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: Е. А. Айрапетова
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

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.

1 Описание

Как сказано в [2]: «Жадные алгоритмы - алгоритмы, предполагающие принятие локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным. В общем случае жадные алгоритмы могут не находить глобального оптимума, однако в некоторых задачах позволяют это сделать».

Во время считывания массива из стандартного ввода программа считает единичек, двоек и троек. Таким образом мы можем представить себе массив, который должен получиться в результате сортировки - в первую очередь в нем будут записаны все единицы, далее - все двойки и в последнюю очередь - тройки.

Далее мы проходимся по массиву слева направо. Если в промежутке $[0..n_1]$, где n_1 - общее количество единиц, программа встречает двойку, значит, существует единица, которая не на своем месте. Пытаемся ее найти во втором промежутке $[n_1 + 1..n_2 - n_1]$ (n_2 - количество двоек), так как в таком случае при обмене сразу два элемента встанут на свои места и нам понадобиться сделать лишь одно действие. Если же во втором блоке не нашлось единицы, то ищем в третьем блоке, где она точно есть. Аналогично для остальных промежутков.

Сложность этого алгоритма $O(n^2)$.

2 Исходный код

```
#include <iostream>
#include <string>
#include <algorithm>
#include <chrono>

using namespace std;

template<class T, class S>
void print(T* objects, S N) {
  for (S i = 0; i < N; i++) {
    cout << objects[i] << " ";
  }

  cout << endl;
}

template<class T>
void print(T object, bool end = true) {
  if (end == true)
    cout << object << endl;
  else
    cout << object << " ";
}

template<class T, class... Args>
void print(T& first, Args... next) {
  print(first, false);
  print(next...);
}

int main() {
  size_t n = 0;
  cin >> n;

  short* numbers = new short[n];
  size_t one_two_three[3] = {0, 0, 0};
  for (size_t i = 0; i < n; i++) {
    cin >> numbers[i];
    one_two_three[numbers[i]-1]++;
  }

  size_t result = 0;
  for (size_t i = 0; i < n; i++) {
    if (i < one_two_three[0]) {
      if (numbers[i] == 2) {
        for (size_t j = one_two_three[0]; j < n; j++) {
          if (numbers[j] == 1) {
```

```
48            numbers[i] = 1;
49            numbers[j] = 2;
50            result++;
51            break;
52          }
53        }
54      }
55      else if (numbers[i] == 3) {
56        for (size_t j = one_two_three[0] + one_two_three[1]; j < n; j++) {
57          if (numbers[j] == 1) {
58            numbers[i] = 1;
59            numbers[j] = 3;
60            result++;
61            break;
62          }
63        }
64        if (numbers[i] == 3) {
65          for (size_t j = one_two_three[0]; j < one_two_three[0] + one_two_three[1]; j
                ++) {
66            if (numbers[j] == 1) {
67              numbers[i] = 1;
68              numbers[j] = 3;
69              result++;
70              break;
71            }
72          }
73        }
74      }
75    }
76    else if (i < one_two_three[0] + one_two_three[1]) {
77      if (numbers[i] == 3) {
78        for (size_t j = one_two_three[0] + one_two_three[1]; j < n; j++) {
79          if (numbers[j] == 2) {
80            numbers[i] = 2;
81            numbers[j] = 3;
82            result++;
83            break;
84          }
85        }
86      }
87    }
88  }
89
90  cout << result << endl;
91  delete[] numbers;
92  return 0;
93 }
```

3 Консоль

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ g++ lab8.cpp -o lab8
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ cat test1.txt
3
3 2 1
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./lab8 <test1.txt
1
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ cat test2.txt
10
1 1 2 1 2 2 1 2 1 3
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./lab8 <test2.txt
2
```

4 Тест производительности

Тест производительности представляет из себя следующее: программа засекает, за сколько выполняется программа для каждого из 4 тестов. Первой строкой выводится количество обменов, которое нужно произвести, второй - время, за которое программа это посчитала.

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./benchmark <test1.txt
1
TIME: 1 ms
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./benchmark <test2.txt
2
TIME: 1 ms
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./benchmark <test3.txt
16
TIME: 1 ms
jane@Evgenia:/mnt/c/Files/ДА/ЛР8$ ./benchmark <test4.txt
25
TIME: 1 ms
```

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомилась с жадными алгоритмами и научилась их реализовывать. Данный алгоритм не будет оптимальным для любой задачи, однако за квадратичное время он находит решение.

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] *Жадные алгоритмы — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Жадные_алгоритмы` (дата обращения: 26.05.2021).