Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: П. А. Мохляков
Преподаватель: Н. С. Капралов
Группа: М8О-308Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №8

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти. Реализовать программу на языке C или C++, соответсвующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.

Вариант: Дана последовательность длины N из целых чисел 1, 2, 3. Необходимо найти минимальное количество обменов элементов последовательности, в результате которых последовательность стала бы отсортированной.

Формат входных данных: Число N на первой строке и N чисел на второй строке.

Формат результата: Минимальное количество обменов.

1 Описание

Жадные алгоритмы — это алгоритмы, заключающиеся в принятии локально оптимальных решений, допуская, что общее решение будет оптимальным.

В нашем случае лучшим вариантом будет сразу ставить нужную цифру на нужное место. Для этого сначала подсчитаем количество каждой из цифр, чтобы знать индексы начала каждой цифры в массиве.

Далее проходимся по введенному массиву, все цифры не являющиеся единицами, но при этом находящиеся на отрезке единиц должны быть заменены на них. Поэтому всречая другие цифры мы увеличиваем колличество перестановок.

В разделе двоек мы аналогично считаем только тройки.

При этом может быть ситуация когда, все тройки находятся в разделе единиц, а все единицы в раздел двоек. В таком случае чтобы поставить каждую тройку на место нужно еще одно действие, которое перемещало бы ее с участка двоек на участок троек. Для подсчета количества таких троек нужно из количества двоек на участке троек вычесть количество троек на участке двоек.

Таким образом, сложив три этих числа мы получаем количество перестановок, не изменяя исходный массив.

2 Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>


int main(){
    int n;
    std::cin >> n;
    std::vector<int> nums(n);
    int sum[3] = {0,0,0};
    int count = 0;
    int size = nums.size();
    int countthree = 0;
    for(auto &i:nums){
        std::cin >> i;
        ++sum[i - 1];
    }
    for(int i = 0; i < sum[0]; ++i){
        if(nums[i] != 1)
            ++count;
    }
    for(int i = sum[0]; i < sum[0]+sum[1]; ++i){
        if(nums[i] == 3){
            ++count;
            --countthree;
        }
    }
    for(int i = sum[0]+sum[1]; i < size; ++i){
        if(nums[i] == 2)
            ++countthree;
    }
    count += countthree;
    std::cout << count << std::endl;
    return 0;
}
```

3 Консоль

```
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB8 cat test
8
1 2 3 3 1 1 2 2
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB8 g++ main.cpp -o main
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB8 cat test| ./main
4
```

4 Тест производительности

```
Count elements: 2
Greedy algorithm: 9.42e-07
Naive algorithm: 3.01e-07
Count elements: 3
Greedy algorithm: 1.202e-06
Naive algorithm: 3.81e-07
Count elements: 4
Greedy algorithm: 2.524e-06
Naive algorithm: 0.000429494
Count elements: 5
Greedy algorithm: 1.242e-06
Naive algorithm: 602.647
```

Как мы видим наивный алгорит обладает очень большой сложностью, поэтому в следующих тестах он рассматриваться не будет.

```
Count elements: 10
Greedy algorithm: 1.823e-06
Count elements: 100
Greedy algorithm: 2.655e-06
Count elements: 10000
Greedy algorithm: 0.000119414
Count elements: 100000
Greedy algorithm: 0.00112389
Count elements: 1000000
Greedy algorithm: 0.0114983
```

Как видно время работы растет линейно по отношению к объему входных данных, что совпадает с теоритической сложностью $O(n)$.

5 Выводы

Выполнив данную лабораторную работу по курсу «Дискретный анализ» я познакомился с подходом к решению задач именуемый жадными алгоритмами. Данный метод может быть полезен там где не справляется динамическое прораммирование. Однако не все задачи могут быть решены с исползованием жадных алгоритмов.

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.))