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

Выполнил студент группы 08-208 МАИ *Ядров Артем*.

Условие

1. Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

2. Вариант задания определяется типом ключа (и соответствующим ему методом сортировки) и типом значения:

   (a) Поразрядная сортировка.

   (b) Тип ключа: телефонные номера, с кодами стран и городов в формате +<код страны> <код города> телефон.

   (c) 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.

Метод решения

Алгоритм поразрядной сортировки достаточно прост. Достаточно пройтись по всем разрядам справа налево (конечно, интуиция подсказывает, что начать лучше со старшего разряда, но в третьем издании книги Кормена "Алгоритмы. Построение и анализ"в главе 8.3 "Поразрядная сортировка"отлично показано, что начинать лучше с крайнего правого разряда) и применить устойчивую сортировку. Лично я использовал сортировку подсчетом, так как мы имеем ограниченное количество элементов (10 цифр).

Описание программы

Весь код содержится в файле **main.cpp**: класс **TObject**, в котором хранится ключ, разделенный на разряды (код страны, код города и непосредственно сам номер телефона), и значение. С целью экономии памяти я использовал char[] вместо std::string.

Дневник отладки

С самого начала я использовал Make для отправки своего решения. Однако при отладке непосредственно у себя на устройстве я использовал ключ -fsanitize=address, который потом на чекере использовал много памяти, из-за чего я превышал заданную память. Также была попытка использовать std::string, что достаточно сильно влияло на расход

памяти. В итоге я решил использовать ограниченный массив char[], так как заранее известны ограничения на ключ и значение.

Тест производительности

N	Time (seconds)
10000	0.035
100000	0.24
1000000	2.268

Недочёты

Из недочетов это то, что функция сортировки изменяет исходный массив, а не создает новый. Я боялся, что при создании нового достаточно большого массива я могу израсходовать память. Также непосредственно в самой функции много одинакового кода, который отличается лишь в паре строк. Можно было вынести этот код в отдельную функцию и на вход давать параметр, отвечающий за эти строки.

Выводы

Известно, что сортировка подсчетом эффективно лишь тогда, когда диапазон ключей достаточно мал. Однако, что делать, если диапазон ключей во много раз превышает количество элементов? Тогда на помощь приходит поразрядная сортировка. Пусть имеется n $d-$ *значных* чисел, в которых каждая цифра принимает одно из k возможных значений. Тогда алгоритм Radix-Sort(A, d) позволяет выполнить корректную сортировку этих чисел за время $\Theta(d(n+k))$, если устойчивая сортировка, используемая в алгоритме имеет время работы $\Theta(n+k)$. Сам алгоритм довольно прост: проходя по всем разрядам мы сортируем по массив по каждому разряду выбранной сортировкой.