20.1. Отвечать на запросы: число вхождений строки s в t. Предподсчет за O(|t|), ответ за O(|s|).

Решение:

Заменяем $\mathbf{t_1}$ = **t$**. Строим суффиксное дерево по $\mathbf{t_1}$. Запускаем **dfs** с корня и для каждой вершины будем также хранить число (кол-во листов в её поддереве). Число, которое будет хранится в вершине до которого мы дошли, находя **s** и будет ответом на задачу.

Функция DFS будет возвращать число и работать так:

Сначала заходим в вершину, если это лист заменяем **NumLeafs** (число, про которое говорилось выше) на 1. И не зависимо от вершины возвращаем **NumLeafs** для данной вершины. А родитель ловит и прибавляет данное число к своему.

14.2. Дан набор строк s_i. Найти самую короткую строку, которая не является префиксом никакой из них?

Решение:

```
struct Trie {
    map<char, Trie> next;
    bool isVertex = false;
};
```

Будем хранить бор в таком виде.

```
string rec(Trie &t) {
    cout << 1;
    if (!t.isVertex) return "";

    cout << 2;
    string ans;
    int length = INT_MAX;

    for (char i = 'a'; i <= 'z'; ++i) {
        string st = rec(t.next[i]);
        if (st.size() + 1 < length) {
            ans = i + st;
            length = ans.size();
        }
    }

    return ans;
}
```

Для решение напишем такую простую рекурсию, которая делает обход по бору и ищет минимальную по длине строку которой нет в нём.