МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное

образовательное учреждение

высшего образования

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

**по дисциплине**

**“ Алгоритмы построение и анализ”**

**Суффиксный массив**

Выполнил: студент гр. ФИб-3302-51-00 ё \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ПМиИ Разова Е. В. \_\_\_\_\_\_\_\_\_\_\_

Киров 2020

**Задание 1.** Дать ответы на следующие вопросы (по материалу лекции <https://www.youtube.com/watch?v=CIibwErm24I&list=PLlb7e2G7aSpRp0_fj1OkEqaC-ml0t0PHf&index=7> ):

- опредение суффиксного массива;

**Ответ**: Это отображение из множества рангов множество позиций. Отсортированный массив всех суффиксов строки.

- определение обратного суффиксного массива;

**Ответ**: Т.к. суффиксный массив является перестановкой, можно взять обратную перестановку и получить обратный суффиксный массив

**Задание 2.** Описать построение суффиксного массива для строки *abaaaabaaaabaaba* с помощью:  
1. наивного алгоритма построения суффиксного массива с временной сложностью O(n2logn);

**Ответ**:

Построим суфиксный массив по исхожной строке, затем отсортируем его сортировкой quicksort, вызывая функцию сравнения strcmp

Пример для строки *abaaaabaaaabaaba*

***abaaaabaaaabaaba 0 - a 15***

***baaaabaaaabaaba 1 -* *aaaabaaaabaaba 2***

***aaaabaaaabaaba 2 - aaaabaaba 7***

***aaabaaaabaaba 3 - aaabaaaabaaba 3***

***aabaaaabaaba 4 - aaabaaba 8***

***abaaaabaaba 5 - aaba 12***

***baaaabaaba 6 - aabaaaabaaba 4***

***aaaabaaba 7 - aabaaba 9***

***aaabaaba 8 - aba 13***

***aabaaba 9 - abaaaabaaaabaaba 0***

***abaaba 10 - abaaaabaaba 5***

***baaba 11 - abaaba 10***

***aaba 12 - ba 14***

***aba 13 - baaaabaaaabaaba 1***

***ba 14 - baaaabaaba 6***

***a 15 - baaba 11***

2. бинарного алгоритма построения суффиксного массива с временной сложностью O(nlogn).

**Ответ:** Идея состоит в том, чтобы использовать тот факт, что строки, которые необходимо отсортировать, являются суффиксами одной строки.

Сначала мы сортируем все суффиксы по первому символу, затем по первым 2 символам, затем по первым 4 символам и так далее, пока количество рассматриваемых символов меньше 2n.

Присвоим ранг всем суффиксам, используя значение ASCII первого символа.

**Построим суфиксный массив ранк**

***Abaaaabaaaabaaba 0 - 0***

***Baaaabaaaabaaba 1 -*  *1***

***Aaaabaaaabaaba 2 - 0***

***Aaabaaaabaaba 3 0***

***Aabaaaabaaba 4 0***

***Abaaaabaaba 5 0***

***Baaaabaaba 6 - 1***

***aaaabaaba 7 0***

***aaabaaba 8 - 0***

***aabaaba 9 - 0***

***abaaba 10 0***

***baaba 11 - 1***

***aaba 12 - 0***

***aba 13 - 0***

***ba 14 - 1***

***a 15 - 0***

Для каждого символа мы также сохраняем ранг следующего соседнего символа, т. Е. Ранг символа в str [i + 1] (это необходимо для сортировки суффиксов по первым 2 символам). Если последний то -1.

**Построим суфиксный массив ранк след ранк**

***Abaaaabaaaabaaba 0 - 0 1***

***Baaaabaaaabaaba 1 -*  *1 0***

***Aaaabaaaabaaba 2 - 0 0***

***Aaabaaaabaaba 3 0 0***

***Aabaaaabaaba 4 0 0***

***Abaaaabaaba 5 0 1***

***Baaaabaaba 6 - 1 0***

***aaaabaaba 7 0 0***

***aaabaaba 8 - 0 0***

***aabaaba 9 - 0 0***

***abaaba 10 0 1***

***baaba 11 - 1 0***

***aaba 12 - 0 0***

***aba 13 - 0 1***

***ba 14 - 1 0***

***a 15 - 0 -1***

Потом отсортировать все суффиксы по рангу и смежному рангу. Ранг считается первой цифрой и, а смежный ранг считается второй цифрой. Когда по первым четырем то str [i + 1] и т.д.

***a 15 0 -1***

***Aaaabaaaabaaba 2 0 0***

***aaaabaaba 7 0 0***

***Aaabaaaabaaba 3 0 0***

***aaabaaba 8 - 0 0***

***aaba 12 - 0 0***

***Aabaaaabaaba 4 0 0***

***aabaaba 9 - 0 0***

***aba 13 - 0 1***

***Abaaaabaaaabaaba 0 - 0 1***

***Abaaaabaaba 5 0 1***

***abaaba 10 0 1***

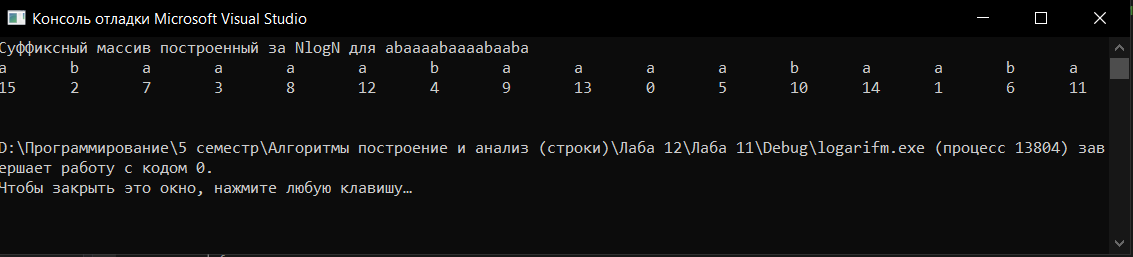
***ba 14 - 1 0***

***Baaaabaaaabaaba 1 -*  *1 0***

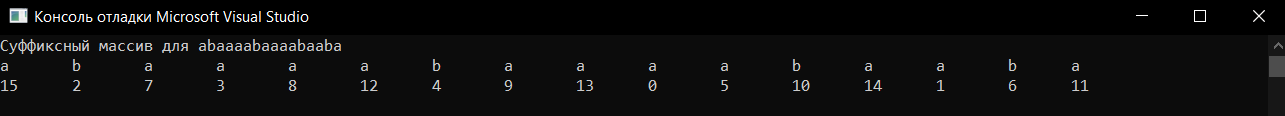
***Baaaabaaba 6 - 1 0***

***baaba 11 - 1 0***

**Задание 3.** Реализовать алгоритм построения суффиксного массива с временной сложностью O(n2logn).

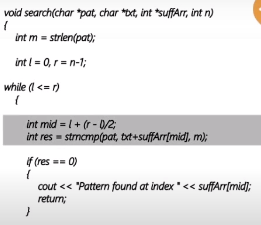


**Задание 4.** Реализовать алгоритм построения суффиксного массива с временной сложностью O(nlogn).

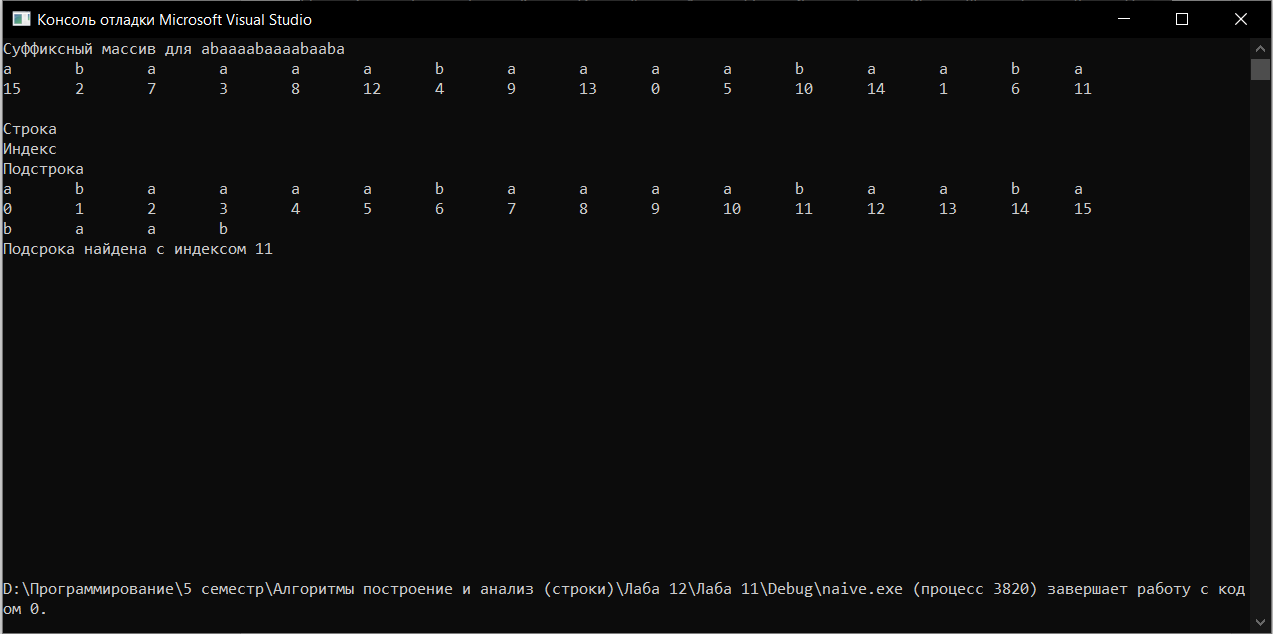


**Задание 5.** Реализовать алгоритм поиска шаблона при помощи суффиксного массива.

Алгоритм поиска (бинарный поиск):



Результат:



**Листинг программы с комментариями.**

Суффиксный массив с временной сложностью O(n2logn) + поиск шаблона.

#include <iostream>

#include <cstring>

#include <algorithm>

using namespace std;

// Структура для cуффиксного массива

struct suffix

{

int index; // Чтобы хранить исходный индекс

int rank[2]; // Чтобы хранить следующую пары рангов

};

// Функция сравнения, для сравнения двух суффиксов

// Сравнивает две пары, возвращает 1, если первая пара меньше

int sravnenie(struct suffix a, struct suffix b)

{

return (a.rank[0] == b.rank[0]) ? (a.rank[1] < b.rank[1] ? 1 : 0) :

(a.rank[0] < b.rank[0] ? 1 : 0);

}

// построение суфиксного массива

int \*buildSuffixArray(char \*txt, int n)

{

//Структура для хранения суффиксов и их индексов

struct suffix suffixes[16];

// Сохраняем суффиксы и их индексы в массиве структур, что бы потом отсортировать суффиксы по алфавиту

//и сохранить их старые индексы при сортировке

for (int i = 0; i < n; i++)

{

suffixes[i].index = i;

suffixes[i].rank[0] = txt[i] - 'a';

suffixes[i].rank[1] = ((i + 1) < n) ? (txt[i + 1] - 'a') : -1;

}

sort(suffixes, suffixes + n, sravnenie);

// На этом этапе все суффиксы отсортированы по первым двум символам.

//Затем оритурем суффиксы по первым 4 символам,по 8 символам и д

int ind[16]; // Этот массив нужен для получения индекса в суффиксах из исходного индекса.нужно для получения следующего суффикса.

for (int k = 4; k < 2 \* n; k = k \* 2)

{

// Присвоение значений ранга и индекса первому суффиксу

int rank = 0;

int prev\_rank = suffixes[0].rank[0];

suffixes[0].rank[0] = rank;

ind[suffixes[0].index] = 0;

// Присвоение ранга суффиксам

for (int i = 1; i < n; i++)

{

// Если первый и следующий ранги совпадают с рангом предыдущего суффикса в массиве,

//присваиваем этому суффиксу тот же новый ранг либо наоборот увеличиваем ранг

if (suffixes[i].rank[0] == prev\_rank &&

suffixes[i].rank[1] == suffixes[i - 1].rank[1])

{

prev\_rank = suffixes[i].rank[0];

suffixes[i].rank[0] = rank;

}

else

{

prev\_rank = suffixes[i].rank[0];

suffixes[i].rank[0] = ++rank;

}

ind[suffixes[i].index] = i;

}

// назначение следующиго ранг каждому суффиксу

for (int i = 0; i < n; i++)

{

int nextindex = suffixes[i].index + k / 2;

suffixes[i].rank[1] = (nextindex < n) ?

suffixes[ind[nextindex]].rank[0] : -1;

}

// сортируем суффиксы по первым k символам

sort(suffixes, suffixes + n, sravnenie);

}

// храним индексы всех отсортированных суффиксов в массиве суффиксов

int \*suffixArr = new int[n];

for (int i = 0; i < n; i++)

suffixArr[i] = suffixes[i].index;

return suffixArr;

}

void printArr(int arr[], int n, char \*txt)

{

for (int i = 0; i < n; i++)

cout << txt[i] << "\t";

cout << endl;

for (int i = 0; i < n; i++)

cout << arr[i] << "\t";

cout << endl;

cout << endl;

}

int main()

{

setlocale(0, "");

char txt[] = "abaaaabaaaabaaba";

int n = strlen(txt);

int \*suffixArr = buildSuffixArray(txt, n);

cout << "Суффиксный массив построенный за NlogN для " << txt << endl;

printArr(suffixArr, n, txt);

return 0;

}

Суффиксный массив с временной сложностью O(nlogn).

#include <iostream>

#include <cstring>

#include <algorithm>

using namespace std;

// Структура для cуффиксного массива

struct suffix

{

int index; // Чтобы хранить исходный индекс

int rank[2]; // Чтобы хранить следующую пары рангов

};

// Функция сравнения, для сравнения двух суффиксов

// Сравнивает две пары, возвращает 1, если первая пара меньше

int sravnenie(struct suffix a, struct suffix b)

{

return (a.rank[0] == b.rank[0]) ? (a.rank[1] < b.rank[1] ? 1 : 0) :

(a.rank[0] < b.rank[0] ? 1 : 0);

}

// построение суфиксного массива

int \*buildSuffixArray(char \*txt, int n)

{

//Структура для хранения суффиксов и их индексов

struct suffix suffixes[16];

// Сохраняем суффиксы и их индексы в массиве структур, что бы потом отсортировать суффиксы по алфавиту

//и сохранить их старые индексы при сортировке

for (int i = 0; i < n; i++)

{

suffixes[i].index = i;

suffixes[i].rank[0] = txt[i] - 'a';

suffixes[i].rank[1] = ((i + 1) < n) ? (txt[i + 1] - 'a') : -1;

}

sort(suffixes, suffixes + n, sravnenie);

// На этом этапе все суффиксы отсортированы по первым двум символам.

//Затем оритурем суффиксы по первым 4 символам,по 8 символам и д

int ind[16]; // Этот массив нужен для получения индекса в суффиксах из исходного индекса.нужно для получения следующего суффикса.

for (int k = 4; k < 2 \* n; k = k \* 2)

{

// Присвоение значений ранга и индекса первому суффиксу

int rank = 0;

int prev\_rank = suffixes[0].rank[0];

suffixes[0].rank[0] = rank;

ind[suffixes[0].index] = 0;

// Присвоение ранга суффиксам

for (int i = 1; i < n; i++)

{

// Если первый и следующий ранги совпадают с рангом предыдущего суффикса в массиве,

//присваиваем этому суффиксу тот же новый ранг либо наоборот увеличиваем ранг

if (suffixes[i].rank[0] == prev\_rank &&

suffixes[i].rank[1] == suffixes[i - 1].rank[1])

{

prev\_rank = suffixes[i].rank[0];

suffixes[i].rank[0] = rank;

}

else

{

prev\_rank = suffixes[i].rank[0];

suffixes[i].rank[0] = ++rank;

}

ind[suffixes[i].index] = i;

}

// назначение следующиго ранг каждому суффиксу

for (int i = 0; i < n; i++)

{

int nextindex = suffixes[i].index + k / 2;

suffixes[i].rank[1] = (nextindex < n) ?

suffixes[ind[nextindex]].rank[0] : -1;

}

// сортируем суффиксы по первым k символам

sort(suffixes, suffixes + n, sravnenie);

}

// храним индексы всех отсортированных суффиксов в массиве суффиксов

int \*suffixArr = new int[n];

for (int i = 0; i < n; i++)

suffixArr[i] = suffixes[i].index;

return suffixArr;

}

void printArr(int arr[], int n, char \*txt)

{

for (int i = 0; i < n; i++)

cout << txt[i] << "\t";

cout << endl;

for (int i = 0; i < n; i++)

cout << arr[i] << "\t";

cout << endl;

cout << endl;

}

int main()

{

setlocale(0, "");

char txt[] = "abaaaabaaaabaaba";

int n = strlen(txt);

int \*suffixArr = buildSuffixArray(txt, n);

cout << "Суффиксный массив построенный за NlogN для " << txt << endl;

printArr(suffixArr, n, txt);

return 0;

}