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: Д.В. Семенов
Преподаватель: А.А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2018

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: Карманная сортировка.

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

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

1 Описание

Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа алгоритмом Карманной сортировки (Bucket Sort) за линейное время и вывод отсортированной последовательности.

Как сказано в [1]: «В случае Карманной Сортировки (bucket sort) предполагется, что входные данные подчиняются равномерному закону распределения. Время работы в среднем случае при этом оказывается равном O(n). При карманной сортировке предполагается, что входные числа генерируются случайным процессом и равномерно распределены в интервале $[0, 1)$.

Карманная сортировка разбивает интервал $[0, 1)$ на n одинаковых интервалов, или карманов (buckets), а затем распределяет n входных чисел. Поскольку последние равномерно распределены в интервале $[0, 1)$, мы предполагем, что в каждой из карманов попадает не очень много элементов. Чтобы получить выходную последовательность, нужно просто выполнить сортировку чисел в каждом кармане, а затем последовательно перечислить элементы каждого кармана.

При составлении кода карманной сортировки предполагается, что на вход подается массив , состоящий из n элементов, и что величина каждого принадлежащего массиву элемента $[i]$ удовлетворяет неравенству $0 =< A[i] < 1$. Для работы нам понадобится вспомогательный массив связанных списков (карманов) $B[0..n-1]$».

2 Исходный код

Основные этапы написания кода:

1. Считывание исходных данных
2. Разделение по карманам
3. Сортировка каждого отдельного кармана
4. Вывод результата

Мы храним ключ и значение в структуре KeyValue, реализованной в "KeyValue.h".

Во время считывание вызываем конструктор KeyValue() и кладем получившийся объект в вектор.

Для распределения всех чисел в полуинтервале $[0; 1)$, мы делим ключи на максимально возможное значения ключа + 10, чтобы не получить значение 1

```
#include "vector.cpp"
#include "KeyValue.h"
#include <cmath>
#include <limits.h>
#include <iostream>

int main(){
    std::cout.precision(20);
    long double key;
    std::string value;
    NVector::TVector<KeyValue> input;

  //reading structure
    while(std::cin >> key >> value){
        if(input.Push_back(KeyValue(key, value)) == false){
            std::cout << "ERROR push_back" << std::endl;
            return 0;
        }
    }

    const long double num_of_el = input.Size();

    NVector::TVector<NVector::TVector<KeyValue>> matrix;
    matrix.Resize(num_of_el);

    // making counting - filling matrix B
    for(long long i = 0; i < num_of_el; ++i){
        long double tmp = (long double)input[i].m_key / ((long double)ULLONG_MAX + 10);
        if(matrix[(long long)(( num_of_el * tmp ))].Push_back(input[i]) == false){
            std::cout << "ERROR push_back" << std::endl;
```

```
            return 0;
        }
    }

  //sorting every vector in matrix

    for(long long i = 0; i < matrix.Size(); i++){
        if(matrix[i].Size() != 0) {
            for(long long k = 1; k < matrix[i].Size(); k++){
            KeyValue tmp;
            tmp.m_key = matrix[i][k].m_key;
            tmp.m_value = matrix[i][k].m_value;
            long long j = k-1;
                while(j >= 0 && matrix[i][j].m_key > tmp.m_key){
                    matrix[i][j+1].m_key = matrix[i][j].m_key;
                    matrix[i][j+1].m_value = matrix[i][j].m_value;
                    matrix[i][j].m_key = tmp.m_key;
                    matrix[i][j].m_value = tmp.m_value;
                    j--;
                }
            }
        }
    }

  //printing result
    for(unsigned long long i = 0; i < num_of_el; i++){
        if(matrix[i].Size() != 0){
            for(unsigned long long j = 0; j < matrix[i].Size(); j++){
                std::cout << matrix[i][j].m_key << "\t" << matrix[i][j].m_value << std::
                    endl;
            }
        }
    }

}
```

3 Консоль

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_da/1lab_debug/testing$ sudo ./wrapper.sh
Execute tests/01.t
OK
Execute tests/02.t
OK
Execute tests/03.t
OK
```

Пример теста 01.t:

3689348814741910528 vKLwRKbNx
14757395258967642112 eaTYYzeyW
11068046444225732608 oXVGXjwKN
0 aBlXfuZsC
7378697629483821056 XFgYykwmA

Полученный ответ 01.a:

0 aBlXfuZsC
3689348814741910528 vKLwRKbNx
7378697629483821056 XFgYykwmA
11068046444225732608 oXVGXjwKN
14757395258967642112 eaTYYzeyW

4 Тест производительности

```
import random
import string
from decimal import Decimal

MAX_KEY_VAL = 18446744073709551615
MAX_VAL_LEN = 10

def generate_random_value():
    return "".join([random.choice(string.ascii_letters) for _ in range(1, MAX_VAL_LEN)
        ])

if __name__ == "__main__":
    for num in range(1, 2):
        values = list()
        output_filename = "tests/{:02d}.t".format(num)
        N = random.randint(5, 6)
        keyadd = MAX_KEY_VAL/N
        key = 0
        for _ in range( N ):
            value = generate_random_value()
            values.append( (key, value) )
            key = key + keyadd

        random.shuffle(values)

        with open( output_filename, 'w') as output:
            for item in values:
                output.write( "{0:.0f}\t{1:}\n".format( item[0], item[1] ) )

        output_filename = "tests/{:02d}.a".format( num )
        with open( output_filename, 'w') as output:
            values = sorted( values, key=lambda x: x[0] )
            for value in values:
                output.write( "{0:.0f}\t{1:}\n".format(value[0], value[1]) )
```

Первый результат - результат карманной сортировки. Второй - сортировки вставкой.

Кол-во тестов	Bucket Sort (s)	Insert Sort (s)
100	6.999999999999999895e-06	0.00023399999999999576
10000	0.0004550000000000000008	2.1180940000000001433
40000	0.0017530000000000000009	33.79475999999999658

Как мы видим, на равномерно распределенных последовательностях наша сортировка работает гораздо быстрее.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился понимать различные методы линейной сортировки, в особенности Карманную сортировку. Научился процессу разработки программ с использованием своих unit-тестов. Понял, что нужно обрабатывать все нежелательные моменты в коде, такие как недостаток памяти. Научился составлять benchmark'и.

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] — *Википедия.*
URL: `http://ru.wikipedia.org/wiki/Блочная_сортировка` (дата обращения: 29.09.2018).