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. Л. Яхин
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2025

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: `даты в формате DD.MM.YYYY`.

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

1 Описание

Требуется написать реализацию алгоритма поразрядной сортировки.

Поразрядная сортировка используется для наборов данных, в которых ключ можно поделить на "разряды"такие как, например, цифры в десятичном числе или небольшие группы бит в двоичном числе. Если такие разряды могут обладают достаточно небольшим набором значений, то для каждого из них удобно использовать сортировку подсчетом. Таким образом, применив сортировку подсчетом для каждого разряда можно отсортировать набор данных.

Тогда временная сложность алгоритма будет $O(n\omega)$, где

n — количество сортируемых элементов

ω — количество разрядов в одном ключе

2 Исходный код

Программа состоит из нескольких файлов.

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру $TKeyValuePair$, в которой будем хранить ключ и значение. Для хранения ключа тоже создадим структуру $TDate$ в которой реализуем поразрядное представление даты. Также напишем свою реализацию динамического массива $TVector$ для хранения данных.

main.cpp	
int main()	Точка входа в программу. Объявление контейнера, ввод данных, вызов сортировки, вывод отсортированного массива
sort.hpp	
void RadixSort(TVector< TKeyValuePair<TDate, unsigned long long> >& vector)	Поразрядная сортировка.
vector.hpp	
class TVector	Динамический массив
pair.hpp	
struct TKeyValuePair	Пара ключ-значение
date.hpp	
class TDate	Дата с поразрадным представлением

main.cpp

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4
5  #include "pair.hpp"
6  #include "date.hpp"
7  #include "vector.hpp"
8  #include "sort.hpp"
9
10
11 int main()
12 {
13   unsigned long long value;
14
15   std::string strBuf;
16
17   TKeyValuePair<TDate, unsigned long long> pairBuf;
18
19   TVector< TKeyValuePair<TDate, unsigned long long> > vector;
```

```

  std::ifstream fin("input.txt");
  std::ofstream fout("output.txt");

  while (fin >> strBuf >> value) {
    TDate dateBuf(strBuf);
    pairBuf.key = dateBuf;
    pairBuf.value = value;

    vector.PushBack(pairBuf);
  }

  RadixSort(vector);

  for (int i = 0; i < vector.Size(); ++i) {
    fout << vector[i].key.GetDateStr() << '\t' << vector[i].value << '\n';
  }

  fin.close();
  fout.close();

  return 0;
}
```

sort.hpp

```
#pragma once
const int COUNT_SIZE = 10;

void RadixSort(TVector< TKeyValuePair<TDate, unsigned long long> >& vector) {
  int prefSumCountArray[COUNT_SIZE]; // index for next element with this digit at the
      place
  TVector< TKeyValuePair<TDate, unsigned long long> > tmpVector(vector.Size());

  // key[] = {y, y, y, y, m, m, d, d}
  for (int digitOffset = DATE_SIZE - 1; digitOffset >= 0; --digitOffset) {

    for (int i = 0; i < COUNT_SIZE; ++i) { // init auxiliary arrays with zeros
      prefSumCountArray[i] = 0;
    }
    for (int i = 0; i < vector.Size(); ++i) { // Counting keys with same digit
      ++prefSumCountArray[vector[i].key[digitOffset] - '0'];
    }
    for (int i = 1; i < COUNT_SIZE; ++i) { // Prefix sum for index of beginning of a
        series of same keys
      prefSumCountArray[i] += prefSumCountArray[i - 1];
    }
    for (int i = vector.Size() - 1; i >= 0; --i) { // Forming result
      tmpVector[prefSumCountArray[vector[i].key[digitOffset] - '0'] - 1] = vector[i];
      --prefSumCountArray[vector[i].key[digitOffset] - '0'];
```

```
    }
    for (int i = 0; i < vector.Size(); ++i) { // Result
      vector[i] = tmpVector[i];
    }
  }
}
```

vector.hpp

```
template<typename T>
class TVector
{
private:
  int size;
  int capacity; // max_length before reallocate
  T* dataPtr;

public:
  // Constructors
  TVector();
  TVector(const int size);
  TVector(const int size, T initValue);

  TVector(const TVector& other);
  TVector& operator =(const TVector& other);

  TVector(TVector&& other);
  TVector& operator =(TVector&& other);

  // Destructor
  ~TVector();

  //
  int Size(); // Returns length
  bool Empty(); // length == 0
  T& operator [](const int index);
  void PushBack(const T& element); // Returns 0 in case of failure
  T PopBack(); // Removes last element and returns it

};
```

pair.hpp

```
template<typename TKey, typename TValue>
struct TKeyValuePair
{
  TKey key;
  TValue value;

  TKeyValuePair() = default;

```

```
  bool operator <(const TKeyValuePair& other) const;
};
```

date.hpp

```
class TDate
{
private:
  std::string strDate;
  char digits[DATE_SIZE];
public:
  TDate() = default;
  TDate(const std::string& str);
  char operator [](int index);
  bool operator <(TDate& other);

  std::string GetDateStr() const;
};
```

3 Консоль

```
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# g++ -std=c++17
-Wall -Wextra -I. -o solution main.cpp
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# cat input.txt
1.1.1   13207862122685464576
01.02.2008      7670388314707853312
1.1.1   4588010303972900864
01.02.2008      12992997081104908288
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# ./solution
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# cat output.txt
1.1.1   13207862122685464576
1.1.1   4588010303972900864
01.02.2008      7670388314707853312
01.02.2008      12992997081104908288
```

4 Тест производительности

Тест производительности представляет из себя следующее: поразрядная сортировка сравнивается с $std :: stable\_sort()$, чья сложность, согласно стандарту C++17, $O(n * \log(n))/O(n * (\log(n))^2)$ (В GCC реализация — merge sort). Тестовый набор данных содержит в себе 1000000 (10^6) строк — пар ключ-значение.

```
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# g++ -std=c++17
-Wall -Wextra -I. -o radix main.cpp
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# ./radix
RadixSort time: 1.12129
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# g++ -std=c++17
-Wall -Wextra -I. -o std_sort std_sort.cpp
root@77eb62ed9309:/workspaces/DA-Laboratory-Work/Lab_n1/solution# ./std_sort
std::stable_sort time: 2.21333
```

Как видим, поразрядная сортировка оказалась значительно боллее эффективным методом для такого набора данных.

5 Выводы

Поразрядная сортировка — эффективный метод упорядочивания элементов массива, однако доступный только на узком множестве возможных данных.

В ходе лабораторной работы был с нуля реализован динамический массив. Этот опыт может быть полезен для понимания внутреннего устройства стандартной библиотеки $C++$ и более грамотного её использования. Также был составлен отчет при помощи системы tex, которая позволяет автоматизировать многие процессы создания документов.

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: `https://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 09.04.2025).

[3] *Поразрядная сортировка — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Поразрядная_сортировка` (дата обращения: 09.04.2025).