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: А. Т. Бахарев
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2018

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: `MD5-суммы (32-разрядные шестнадцатиричные числа).`

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

1 Описание

Требуется написать реализацию алгоритма поразрядной сортировки.

Идея сортировки заключается в разбиении сортируемых элементов на разряды. Затем выполняется устойчивая сортировка подсчетом для каждого разряда. При этом, для строк подходит версия MSD(Most Significant Digit) - сортировка начинается от самого старшего разряда. Для чисел же нужно использовать LSD-версию(Least Significant Digit). В данной версии, сортировка начинается от самого младшего разряда.

Свойства сортировки подсчетом:

- Не является сортировкой сравнением: ни одна пара элементов не сравнивается друг с другом
- Линейная (вернее, $O(k+n)$, но при $k = O(n)$ время выполнения $O(n)$)
- Устойчивая (стабильная)
- Требует дополнительную память под массивы C и B размером k и n соответственно

Описание работы программы Сначала происходит ввод данных. Так как заранее неизвестно, сколько элементов будет обрабатываться, то исходный массив динамически расширяется. Как только встречается символ EOF, ввод считается завершенным и массив передается функции $sort$, которая сортирует данные. Затем происходит печать того же массива, который уже отсортирован.

2 Исходный код

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
typedef struct TElement TElement;
struct TElement
{
    char Buffer[33];
    unsigned long long int n;
};
void sort(TElement*, int& amount_of_elems);
int main(int argc, char *argv[])
{
    int size_of_array = 1; // default value
    int amount_of_elems = 0;
    TElement* array = new TElement[size_of_array]; // array of elems
    while(1)
    {
        char ch = getchar();
        if(ch == EOF || ch == '\0')
            break;
        else
            ungetc(ch, stdin);
        if(amount_of_elems == size_of_array)
        {
            TElement* tmp_buffer = new TElement[size_of_array];
            memcpy(tmp_buffer, array, size_of_array*sizeof(TElement));
            array = new TElement[size_of_array * 2];
            memcpy(array, tmp_buffer, size_of_array*sizeof(TElement));
            size_of_array *= 2;
            delete[] tmp_buffer;
        }
        scanf("%s", array[amount_of_elems].Buffer);
        scanf("%llu", &array[amount_of_elems].n);
        ++amount_of_elems;
        getchar();
    }
    sort(array, amount_of_elems);
    for(int i = 0; i < amount_of_elems; ++i)
    {
        printf("%s\t", array[i].Buffer);
        printf("%llu\n", array[i].n);
    }
    delete[] array;
    return 0;
}
void sort(TElement* array, int& amount_of_elems)
```

```
48 {
49     const int radix = 16;
50     const int strLen = 32;
51     TElement* A = new TElement [amount_of_elems];
52     for(int digit = strLen - 1; digit >= 0; --digit)
53     {
54         int C[radix] = {0};
55         for(int i = 0; i < amount_of_elems; ++i)
56         {
57             A[i] = array[i];
58             int c = array[i].Buffer[digit];
59             if (c >= '0' && c <= '9') c = c - '0';
60             else c = c - 'a' + 10; // According to ASCII table
61             ++C[c];
62         }
63         for(int i = 1; i < radix; ++i)
64             C[i] = C[i] + C[i - 1];
65         for(int i = amount_of_elems - 1; i >= 0; --i)
66         {
67             TElement val = A[i];
68             int c = A[i].Buffer[digit];
69             if (c >= '0' && c <= '9') c = c - '0';
70             else c = c - 'a' + 10; // According to ASCII table
71             int position = C[c] - 1;
72             array[position] = val;
73             --C[c];
74         }
75     }
76     delete[] A;
77 }
```

3 Консоль

```
alex$ g++ -std=c++11 -o da_1 -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long
-lm main.cpp sort.cpp
alex$ cat 01.t
00000000000000000000000000000000 13207862122685464576
ffffffffffffffffffffffffffffffff 7670388314707853312
00000000000000000000000000000000 4588010303972900864
ffffffffffffffffffffffffffffffff 12992997081104908288
alex$ ./da_1 <01.t
00000000000000000000000000000000 13207862122685464576
00000000000000000000000000000000 4588010303972900864
ffffffffffffffffffffffffffffffff 7670388314707853312
ffffffffffffffffffffffffffffffff 12992997081104908288
```

4 Тест производительности

Тест производительности представляет из себя следующее: Производится замер времени для работы поразрядной сортировки и сравнивается со временем работы сортировки std::sort().

```
alex$ bash benchmark.sh
g++ -std=c++11 -o da_1 -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long
-lm main.cpp sort.cpp
Time for Radix Sort

real 0m20.225s
user 0m19.748s
sys 0m0.472s
Time for std::sort()

real 0m28.851s
user 0m27.836s
sys 0m1.000s
alex$
```

Как видно, поразрядная сортировка выиграла по времени у быстрой сортировки(std::sort()
Это связано с тем, что сложность первой сортировки - линейная, а у std::sort()
она равна $O(NlogN)$.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился реализовывать и применять на практике эффективные сортировки за линейное время. Те алгоритмы, которые я знал до этого, значительно проигрывают по времени и по памяти. Однако чтобы это увидеть, надо обрабатывать большие обьемы данных, иначе преимущество алгоритма не так очевидно. Также, я научился писать бенчмарки для оценки времени работы программы и определять сложность работы алгоритмов, а это, на мой взгял, очень ценные знания.

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: http://ru.wikipedia.org/wiki/Сортировка_подсчётом (дата обращения: 16.12.2017).