Лабораторная работа № 1 по курсу дискретного анализа: сортировка за линейное время

Выполнил студент группы 08-307 МАИ *Лопатин Александр.*

Условие

Требуется разработать программу, осуществляющую ввод пар “ключ-значение” и их сортировку за линейное время:

1. На каждой непустой строке входного файла располагается пара “ключ-значение”, разделённые знаком табуляции. В выходных данных должны быть отсортированные строки исходной последовательности (за исключением пустых)

2. Вариант задания: 7-3

Ключи— Автомобильные номера в формате A 999 BC (используются буквы латинского алфавита).

Значения— Числа от 0 до $2^{64} - 1$.

Метод решения

1. Данные на вход программе подаются через перенаправление вывода из файла, и, как следствие, весьма удобно считывать циклом while(особенно это важно при неизвестном количестве строк).
   Когда будет считан символ EOF, цикл завершится.

2. Предусмотрена работа программы с неизвестным количеством входных данных.

3. Для работы алгоритма был введен вспомогательный массив: для каждой цифры j, которая может стоять на i-ом разряде, значением j-того элемента этого массива будет количество таких элементов пар ”ключ-значение”, что у ключа на i-ом разряде стоит цифра j.

4. Алгоритм сортировки принимает на вход ссылку на массив пар ”ключ-значение” и его размер. Результатом работы алгоритма будет отсортированный массив, содержащийся по начальной ссылке (т.е. алгоритм сортирует сам массив, не создавая его копию).

Описание программы

- **lab1.cpp**
  Основной файл, содержит в себе собственно функцию “main” и функцию сортировки “RadixSort”

- **lab1.h**
  Заголовок основного файла, в котором находится описание всех используемых структур и констант.

Дневник отладки

При создании следующей таблицы была использована история локального гит-репозитория.

Время	Коммит	Описание
11 21:22:25	init	Начало работы, есть только шаблоны файлов и функций
11 22:05:16	parsing	Заготовки под функцию парсинга, поиск необходимых методов и тестирование функции + 3 таких же коммита
11 22:16:57	оптимизация указателей	Изменение структуры ключа, замена строки ссылкой на элемент отдельного массива строк
12 13:31:19	убрал утечки памяти	Не освобождалась вся память, в самой структуре ключа я выделил память для указателя на строку, потом это значение перезаписывалось с ввода, а выделенная ранее память не освобождалась
12 13:55:02	доделать сортировку	Была проблема с логикой алгоритма сортировки, цикл завершался досрочно, либо значения не отсортировывались в нужном порядке + 3 коммита мелкие исправления
15 21:33:11	cut last symbol	Заметил, что обрезается последний символ строки-значения + 18 коммитов на то, чтобы понять, что необходимо создать механизм динамического расширения выделенной памяти
19 14:36:07	WORKING!!!!	Рабочая версия программы, чекер отправил OK, далее проходила незначительная оптимизация кода и кодстайла
22 11:39:58	string->char*	Узнал что такое стандартные контейнеры STL в C++, пришлось переводить все вхождения на массив char-ов
22 21:47:39	Conditional jump or move	Надо было учесть, что строки обязаны заканчиваться символом \0, для корректной обработки строк

Так же при проверке работы программы учитывалось время её исполнения (утилита time), осуществлялся контроль различных ошибок и утечек памяти (утилита valgrind) и отдельно для тестов применялась утилита memusage.

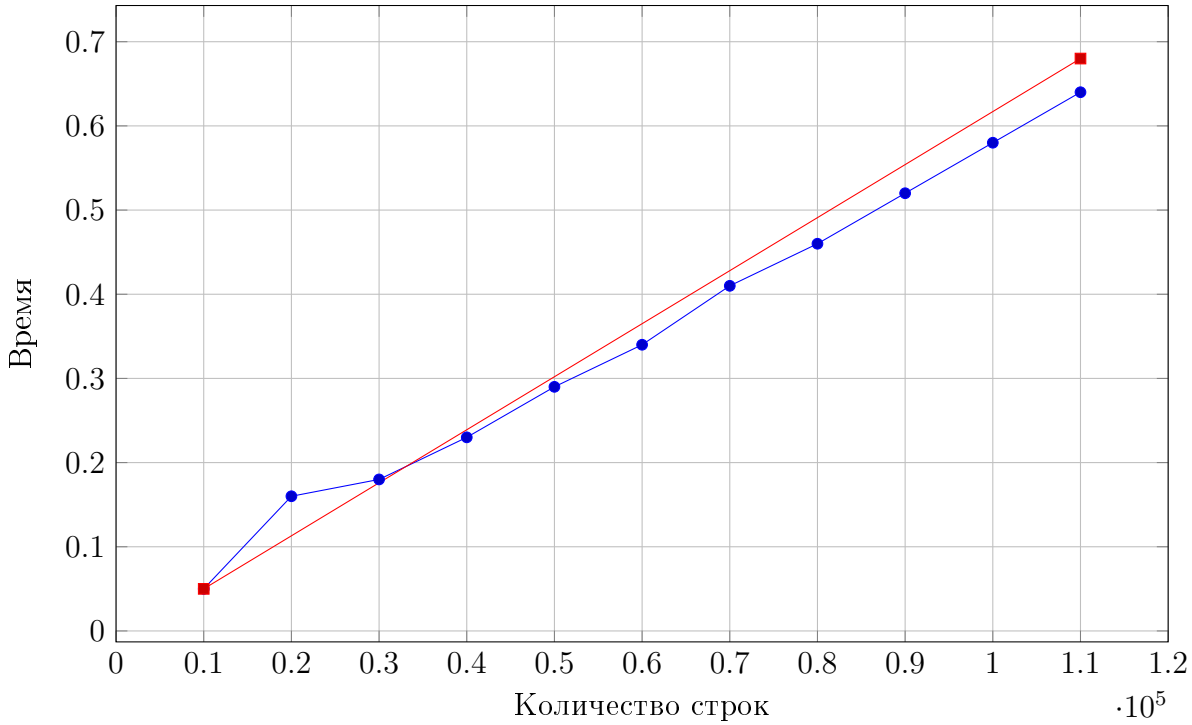
Тест производительности

Тесты создавались с помощью небольшой программы на языке Python:

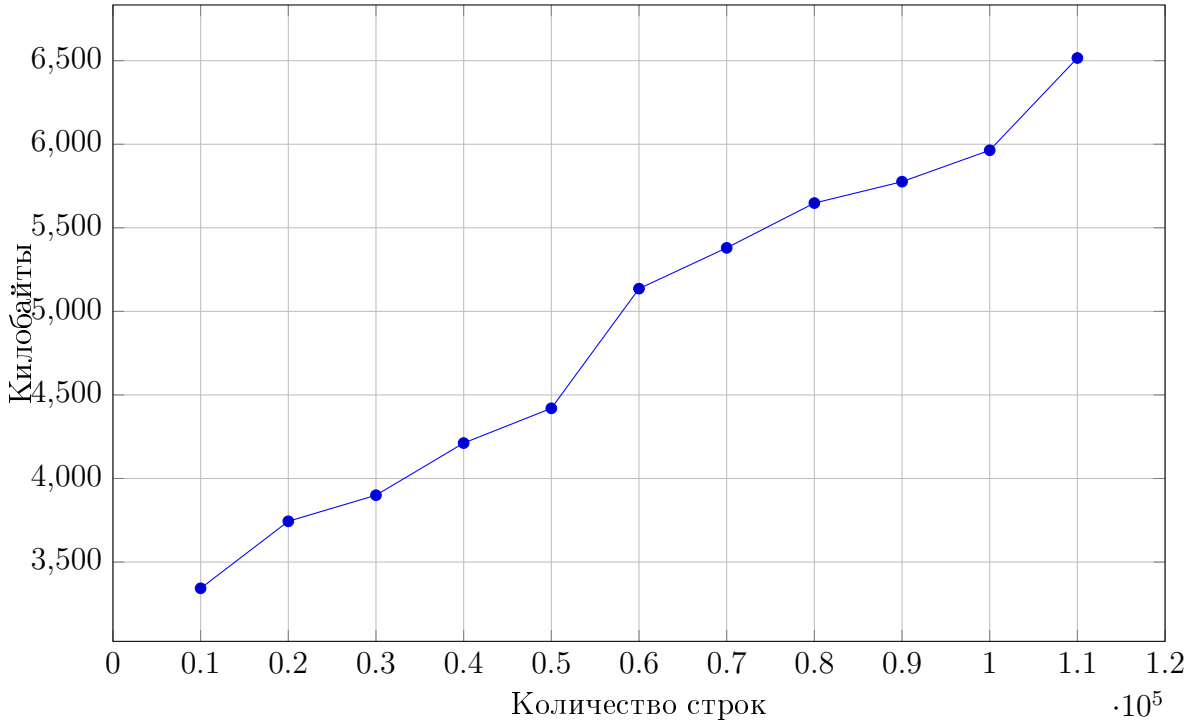
```
import random

for i in range(10000, 120000, 10000):
    file = open('tests/test'+str(i)+'.txt','w')
```

Размер файла	Имя файла и количество тысяч строк в нём
4,1M	100k
8,1M	200k
13M	300k
17M	400k
21M	500k
25M	600k
29M	700k
33M	800k
37M	900k

```
for j in range(1, i):
    file.write(str(random.randint(0,9))
        +chr(random.randint(65,90))
        +chr(random.randint(65,90))
        +chr(random.randint(65,90))
        +str(random.randint(0,9))
        +str(random.randint(0,9))
        +'\t'
        +str(random.randint(0,4294967295))
        +'\n')
file.close()
```



Итого, по графику результатов времени выполнения программы, её сложность близка к линейной.



Выводы

Данный алгоритм хорошо подойдет для сортировки объектов с ограниченным количеством разрядов (например, переменных типа int или, как в этом варианте, автомобильных номеров), но деградирует с линейной до квадратической при неограниченном (например, string).