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: Ибрагимов Р. Р.
Группа: М8О-303Б-22
Вариант: 2
Преподаватель: Макаров Н.К.
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Постановка задачи
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 ввода
Число N на первой строке и N чисел на второй строке.

Формат вывода
Минимальное количество обменов.

Метод решения
В отсортированной последовательности сначала будут идти все 1, затем все 2, а в конце 3. Подсчитав количество вхождений 1, 2 и 3, можно установить границы зон, в которых располагаются все 1, все 2, все 3 соответственно в отсортированной последовательности. Далее посчитаем конкретно в границах каждой зоны количество вхождений элементов исходной последовательности. Интерес представляют неверно расположенные элементы (2 и 3 в зоне 1, 1 и 3 в зоне 2, 1 и 2 в зоне 3).
Необходимо минимизировать количество обменов. Ответ будет складываться из обменов двух и трёх элементов. Для подсчета обмена двух элементов будем брать минимум из количества неверно расположенных пар в зонах друг друга (например, 2 в зоне 1 и 1 в зоне 2). Оставшиеся числа будут учтены в обмене трёх элементов. Соответственно, определяем количество оставшихся троек и умножаем его на 2 (так как для обмена трёх элементов нужно два обмена).
Данный алгоритм решения жадный, потому что для неверно расположенных чисел определяется минимальное количество необходимых перестановок на текущий момент.
Сложность алгоритма решения: O(n), где n — длина последовательности чисел.

Исходный код
```
#include <iostream>
#include <vector>

int main() {
    size_t n;
    std::cin >> n;

    std::vector<int> sequence(n);
    for (size_t i = 0; i < n; ++i) {
        std::cin >> sequence[i];
    }

    std::vector<size_t> count(3, 0);
    for (size_t i = 0; i < n; ++i) {
        ++count[sequence[i] - 1];
    }
```

```
    for (size_t i = 1; i < count.size(); ++i) {
        count[i] += count[i - 1]; // теперь в соответстующих count[i]
лежат границы для зон 1, 2, 3
    }

    std::vector<std::vector<size_t>> countInZones(3,
std::vector<size_t>(3, 0));

    for (size_t i = 0; i < n; ++i) {
        size_t zone = (i < count[0]) ? 0 : (count[0] <= i && i <
count[1]) ? 1 : 2;
        ++countInZones[zone][sequence[i] - 1];
    }

    size_t swapTwoElements = std::min(countInZones[1][0],
countInZones[0][1]) +
                             std::min(countInZones[2][1],
countInZones[1][2]) +
                             std::min(countInZones[0][2],
countInZones[2][0]);

    size_t remainingWrongElements = (std::max(countInZones[1][0],
countInZones[0][1]) - std::min(countInZones[1][0], countInZones[0][1]) +
                                    std::max(countInZones[2][1],
countInZones[1][2]) - std::min(countInZones[2][1], countInZones[1][2]) +
                                    std::max(countInZones[0][2],
countInZones[2][0]) - std::min(countInZones[0][2], countInZones[2][0])) /
3;

    size_t swapThreeElements = 2 * remainingWrongElements;

    size_t minSwap = swapTwoElements + swapThreeElements;

    std::cout << minSwap << '\n';

    return 0;
}
```

Тесты

Номер теста	Ввод	Вывод
1	3 3 2 1	1
2	5 1 1 2 2 3	0
3	11 1 3 2 1 1 3 2 1 2 3 2	4

Выводы

В ходе выполнения данной лабораторной работы была решена задача на использование жадного алгоритма. Кажется, что жадные алгоритмы требуют больше доказательной части верности алгоритма и следование принципу выбора лучшего действия в данный момент, что на самом деле не всегда может приводить к верному решению.