

Лабораторная работа № 1 по курсу дискретного анализа: сортировка за линейное время

Выполнил студент группы М8О-208Б-21 МАИ *Королев Илья*.

Условие

Кратко описывается задача:

1. Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

Тип ключа: почтовые индексы.

Тип значения: строки переменной длины (до 2048 символов).

2. Вариант задания 2-2. Сортировка подсчётом.

Метод решения

Решение реализовано согласно варианту, сортировкой подсчётом. Сначала создается динамический массив `counts`, в котором будут храниться счетчики вхождений каждого ключа. Затем происходит подсчет количества вхождений каждого ключа в исходном массиве и запись значений в массив `counts`. После этого в массиве `counts` находятся позиции, на которые должны быть помещены элементы в отсортированном массиве. Далее выделяется динамический массив `sorted arr`, в который будут записываться отсортированные значения. Затем элементы исходного массива записываются в отсортированный массив с учетом найденных позиций. Полученный отсортированный массив копируется обратно в исходный массив.

Описание программы

Все реализовано в одно файле. Рассмотрим функцию `Main()`

1. В начале функции выделяется динамическая память для массива `data_of_key_value`, в котором будут храниться все значения. Затем создается массив структур `KeyValuePair` размера `1e7`. Структура хранит в себе ключ и индексы начала и конца значения в массиве `data_of_key_value`. Это сделано для того, чтобы потребление памяти.
2. Далее в цикле считываются пары значений (`int`, `char`), где `int` - ключ, а `char` - значение. Считанные значения записываются в массив `data_of_key_value`. Значение `len` хранит текущую длину массива `data_of_key_value`.

3. Далее создается структура `one_str` и заполняется значениями:
 - ключом, считанным из входных данных;
 - левой границей диапазона значений в массиве `data_of_key_value`;
 - правой границей диапазона значений в массиве `data_of_key_value`.
4. Затем заполненная структура `one_str` записывается в массив `arr` и инкрементируется `idx`.
5. После цикла `while` вызывается функция `CountingSort`, которая сортирует массив `arr`. Затем циклом `for` выводятся отсортированные значения из массива `arr`, полученные из массива `data_of_key_value`.
6. В конце освобождается выделенная динамическая память для массивов `arr` и `data_of_key_value`.

Дневник отладки

1. В начале разработки были общие проблемы, связанные с тем как это реализовывать. Падало на 5м тесте с WA. Думал что проблема была в том, что я не обрабатываю строки с ключами но без значений: "123123 ". Потом проинтерпретировал условие "непустая строчка" понял, что скорее всего иметься ввиду, что может быть пуста строчка. То есть надо дополнительно обрабатывать пустые строчки. В этот момент я считывал с файла. Пытался разобраться как это сделать. Решил эту проблему вот так.

```

    bool isWhitespaceLine = true;
    for (char* p = buf; *p != '\0'; ++p) {
        if (*p != ' ' && *p != '\t' && *p != '\n' && *p != '\r') {
            isWhitespaceLine = false;
            break;
        }
    }

    if (isWhitespaceLine) {
        numLines--;
        delete [] value;
        continue;
    }

```

2. Была проблема с большими затратами памяти. Как я понимаю сейчас, это было связано с тем, что я криво работал с файлами и буфером, который нужен был для считывания количества строк. While я не мог так как в тот момент работал

со статической памятью. Каждый раз выделял по `char [2048]` для строки и передавал в структуру. Алгоритм падал на ML. В этот момент я решил, что буду: 1 работать терминальным вводом; 2 думать над тем, как не хранить в структуре значения. 1 оказалось гораздо проще, чем с файлом. 2 решил хранить в структуре только индекс начала и конца значения в `data_of_key_value`. Вывод реализовал так (позорный, но на тот момент по другому не мог)):

```
for (int i = 0; i < idx; ++i) {
    int temp = arr[i].key;
    int digits = 0;
    do{
        temp /= 10;
        digits++;
    }while (temp != 0);
    int zerosToAdd = 6 - digits;
    for (int i = 0; i < zerosToAdd; i++) {
        std::cout << "0";
    }
    //
    //printf("%06d\t", arr[i].key);
    std::cout << arr[i].key << "\t";
    for (int j = arr[i].value.left; j < arr[i].value.right; ++j) {
        std::cout << data_of_key_value[j];
    }
    std::cout << "\n";
}
```

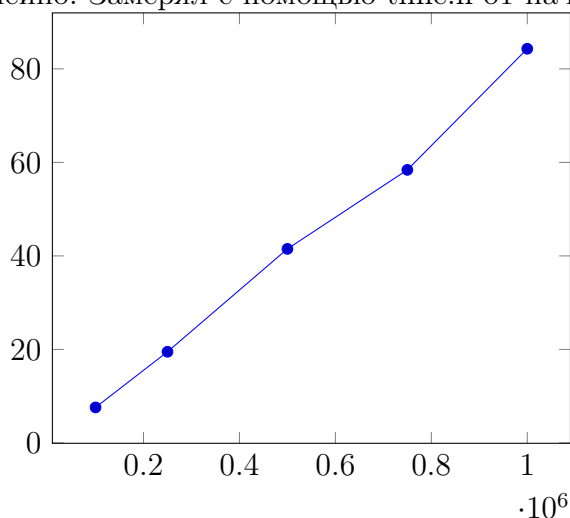
3. Все выше меня привело к тому, что я набрал около 1.09s и ошибку TL на 13м тесте. Потом исправил вывод на адекватный(я так думаю):

```
for (int i = 0; i < idx; ++i) {
    printf("%06d\t", arr[i].key);
    for (int j = arr[i].value.left; j < arr[i].value.right; ++j) {
        std::cout << data_of_key_value[j];
    }
    std::cout << "\n";
}
```

Почему-то это не ускорило. Тогда я решил переписать на си(поменять несколько строчек). Решил я так, потому что из с++ я так и не понял что могу взять. Все это есть в си. Успешно переписал. Время сократилось на 0,4 секунды. Тут та все и зашло.

Тест производительности

По оси x количество строк в файле, по оси y время в миллисекундах. Видим что время линейно. Замерял с помощью time.h от начала main и до return 0.



Недочёты

В целом, по ходу работы я расписал с какими трудностями я столкнулся. Сейчас же все работает.

Выводы

Я понял, что сортировка подсчетом используется в задачах, где необходимо отсортировать последовательность из небольшого диапазона значений. Например, когда необходимо отсортировать массив целых чисел, в котором каждое число находится в диапазоне от 1 до 1000000. Также с её помощью можно определить наиболее часто встречающиеся элементы в массиве.

Сложность по времени алгоритма сортировки подсчетом составляет $O(n+k)$, где n - количество элементов в массиве, k - диапазон значений элементов. При этом, необходимо выделить дополнительную память для массива подсчета, размер которого также зависит от диапазона значений элементов.

Лабораторная работа заставила меня чуть-чуть попотеть, так как за пределами лабораторных работ программирую в основном на питоне. Плюс ко всему нельзя юзать ничего, что есть в c++, а так как опыта в разработке на плюсах мало(его нет), то встал вопрос: что есть что? Мой код также успешно прошел на задачу 2-1. Надо было чуть-чуть поменять объем, выделяемый на значения (до 64).

В целом я был рад проделать данную лабораторную работу, хотя когда не заходило на 5м тесте, ~~было не очень приятно~~ хотелось перестать выполнять данную лр.