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: И. М. Королев
Преподаватель: Н. С. Капралов
Группа: М8О-208Б

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 сотрировка.

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

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

1 Описание

Требуется написать реализацию алгоритма порязрядной сортировки. На ввод подаётся пара ключ-значение, каждый элемент из которых принимает значения от 0 до $2^{64} - 1$. Выходные данные состоят из тех же строк, что и входные, за исключением пустых и порядка следования.
Основная идея поразрядной сортировки заключается в том, что для каждого разряда числа от младшего к старшему выполняется сортировка подсчётом. [1]. Сортировка подсчётом находит для входного элемента x количество элементов, разряд которых меньше разряда элемента x. При большом количестве разрядов в числе поразрядная сортировка выполняется не по каждому разряду, а по блокам, каждый из которых содержит 1 байт от числа. То есть ключ, который принимает числа от 0 до $2^{64} - 1$, будет типа unsigned long long, который занимает 8 байт. В каждом байте будет содержаться число от 0 до 255. Сортировка подсчётом создаст массив на 256 элементов и будет заполнять его. После заполнения, будет выполнено прохождение по начальному массиву с конца до его начала. Будет находится количество чисел, которые должны стоять до рассматриваемого элемента по индексу от его значения в массиве, после чего, он будет поставлен на определённое место в массиве. После заполнения массива по байтам, массив, который содержит ключи и значения, будет отсортирован по первому байту. И так поразрядная сортировка будет вызывать сортировку подсчётом по байтам 8 раз. В итоге массив из ключей и значений будет отсортирован.

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новый класс $TPair$, в котором будем хранить ключ и значение. Так как этих пар много, то создадим класс $TVector$, который будет принимать эти пары. У него есть переменные $Size$, которая хранит размер вектора, и $Capasity$, которая хранит вместимость этого вектора. Если размер вектора становится больше вместимости, то выделяется новая память с большей вместимостью, в которую копируется вектор со значениями, а старая память освобождается. После ввода всех значений, выполняется поразрядная сортировка, которая с помощью сортировки подсчётом сортирует числа по отдельным байтам. Была также написана функция обмена местами двух векторов в качестве параметров. Это нужно, чтобы каждый раз не заполнять один вектор отсортированными значениями из второго.

```
template<class T1, class T2>
class TPair {
public:
  T1 Key; //
  T2 Value; //
  TPair();
  TPair(T1 k, T2 v);
  ~TPair();
};

template<class T>
class TVector{
private:
  size_t Capasity;
  size_t Size;
public:
  T *Data;
  TVector();
  TVector(size_t size);
  int Length(); //
  void PushBack(T elem); //
  void Clear(); //
  ~TVector();
}
```

sorts.hpp	
void CountingSort(NVector::TVector <NPair::TPair<unsigned long long, unsigned long long» & vector, NVector::TVector<NPair::TPair<unsigned long long, unsigned long long» & v, long long n, int pos)	Функция сортировки подсчётом.
void Swapping(NVector::TVector <NPair::TPair<unsigned long long, unsigned long long» & vector, NVector::TVector<NPair::TPair<unsigned long long, unsigned long long» & v, long long n, int pos)	Функция перестановки векторов, местами в качестве параметров в сортировке подсчётом.
void RadixSort(NVector::TVector <NPair::TPair<unsigned long long, unsigned long long» & vector, long long n)	Функция поразрядной сортировки
main.cpp	
int main()	Функция, которая запускает выполнение программы.

3 Консоль

```
harry@harry-VirtualBox:~/DA$ g++ lab1.cpp -o lab1
harry@harry-VirtualBox:~/DA$ cat test1
2 324324234
342 33348983
2323 23232342
0 34234234
132 2323223
2 454353453
98884 3432432
10 1000323
54 454542
432425436 909432
3 576
harry@harry-VirtualBox:~/DA$ ./lab1 <test1
0 34234234
2 324324234
2 454353453
3 576
10 1000323
54 454542
132 2323223
342 33348983
2323 23232342
98884 3432432
432425436 909432
```

4 Тест производительности

Тест производительности представляет из себя следующее: сортировка ключей с помощью алгоритма поразрядной сортировки сравнивается с сортировкой sorted(key, value). Учитывается время только сортировок, то есть время выполнения остальной программы не учитывается. Тест состоит из пар ключ-значение, которые принимают значение в диапазоне от 0 до $2^{64} - 1$. Было проведено 3 теста с количеством пар 1000, 100000 и 1000000 соответственно.

```
harry@harry-VirtualBox:~/DA$ g++ lab1.cpp -o lab1
harry@harry-VirtualBox:~/DA$ python3 testgen.py
Sort: 0.0009882450103759766 seconds
harry@harry-VirtualBox:~/DA$ cat 01.t | ./lab1
Sort time: 0.000360393 sec
harry@harry-VirtualBox:~/DA$ python3 testgen.py
Sort: 0.20357751846313477 seconds
harry@harry-VirtualBox:~/DA$ cat 01.t | ./lab1
Sort time: 0.0334298 sec harry@harry-VirtualBox:~/DA$ python3 testgen.py
Sort: 2.8298895359039307 seconds
harry@harry-VirtualBox:~/DA$ cat 01.t | ./lab1
Sort time: 0.321573 sec
```

Как видно, поразрядная сортировка выполнилась быстрее на всех тестах, чем сортировка sorted(key, value) на python. Также, поразрядная сортировка для больших чисел выполняется побайтово, потому что это ускоряет процесс сортировки, так как если у числа большое количество разрядов, то сортировка по каждому разряду будет дольше из-за увеличения количесва итераций в цикле сортировки. Тогда получается, что сложность поразрядной сортировки будет $O(\frac{b}{R} * (n + 2^R))$, где n - длина вектора, b - длина числа в битах, R - длина разряда в битах. Если выполнять сортировку поразрядно, то её сложность будет $O(d * n)$, где d - количество разрядов.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился создавать свои пространства имён, работать с выделением новой памяти и её освобождением, сортировать данные по ключам с помощью поразрядной сортировки побайтово, которая работает вместе с сортировкой подсчётом. Я узнал, что при выделении памяти может быть её утечка и это нужно контролировать, иначе могут возникнуть проблемы. Поэтому очень важно понимать, что такое может возникнуть и попытаться это предотвратить. Также было выяснено, что обмен данными между двумя векторами можно произвести посредством присвоения указателя одного вектора другому. Этот способ является более эффективным, чем прохождение по всему вектору и заполнение другого его значениями.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *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] Роберт Лафоре. *Объектно-ориентированное программирование в C++. Классика Computer Science, 4-е издание.* — Издательский дом «Питер», 2018. Перевод с английского: А. Кузнецов, М. Назаров, В. Шрага. — 928 с. (ISBN 978-5-596-00353-7 (рус.))

[3] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 28.09.2020).

[4] *Поразрядная сортировка — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Поразрядная_сортировка` (дата обращения: 28.09.2020).