Лабораторная работа №1 по курсу Дискретный Анализ. Сортировка за линейное время

Выполнил студент группы М8О-207Б-21 МАИ *Друхольский Александр.*

Условие

Кратко описывается задача:

1. 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.

2. Вариант задания: 4-1. 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. Сортировка: поразрядная

Метод решения

В основу алгоритма легла поразрядная сортировка. Данный вид сортировки оказался весьма удобным для ключа в формате даты, ведь все ограничения на эту сортировку соблюдены (целые неотрицательные числа). Суть сортировки в том, что ключ делится на разряды (единицы, десятки и так далее). Далее последовательность сортируется по первому разряду, в результате чего перегруппировывается. Далее работаем со следующим разрядом. В результате, сложность такого алгоритма будет $O(n)$, так как данная сортировка не требует сравнения значений между собой (мы используем массив из 10 ячеек и в соответствии с цифрой в разряде отправляем весь ключ в нужную ячейку с номером равным этой цифре, потом переводим этот массив в одномерный массив, сохраняя порядок попадания ключей в каждую ячейку). Однако такая сортировка требует дополнительной памяти (в два раза больше входящих данных), которая представлена массивом, упомянутым выше. В каждой ячейке такого массива может находиться до n ключей. n - количество входных данных (строки). Сортировка является стабильной, то есть порядок одинаковых ключей сохранится, так как мы учитываем порядок попадания ключей в каждую из 10-ти ячеек дополнительного массива.

Описание программы

Основные моменты:

1. Структура *Pair*, которая соответствует каждой входной строке.

    1.1 *int day* содержит целое число ДЕНЬ

1.2 *int month* содержит целое число МЕСЯЦ

1.3 *int year* содержит целое число ГОД

1.4 *char* key* содержит строку КЛЮЧ

1.5 *char* value* содержит строку ЗНАЧЕНИЕ

2. Основные функции

1.1 *Pair* ParseDate(Pair *data, long long size)* разбивает исходные ключи на три значения: день, месяц и год. Возвращает массив Pair с заполненными полями *day*, *month*, *year*.

1.2 *Pair* RadixSort(Pair* data, long long datasize)* выполняет непосредственно сортировку уже обработанного массива. По каждому из трёх значений сортируем поразрядно. Сначала сортируем по значению *day*, далее по *month* и *year*. В функции предусматриваем рациональное выделения памяти под двумерный массив *Pair ** index*, в который мы и используем для поразрядной сортировки.

3. Дополнительные функции

1.1 *int IsSymbol(char c)* нужна для проверки символа на то, является он цифрой или точкой (необходима для парсинга).

1.2 *void MyStrcpy(char *dst, const char *src)* является аналогом встроенной в библиотеку *string.h* функции *strcpy*.

1.3 *size_t MyStrlen(const char *s)* определяет длину строки.

4. Функция *int main(void)*, которая состоит из ввода данных, вызова сортирующих и побочных функций, подсчёта времени и вывода результата.

Все структуры сделаны динамическими, чтобы удовлетворять условию по ограничению памяти. В версии программы с замером времени убран вывод результата.

Дневник отладки

WA - 8 Проблема была в алгоритме. Изначально я сортировал в порядке: год, месяц, день, а надо было наоборот.

RE - 13 Исправил выделение памяти функцией *realloc*. Изначально увеличивал *capacity* в два раза, что на больших данных ломала код. Сделал увеличение меньше, ошибка пропала.

ML - 13 Получил эту ошибку, пока пытался починить *runtime error*, переборщил с использование long int и long long.

CE Запустил не на том компиляторе.

Тест производительности

Замерялось время выполнения, включая ввод данных.

№	Кол-во строк	Время, с
1	10	0.000071
2	100	0.000150
3	1000	0.001018
4	10000	0.011547
5	100000	0.057756
6	1000000	0.651274

Каждый последующий тест больше предыдущего в 10 раз, что делает результат замера более наглядным.

Выводы

Сортировка подсчётом показалась мне очень хорошим средством сэкономить время при работе с данными, на которые наложены подходящие ограничения. Алгоритм, по моему мнению, не является сложным. Единственное неудобство связано с особенностями моих входных данных, так как пришлось предусматривать разное написание даты и парсить данные, чтобы выполнить сортировку, что тоже требует времени, но потенциально мы экономим больше времени, чем теряем на первичную обработку.