«Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Лодейщикова В.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Отчёт

Лабораторной работе №5

«Рекурсивные алгоритмы»

Студент группы ПИ 92 В.М. Шульпов

Преподаватель доцент, к.ф-м.н. Лодейщикова В.В.

Барнаул 2021

**Цели и задачи:** изучение рекурсивного программирования

**Методика выполнения работы:**

* Изучить технологию использования стека при рекурсивном программировании.
* Написать рекурсивную программу решения поставленной задачи.
* Протестировать программу.

**Требования к отчету:**

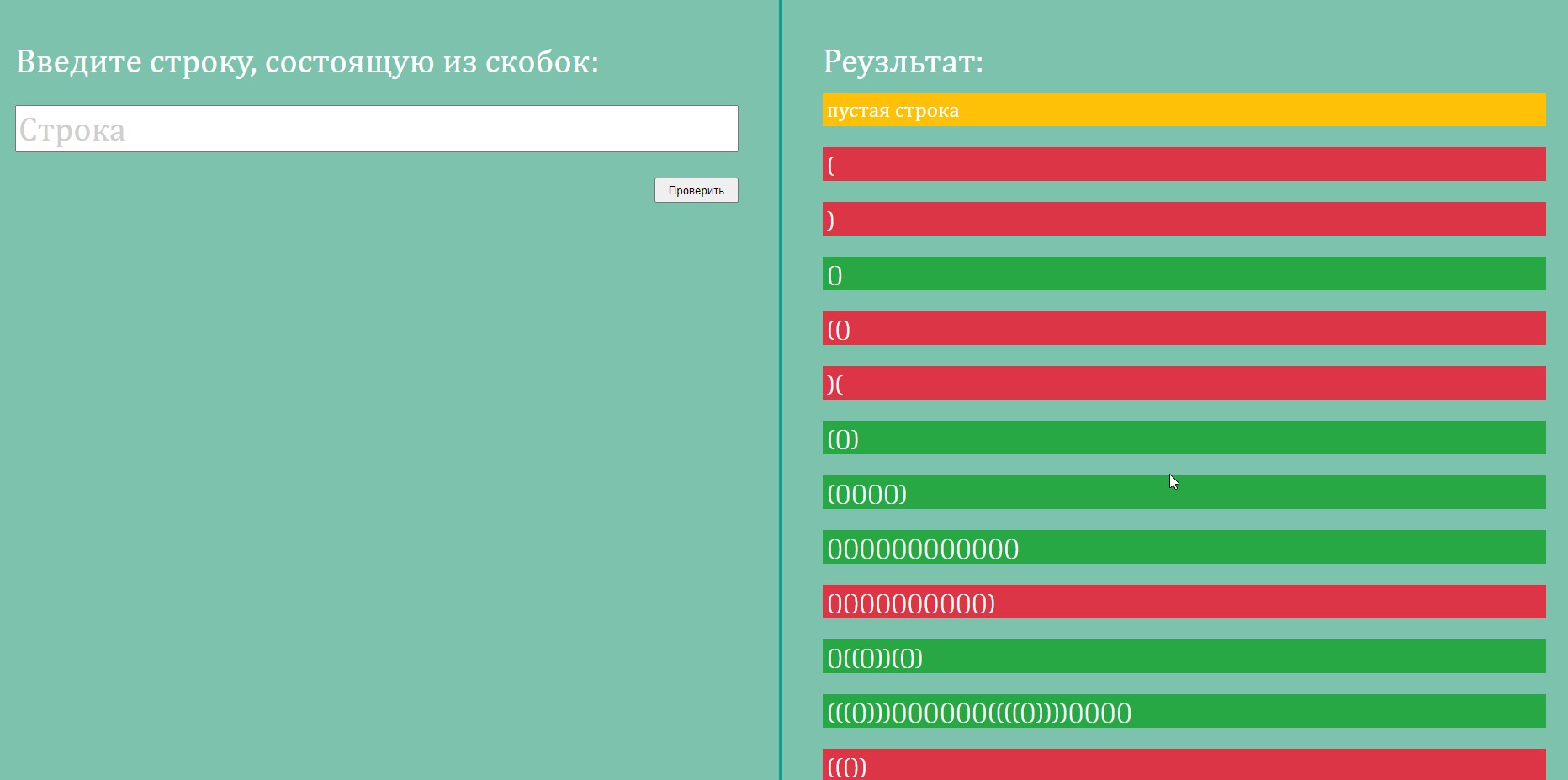
Отчет по лабораторной должен содержать титульный лист, задание, алгоритмы решения, текст программы, тесты.

|  |  |
| --- | --- |
| Вариант 20 | Написать рекурсивную функцию, проверяющую правильность расстановки скобок в строке. При правильной расстановке выполняются условия:  (а) количество открывающих и закрывающих скобок равно.  (б) внутри любой пары открывающая – соответствующая закрывающая скобка, скобки расставлены правильно.  Примеры неправильной расстановки: )(,())(, ())(() и т.п. |

Будем использовать стек. У него есть правило FILO (первый вошёл, последний вышел). Также используем рекурсию.

Алгоритм:

Функция «вырезает» пару рядом стоящих скобок, а затем запускает функцию снова, но уже для обрезанной скобочной последовательности. В этом случае функция ищет и удаляет все рядом стоящие пары скобок. Нужно оставшуюся строку еще раз «скармливать» этой функции. Не используется стек, алгоритм является не самым эффективным, так как приходится повторно проходить по строке много раз. В худшем случае появляется квадратичная сложность (O(n\*n)).

Тесты: 

Код функции из программы:

//рекурсивный алгоритм без стека

function isValid2(string, obj) {

    let wasExit = false;

    for(let i=0; i<string.length-1; i++) {

        obj.iter++;

        if(string[i] == '(' && string[i+1] == ')') {

            string = string.slice(0,i) + string.slice(i+2);

            wasExit = true;

            break;

        }

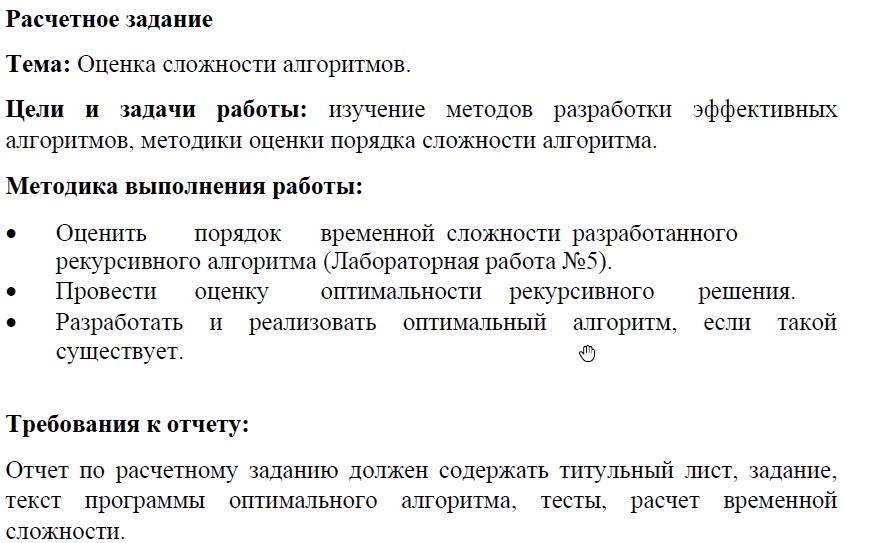
    }

    if (wasExit) return obj.result=isValid2(string, obj); //если вышли по break

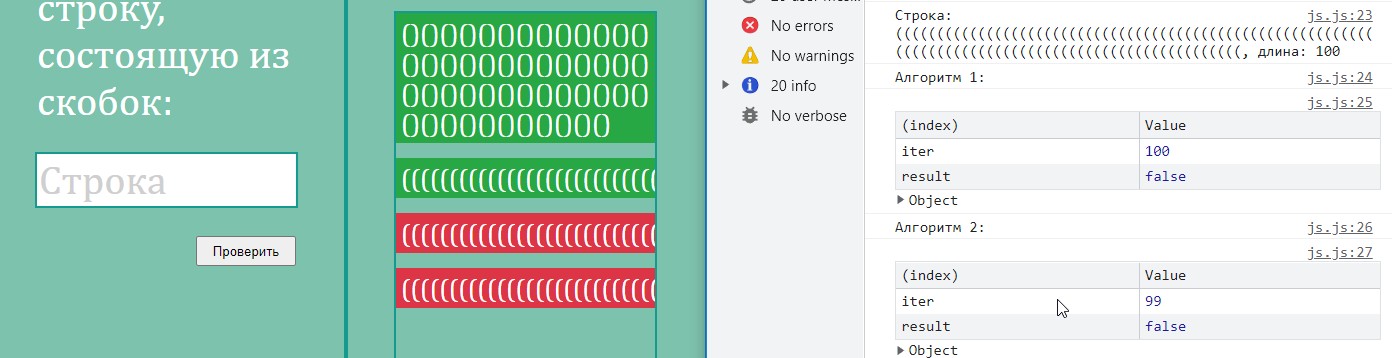
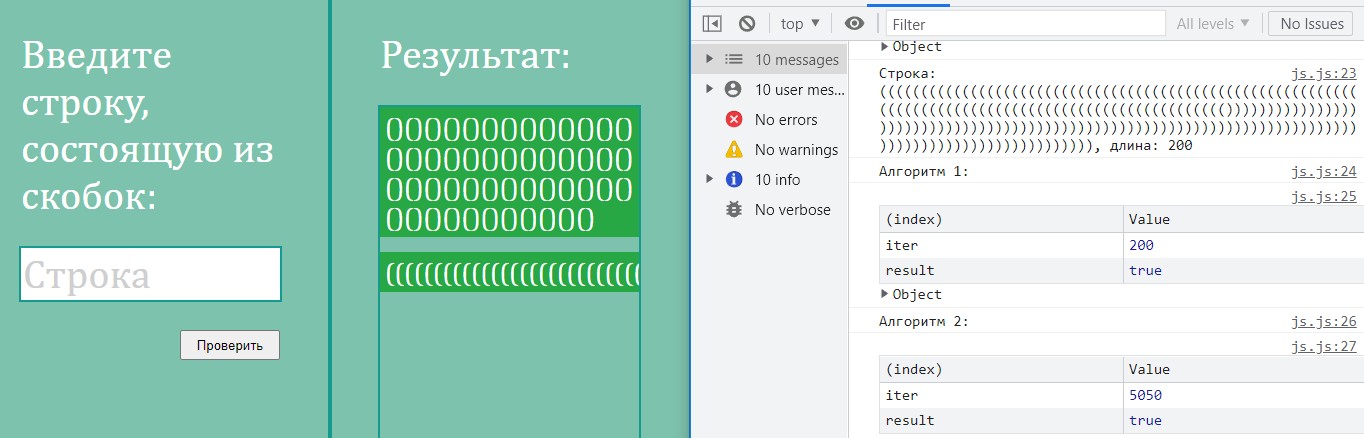
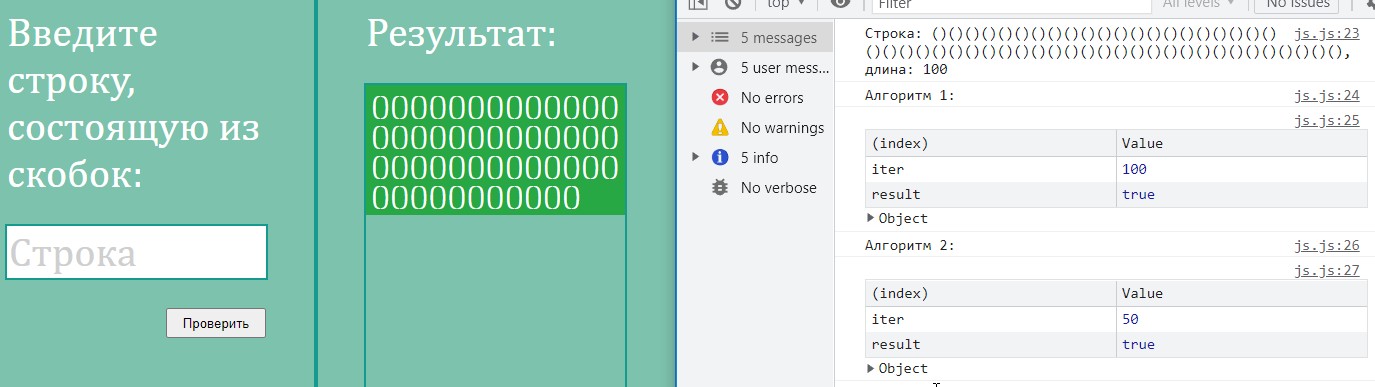
    if(string==='') return obj.result=true;

    else return obj.result=false;

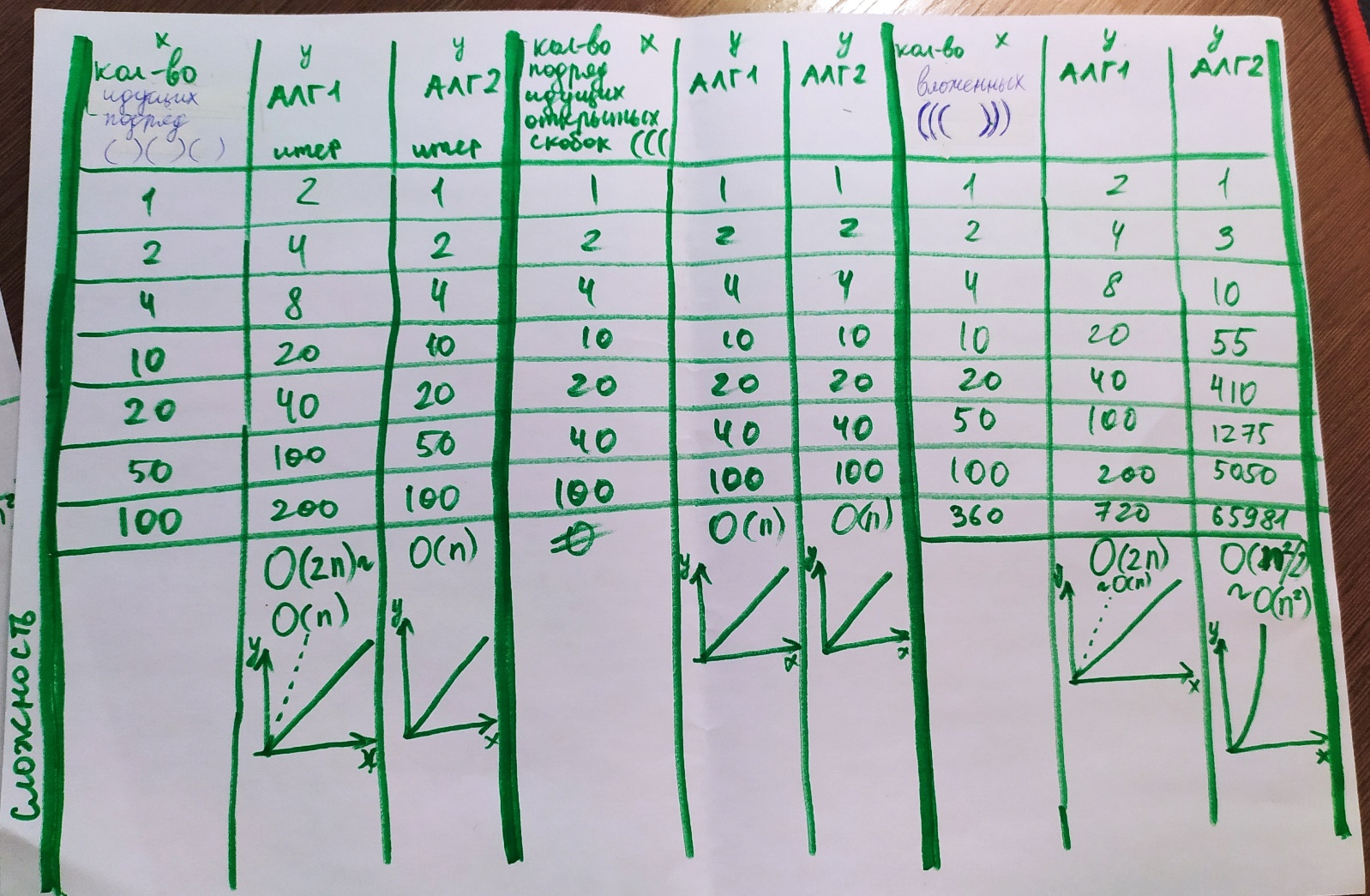
}



Оценка сложности:

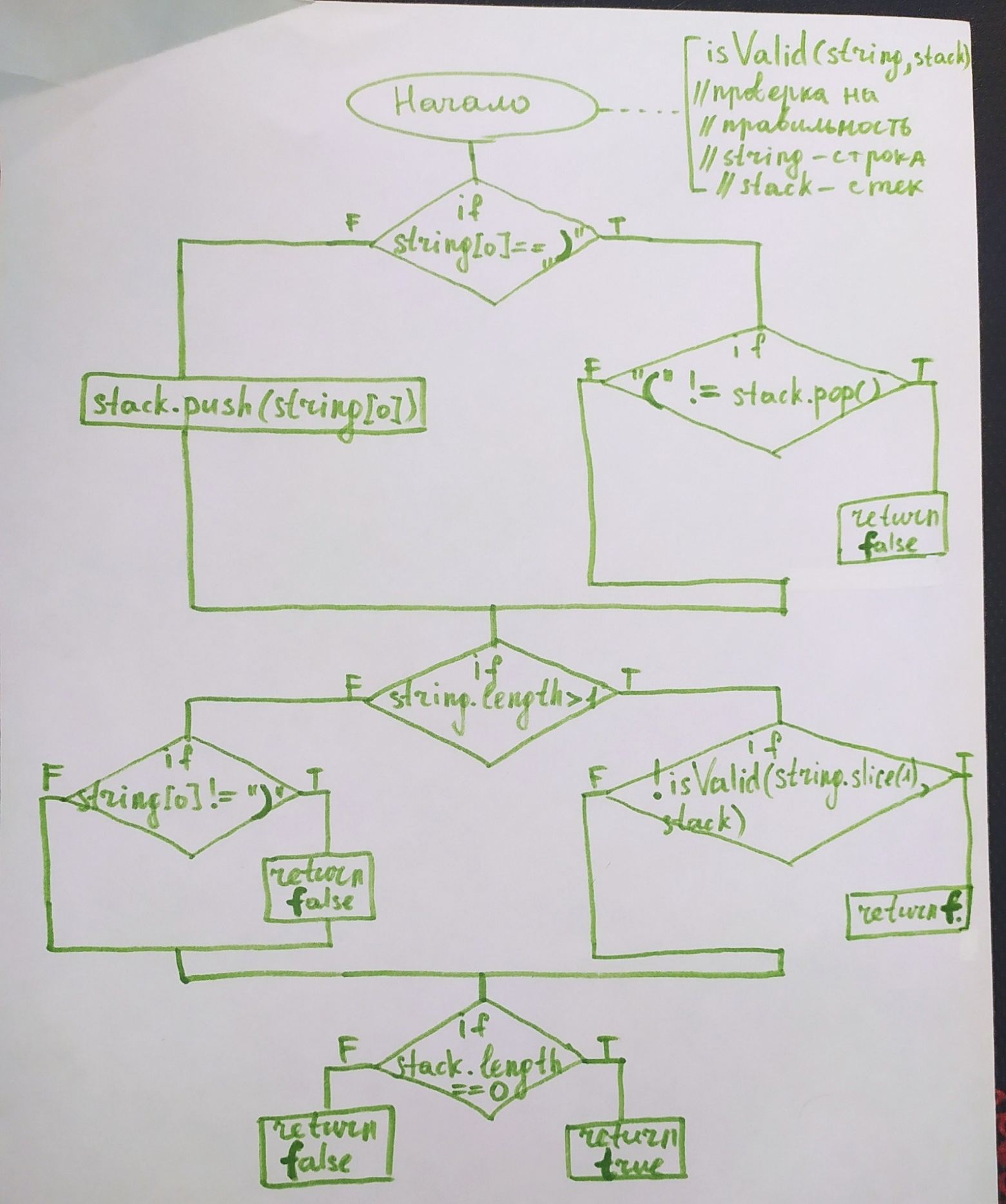


Расчет сложности. Алгоритм 1 более оптимизированный.



Алгоритм оптимизированный:

* Кладём в стек открывающиеся скобки, пока не встретим закрывающую.
* Когда встретили одну закрывающую, убираем из стека одну открывающую.
* Если пытались достать из пустого стека или после отработки алгоритма стека остался пустой – скобочная последовательность неверна.



Данный алгоритм можно реализовать и без рекурсии, используя циклы. Сложность останется прежней.

Код программы:

//рекурсивный алгоритм со стеком

function isValid(string, stack, obj) {

    obj.iter++;

    if (string[0]==')') {

        if ('(' != stack.pop()) return obj.result=false;

    }

    else stack.push(string[0]);

    if (string.length>1) {

            if (!isValid(string.slice(1), stack, obj)) return obj.result=false;

    }

    else {

        if (string[0] != ')') return obj.result=false;

    }

    return obj.result = (stack.length == 0);

}

Код программы без рекурсии (идея алгоритма та же, плюсом добавлена возможность вводить скобки разного типа, сложность такая же):

function isValid(s) {

    const brackets = {

      ")": "(",

      "]": "[",

      "}": "{"

    };

    const st = [];

    for (let i = 0; i < s.length; i++) {

      if (isClosedBracket(s[i])) {

        if (brackets[s[i]] !== st.pop()) return false;

      } else {

        st.push(s[i]);

      }

    }

    return st.length === 0;

  }

  function isClosedBracket(ch) {

    return [")", "]", "}"].indexOf(ch) > -1;

  }