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: В. Р. Орусский
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
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: Сортировка подсчётом.

Вариант ключа: `Числа от 0 до 65535.`

Вариант значения: `Строки переменной длины (до 2048 символов).`

1 Описание

Основная идея сортировки подсчётом - сортировка элементов с помощью сортировки их идентификатора (ключа). При сортировке используется вспомогательный массив, который по умолчанию заполнен нулями, имеет длину, равную значению максимального ключа. Далее в этом массиве будут храниться индексы, в которых должны находиться исходные элементы. Для стабильной сортировки, размещение элементов происходит начиная с конца.

Время выполнения: O(n + k)

Память: O(k)

Где, n - кол-во элементов в исходном массиве k - максимальное значение ключа в исходном массиве

2 Исходный код

Для работы с парой «ключ-значение», будем использовать «pair», реализованную в STL. Так как из условия задачи, нам известно, что значение ключа будет не более 65535, то для экономии памяти будем использовать тип данных «uint16_t», что аналогично типу «unsigned short». Для хранения нужных индексов будем использовать массив «count_keys».

```
1  template<class T>
2  void CountingSort(vector<T> &data) {
3    if (data.empty()) return;
4
5    uint16_t max_key = 0;
6    for (size_t i(0); i < data.size(); ++i) {
7      max_key = std::max(max_key, data[i].first);
8    }
9
10   vector<size_t> count_keys(max_key + 1);
11   for (size_t i(0); i < data.size(); ++i) {
12     ++count_keys[data[i].first];
13   }
14
15   for (size_t i(1); i < count_keys.size(); ++i) {
16     count_keys[i] += count_keys[i - 1];
17   }
18
19   vector<T> res(data.size());
20   for (int64_t i = data.size() - 1; i >= 0; --i) {
21     res[--count_keys[data[i].first]] = data[i];
22   }
23
24   for (size_t i(0); i < res.size(); ++i) {
25     cout << res[i] << '\n';
26   }
27 }
```

3 Консоль

```
slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ g++ full.cpp


slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ ./a.exe <tests/04.t
2       t
2       o
2       u9
2       y
3       m5
4       kv
4       7
5       g0
6       x
9       v
10      z
10      pn
11      u
14      z
14      k
15      eb
```

4 Тест производительности

Тест производительности представляет из себя следующее: сортировка элементов, представленных парами ключ-значение. Тип ключа: числа от 0 до 65535. Тип значения: строки переменной длины (до 2048 символов).

Сравнение происходит между алгоритком сортировки подсчётом и «stable _sort» из std. Для тестов будем использовать файлы на 50, 500, 10K, 100K строк. Длина самих строк максимально вариативна, но удовлетворяет условию в 2048 символов.

```
slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ ./a.exe <./tests/00.t
Counting sort time: 503
STL stable sort time: 6

slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ ./a.exe <./tests/01.t
Counting sort time: 924
STL stable sort time: 100

slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ ./a.exe <./tests/02.t
Counting sort time: 4740
STL stable sort time: 1059

slava@DESKTOP-9JJF73M MINGW64 /d/Slavik/Coding/C++/DiskretAnalysis/Lab#1 (master)
$ ./a.exe <./tests/03.t
Counting sort time: 1546
STL stable sort time: 228
```

Видно, что «stable _sort» работает в разы быстрее.. Почему? Хороший вопрос, один из моментов - медленная перемещение пары из одного массива в другой, это можно исправить с помощью «std::move», а других причин не знаю.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я изучил работу шаблонов, лямбда-функций в C++. Узнал о линейных сортировках, которые работают за линейное время, но только если, существует какая-либо численная идентификация объекта. Хочется отметить полезный навык поиска SegFault ошибок при написании данной ЛР.

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] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 16.12.2013).