# Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы М8О-307Б-20 МАИ *Чекменев Вячеслав.*

## Условие

1. Необходимо реализовать алгоритм построения суффиксного массива за n*logn. Построив такой массив, найти в нем образцы используя двоичный поиск.

2. Вариант задания: поиск в известном тексте неизвестных заранее образцов

## Метод решения

Реализовать построение суффиксного массива с помощью циклических строк, классов эквивалентности и цифровой сортировки. Для этого надо написать сортировку подсчетом и увеличивая размер циклической строки сортировать их по их классам эквивалентности. После построить массив для хранения всех lcp между соседними суффиксами. Потом методом бинарного поиска идти по массиву суффиксов и использовать lcp для пропуска уже проверенных символов.

## Описание программы

Программа состоит из трёх файлов: *TSuffArray.hpp* с реализацией суффиксного массива и необходимых алгоритмов, *TCountingSort.hpp* с реализацией сортировки подсчетом, *main.cpp* и *Makefile* для сборки программы.

## Дневник отладки

1. Написал суффиксный массив и наивный поиск, не прохожу по по времени на 9 тесте

2. Добавил lcp, получил опять TL, но на 11 тесте

3. исправил поиск остальных вхождений, сделал это с помощью lcp, все равно TL на 11

4. стал добавлять в массив вхождений номера суффиксов сразу после нахождения, прошел тесты

## Тест производительности

| Кол-во символов | время |
|---|---|
| 1000 | 0m0,032s |
| 10000 | 0m0,150s |
| 100000 | 0m0,558s |
| 1000000 | 0m5,698s |

Из тестов ясно видно, что алгоритм работает за $O(nlogn)$.

## Недочёты

Не получилось реализовать поиск все значений lcp через обход дерева, поэтому пришлось часто его считать в ручную.

## Вывод

Проделав лабораторную работу, я ознакомился с такой структурой данных как суффиксный массив, в отличие от суффиксного дерева, он не требует работы с указателями, что сильно уменьшает стоимость операций для работы с ним. Суффиксный массив можно строить и напрямую из суффиксного дерева за O(n), однако такой алгоритм предполагает то, что мы опять будем работать с указателями, что мне хотелось избежать. Также можно использовать алгоритм, который работает за O(n*logn), однако он менее очевидный. Однако я использовал его, так как ни разу не работал с такой структурой, в отличие сжатого trie.