



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

**ОТЧЕТ  
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7**

Алгоритмические стратегии. Разработка и программная реализация задач с применением  
метода сокращения числа переборов  
**по дисциплине**  
«Структуры и алгоритмы обработки данных»

Выполнил студент группы ИКБО-01-21

Луковников Д.Р.

Принял преподаватель

Туманова М.Б.

Практическая

«\_\_»\_\_\_\_\_ 2022 г.

\_\_\_\_\_

работа выполнена

«Зачтено»

«\_\_»\_\_\_\_\_ 2022 г.

\_\_\_\_\_

Москва 2022

## СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ.....	4
ХОД РАБОТЫ.....	5
1.1    Постановка задачи .....	5
1.2    Математическая модель решения.....	5
1.3    Программная реализация .....	5
1.4    Сравнение числа переборов .....	7
ТЕСТИРОВАНИЕ.....	8
ВЫВОДЫ .....	9
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	10

## **ЦЕЛЬ РАБОТЫ**

Освоить навыки применения алгоритмической стратегии. Разработать и реализовать программу с применением метода сокращений числа переборов, а именно метод динамического программирования.

## ХОД РАБОТЫ

### 1.1 Постановка задачи

Дана строка из заглавных букв латинского алфавита. Найти длину наибольшего палиндрома, который можно получить вычеркиванием некоторых букв из данной строки.

Применить метод Динамического программирования.

### 1.2 Математическая модель решения

Решение основано на матрице, которая служит для перебора возможных под-последовательностей. Перебор начинается с подпоследовательности длиной 1, для такой строки, ничего вычёркивать не требуется, такая строка и так будет являться полиномом и как следствие искомой строкой подпоследовательности.

Для последовательности из 2-х элементов существует 2 варианта, элементы равны или нет, в первом случае ничего не делаем, а во втором вычёркиваем любой из них. Остальные же подпоследовательности вычисляются по следующим алгоритмам, рисунок 1.

$$L[i][j] = \begin{cases} 1, & i = j \\ 0, & i > j \\ L[i + 1][j - 1] + 2, & s[i] = s[j] \\ \max(L[i][j - 1], L[i + 1][j]), & s[i] \neq s[j] \end{cases}$$

Рисунок 1 – Получение под-последовательности

### 1.3 Программная реализация

При написании программы использовались 2 способа ввода: запись строки сразу в переменную и генерация случайной строки.

Код приведён в листинге 1.

### Листинг 1 – main.cpp

```
/*
 * Дана строка из заглавных букв латинского алфавита.
 * Найти длину наибольшего палиндрома, который можно получить вычеркиванием
 некоторых букв из данной строки.
 *
 * Решить задачу с помощью динамического программирования.
 *
 * Математическая модель решения:
 * 1. Построить таблицу, в которой в ячейке [i, j] будет храниться длина
 наибольшего палиндрома, который можно получить из подстроки s[i..j].
 * 2. Заполнить таблицу по диагоналям, начиная с главной.
 * 3. Найти максимальное значение в таблице.
 *
 * Как работает матрица:
 * 1. Если i == j, то это один символ, а значит это палиндром.
 * 2. Если i + 1 == j, то это два символа, а значит это палиндром, если они
 равны.
 * 3. Если i + 1 < j, то это больше двух символов, а значит это палиндром,
 если s[i] == s[j] и s[i + 1..j - 1] - палиндром.
 */

#include <iostream>
#include <string>

using namespace std;

int main() {
    srand(time(nullptr));
    string s;

    // Генерация строки
    for (int i = 0; i < 100; i++) {
        s += 'A' + rand() % 26;
    }
    s = "ABCCBEA";
    s = "AABARA";

    // Вывод строки
    cout << s << endl;

    // Создание таблицы
    int n = s.length();
    int dp[n][n];

    // Заполнение таблицы
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            dp[i][j] = 0;

    // Заполнение главной диагонали
    for (int i = 0; i < n; i++)
        dp[i][i] = 1;

    // Заполнение диагоналей, параллельных главной
    for (int i = 0; i < n - 1; i++)
        // Если символы совпадают, то длина палиндрома равна 2
        if (s[i] == s[i + 1])
            dp[i][i + 1] = 2;
        else
            // Иначе длина палиндрома равна 1
```

### Продолжение Листинга 1

```
        dp[i][i + 1] = 1;
// Заполнение остальных ячеек
for (int i = 2; i < n; i++)
    // i - длина палиндрома
    for (int j = 0; j < n - i; j++)
        // j - начало палиндрома
        if (s[j] == s[j + i]) // Если символы совпадают
            dp[j][j + i] = dp[j + 1][j + i - 1] + 2; // Длина палиндрома
// равна длине палиндрома без первого и последнего символа + 2
        else
            dp[j][j + i] = max(dp[j + 1][j + i], dp[j][j + i - 1]); //
// Иначе длина палиндрома равна максимальной длине палиндрома, полученного из
// двух подпалиндромов
cout << dp[0][n - 1] << endl; // Вывод ответа

// Вывод таблицы
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)
        cout << dp[i][j] << " ";
    cout << endl;
}

// Вывод самого длинного палиндрома
int i = 0, j = n - 1;
while (i < j) {
    if (s[i] == s[j]) {
        cout << s[i];
        i++;
        j--;
    } else if (dp[i + 1][j] > dp[i][j - 1])
        i++;
    else
        j--;
}
return 0;
}
```

## 1.4 Сравнение числа переборов

Каждый элемент массива вычисляется 1 раз за  $O(1)$  обращаясь к уже вычисленным элементам. Так как размер массива  $n \times n$ , то алгоритм работает за  $O(n^2)$ .

В случае решения «в лоб», необходимо было бы в каждой подпоследовательности перебрать каждый элемент и затем сопоставить все подпоследовательности вместе, что даёт сложность  $O(2^n)$ .

## ТЕСТИРОВАНИЕ

Для начала протестируем на данных, введенных вручную, рисунок 2

```
/Users/minusd/CLionProjects/siaod_tasks
ABCCBEA
6
1 1 1 2 4 4 6
0 1 1 2 4 4 4
0 0 1 2 2 2 2
0 0 0 1 1 1 1
0 0 0 0 1 1 1
0 0 0 0 0 1 1
0 0 0 0 0 0 1
ABC
```

Рисунок 2 – Тестирование с введенной строкой

И проверим работу на случайно сгенерированной строке, рисунок 3.

```
/Users/minusd/CLionProjects/siaod_tasks/cmake-build-debug/siaod_tasks
GDAFSXYIJKINTGKNDFJEYMRUGFITMSVXIDCEDUMGQVIGIFJQDFHYIYFGBFLZNZTQNUBNEFECTWQJDVTAQFSTXQYMPQWTTTHFCU
31
1 1 1 1 1 1 1 1 1 3 3 3 5 5 5 5 5 5 7 7 7 9 9 11 11 11 11 11 11 11 11 13 13 13 13 13 15 15 15 15
0 1 1 1 1 1 1 1 1 3 3 3 3 3 3 5 5 5 5 7 7 7 9 9 9 11 11 11 11 11 11 11 11 13 13 13 13 13 13 13 13 13
```

Рисунок 3 – Тестирование со сгенерированной строкой

Как видно из тестов, программа работает корректно.



## **ВЫВОДЫ**

При выполнении работы были получены навыки разработки программ с использованием метода динамического программирования. Была написана и протестированная программа согласно заданному варианту.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
2. Документация по языку C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ruru/cpp/cpp/> (дата обращения 01.12.2022).
3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=4020> (дата обращения 01.12.2022)
4. Статья - Задача о наибольшей подпоследовательности-палиндроме [Электронный ресурс]. URL: [https://neerc.ifmo.ru/wiki/index.php?title=Задача\\_о\\_наибольшей\\_подпоследовательности-палиндроме](https://neerc.ifmo.ru/wiki/index.php?title=Задача_о_наибольшей_подпоследовательности-палиндроме) (дата обращения 05.12.2022).