

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут ІКНІ
Кафедра ПЗ**

ЗВІТ

До лабораторної роботи № 7

З дисципліни: *“Алгоритми та структури даних”*

На тему: *“Порівняння методів сортування”*

Лектор:

доц. каф. ПЗ
Коротєєва Т.О.

Виконав:

ст. гр. ПЗ – 22
Ясногородський Н.В.

Прийняв:

асист. каф. ПЗ
Франко А.В.

« ____ » _____ 2022 р.
 Σ = _____

Львів – 2022

Тема роботи: Порівняння методів сортування.

Мета роботи: Порівняти вивчені раніше алгоритми сортування. Побудувати таблицю і графік швидкодії алгоритмів сортування. Зробити висновки щодо використання цих алгоритмів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Алгоритм сортування — це алгоритм, що розв'язує задачу сортування, тобто здійснює впорядкування лінійного списку (масиву) елементів.

На практиці елементи, що впорядковуються, рідко бувають просто числами. Набагато частіше, кожен такий елемент є записом (англійською «Record»). В кожному записі є ключ (англійською «Key»), за яким власне і здійснюється впорядкування, в той же час є й інша супутня інформація. Алгоритм сортування на практиці має бути реалізований так, щоб разом з ключами переміщати і супутню інформацію. Якщо кожен запис містить супутню інформацію великого об'єму, то з метою звести до мінімуму переписування великих об'ємів інформації, впорядкування відбувається не у самому масиві елементів, а в масиві вказівників на елементи.

Сам метод сортування не залежить від того, чи впорядковуються тільки числа, чи також і супутня інформація, тому при описі алгоритмів для простоти припускають, що елементи є числами.

Для алгоритму сортування основними характеристиками є обчислювальна та ємнісна складність. Крім цих двох характеристик, сортування поділяють на стабільні та нестабільні, з використанням додаткової інформації про елементи, чи без використання.

Для значної кількості алгоритмів середній і найгірший час впорядкування масиву з n елементів є $O(n^2)$, це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд (різниця між індексами елементів не перевищує деякого заданого числа). Такі алгоритми зазвичай є стабільними, хоча і неефективними для великих масивів.

Інший клас алгоритмів здійснює впорядкування за час $O(n \cdot \log(n))$. В цих алгоритмах використовується можливість обміну елементів, що знаходяться на будь-якій відстані один від одного.

Теорема про найкращий час сортування стверджує, що якщо алгоритм сортування в своїй роботі спирається тільки на операції порівняння двох об'єктів (\leq) і не враховує жодної додаткової інформації про елементи, то він не може впорядкувати масив елементів швидше ніж за $O(n \cdot \log(n))$ в найгіршому випадку.

ЗАВДАННЯ

Запустити на виконання кожен з написаних раніше програм щонайменше сім разів, отримати таким чином значення часу сортування масивів щонайменше семи різних розмірів кожним з шести вивчених методів. В якості набору значень розмірів масивів використати таку послідовність чисел:

- 1) 1024;
- 2) 4096;
- 3) 16384;
- 4) 65536;
- 5) 262144;
- 6) 1048576;
- 7) 4194304 (в разі якщо сортування відбувається довше, ніж 5 хвилин — перервати роботу програми та вважати час сортування нескінченно великим).

РЕЗУЛЬТАТИ

Running sorting with array of size 1024:
Bubble Sort: 0.08853sec
Selection Sort: 0.05135sec
Shell Sort: 0.00390sec
Quick Sort: 0.00276sec
Merge Sort: 0.00323sec
Count Sort: 0.00021sec

Running sorting with array of size 4096:
Bubble Sort: 1.15106sec
Selection Sort: 0.67418sec
Shell Sort: 0.02440sec
Quick Sort: 0.01739sec
Merge Sort: 0.01528sec
Count Sort: 0.00070sec

Running sorting with array of size 16384:
Bubble Sort: 18.11222sec
Selection Sort: 10.75722sec
Shell Sort: 0.16402sec
Quick Sort: 0.07189sec
Merge Sort: 0.07250sec
Count Sort: 0.00263sec

Running sorting with array of size 65336:
Bubble Sort: 287.63837sec
Selection Sort: 170.08465sec
Shell Sort: 0.34423sec
Quick Sort: 0.36625sec
Merge Sort: 0.34404sec
Count Sort: 0.01042sec

Running sorting with array of size 262144:
Bubble Sort: timeout 300sec
Selection Sort: timeout 300sec
Shell Sort: 11.85286sec
Quick Sort: 1.84188sec
Merge Sort: 1.65283sec
Count Sort: 0.04118sec

Running sorting with array of size 1048576:
Bubble Sort: timeout 300sec
Selection Sort: timeout 300sec
Shell Sort: 71.94778sec
Quick Sort: 9.30944sec
Merge Sort: 7.93281sec
Count Sort: 0.16384sec

Running sorting with array of size 4194304:
Bubble Sort: timeout 300sec
Selection Sort: timeout 300sec
Shell Sort: timeout 300sec
Quick Sort: 47.83058sec
Merge Sort: 38.12475sec
Count Sort: 0.65670sec



Лінійний графік залежності часу виконання і кількості елементів для кожного виду сортування



Стовпчикова діаграма залежності часу виконання і кількості елементів для кожного виду сортування

ВИСНОВКИ

Було порівняно попередньо вивчені алгоритми сортування. Тестові дані складаються з масивів розмірністю 1024, 4096, 16384, 65536, 262144 цілочисельних елементів:

- **Сортування бульбашки:** найпростіший алгоритм сортування, але водночас один із найгірших по часу, оскільки складність алгоритму в середньому і найгіршому випадку складає $O(n^2)$. Час виконання алгоритму компенсується його простотою.
- **Сортування вибіркою:** працює добре для невеликої к-ті елементів. Даний алгоритм виконує менше перестановок, ніж бульбашка. Складність в найгіршому випадку сягає $O(\frac{n * (n - 1)}{2})$. Отже, алгоритм є швидшим за бульбашку, проте значно повільніший за швидке сортування та сортування Шелла до певної к-ті елементів.
- **Сортування Шелла:** має складність в найгіршому випадку $O(n^{1.5})$. Різниця в швидкості з іншими алгоритмами помітна лише на великій к-ті елементів. Саме тоді Шелл показує себе значно гірше, ніж швидке сортування чи сортування злиттям
- **Швидку сортування:** має складність $O(n^2)$ в найгіршому випадку і $O(n * \log n)$ в середньому і найкращому випадку. На результат значно впливає вибір півотного числа. У порівнянні з іншими алгоритмами, показує чудові результати.
- **Сортування злиттям:** також є одним з найкращих алгоритмів у вибірці, оскільки він працює за $O(n * \log(n))$ у всіх випадках. Хоч він і потребує виділення додаткової пам'яті розміром з вхідний масив, на практиці сортування злиттям показало чудові показники.
- **Сортування підрахунком:** складність $O(n + k)$. Результати для малих чисел можна пояснити тим, що використовувалися числа з великим діапазоном. Цей алгоритм працює найкраще з малим діапазоном чисел. Попри це, якщо пам'ять не є проблемою, алгоритм покаже себе чудово навіть для дуже великих масивів.