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. П. Будникова
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б-19

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 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.

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

Формат результата: Минимальное количество обменов.

1 Описание

Как сказано в [1]: «Жадный алгоритм — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.» В данном алгоритме в самом начале считывается число, которое обозначает длину строки. Выделяется память под строку и считывается строка. При считывании определяется количество троек, двоек и единиц в строке. Так как последовательность должна быть отсортированной, то на первом месте должны стоять единицы(если они есть), на втором двойки(если они есть)б на третьем(если они есть). С помощью подсчета вхождений каждой цифры определяется границы блоков. Сначала осуществляется проход от 0, до места, где должен кончатся блок единиц, если в этом блоке встречается двойка, то оптимальным ходом будет обмен с единицей, находящейся в блоке двоек, если там нет единицы, то происходит обмен с единицей в блоке троек, счетчик обменов при каждом обмене увеличивается на 1. Аналогично для тройки в блоке единиц. Далее необходимо пройтись по блоку двоек и подсчитать количество элементов, не равных двойке(кол-во троек), а далее прибавить это число к количеству обменов. Выбирая на каждом шаге оптимальный вариант обмена, в результате мы получаем минимальное количество обменов, необходимых для сортировки последовательности. Если одна из цифр отсутствует в последовательности: Если отсутствует единица, то ее граница будет в 0 и проход будет осуществляться только по блоку двоек(если они существуют), если и двоек не существуют, то проходов не будет происходить, так как в строке присутствуют только тройки. Если отсутствует двойка, то будет происходить проход только по блоку единиц(если он есть), при отсутствии проход аналогично не нужен. При отсутствии троек проход осуществляется по блоку единиц с обменом, а потом по блоку двоек, но в блоке двоек(если такой существует), если блок двоек существует, то он будет заполнен двойками на предыдущем шаге, счетчик обменов не увеличится. В конце наборы программы печатается результат и очищается выделенная память.

2 Исходный код

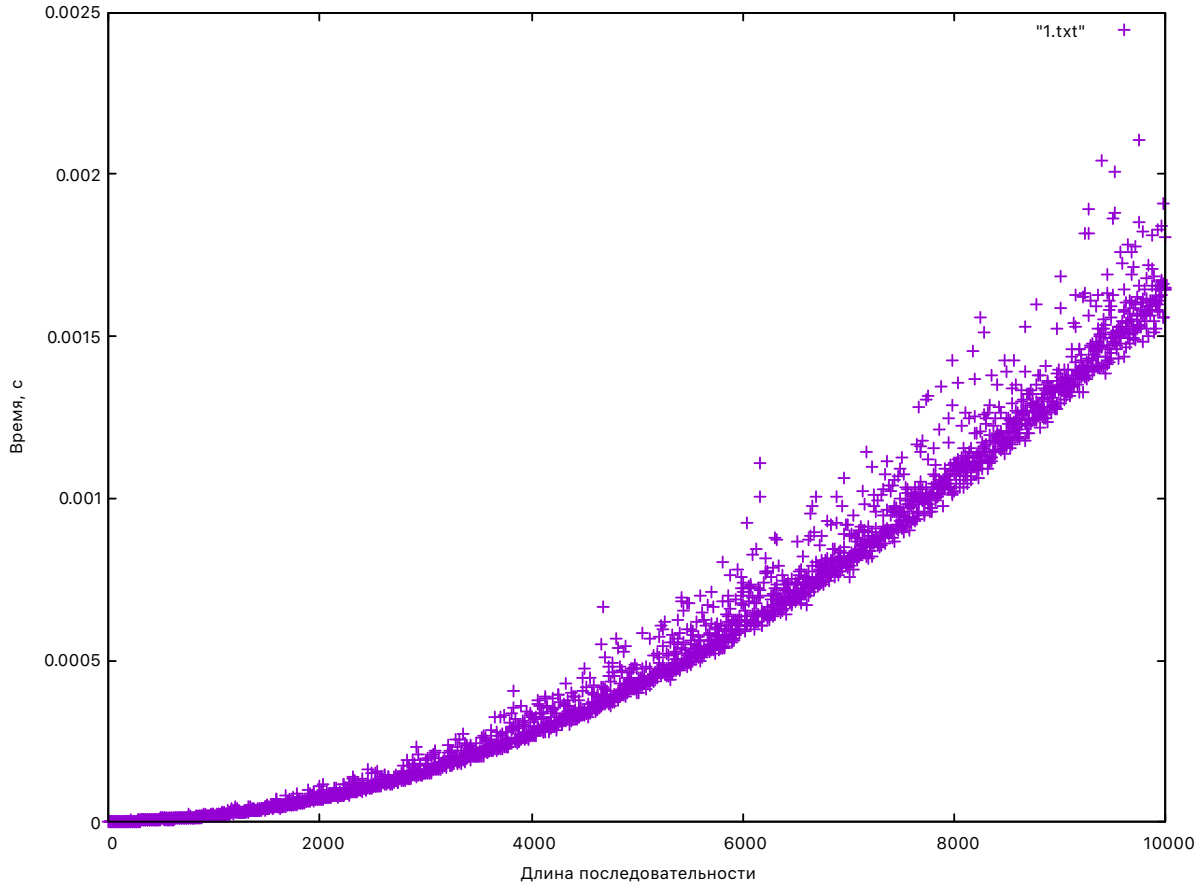
```cpp
#include <iostream>
#include <vector>

size_t count[4] = {0, 0, 0, 0};
char * a;

short Find(char find) {
    short num = find - '0';
    for (size_t i = count[num - 1]; i < count[num]; ++i) {
        if (a[i] == '1') return i;
    }
    num = (num - 1) % 2 + 2;
    for (size_t i = count[num - 1]; i < count[num]; ++i) {
        if (a[i] == '1') return i;
    }
    return -1;
}

int main() {
    size_t n;
    size_t res = 0;
    std::cin >> n;
    a = new char[n];
    for (size_t i = 0; i < n; ++i) {
        std::cin >> a[i];
        ++count[a[i] - '0'];
    }
    count[2] += count[1];
    count[3] += count[2];
    for (size_t i = 0; i < count[1]; ++i) {
        if (a[i] != '1') {
            ++res;
            a[Find(a[i])] = a[i];
            a[i] = '1';
        }
    }
    for (size_t i = count[1]; i < count[2]; ++i) {
        if (a[i] != '2') ++res;
    }
    std::cout << res << std::endl;
    delete[] a;
}
```

3 Консоль

```
Lera:l8 valeriabudnikova$ ./lab8
9
231212133
2
Lera:l8 valeriabudnikova$ ./lab8
9
233321112
4
Lera:l8 valeriabudnikova$ ./lab8
6
211332
2
Lera:l8 valeriabudnikova$ ./lab8
12
121133133222
4
Lera:l8 valeriabudnikova$ ./lab8
1
1
0
Lera:l8 valeriabudnikova$ ./lab8
2
33
0
```

4 Тест производительности

Определим зависимость времени работы программы, от количества входных данным, где единиц, двоек и троек равное количество:



Графики при отсутствии какого либо символа в алфавите выглядят аналогично. Так как в самом плохом случае мы будем проходится по строке два раза, то сложность данного алгоритма $O(n^2)$. Пример наихудшего случая: Подается последовательность "22...2233...3311...11 когда прийдется проходить по первым двум ячейкам, обменивая элементы и производить поиск в других ячейках. Также количество обменов такой длины будет максимальным для строк такой длины.

Так как для хранения последовательности мы выделяем ровно n ячеек, далее алгоритм работает с данными ячейками, то пространственная сложность $O(n)$.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомилась с жадными алгоритмами и научилась реализовывать алгоритм для подсчета количества перестановок в строке для получения отсортированной последовательности.

Список литературы

[1] *Жадный алгоритм — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Жадный_алгоритм` (дата обращения: 12.05.2021).