ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Longest Valid Parentheses»

Выполнил работу

Макаров Эльдар

Академическая группа J3114

Принято

Должность, звание Фамилия Имя преподавателя

Санкт-Петербург

2024

1. Введение

Цель работы: изучение и реализация алгоритмов для нахождения длины самой длинной валидной последовательности скобок.

Задачи:

Изучить теоретические аспекты алгоритма.

Реализовать алгоритм на языке C++.

Проанализировать производительность программы.

Оценить временную и пространственную сложность.

Сравнить экспериментальные результаты с теоретическими оценками.

1. Теоретическая подготовка

Динамическое программирование — это способ решения сложных задач путём разбиения их на более простые подзадачи.

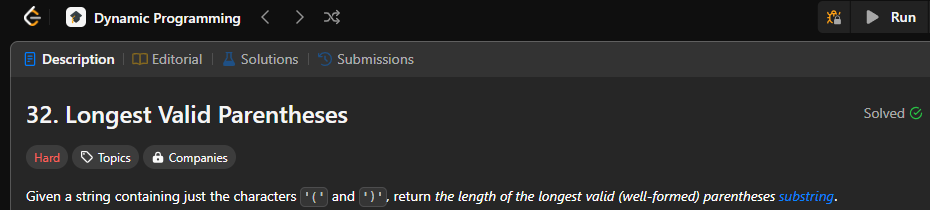
Сама задача состоит в нахождении самой длинной строки скобочек, которая является валидной. Пройдя по строке слева направо и, справа налево получается учесть все возможные последовательности.

В процессе работы используются счётчики для подсчёта количества открывающих и закрывающих скобок. Я использовал счётчики для подсчета количества левых и правых скобок.

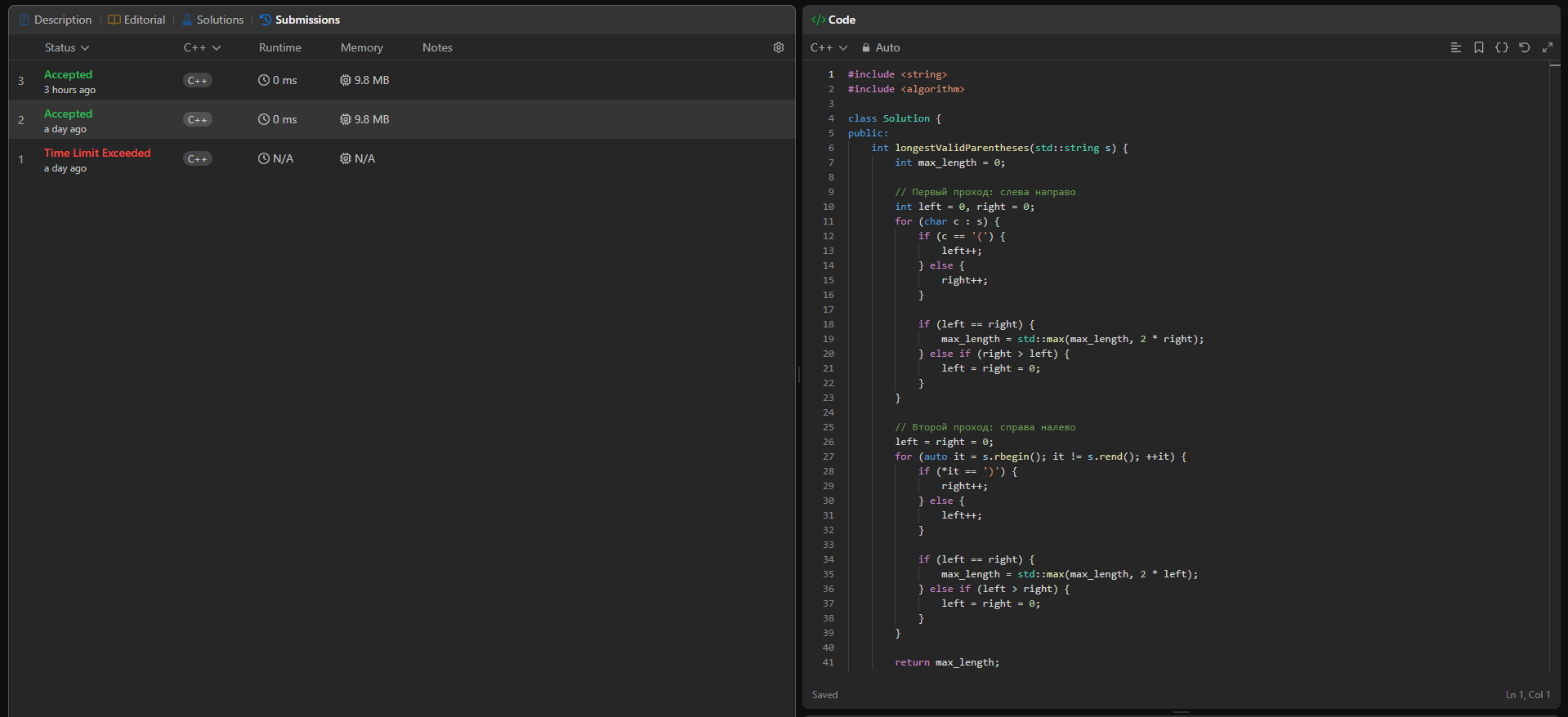
Используемые типы данных:

(std::string) - для хранения последовательности скобок.

(int) - для счётчиков, хранящих количество левых и правых скобок и для хранения максимальной длины.

Задача:

Доказательство выполнения:



1. Реализация

Этап 1: Анализ задачи

Как сказано ранее, сама задача состоит в нахождении самой длинной строки скобочек, которая является валидной. Изначально я подумал о том, что можно попытаться найти правую скобку для каждой левой скобки в подстроке, где подстроку мы выбираем перебором. Я смог выполнить задачу таким алгоритмом, однако он оказался слишком неэффективен, после чего я начал сначала и понял, что если идти по валидной строке, то количество правых скобок будет всегда меньше или равно количеству левых скобок.

Этап 2: Разработка алгоритма

Я не стану рассказывать о своей первой неудачной попытке, так как считаю, что это лишнее.

Мой алгоритм основан на подсчете левых и правых скобок, если правых становится больше, то валидная строка заканчивается, значение ее длинны записывается в переменную и алгоритм продолжает проходить по элементам строки.

Кроме того, существует необходимость пройти по строке в обратном порядке, иначе, некоторые строки будут учитаны неправильно. При прохождении строки в обратном порядке алгоритм тот же, но левые и правые скобки меняются местами.

Этап 3: Написание кода

Код:  
#include <string>

#include <algorithm>

class Solution {

public:

    int longestValidParentheses(std::string s) {

        int max\_length = 0;

        // Первый проход: слева направо

        int left = 0, right = 0;

        for (char c : s) {

            if (c == '(') {

                left++;

            } else {

                right++;

            }

            if (left == right) {

                max\_length = std::max(max\_length, 2 \* right);

            } else if (right > left) {

                left = right = 0;

            }

        }

        // Второй проход: справа налево

        left = right = 0;

        for (auto it = s.rbegin(); it != s.rend(); ++it) {

            if (\*it == ')') {

                right++;

            } else {

                left++;

            }

            if (left == right) {

                max\_length = std::max(max\_length, 2 \* left);

            } else if (left > right) {

                left = right = 0;

            }

        }

        return max\_length;

    }

};

1. Экспериментальная часть

Подсчет асимптотики:

Первый цикл (слева направо): O(n), где n – длина строки.

Второй цикл (справа налево): O(n), где n – длина строки.

Итоговая сложность: O(n).

Я протестировал код на правильную работу разных строках:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input | (() | )()()) |  | ))(()) | ()() | )(())()()()()(())( |
| Output | 2 | 4 | 0 | 4 | 4 | 16 |

Результаты работы кода на случайных строках:

|  |  |  |
| --- | --- | --- |
| Длина строки | Время работы кода (с) | Память (bytes) |
| 10 | 7.6e-06 | 19 |
| 100 | 4.44e-05 | 109 |
| 1000 | 0.0004013 | 1192 |
| 10000 | 0.0039886 | 13532 |
| 100000 | 0.0400625 | 102731 |
| 1000000 | 0.426538 | 1170122 |
| 10000000 | 4.09972 | 13328374 |

На графике (изображение 1) видно, что время выполнения линейно зависит от длины строки, что подтверждает теоретическую сложность O(n).

1. Заключение

В ходе выполнения лабораторной работы мною был реализован алгоритм нахождения максимальной длины валидной подстроки скобок. Цель работы была достигнута благодаря динамическому программированию, что позволило выбрать подходящий подход для решения задачи.

Алгоритм был протестирован на различных входных данных, включая случайно сгенерированные строки разной длины. Результаты экспериментов показали соответствие теоретическим оценкам сложности алгоритма и подтвердили его корректность. График зависимости времени работы от длины строки (изображение 1) и график зависимости затрачиваемого объема памяти от длины строки (изображение 2) продемонстрировали правильное поведение алгоритма в соответствии с расчетной временной сложностью и пространственной сложностью.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения уменьшения затрат использования памяти.