**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: Динамическое программирование. Палиндром.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5382 |  | Никитин В.А. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017 г.

**Цель работы.**

Ознакомиться с методом динамического программирования, получить навыки динамического программирования на языке программирования С++.

**Постановка задачи.**

Необходимо написать алгоритм, который будет решать следующую задачу:

Палиндром - строка, которая одинаково читается как слева направо, так и справа налево. Составить алгоритм, который по заданной строке определяет минимальное количество символов, которые необходимо вставить в строку для образования палиндрома.

**Вход:** строка длиной n (3<=n<=5000), состоящая из прописных и строчных букв латинского алфавита и цифр от 0 до 9. Прописные и строчные буквы различаются.

**Выход:** целое число - минимальное количество вставляемых символов.

**Основные теоретические положения.**

**Динамическое программирование** в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Метод динамического программирования сверху — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. Динамическое программирование снизу включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

**Спецификация программы.**

*Назначение программы*.

Программа предназначена для решения поставленной задачи методом динамического программирования.

*Описание программы*.

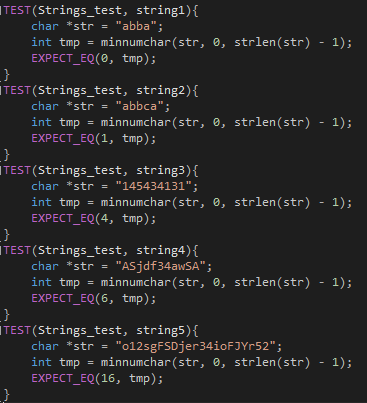
Программа написана на языке C++. На вход программе подается целое число n: 3<=n<=5000, и строка длиной n. Выходными данными является количество символов, необходимое для образования палиндрома. Для простоты проверки результатов в программу были добавлены Google Test’ы, которые проверяют результат выполнения алгоритма, с ожидаемым результатом, для конкретных последовательностей символов.

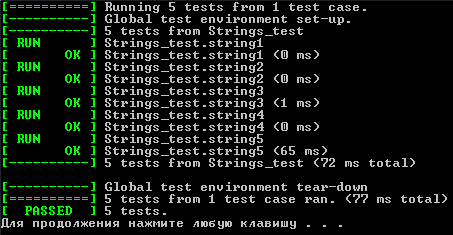
*Реализация*.

Функции:

* int min(int a, int b) // Возвращает минимальное значение из a b.
* int minnumchar(char str[], int l, int r)//рекурсивная функция проверяющая последовательность на условие палиндрома и подсчитывающая количество символов, необходимых к добавлению в последовательность.
* TEST(Strings\_test, string1)… TEST(Strings\_test, string5) // пять Гугл тестов для разных последовательностей символов.

**Тестирование.**





**Выводы**

В ходе выполнения лабораторной работы был исследован алгоритм метод динамического программирования и написана программа, подсчитывающая количество недостающих символов в строке до преобразования ее в палиндром.

**Приложение А. Исходный код.**

#include "stdafx.h"

#include <stdio.h>

#include <iostream>

#include <string.h>

#include "gtest/gtest.h"

using namespace std;

int min(int a, int b)

{

if (a < b)return a;

else return b;

}

int minnumchar(char str[], int l, int r)

{

if (l == r) return 0;

if (l > r) { return INT\_MAX; }

if (l == r - 1){

if (str[l] == str[r]) return 0;

else 1;

}

if (str[l] == str[r]) return minnumchar(str, l + 1, r - 1);

else return (min(minnumchar(str, l, r - 1), minnumchar(str, l + 1, r)) + 1);

}

TEST(Strings\_test, string1){

char \*str = "abba";

int tmp = minnumchar(str, 0, strlen(str) - 1);

EXPECT\_EQ(0, tmp);

}

TEST(Strings\_test, string2){

char \*str = "abbca";

int tmp = minnumchar(str, 0, strlen(str) - 1);

EXPECT\_EQ(1, tmp);

}

TEST(Strings\_test, string3){

char \*str = "145434131";

int tmp = minnumchar(str, 0, strlen(str) - 1);

EXPECT\_EQ(4, tmp);

}

TEST(Strings\_test, string4){

char \*str = "ASjdf34awSA";

int tmp = minnumchar(str, 0, strlen(str) - 1);

EXPECT\_EQ(6, tmp);

}

TEST(Strings\_test, string5){

char \*str = "o12sgFSDjer34ioFJYr52";

int tmp = minnumchar(str, 0, strlen(str) - 1);

EXPECT\_EQ(16, tmp);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

testing::InitGoogleTest(&argc, argv);

RUN\_ALL\_TESTS();

system("pause");

return 0;

}