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: С. Я. Симонов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2020

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

Требуется написать реализацию алгоритма поразрядной сортировки. Сравнение производится поразрядно: сначала сравниваются значения одного крайнего разряда, и элементы группируются по результатам этого сравнения, затем сравниваются значения следующего разряда, соседнего, и элементы либо упорядочиваются по результатам сравнения значений этого разряда внутри образованных на предыдущем проходе групп, либо переупорядочиваются в целом, но сохраняя относительный порядок, достигнутый при предыдущей сортировке. Затем аналогично делается для следующего разряда, и так до конца [2].

Для реализации алгоритма поразрядной сортировки нужно использовать устойчивый алгоритм сортировки. В данной работе я использовал сортировку подсчётом. Идея этого алгоритма в том, чтобы для каждого элемента x предварительно подсчитать, сколько элементов входной последовательности меньше x, после чего записать x напрямую в выходной массив в соответствии с этим число [1].

2 Исходный код

В каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру KV , в которой будем хранить ключ и значение. Будем считывать значения циклом с помощью стандартной функции getline, после чего с помощью функции ValueToKey, указанной в таблице, будем преобразовывать номер телефона из строки в ключ типа unsigned long long. Параллельно будем запоминать максимальное значение ключа, чтобы использовать это значение для сортировки. После считывания с помощью функции VectorPushBack все ключи и значения заносятся в вектор. Для ускорения считывания мы используем std::ios_base::sync_with_stdio. Эта команда отключает синхронизацию iostream с stdio. Также используется cin.tie(NULL) для отключения привязки cin к cout. После считывания всех данных функция radixsort с помощью ранее полученного максимального ключа поразрядно сортирует все элементы по ключам. radixsort использует сортировку подсчётом ContingSort. В этой функции мы создаём вектор result со значениями типа KV , в который попадают все элементы после сортировки. Сначала создаётся массив temp размера 10 и заполняется нулями. Потом подсчитываются все элементы рассматриваемого разряда. Потом подсчитываем количество элементов, не превосходящих значение данного разряда, а затем отсортированные значения помещаются в вектор result, после чего эти значения становится значениями вектора v, который подавался на вход. И так повторяется до тех пор, пока не пройдём все разряды ранее полученного максимального ключа. В конце выводятся значения отсортированного вектора v.

main.cpp	
void v_create(vector<T> &v, size_t size)	Функция создания вектора размера size
void VectorPushBack(vector<T> &v, T val)	Сложение двух больших чисел
void v_delete(vector<T> &v)	Функция удаления вектора
void VectorSwap(T &v1,T &v2)	Функция присваивания вектору v2 значений вектора v1
void CountingSort(vector<T> &v, unsigned long long div)	Функция сортировки подсчётом
void radixsort(vector<T> &v, unsigned long long max)	Функция поразрядной сортировки
bool goodval(char a)	Функция, возвращающая true, если a является цифрой от 0 до 9
void ValueToKey(KV &data)	Функция, для перевода телефонного номера в значение типа unsigned long long

```
struct KV {
    unsigned long long key;
    char value[256];
};
template <class T>
struct vector {
T* body;
    size_t size;
    size_t CAP;
};
```

3 Консоль

```
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ make
g++ -std=c++11 -Wall -pedantic main.o -o lab1
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tests/test01.txt
+7-495-1123212 13207862122685464576
+375-123-1234567 7670388314707853312
+7-495-1123212 4588010303972900864
+375-123-1234567 12992997081104908288
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ ./lab1 < tests/test01.txt
+7-495-1123212 13207862122685464576
+7-495-1123212 4588010303972900864
+375-123-1234567 7670388314707853312
+375-123-1234567 12992997081104908288
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tests/test02.txt
+7-4 1
+375-1 2
+7 3
+375000000000 4
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ ./lab1 < tests/test02.txt
+7 3
+7-4 1
+375-1 2
+375000000000 4
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tests/test03.txt
+7-954-9874895 1
+7-954-9874895 2
+7-954-9874895 2
+7-954-9874895 3
+7-954-9874895 3
+7-954-9874895 3
+7-954-9874895 4
+7-954-9874895 4

+7-954-9874895 4
+7-954-9874895 4
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ ./lab1 < tests/test03.txt
+7-954-9874895 1
+7-954-9874895 2
+7-954-9874895 2
+7-954-9874895 3
```

+7-954-9874895 3
+7-954-9874895 3
+7-954-9874895 4
+7-954-9874895 4
+7-954-9874895 4
+7-954-9874895 4
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tests/test04.txt
+7-23-1314131231 213123
+321-23123-21312312 312312
+131-2313-21313131 3232
+321-123 32
+3 23123345
+13-23 453645647
+1123-2131-3123 342167
+123-12312313-1213 9823
+1 47671280321
+21-213-23 3123174790
+221-32132132 476889
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ ./lab1 < tests/test04.txt
+1 47671280321
+3 23123345
+13-23 453645647
+321-123 32
+21-213-23 3123174790
+221-32132132 476889
+1123-2131-3123 342167
+7-23-1314131231 213123
+123-12312313-1213 9823
+131-2313-21313131 3232
+321-23123-21312312 312312

4 Тест производительности

Для тестирования будем использовать генератор тестов, написанный на языке программирования Python. На вход программе будет подаваться 1 миллион строк, причём максимальным значением вводимого ключа будет максимальное значение типа unsigned long long, а максимальная длина значения будет 30 букв. Время работы программы будем замерять с помощью стандартной библиотеки chrono. В файл с названием tmp будет выводиться результат работы программы.

Время сортировки 1 миллиона строк:
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 5542 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 3263 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 4139 ms
Время сортировки 10 миллионов строк:
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 48436 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 55515 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 50659 ms
Теперь применим функцию std :: sort.
Время сортировки 1 миллиона строк:
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 6783 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 4304 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 5675 ms
Время сортировки 10 миллионов строк:
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 49768 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 52021 ms
serjant@DESKTOP-AGCG4T3:/mnt/c/labs/DA/lab1$ cat tmp | grep "The time"
The time: 49034 ms
Несмотря на то, что std :: sort в среднем работает за O(NlogN), а поразрядная сортировка за O(N), с входными данными типа unsigned long long данные реализации сортировок справляются за примерно одинаковое время. Скорее всего это произошло потому, что точная сложность поразрядной сортировки равна O(d(N + k)), где d -

количество разрядов, а k - максимальное значение ключа. Так как мы сортируем числа типа unsigned long long, причём тесты подобраны так, что в среднем сгенерированный ключ - это 15-значное число, то числа d и k являются слишком большими для того, чтобы поразрядная сортировка работала быстрее стандартной. Для ускорения сортировки следует использовать битовые операции, так как битовый сдвиг - это намного более простая и быстрая операция, чем деление на 10. Тогда сравнение будет производиться по двоичным разрядам путём выделения разряда с помощью двоичной маски.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я реализовал поразрядную сортировку, использующую сортировку подсчётом в качестве внутренней сортировки.

Также в ходе выполнения работы я научился пользоваться утилитой valgrind для обнаружения утечек памяти, а при анализе производительности алгоритма поразрядной сортировки я выяснил, что применение битовых операций и сравнения по двоичным разрядам может существенно уменьшить время сортировки.

Список литературы

[1] *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, КлиффордШтайн.* Алгоритмы: построение и анализ, 2-е издание. — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Поразрядная сортировка — Википедия.*

URL: https://ru.wikipedia.org/wiki/Поразрядная_сортировка (дата обращения: 01.11.2019).