Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: Н. И. Лохматов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата: 22.03.2023
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №1

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

Вариант сортировки: Поразрядная сортировка.

Вариант ключа: `Телефонные номера, с кодами стран и городов в формате +<код страны> <код города> телефон.`

Вариант значения: `Числа от 0 до` $2^{64} - 1$.

1 Описание

Основная идея поразрядной сортировки заключается в том, чтобы поразрядно сравнивать некие объекты (изначально, целые числа, но в памяти компьютера любая информация записывается целыми числами), запись которых можно поделить на «разряды», содержащие сравнимые значения, например, строки, и вообще любые данные, представленные в виде набора байтов. Сравнение может происходит в сторону менее значащих цифр (LSD) или более значащих цифр (MSD). Далее нужно сортировать эти цифры любой устойчивой сортировкой.

Основная идея сортировки подсчетом заключается в том, чтобы для каждого входного элемента x определить количество элементов, которые меньше x.

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру KV, в которой будем хранить ключ и значение, а также флаг, который нам позже понадобится. Ключ и значение будут представлять из себя значение типа $unsignedlonglong$. Запись производится из файла. Пробежимся циклом $while$ по файлу. Внутри цикла пробегаеимя по ключам, обходя стороной знаки $+$ и $-$. Собираем новую строку, а потом переводим её в число с помощью $stol$. В переменную $flag$ записываем количество нулей после $+$ и до первого ненулевого символа. Считаем максимальную длину ключа. Записываем ключ-число, значение и флаг в вектор, который реализовали отдельно. Если вектор получился пустым, прекращаем выполнение программы. Далее создаём вектор, в который будем записывать отсортированные значения и передаём его вместе с исходным вектором, а также максимальной длиной ключа в функцию поразрядной сортировки. Непосредственно в самой функции создаём вспомогательный вектор, а также переменные a и b, которые нужны будут для отделения разрядов у числа. Проходимся по ключам в исходном векторе и отделяем разряды, попутно находя максимальный. Теперь нам нужно их отсортировать. Для сортировки разрядов мною была выбрана сортировка подсчётом. Передаём в неё исходный вектор, вектор для результата, максимальную цифру, а также a и b. В функции сортировки подсчётом создаём вспомогательный вектор длиной, на единицу больше максимальной цифры из данного разряда, а после заполняем этот вектор нулями. Проходимся по цифрам данного разряда в начальном векторе слева направо. К элементу вектора B, находящемуся на позиции, равной этой цифре, прибавляем единицу. Далее пробежимся по вектору B, делая что-то вроде префиксной суммы: к 1 элементу прибавляем 0 элемент, ко второму - первый и так далее. Теперь пройдёмся по элементам начального вектора, но уже справа налево (для сохранения свойства устойчивости), будем заполнять вектор результата. Создадим вспомогательный индекс, куда будем записывать данные разряды, но уже вектора B, уменьшенные на единицу. В вектор результата на позицию, равную индексу, записываем значение, стоящее на позиции вектора B, но уже нашего начального вектора. Уменьшаем значение на данной позиции у вектора B. Мы отсортировали вектор по ключам по последнему (самому маленькому) разряду. Повторим сортировку количество раз, равное длине максимального ключа. Когда ключи отсортированы по всем разрядам, необходимол вывести результат в консоль. Проходимся по вектору результата, выводя данные, не забывая о флаге, показывающем, сколько нулей нужно дописать в начале. Также добавляем знаки $+$ и $-$.

Оценим сложность нашей сортировки. Пусть n - количество строк, подающихся на вход, m - максимальная длина ключа, k - максимальное число разряда. Тогда сложность - $\Theta(m * (n + k))$.

sort.cpp	
void CountingSort(TVec<KV> &input, unsigned long long maxNum, long a, long b, TVec<KV> &result)	Сортировка подсчётом для сортировки разрядов.
void RadixSort(TVec<KV> &elems, size_t maxLength, TVec<KV> &result)	Поразрядная сортировка.
struct KV	Структура, которую будем помещать в вектор.

Структура $< KV >$:

```
1   struct KV {
2       KV() = default;
3       KV(unsigned long long key, unsigned long long value, int flag): key(key), value
            (value), flag(flag) {}
4       unsigned long long key;
5       unsigned long long value;
6       int flag;
7   };
```

Проходим по файлу, обрабатываем ключи и записываем необходимые значения в вектор:

```
1   while (cin >> key >> value) {
2       string a;
3       int flag = 0;
4       bool check = true;
5
6       for (int i = 0; i <= key.length() - 1; i++) {
7           if (key[i] != '+' && key[i] != '-') {
8               a += key[i];
9
10              if (key[i] == '0' && check) {
11                  flag++;
12              } else {
13                  check = false;
14              }
15          }
16      }
17
18      unsigned long long newKey = stol(a);
19
20      elems.Push( KV(newKey, value, flag) );
21      maxLength = max(maxLength, key.length());
22  }
```

А здесь выводим получившийся вектор, добавляя необходимые символы:

```
    for (unsigned int i = 0; i < result.Size(); i++) {
        string str = to_string(result[i].key);

        for (int j = 0; j < result[i].flag; j++) {
            str = '0' + str;
        }

        int len = str.length();

        cout << '+'
        << str.substr(0, len - 10)
        << '-'
        << str.substr(len - 10, 3)
        << '-'
        << str.substr(len - 7, 7)
        << '\t'
        << result[i].value
        << '\n';
    }
```

3 Консоль

```
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ g++ -std=c++17
sort.cpp -o sort
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ cat tests/03.t
+873-552-5491615        16553146571953280423
+123-892-1443852        13725978830006833910
+305-428-6277557        14115730502652581578
+946-562-1832356        10435344994214288714
+727-463-5547019        15098790682681338015
+839-821-1198590        10653330933284444965
+188-600-0103752        17935578389373855645
+072-077-4659125        10362624390623909626
+450-687-5667568        3678154125565467079
+658-424-2591745        13981219535755493892
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./sort <tests/03.t
+072-077-4659125        10362624390623909626
+123-892-1443852        13725978830006833910
+188-600-0103752        17935578389373855645
+305-428-6277557        14115730502652581578
+450-687-5667568        3678154125565467079
+658-424-2591745        13981219535755493892
+727-463-5547019        15098790682681338015
+839-821-1198590        10653330933284444965
+873-552-5491615        16553146571953280423
+946-562-1832356        10435344994214288714
```

4 Тест производительности

Тест производительности представляет из себя следующее: сортировка вектора поразрядной сортировкой, реализованной мной, и STL-сортировкой. Входные данные представляют из себя файлы, размеры которых варьируются от 1 до 10000000 строк. Время замеряется только при работе функций сортировок.

```
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/02.t
Radix sort time: 6us
STL stable sort time: 1us
Koeff: 6.0
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/03.t
Radix sort time: 13us
STL stable sort time: 2us
Koeff: 6.1
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/04.t
Radix sort time: 96us
STL stable sort time: 12us
Koeff: 8.0
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/05.t
Radix sort time: 943us
STL stable sort time: 130us
Koeff: 7.33
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/06.t
Radix sort time: 21579us
STL stable sort time: 1647us
Koeff: 13.168
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/07.t
Radix sort time: 128123us
STL stable sort time: 22560us
Koeff: 5.15323
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/08.t
Radix sort time: 1082876us
STL stable sort time: 289259us
Koeff: 3.215099
separatrix@separatrix-VirtualBox:~/Рабочий стол/da labs/lab1$ ./benchmark <tests/09.t
Radix sort time: 10546895us
STL stable sort time: 3234572us
Koeff: 3.843179
```

Как видно, на всех тестах STL-сортировка выигрывает, но при увеличении числа элементов разница во времени уменьшается и поразрядная догоняет STL-сортировку.

Для большей наглядности я также вывожу коэффициент, показывающий, во сколько раз STL-сортировка бысрее.

Сложность STL-сортировки - $\Theta(n*log(n))$ (где n - количество элементов) [?]. Поразрядная сортировка работает за $\Theta(m*(n+k))$.

«Если $b = O(\log_2 n)$, как это часто бывает, и мы выбираем $r \approx \log_2 n$, то время работы алгоритма поразрядной сортировки равно $\Theta(n)$, что выглядит предпочтительнее среднего времени выполнения быстрой сортировки $\Theta(n \log_2 n)$. Однако в этих выражениях, записанных в Θ-обозначениях, разные постоянные множители. Несмотря на то, что для поразрядной сортировки n ключей может понадобиться меньше проходов, чем для их быстрой сортировки, каждый проход при поразрядной сортировке может длиться существенно дольше».

Константа у поразрядной сортировки довольно большая, но существует число n_0 такое, что все тесты, в которых количество элементов больше, чем n_0, будут сортироваться быстрее поразрядной сортировкой, чем STL-сортировкой.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я, во-первых, познакомился с двумя сортировками за линейное время: поразрядной и сортировкой подсчётом. Я не просто узнал о них в теории, но также сам их реализовал. Также я научился оценивать сложность алгоритмов, писать тесты для программ и даже реализовал свой вектор. Ещё примечательно, что это моя первая серьёзная программа, написанная на языке $C{+}{+}$. Уверен, опыт от проделанной работы ни раз пригодится мне в дальнейшем.

Список литературы

[1] *Поразрядная сортировка LSD (Radix Sort) - Habr.*
URL: https://habr.com/ru/post/484224/ (дата обращения: 12.03.2023).

[2] *Сортировка подсчётом - Википедия.*
URL: http://ru.wikipedia.org/wiki/Сортировка_подсчётом (дата обращения: 12.03.2023).