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: В. В. Косогоров
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №8

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.
Вариант алгоритма: 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

Говорят, что к оптимизационной задаче применим принцип жадного выбора, если последовательность локально оптимальных выборов даёт глобально оптимальное решение. В типичном случае доказательство оптимальности следует такой схеме:
1) Доказывается, что жадный выбор на первом шаге не закрывает пути к оптимальному решению: для всякого решения есть другое, согласованное с жадным выбором и не хуже первого.
2) Показывается, что подзадача, возникающая после жадного выбора на первом шаге, аналогична исходной.
3) Рассуждение завершается по индукции.[1].

2 Исходный код

Для поиска оптимального решения будем считать количество единиц и двоек в последовательности. Это позволит нам разделить последовательность на «зоны», в которые будут попадать числа после сортировки - единицы в зону 1, двойки в зону 2, тройки в зону 3 (эта зона не учитывается, так как тройки попадут туда автоматически после сортировки в первых двух зонах).
Заметим, что оптимальной заменой является такая замена, при которой число из неподходящей ему зоны переходит в соответствующую своему значению зону, а на его место становится подходящее для данной зоны число. Поэтому сначала будем совершать оптимальные замены, что приведет к наименьшему количеству замен.

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

int main() {
    size_t n;
    std::cin >> n;
    std::vector<int> arr(n);
    size_t count1 = 0, count2 = 0;
    for (size_t i = 0; i < n; i++) {
        std::cin >> arr[i];
        if (arr[i] == 1) {
            ++count1;
        } else if (arr[i] == 2) {
            ++count2;
        }
    }

    size_t res = 0;

    for (size_t i = 0; i < count1 + count2; i++) {
        if (i < count1) { //zone 1
            if (arr[i] == 3) {
                for (size_t j = n - 1; j >= count1; j--) {
                    if (arr[j] == 1) {
                        std::swap(arr[i], arr[j]);
                        ++res;
                        break;
                    }
                }
```

```
            } else if (arr[i] == 2) {
                for (size_t j = count1; j < n; j++) {
                    if (arr[j] == 1) {
                        std::swap(arr[i], arr[j]);
                        ++res;
                        break;
                    }
                }
            }
        } else { // zone 2
            if (arr[i] == 3) {
                for (size_t j = count1 + count2; j < n; j++) {
                    if (arr[j] == 2) {
                        std::swap(arr[i], arr[j]);
                        ++res;
                        break;
                    }
                }
            }
        }
    }
    std::cout << res << '\n';
    return 0;
}
```

3 Консоль

```
 .../da_labs/lab_08 ./a.out
1
1
0

 .../da_labs/lab_08  ./a.out
5
1 1 2 2 3
0

 .../da_labs/lab_08 ./a.out
5
3 2 1 1 2
3

 .../da_labs/lab_08  ./a.out
15
1 2 3 3 1 3 2 1 1 1 2 3 2 1 3
6
```

4 Тест производительности

Для тестирования будем использовать генератор тестов, написанный на Python.

Тест 1: Длина последовательности - 10 000.
time: 0.013 seconds

Тест 2: Длина последовательности - 100 000.
time: 1.8 seconds

Тест 3: Длини последовательности - 1 000 000.
time: 145.40231 seconds

Как видно из проведённых тестов, при увеличении длины последовательности в 10 раз, время работы программы увеличивается примерно в 100 раз, что показывает квадратичную сложность алгоритма.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я реализовал жадный алгоритм поиска оптимального количества обменов при сортировке последовательности с заданными значениями элементов. Жадные алгоритмы - хороший способ поиска оптимальных решений многих задач.

Список литературы

[1] *Greedy algorithm*
URL: `https://en.wikipedia.org/wiki/Greedy_algorithm` (дата обращения: 10.05.2020)