Московский авиационный институт (национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: С. Ю. Свиридов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-22

Дата:
Оценка:
Подпись:

Москва 2024

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: `Телефонные номера, с кодами стран и городов в формате +<код страны><код города> телефон.`

Вариант значения: `Строки переменной длины (до 2048 символов).`

1 Описание

Основная идея поразрядной сортировки заключается в сортировке чисел устойчивой сортировкой по каждому разряду, начиная с младшего.

Со стандартного ввода будет считываться номер и записываться как строка. Затем эта строка дробится на 3 разряда и записывается в целочисленные массивы, неразрывно связанные с каждой парой «ключ-значение».

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру `Pair`, в которой будем хранить ключ в виде строки типа `char*` и значение как статический массив char'ов. Таким образом получим массив структур, хранящий в себе такое количество пар «ключ-значение», какое ввел пользователь.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>

typedef struct {
    char* number;
    char* value;
    int NUM[3];
} Pair;
```

Как было сказано выше, перед началом сортировки нужно, чтобы исходная входная строка, записанная в поле `char* number` была поделена на 3 разряда и записана соответствующим образом в массив `int NUM[3]`. Для этого существует функция `void CreatingDigits(Pair* arr, int n)`, принимающая на вход массив структур `Pair` и его размер.

```
void CreatingDigits(Pair* arr, int n) {
    for (int i = 0; i < n; i++) {
        char temp[20];
        strcpy(temp, arr[i].number);
        char *token = strtok(temp, "-");
        if (token != NULL) {
            arr[i].NUM[0] = atoll(token);
        }
        else {
            arr[i].NUM[0] = 0;
        }

        token = strtok(NULL, "-");
        if (token != NULL) {
            arr[i].NUM[1] = atoll(token);
        }
        else {
            arr[i].NUM[1] = 0;
        }

```

```
        token = strtok(NULL, "\0");
        if (token != NULL) {
            arr[i].NUM[2] = atoll(token);
        }
        else {
            arr[i].NUM[2] = 0;
        }
    }
}
```

Сортировка подсчётом помимо массива и его размера принимает номер разряда `exp` и позицию в массиве `NUM`. То есть мы сортируем числа, находящиеся в каждой структуре на позиции `NUM[var]` и переменная `exp` позволяет нам определить номер разряда для каждого числа в `NUM[var]` и использовать его для сортировки. Сортировка начинается с младших разрядов - с единиц.

```
void counting_sort(Pair *arr, int n, int exp, int var) {
    Pair output[n];
    int count[10] = {0};
```

Перед сортировкой создаем временный массив `Pair output[n]` и массив для подсчета 10 уникальных цифр `int count[10]`, который заполняем нулями.

```
    for (int i = 0; i < n; i++) {
        int index = (arr[i].NUM[var] / exp) % 10;
        count[index]++;
    }

    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }
```

После этого мы проходимся по массиву `arr` и находим значение разряда для каждого числа в `NUM[var]`. В этом же цикле увеличиваем значение соответсвующего разряду индекса в массиве `count` на единицу. В следующем цикле мы находим сумму префиксов.

```
        for (int i = n - 1; i >= 0; i--) {
        int index = (arr[i].NUM[var] / exp) % 10;
        output[count[index] - 1] = arr[i];
        count[index]--;
    }

    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
```

```
}
```

Далее формируем с конца исходный массив, используя сумму префиксов массива `count` и уменьшая на единицу счетчик для соответствующего числа. Ну и в самом конце копируем получившийся массив `output` в исходный массив `arr`. Таким образом на выходе получаем отсортированный исходный массив.

Во время поразрядной сортировке массив разбивается на отдельные разряды, в нашем случае на код страны, код города и телефон. Далее каждый из этих разрядов сортируется подсчетом. В этой реализации используется тот же приницип

```
void radix_sort(Pair* arr, int n) {
    int max1 = MaxLength(arr, n, 2);
    for (int exp = 1; max1 / exp > 0; exp *= 10) {
        counting_sort(arr, n, exp, 2);
    }

    int max2 = MaxLength(arr, n, 1);
    for (int exp = 1; max2 / exp > 0; exp *= 10) {
        counting_sort(arr, n, exp, 1);
    }

    int max3 = MaxLength(arr, n, 0);
    for (int exp = 1; max3 / exp > 0; exp *= 10) {
        counting_sort(arr, n, exp, 0);
    }

    for (int i = 0; i < n; i++) {
        printf("%s\t%s\n", arr[i].number, arr[i].value);
    }
}
```

Номер телефона поделен по разрядам, каждый из которых хранится в массиве `int NUM[3]`: код страны хранится в ячейке с индексом 0, код города в ячейке с индексом 1 и телефон хранится в ячейке с индексом 2. Для каждого индекса ячейки в массиве структур `arr` вызывается сортировка подсчетом, то есть сортируются сначала телефоны, затем коды городов и затем коды стран. Перед вызовом функции сортировки подсчета, ищется максимальное значение числа в текущих ячейках с помощью функции `MaxLength`. Затем в цикле `for` сортировка подсчетом вызывается столько раз, сколько разрядов имеет максимальное число. После сортировки всех разрядов ключей происходит печать отсортированных пар «ключ-значение».

3 Консоль

```
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ gcc -lm -O2
-fno-stack-limit -std=c17 -x c myvar2.c -olab1
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ ./lab1
+7-495-1123212  xGfxrxGGxrxMMMMfrrrG
+375-123-1234567        xGfxrxGGxrxMMMMfrrr
+7-495-1123212  xGfxrxGGxrxMMMMfrr
+375-123-1234567        xGfxrxGGxrxMMMMfr


+7-495-1123212  xGfxrxGGxrxMMMMfrrrG
+375-123-1234567        xGfxrxGGxrxMMMMfrrr
+7-495-1123212  xGfxrxGGxrxMMMMfrr
+375-123-1234567        xGfxrxGGxrxMMMMfr
+7-495-1123212  xGfxrxGGxrxMMMMfrrrG
+7-495-1123212  xGfxrxGGxrxMMMMfrr
+7-495-1123212  xGfxrxGGxrxMMMMfrrrG
+7-495-1123212  xGfxrxGGxrxMMMMfrr
+375-123-1234567        xGfxrxGGxrxMMMMfrrr
+375-123-1234567        xGfxrxGGxrxMMMMfr
+375-123-1234567        xGfxrxGGxrxMMMMfrrr
+375-123-1234567        xGfxrxGGxrxMMMMfr
```

4 Тест производительности

Тесты производительности представляют из себя следующее: поразрядная сортировка сравнивается с qsort из библиотеки `stdlib` языка C. Быстрая сортировка будет сортировать целые числа, сформированные из номеров телефона, а поразрядная будет сортировать сами номера телефона. Учитывается только время непосредственной сортировки. Тестов будет три: на 10^3 и 10^4 и 10^5элементов.

```
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ gcc myvar2_bench.c
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ ./a.out
Выберите размер массива: 1000
Время работы быстрой сортировки -0.000134 сек
Время работы поразрядной сортировки -0.000203 сек
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ ./a.out
Выберите размер массива: 10000
Время работы быстрой сортировки -0.000700 сек
Время работы поразрядной сортировки -0.001998 сек
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog4sem/discran/laba1$ ./a.out
Выберите размер массива: 100000
Время работы быстрой сортировки -0.008112 сек
Время работы поразрядной сортировки -0.037152 сек
```

На всех тестах `qsort` существеннее быстрее поразрядной сортировки. Сложность быстрой сортировки - $O(n * log(n))$, в то время как сложность поразрядной - $O(d(n + k))$. Поразрядная сортировка может быть более эффективной в случае, если сортируемые числа находятся в определенном диапазоне значений, потому что поразрядная сортировка не требует сравнения элементов. Результаты бенчмарка ожидаемы, так как быстрая сортировка обычно быстрее поразрядной, особенно для больших массивов и неограниченных диапазонов значений. `Qsort` из библиотеки `stdlib` языка C - эффективный выбор для сортировки целых чисел.

5 Выводы

Изучив поразрядную сортировку в рамках первой лабораторной работы по курсу «Дискретный анализ», я приобрел новые знания и навыки. Понял, как эффективно использовать этот алгоритм для сортировки структур по ключу. Также я понял важность бенчмарка, который позволяет оценивать производительность программы, сравнивая ее с другой, и применил его на практике, сравнив свою сортировку с сортировкой из стандартной библиотеки языка. Я думаю, что полученные знания и навыки обязательно пригодятся мне в дальнейшем.

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] *Radix Sort – Data Structures and Algorithms Tutorials*
URL: https://www.geeksforgeeks.org/radix-sort/ (дата обращения: 11.03.2024).