Алгоритмічні мови та програмування

Лабораторна робота № 6

«Алгоритми сортування»

Теоретичні відомості та рекомендації:

Сортування масиву чи списку потребує визначення двох основних операцій – порівняння (визначення чи є обрана пара значень правильно відсортованою) та обміну (поміняти місцями два елемента). Велика кількість алгоритмів сортування спричинена тим, що тривіальні методи надто повільні на великих масивах і розробники постійно шукають оптимізації, що прискорюють алгоритм у специфічних випадках – частково відсортованих масивахЮ масивах з повторами, з відомими функціями розподілу чи при наявності апріорної інформації про множини, до яких належать елементи.

Для порівняння алгоритмів сортування використовують такі основні показники:

- стійкість – здатність не виконувати обмін однакових елементів
- природність – здатність ефективно обробляти вже відсортовані чи частково відсортовані масиви. Повністю природний алгоритм на відсортованому масиві робить n порівнянь та жодного обміну.
- асимптотика часу (швидкість, або обчислювальна складність) – оцінка кількості виконаних операцій, оцінена зверху (через О-велике) у гіршому, середньому та кращому випадках.
- пам’ять – кількість додстково витраченої на виконання алгоритму пам’яті віднесеної до розміру елементу що сортується.

Описи найбільш вживаних алгоритмів сортування.

Сортування бульбашкою / Bubble sort

Будемо йти по масиву зліва направо. Якщо поточний елемент перевищує наступний, міняємо їх місцями. Робимо так, поки масив не буде відсортований. Зауважимо, що після першої ітерації найбільший елемент буде знаходитися в кінці масиву, на правильному місці. Після двох ітерацій на правильному місці стоятимуть два найбільших елемента, і так далі. Очевидно, не більше ніж після n ітерацій масив буде відсортований. Таким чином, оцінка числа операцій у гіршому і середньому випадку - O (n2), в кращому випадку - O (n).

Шейкерне сортування / Shaker sort

(Також відоме як сортування перемішуванням або коктейльне сортування). Зауважимо, що сортування бульбашкою працює повільно на тестах, в яких маленькі елементи стоять в кінці (їх ще називають «черепахами»). Такий елемент на кожному кроці алгоритму буде зрушуватися всього на одну позицію вліво. Тому будемо одночасно йти не тільки зліва направо, а й справа наліво. Будемо підтримувати два вказівника begin і end, що позначають, який відрізок масиву ще не впорядкований. На черговій ітерації при досягненні end віднімаємо з нього одиницю і рухаємося справа наліво, аналогічно, при досягненні begin додаємо одиницю і рухаємося зліва направо. Оцінка числа операцій у алгоритму така ж, як і у сортування бульбашкою, проте реальний час роботи краще.

Сортування гребінцем / Comb sort

Ще одна модифікація сортування бульбашкою. Для того, щоб позбутися від «черепах», будемо переставляти елементи, які стоять на відстані. Зафіксуємо його і будемо йти зліва направо, порівнюючи елементи, які стоять на цій відстані, переставляючи їх, якщо необхідно. Очевидно, це дозволить «черепахам» швидко дістатися в початок масиву. Оптимально спочатку взяти відстань рівну довжині масиву, а далі ділити її на деякий коефіцієнт, що дорівнює приблизно 1.247. Коли відстань стане дорівнювати одиниці, виконується сортування бульбашкою. У кращому випадку Оцінка числа операцій дорівнює O (nlogn), в гіршому - O (n2).

Сортування вставками / Insertion sort

Створимо масив, в якому після завершення алгоритму буде лежати результат. Будемо по черзі вставляти елементи з вихідного масиву так, щоб елементи в масиві-результаті завжди були відсортовані. Оцінка числа операцій в середньому і найгіршому випадку - O(n2), в кращому - O(n). Для реалізації алгоритму зручно використати зв’язаний список з одним елементом даних та одним посиланням на наступний елемент. Тоді першим проходом по масиву для кожного елементу створюється елемент списку та вставляється у порібне місте, другим проходом по списку формується відсортований масив.

Сортування Шелла / Shellsort

Використовуємо ту ж ідею, що і сортування з гребінцем, і застосуємо до сортування вставками. Зафіксуємо деяку відстань. Тоді елементи масиву розіб'ються на класи - в один клас потрапляють елементи, відстань між якими кратно зафіксованим віддалі. Відсортуємо сортуванням вставками кожен клас. На відміну від сортування гребінцем, невідомий оптимальний набір відстаней. Існує досить багато послідовностей з різними оцінками. Найпростішою є послідовність Шелла - перший елемент дорівнює довжині масиву, кожен наступний вдвічі менше попереднього. Оцінка числа операцій в гіршому випадку - O (n2). Також застосовують послідовності Хіббарда, Седжвіка, Пратта та інші. Ви можете обрати послідовність за смаком. Послідовності проріджування необхідно розраховувати до розміру масива та застосовувати від більшого до меньшого.

Сортування гнома / Gnome sort

Алгоритм схожий на сортування вставками. Підтримуємо покажчик на поточний елемент, якщо він більше попереднього або він перший - зміщуємо покажчик на позицію вправо, інакше змінюємо поточний і попередній елементи місцями і зміщується вліво.

Сортування вибором / Selection sort

На черговій ітерації будемо знаходити мінімум в масиві після поточного елемента і міняти його з ним, якщо треба. Таким чином, після і-ої ітерації перші і елементів стоятимуть на своїх місцях. Оцінка числа операцій: O (n2) в кращому, середньому і найгіршому випадку. Потрібно відзначити, що це сортування можна реалізувати двома способами - зберігаючи мінімум і його індекс або просто переставляючи поточний елемент з даним, якщо вони стоять в неправильному порядку. Перший спосіб виявляється дещо швидшим.

Швидке сортування / Quicksort

Виберемо деякий опорний елемент. Після цього перекинемо всі елементи, менші його, ліворуч, а більші - направо. Рекурсивно розділяємо такіи самим чином ліву та праву частини масиву аж доки довжина підмасиву на стане меньшою 2. В результаті отримаємо відсортований масив, оскільки кожен елемент меньше опорного опинився лівіше кожного більшого. Оцінка числа операцій: O (nlogn) в середньому і кращому випадку, O (n2). Найгірша оцінка досягається при невдалому виборі опорного елемента.

Сортування злиттям / Merge sort

Сортування, засноване на парадигмі «розділяй і володарюй». Розділимо масив навпіл, рекурсивно відсортуємо частини, після чого виконаємо процедуру злиття: підтримуємо два вказівника, один на поточний елемент першої частини, другий - на поточний елемент другої частини. З цих двох елементів вибираємо мінімальний, вставляємо в результуючий масив і зрушуємо вказівник, що відповідає мінімуму. Злиття працює за O (n), рівнів всього logn, тому Оцінка числа операцій O (nlogn). Ефективно заздалегідь створити тимчасовий масив і передати його в якості аргументу функції. Це сортування рекурсивне, як і qsort, а тому можливий перехід на квадратичну асимптотику при невеликому числі елементів.

Сортування підрахунком / Counting sort

Створимо додатковий масив частот розміру r - l, де l - мінімальний, а r - максимальний елемент масиву. Після цього пройдемо по вихідному масиву і підрахуємо кількість входжень кожного елемента, записуючи її у частотний. Тепер можна пройти по масиву частот і виписати кожне число стільки разі, скільки потрібно у результуючий масив. Оцінка числа операцій - O(n + r - l).

Завдання:

1. Обрати за номером варіанту файл (l6vXX.txt, де XX – номер варіанту) з каталогу вхідних даних для лабораторних робіт №6.
2. Прочитати з файлу масив цілих беззнакових чисел.
3. Обрати за номером варіанту п’ять алгоритмів сортування.
4. Реалізувати кожен алгоритм у вигляді функції із такими характеристиками
    a. Приймає масив у вигляді двох аргументів - вказівника та довжини.
    b. Відсортований масив повертає у тій самій пам’яті, де лежав вихідний.
    c. Приймає також два вказівника на цілі числа, куди записує кількість витрачених операцій та виділеної додаткової пам’яті
    d. Повертає ціле число де 0 означає успішне виконання, а -1 – помилку.
    e. Ніякі операції вводу-виводу усередині функції сортування не проводити!
5. Створити додатково три масиви цілих чисел такого розміру, як вихідний, один заповнити натуральним рядом чисел у прямому, другий – у зворотному напрямку, третій наполовину нулями, наполовину одиницями.

6. Для кожного алгоритму сортування викликати відповідну функцію для прочитаного з файлу масиву (середній випадок), прямого (кращий) та оберненого (гірший випадок) ряду.
7. Записати виміряні значення кількості операцій для кожного алгортму та кожного типу масиву вихідних даних у таблицю – має бути записана асимптотика та пам’ять для середнього, кращого, гіршого випадкв та випаду з повторами.
8. На підставі презентованої таблиці порівняти обрані алгоритми за критеріями стійкості та природності.
9. Скласти висновки.

Варіанти:

№ варіанту	Алгоритми сортування				
1	Вставками	Швидке	Вибором	Злиттям	Шейкер
2	Вибором	Швидке	Гномяче	Вставками	Злиттям
3	Шейкер	Вставками	Гребінцем	Швидке	Злиттям
4	Бульбашка	Вставками	Швидке	Злиттям	Гномяче
5	Гребінцем	Злиттям	Вставками	Гномяче	Підрахунком
6	Злиттям	Бульбашка	Вставками	Гребінцем	Підрахунком
7	Гномяче	Бульбашка	Швидке	Вибором	Вставками
8	Швидке	Гномяче	Вибором	Шелла	Вставками
9	Злиттям	Підрахунком	Гребінцем	Гномяче	Швидке
10	Гребінцем	Шелла	Підрахунком	Злиттям	Шейкер
11	Вставками	Гномяче	Злиттям	Вибором	Гребінцем
12	Бульбашка	Злиттям	Гномяче	Швидке	Гребінцем
13	Злиттям	Вставками	Бульбашка	Гномяче	Швидке
14	Бульбашка	Швидке	Гребінцем	Гномяче	Вставками
15	Швидке	Шейкер	Бульбашка	Вставками	Підрахунком
16	Вибором	Вставками	Шейкер	Підрахунком	Шелла

17	Гребінцем	Бульбашка	Гномяче	Підрахунком	Вставками
18	Гномяче	Вставками	Швидке	Гребінцем	Вибором
19	Гребінцем	Гномяче	Підрахунком	Бульбашка	Вставками
20	Злиттям	Вибором	Гномяче	Гребінцем	Підрахунком
21	Бульбашка	Швидке	Шелла	Гребінцем	Підрахунком
22	Гномяче	Бульбашка	Шейкер	Злиттям	Гребінцем
23	Гномяче	Злиттям	Вставками	Вибором	Шелла
24	Шелла	Вставками	Гномяче	Підрахунком	Гребінцем
25	Шелла	Бульбашка	Швидке	Вибором	Гребінцем