Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Дискретный анализ»

Сортировки за линейное время

Студент: Эсмедляев Федор Романович
Группа: М8О–212Б–22
Вариант: 4-1
Преподаватель: Н.Д.Глушин
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024.

Условие

Вариант: 4-1

J. 4-1

Ограничение времени	3 секунды
Ограничение памяти	300Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.
Вариант задания определяется типом ключа (и соответствующим ему методом сортировки) и типом значения:
Поразрядная сортировка.
Тип ключа: даты в формате DD.MM.YYYY, например 1.1.1, 1.9.2009, 01.09.2009, 31.12.2009.

Тип значения: строки фиксированной длины 64 символа, во входных данных могут встретиться строки меньшей длины, при этом строка дополняется до 64-х нулевыми символами, которые не выводятся на экран.

Формат ввода

На каждой непустой строке входного файла располагается пара «ключ-значение», в которой ключ указан согласно заданию, затем следует знак табуляции и указано соответствующее значение.

Формат вывода

Выходные данные состоят из тех же строк, что и входные, за исключением пустых и порядка следования.

Пример

Ввод

```
1.1.1   n399tann9nnt3ttnaaan9nann93na9
01.02.2008      n399tann9nnt3ttnaaan9r
1.1.1   n399tann9nnt3ttnaaan9nann93na9
01.02.2008      n399tann9nnt3ttnaaan9r
```

Вывод

```
1.1.1   n399tann9nnt3ttnaaan9nann93na9
1.1.1   n399tann9nnt3ttnaaan9nann93na9
01.02.2008      n399tann9nnt3ttnaaan9r
01.02.2008      n399tann9nnt3ttnaaan9r
```

Метод решения

Поскольку с датами разбираться крайне неудобно, давайте все даты из формата DD.MM.YYYY переведем в другой формат, давайте каждую дату представлять, как количество дней, прошедшее от даты 0.0.0. Для этого пройдемся по ключу в строковом представлении и просто отдельно будем брать часть строки до точки т.е. обращаем внимание только на знак “.”.

Эту процедуру можно делать через методы по типу stoll, stoi, но это долго, поэтому просто будем идти посимвольно и умножать на 10. Получившиеся значения умножаем на 1, 30, 365 соответственно (по сути лучше умножать на числа с запасом по типу 1, 32, 370)

Далее просто реализуем поразрядную сортировку. Но чтобы наша сложность оставалась линейной, когда мы будем сортировать разряды, воспользуемся сортировкой подсчетом.

Сложность данного алгоритма: O(14N + 70) = O(N)

Память: O(3N + 10) = O(N)

Описание программы

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <fstream>

using std::cin;
using std::cout;

using ll = int;

void sort(std::vector<std::pair<ll, ll>>& A, ll& i)
{
    const int n = i;
    const int k = 10;

    std::vector<int> C(k, 0);
    std::vector<std::pair<ll, ll>> B(n);
    std::vector<int> mas{ 1, 10, 100, 1000, 10000, 100000, 1000000,
10000000 };
    ll step = 1;

    for (int i = 0; i < 7; i++)
    {
        C.assign(k, 0);

        for (int j = 0; j < n; j++)
        {
            ll d = (A[j].first / mas[i]) % 10;
            C[d]++;
        }


        for (int j = 1; j < k; j++)
        {
            C[j] += C[j - 1];
        }
```

```
        for (int j = n - 1; j >= 0; j--)
        {
            ll d = (A[j].first / mas[i]) % 10;
            B[C[d] - 1] = A[j];
            C[d]--;
        }

        std::swap(A, B);
    }
}

ll day, mon, year;

ll pars(const std::string& s)
{
    ll i = 0;
    ll kol = 0;
    std::vector<ll> mas(3, 0);

    while (i < s.size())
    {
        if (s[i] == '.')
            kol++;
        else
        {
            mas[kol] = mas[kol] * 10 + (s[i] - '0');
        }
        i++;
    }

    return (mas[0] + mas[1]*30 + mas[2]*365);
}

int main()
{
    cin.tie(NULL);
    cout.tie(NULL);
    std::ios_base::sync_with_stdio(false);

    std::string key = "", str = "", keyy;

    std::vector<std::pair<ll, ll>> m(1e6);
    std::vector<std::string> data;
    int i = 0;

    std::ifstream f("input.txt");

    while (
        f >> key
        //keyy != "34re324"
        //std::getline(cin, keyy)
        )
    {
        //std::getline(cin, keyy);
        //if (keyy.empty())
           // continue;

        std::pair<ll, ll> x;

        f >> str;

        x.second = data.size();
        data.push_back(key + '\t' + str);
        std::string s = "", s1 = "";
```

```
            if (key.empty() or str.empty() or key == "" or str == " ")
                continue;

            //if (key == "3")
              //  break;

            x.first = pars(key);

            //ll prom = x.days.size();

           /* for (int i = 0; i < 7 - prom; i++)
            {
                x.days = "0" + x.days;
            }*/

            m[i] = x;
            i++;
        }

        sort(m, i);

        std::ofstream out;
        out.open("output.txt");
        for (int j = 0; j < data.size(); j++)
        {
            out << data[m[j].second] << '\n';
        }
        out.close();

        return 0;
}
```

Дневник отладки

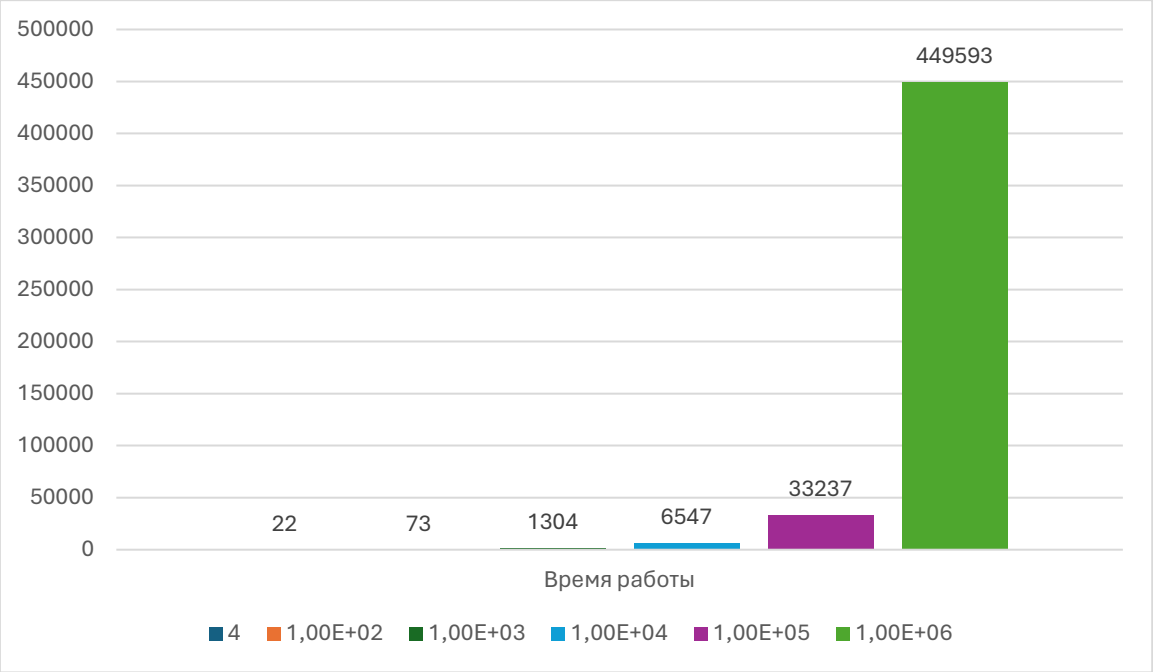
1) Реализация через строки и плохая реализация поразрядной сортировки, как итог сложность O(70N)

2) Переход к лучшему алгоритму поразрядной сортировки, получившаяся сложность O(14N + 70)

3) Оптимизация программы, включая парсинг даты(реализация без stoll), удаление push_back и выделение памяти наперед, сделан assign и std::swap в поразрядной сортировке.

4) Главное исправление – убирание передачи массива структур вида: число, строка. Вместо него теперь: число, число. Это нужно поскольку даже если я не использовал строку при повторном вызове и std::swap происходила передача строк – это очень долго. Соответственно добавил массив с индексами в изначальном массиве, который я прочитал.

Тест производительности



Вывод:

Я научился реализовывать поразрядную сортировку на C++, разобрался в особенностях оптимизации. В частности, какие структуры лучше использовать для экономии времени. Так же сравнил время работы от размера данных, но это не совсем честная оценка, поскольку она зависит так же от вычислительной мощности процессора.