних задачах.

5. Яка різниця між алгоритмами сортування злиттям і швидким сортуванням? У яких випадках краще використовувати кожен з цих алгоритмів?

	Merge Sort	**Quick Sort**
Складність	Завжди O(n log n)	Найгірше — $O(n^2)$, але часто O(n log n)
Швидкість	Стабільна	Швидший у середньому
Пам’ять	Більше використовує	Менше використовує
Стабільність	Так	Ні

- Merge Sort — якщо важлива стабільність (однакові елементи не міняють порядок).
- Quick Sort — якщо треба швидко відсортувати дані у пам’яті.

6. Які фактори слід враховувати при виборі алгоритму сортування для конкретної задачі?

Скільки даних?

Чи треба стабільність?

Чи обмежена пам’ять?

Чи дані вже трохи відсортовані?

Чи критичний найгірший випадок?

Наприклад:

- Якщо даних небагато — можна брати простий алгоритм.
- Якщо даних багато — краще брати Merge Sort або Quick Sort.