Лабораторна робота №10. Швидке сортування

Мета: вивчити метод швидкого сортування. Розробити програмну реалізацію алгоритму швидкого сортування. Провести аналіз алгоритму.

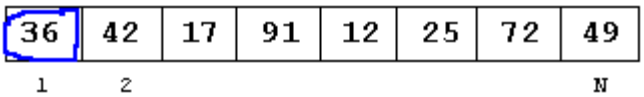
Теоретичні відомості

Метод відноситься до групи методів обміну, тобто до тієї ж, що і найповільніший - метод "бульбашки". Але корінна відмінність його полягає у організації обмінів між далекими за розташуванням елементами, що призводить до істотного прискорення роботи.

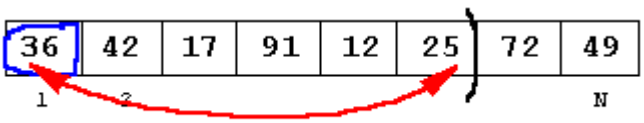
Ось викладення **основної ідеї методу**. Обирається деякий елемент, який назовемо медіаною. Всі елементи масиву на першому етапі розбирають на два підмасиви - елементів, більших від медіани та, відповідно, менших. На цьому, власне, перший етап закінчується. Далі процедура, що реалізує один етап, використовується рекурсивно, із застосуванням до обох частин масиву.

Робота продовжується до тих пір, поки у кожному з підмасивчиків, що на них розбивається масив, не стане лише по одному елементу. Це означатиме, що масив відсортовано.

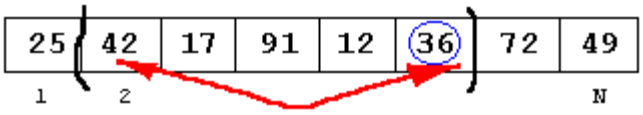
Для організації роботи на першому етапі робиться інтуїтивне припущення, що якщо робити обміни не сусідніх елементів, як у методі "бульбашки", а самих дальніх одне від одного, то це має прискорити роботу програми. Розгляньмо конкретний масив для прикладу:



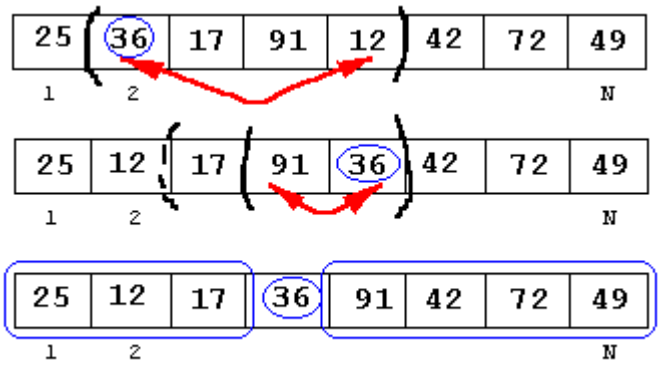
Візьмемо за медіану елемент **a[1]**. Порівняймо його з **останнім** елементом **a[N]**! Якщо їх впорядковано, то **a[1]** порівнюємо з **a[N-1], a[N-2], ...,** поки не зустрінеться елемент, менший від медіани. Тоді їх слід поміняти місцями:



Далі слід порівнювати медіану з наступним елементом, що стоїть за попереднім місцем розташування медіани (в даному випадку - **a[N-2]**). Якщо впорядкованість порушується, треба здійснити обмін:



І взагалі, після кожного обміну слід міняти "кінець" масиву, з яким порівнюється значення медіани. При цьому неаналізована частина завжди буде залишатися "всередині, "зліва" від неї будуть елементи, менші за медіану, а "справа" - більші.



Після того, як медіану порівняно з усіма елементами у масиві, вона, стрибаючи, стане на своє місце, тобто зліва від медіани стануть всі елементи, менші від неї, а справа - більші. Тоді надалі достатньо відсортувати лише праву та ліву частини масиву, бо положення медіани вже не мінятиметься.

Якщо реалізовано фрагмент, що виконує роботу першого етапу щодо всього масиву, треба застосувати його до лівої та правої частин. Dixi.

Аналіз. Для визначення місця медіани її порівнюють з усіма іншими елементами масива рівно по одному разу, тому на першому етапі порівнянь буде N-1. На другому етапі у кожній з підчастин обирається по медіані, для яких робиться загалом N-3 порівняння. На пересічному i-му етапі робитиметься (N+1-2i) порівнянь.

Якщо медіана вдало поділяє масив на дві рівні за довжиною частини, то етапів буде близько

$$\sim \log_2 N,$$

причому

$$C_{\min} = \sum_{i=0}^{[\log_2 N]+1} (N-i).$$

Перед оцінюванням кількості обмінів зауважимо, по-перше, що метод QuickSort є стійким: якщо за медіану береться перший елемент, то у вже відсортованому

масиві ніяких обмінів не відбудеться, але кількість порівнянь буде максимальною:

$$M_{\min} = 0; \quad C_{\max} = (N-1) + (N-2) + \ldots + 2 + 1 = \frac{N(N-1)}{2}.$$

Оцінювання середньостатистичних значень **M** та **C** є нелегкою задачею з огляду на необхідність використання апарату теорії ймовірностей, але обидві величини будуть порядку

$$\sim N \log_2 N.$$

На швидкодію методу істотно впливає, чи медіана реально є серединним елементом. Що ближчим до справжньої медіани виявляється взятий у цій якості елемент, то ближче до поданої оцінки буде реальна кількість операцій. Тому, якщо застосування програми буде носити масовий характер. а кількість елементів масиву є значною, варто на початку етапу оцінити можливе значення медіани, наприклад, взявши середнє арифметичне значення кількох "випадкових" елементів масиву. В принципі, для вибору медіани можна застосувати і інші методики.

Завдання до лабораторної роботи

Для виконання поставленої мети, необхідно вирішити наступні задачі:

1. Реалізувати алгоритм QuickSort.

2. Провести порівняльний аналіз алгоритмів сортування вставками,

алгоритму сортування злиттям, пірамідальне сортування та швидке сортування

3. Реалізувати інтерфейс користувача. Функціональні можливості

інтерфейсу:

– виводити запрошення на введення об’єму випадкової послідовності. Виводити генеровану випадкову послідовність у файл.

– виводити відсортовану послідовність у файл із тим же ім’ям, але іншим розширенням.

Контрольні питання

1. Викладіть основну ідею методу швидкого сортування.
2. Чи можна реалізувати метод QuickSort нерекурсивним чином?