**Київський національний університет імена Тараса Шевченка  
Факультет комп’ютерних наук та кібернетики**

**Алгоритми та складність  
Лабораторна робота №1  
Багатофазне сортування злиттям**

**Звіт**

**Підготував:**студент групи К-29  
Дацюк Віталій Олегович

**Київ-2019**

**1.Постановка завдання**

Іноді сортуємий список буває настільки великим, що він не влазить в оперативну пам’ять комп’ютера цілком. Нагадаємо, що хоч алгоритми які ми вивчаємо мають справу з упорядкуванням ключів, мається на увазі, що дані ключі пов’язані з цілими записами. В багатьох випадках довжина запису значно перевищує довжину ключа. Іноді довжина запису дуже велика і перестановка двох записів займає стільки часу, що аналіз ефективності алгоритму має враховувати як число порівнянь, так і число обмінів.

Іноді допустимо об’явити масив, розміру якого достатньо для розташування усіх потрібних даних, хоч і розміри цього масиву і значно перевищують доступний комп’ютеру об’єм пам’яті. Тоді операційна система користується віртуальною пам’яттю і варто враховувати ефективність її використання. Однак навіть і в цьому випадку об’єм обмінів між оперативною пам’яттю і дисками може бути значним. Навіть в ефективних алгоритмах сортування типу “Швидкого сортування” відношення між частинами розбиття і довжиною блока логічної пам’яті може трапитись таким, що призведе до великого числа перезапису блоків.

Для з’ясування ефективності алгоритмів сортування я підрахував число виконаних ними порівнянь. Однак об’єм роботи по читанню або запису на диск блоків віртуальної пам’яті може значно перевищувати трудоємкість логічних і арифметичних операцій. Ця робота виконується операційною системою і тому у нас немає реальних можливостей подіяти на її ефективність.

Є і інший спосіб: використати файли з прямим доступом і замінити безпосередньо звернення до масиву операціями пошуку в файлі для виходу в потрібну позицію з наступним читанням блоку. В результаті розмір використовуємої логічної пам’яті зменшується, а значить зменшується неконтролюєме навантаження на віртуальну пам’ять. Об’єм операцій вводу-виводу усе одно залишається значним – хоч ми керуємо ним самі або спихаємо це все на операційну систему.

В результаті більшість алгоритмів сортування на великих об’ємах даних виявляються непрактичними. Звернемось тепер до другої можливості: використаємо додаткові файли для зберігання даних та інструмент їх злиття.

Завдання: Відсортувати великий об’єм даних використовуючи додаткові файли та інструмент їх злиття.

**2.Опис алгоритму**

Оцінюю спочатку розумне число записів, які можна зберегти в оперативній пам’яті одночасно. Оголосимо масив, довжина length якого дорівнює цій величині. Цей масив буде використовуватися в двох етапах сортування. На першому кроці ми прочитаємо S записів і відсортуємо їх підходящим сортуванням(в нашому випадку MergeSort). Цей набір вже відсортованих записів перепишемо в файл source\_i.txt (де і – номер файлу). Далі прочитаємо ще S записів, відсортуємо їх і перепишемо в файл source\_i+1.txt. Цей процес продовжується, причому відсортовані блоки записів пишуться в кожний наступний файл.

Після того, як вхідний файл буде повністю розбитий на відсортовані відрізки, ми переходимо до другого кроку – злиття цих відрізків. Кожен з файлів має у собі деяку послідовність відсортованих відрізків, однак, як і в сортуванні злиттям, ми не можемо нічого сказати про порядок в різних записах.

Процес злиття буде аналогічний функції Merge з звичайного сортування злиттям,однак замість того, щоб переписувати записи в новий масив, ми будемо записувати їх в новий файл(result.txt).У кожному файлі ми порівнюємо перші записи, знаходимо найменший ключ та записуємо його в результуючий файл і так далі, доки дані не вичерпаютсья з усіх файлів.

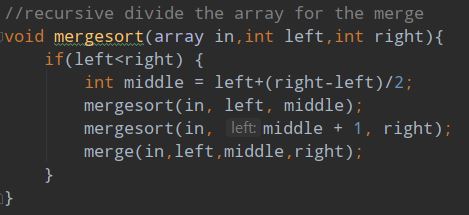
**3.Реалізація алгоритму**

“Злиття”

****

Даний алгоритм сортує вхідний масив злиттям, поділивши його попередньо на дві схожих за розмірами частини.

“Рекурсивне сортування злиттям”



Дана функція є рекурсивна і спрямована на поділ вхідного масиву даних на дві частини, для яких по черзі застосовується дана функція, в кінці якої масив зливається в один.

**4.Особливості алгоритму**

* Дозволяє використовувати будь-яке число N допоміжних послідовностей
* При злитті використовується (N-1) допоміжна послідовність, коли результат записується в вільну (N-ну) послідовність
* Злиття проводиться до тих пір поки всі послідовності не порожні

**Література:**

* <http://www.algolib.narod.ru/Sort/PolyPhaseMerge.html>
* <https://sites.google.com/site/algorithmsandmath/algoritmi-sortuvanna/sortuvanna-zlittam>
* **“The Art of Computer Programming, vol.3. Sorting and Searching” 2-ed Donald E. Knuth**