**Київський національний університет імені Тараса Шевченка**

**факультет комп’ютерних наук та кібернетики**

**кафедра інформаційних систем**

**Лабораторна робота № 1**

**“Зовнішнє k-фазне сортування”**

**Виконав студент 2 курсу**

**групи К-28**

**Руденко Юрій Олегович**

**2018**

• Зміст завдання та уточнення умови

Реалізувати k-фазне зовнішнє сортування даних.

• Уточнення умови та опис алгоритму: int num\_ways – кількість фаз

сортування, тобто максимальна кількість файлів, які будуть використовуватися при сортуванні, int run\_size – кількість елементів, які можуть бути записані в файлі, тобто к-сть елементів, що можуть поміститися у внутрішню пам’ять та бути відсортовані внутрішньо.

char input\_file[],char output\_file[] – імена вхідного та вихідного файлів.

Розділемо вихідний файл на num\_ways додаткових та запишемо в кожен з додаткових не більше run\_size елементів, при чому відсортованих, бо при зчитуванні партії елементів робимо її сортування в внутрішній пам’яті. Після цього використаємо k-фазне злиття додаткових файлів.

При цьому num\_ways = [n/run\_size] + 1(це мінімальна допустимо к-сть)

N к-сть елементів, що нам треба відсортувати

Merge k sorted arrays

Одне з ефективних рішень полягає в тому, щоб спочатку злити масиви в групи з 2 масивфі. Після першого злиття ми маємо k / 2 масивів. Ми знову об'єднуємо масиви в групи, тепер ми маємо k / 4 масивів. Ми продовжуємо робити це, допоки у нас не залишається один масив. Оцінка ефективності цього рішення була б O (nk Log k).

Бо:

Кожне злиття в першій ітерації потребує 2n операцій (злиття двох масивів розміром n). Оскільки всього k / 2 злиттів на першій ітерації, загальний час на першій ітерації буде O (nk). Наступна ітерація також вимагає O (nk) операцій. Буде всього O (Log k) ітерацій, тому оцінка складності O (nk Log k)

Ще одним ефективним рішенням є використання Min Heap. Це рішення має той же таку ж оцінку складності, тобто O (nk Log k). Але для масивів різних розмірів це рішення працює ефективніше.

Нижче наведено детальний алгоритм.

1. Створіть мінімальне бінарне дерево розміру k і вставте перший елемент з кожного масиву в дерево.

2. Повторіть наступні кроки n \* k разів.

     а) Отримати мінімальний елемент з дерева (мінімум завжди в корені) і зберенти його у вихідному файлі.

     б) Замініть корінь піраміди на наступний елемент з масиву, з якого видобутий елемент. Якщо в масиві немає елементів, то замінити корінь нескінченним. Після заміни кореня підримуємо структура дерева через heapify.

• Основні модулі програми:

• Використані джерела:

https://en.wikipedia.org/wiki/External\_sorting  
http://web.eecs.utk.edu/~leparker/Courses/CS302-Fall06/Notes/external-sorting2.html

**https://www.geeksforgeeks.org/external-sorting/**

**https://www.geeksforgeeks.org/merge-k-sorted-arrays/**