

*Государственное образовательное учреждение высшего
профессионального образования*
**«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)**

ЛАБОРАТОРНАЯ РАБОТА №7
ПО КУРСУ «АНАЛИЗ АЛГОРИТМОВ»

Поиск подстроки в строке

Выполнил: Сорокин А.П., гр. ИУ7-52Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019 г.

Оглавление

Введение	2
1 Аналитическая часть	3
1.1 Задачи	3
1.2 Описание алгоритмов	3
1.2.1 Стандартный алгоритм	3
1.2.2 Алгоритм Кнута-Морриса-Пратта	3
1.2.3 Алгоритм Бойера-Мура	3
2 Конструкторская часть	5
3 Технологическая часть	7
3.1 Средства реализации	7
3.2 Реализации алгоритмов	7
3.3 Тесты	9
4 Экспериментальная часть	10
4.1 Примеры работы	10
4.2 Сравнение работы алгоритмов	10
Заключение	12
Литература	13

Введение

Поиск информации - одно из основных использований компьютера, и быстрый поиск точно заданной подстроки в строке является одной из самых простейших задач поиска информации. Однако эта задача является чрезвычайно важной. Данная функция встроена в различные текстовые редакторы и базы данных, что существенно ускоряет процесс поиска информации и редактирование фрагментов. В настоящее время функции поиска подстроки в строке инкапсулированы во многие высокоуровневые языки программирования. Но стоит помнить, что стандартные функции далеко не самые оптимальные и эффективные, и если основной задачей программы является нахождение подстроки в строке, то необходимо знать принципы организации функций поиска. Также не нужно забывать, что область применения функций поиска не ограничивается одними текстовыми редакторами и базами данных, наоборот, расширяется на различные сферы человеческой жизни.

1. Аналитическая часть

1.1 Задачи

Цель лабораторной работы - изучение особенностей работы алгоритмов Кнута-Морриса-Пратта и Бойера-Мура.

Для того чтобы добиться этой цели, были поставлены следующие задачи:

- применить знания программирования для реализации данных алгоритмов;
- получить практические навыки во время выполнения задания;
- экспериментально подтвердить различия во временной эффективности работы стандартного алгоритма поиска подстроки в строке, алгоритма Кнута-Морриса-Пратта и Бойера-Мура.

1.2 Описание алгоритмов

1.2.1 Стандартный алгоритм

Стандартный алгоритм начинает со сравнения первого символа текста с первым символом подстроки. Если они совпадают, то происходит переход ко второму символу текста и подстроки. При совпадении сравниваются следующие символы. Так продолжается до тех пор, пока не окажется, что подстрока целиком совпала с отрезком текста, или пока не встретятся несовпадающие символы. В первом случае задача решена, во втором мы сдвигаем указатель текущего положения в тексте на один символ и заново начинаем сравнение с подстрокой [1].

1.2.2 Алгоритм Кнута-Морриса-Пратта

Это — один из самых известных алгоритмов решения задачи поиска образца P в тексте T , имеющий временную оценку $O(n)$, т. е. в нем поиск образца осуществляется за время, пропорциональное длине текста. В какой-то мере этот результат можно считать «точкой отсчета» в стремлении специалистов по информатике создать новые алгоритмы решения данной классической задачи. Здесь образец, как и в простом алгоритме, последовательно «прикладывается» к тексту и осуществляется пошаговое сравнение символов. Но если в простом алгоритме после несовпадения в какой-то позиции осуществляется сдвиг на одну позицию, то в рассматриваемом, за счет предварительного анализа P , сдвиг выполняется в некоторых случаях более чем на один символ, в отличие от стандартного алгоритма [2].

1.2.3 Алгоритм Бойера-Мура

Алгоритм Бойера-Мура осуществляет сравнение с образцом справа налево, а не слева направо. Исследуя искомый образец, можно осуществлять более эффективные прыжки в

тексте при обнаружении несовпадения. В этом алгоритме кроме таблицы суффиксов применяется таблица стоп-символов. Она заполняется для каждого символа в алфавите. Для каждого встречающегося в подстроке символа таблица заполняется по принципу максимальной позиции символа в строке, за исключением последнего символа. При определении сдвига при очередном несовпадении строк, выбирается максимальное значение из таблицы суффиксов и стоп-символов[1].

2. Конструкторская часть

В данном разделе представлены принципы работы алгоритмов и их схемы. На рисунках 2.1 и 2.2 представлены схемы алгоритмов Кнута-Морриса-Пратта и Бойера-Мура.

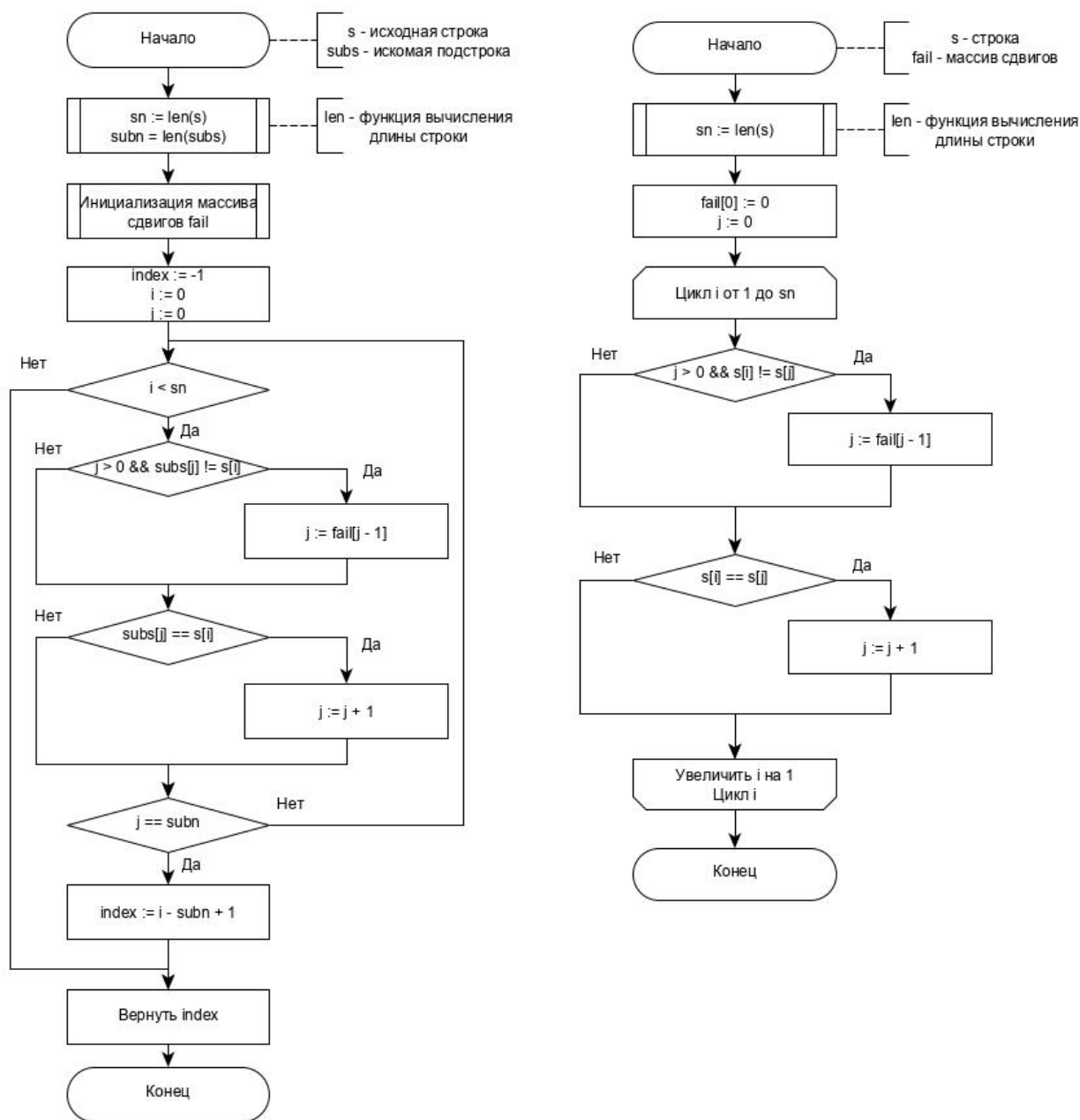


Рис. 2.1: Схема алгоритма Кнута-Морриса-Пратта (слева - основной алгоритм, справа - алгоритм вычисления массива fail)

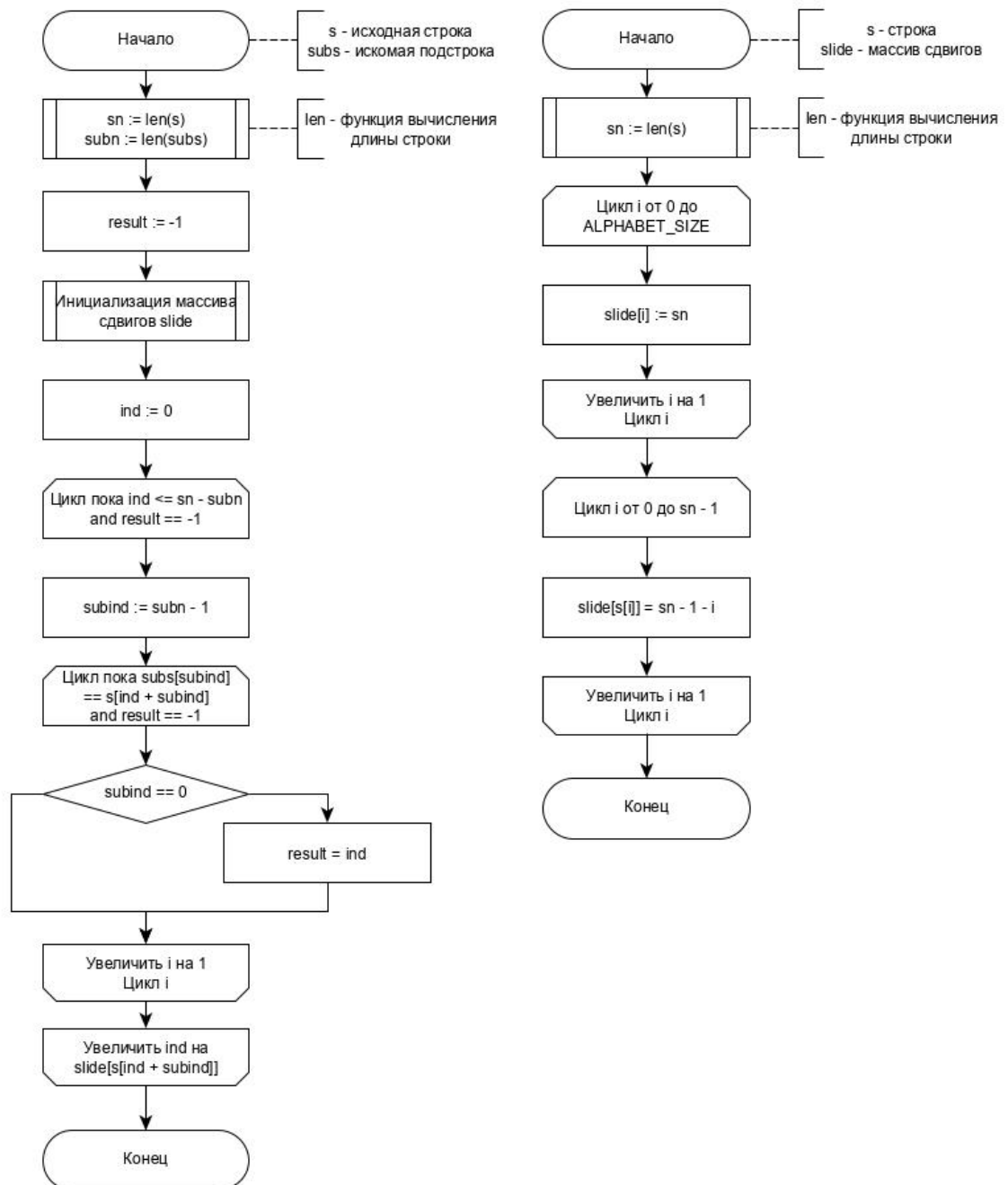


Рис. 2.2: Схема алгоритма Бойера-Мура (слева - основной алгоритм, справа - алгоритм вычисления массива сдвигов)

3. Технологическая часть

В данном разделе приведены требования к программному обеспечению, средствам реализации, а также листинги кода.

3.1 Средства реализации

Для реализации программы был использован язык C++ [3]. Для замера процессорного времени была использована функция `rdtsc()` из библиотеки `stdrin.h`.

3.2 Реализации алгоритмов

В листингах 3.1, 3.2, 3.3, 3.4 представлены коды реализаций рассматриваемых в данной лабораторной работе алгоритмов поиска подстроки в строке.

Листинг 3.1: Алгоритм Кнута-Морриса-Пратта

```
1 int substr_kmp(const std::string &s, const std::string &subs)
2 {
3     if (s.empty() || subs.empty())
4         return -1;
5
6     int sn = s.length(), subn = subs.length();
7     int *fail = new int[subn];
8     fail_compute(subs, fail);
9
10    int index = -1;
11    for (int i = 0, j = 0; i < sn && index == -1; i++)
12    {
13        if (j > 0 && subs[j] != s[i])
14            j = fail[j - 1];
15        if (subs[j] == s[i])
16            j++;
17        if (j == subn)
18            index = i - subn + 1;
19    }
20    delete [] fail;
21    return index;
22 }
```

Листинг 3.2: Вычисление массива сдвигов для алгоритма Кнута-Морриса-Пратта

```
1 void fail_compute(const std::string &s, int *fail)
2 {
3     if (s.empty())
```



```

4     return;
5
6     fail[0] = 0;
7     for (unsigned i = 1, j = 0; i < s.length(); i++)
8     {
9         if (j > 0 && s[i] != s[j])
10            j = fail[j - 1];
11        if (s[i] == s[j])
12            j++;
13        fail[i] = j;
14    }
15 }

```

Листинг 3.3: Алгоритм Бойера-Мура

```

1 int substr_bm(const std::string &s, const std::string &subs)
2 {
3     int sn = s.length(), subn = subs.length();
4     int result = -1;
5     int slide[ALPHABET_SIZE];
6     get_slide(subs, slide);
7
8     int ind = 0;
9     while (ind <= sn - subn && result == -1)
10    {
11        int subind = subn - 1;
12        for (; subs[subind] == s[ind + subind] && result == -1; subind--)
13        {
14            if (subind == 0)
15                result = ind;
16        }
17        ind += slide[(int)s[ind + subind]];
18    }
19    return result;
20 }

```

Листинг 3.4: Вычисление массива сдвигов для алгоритма Бойера-Мура

```

1 void get_slide(const std::string &subs, int *slide)
2 {
3     if (subs.empty())
4         return;
5
6     int subn = subs.length();
7     for (int i = 0; i < ALPHABET_SIZE; i++)
8         slide[i] = subn;
9     for (int i = 0; i < subn - 1; i++)
10        slide[(int)subs[i]] = subn - 1 - i;
11 }

```

3.3 Тесты

Для проверки корректности работы были подготовлены функциональные тесты, представленные в таблице 3.1. В данной таблице λ обозначает пустую строку, а результат, равный -1, означает, что подстрока не входит в исходную строку.

Таблица 3.1: Функциональные тесты

Строка	Подстрока	Ожидание	Кнута-Морриса-Прата	Бойера-Мура
λ	λ	-1	-1	-1
λ	a	-1	-1	-1
a	λ	-1	-1	-1
a	a	0	0	0
abc	a	0	0	0
a	abc	-1	-1	-1
abc	abc	0	0	0
abab	ab	0	0	0
aabab	ab	1	1	1
baaab	ab	3	3	3
baa	ab	-1	-1	-1

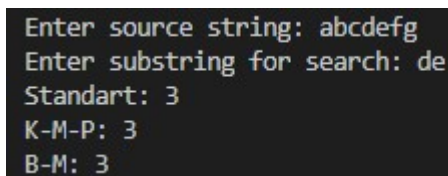
В результате проверки реализации всех алгоритмов прошли тесты.

4. Экспериментальная часть

В данном разделе будет проведён сравнительный анализ реализаций стандартного алгоритма поиска подстроки в строке, алгоритма Кнута-Морриса-Пратта и Бойера-Мура.

4.1 Примеры работы

На рисунке 4.1 представлен пример работы программы, демонстрирующий корректную работу алгоритмов.



```
Enter source string: abcdefg
Enter substring for search: de
Standart: 3
K-M-P: 3
B-M: 3
```

Рис. 4.1: Пример работы программы

4.2 Сравнение работы алгоритмов

Для сравнения времени работы алгоритмов поиска подстроки в строке были использованы случайные строки длиной от 10 до 200 с шагом 10 и подстрока фиксированной длины 2. Эксперимент для более точного результата повторялся 100 раз. Итоговый результат рассчитывался как средний из полученных результатов. Результаты измерений показаны в таблице 4.1 и на рисунке 4.2.

Таблица 4.1: Время работы алгоритмов поиска подстроки в строке в тактах процессора

Размер строки	Стандартный	Алг-м Кнута-Морриса-Пратта	Алг-м Бойера-Мура
10	222	543	2412
20	305	525	2529
30	395	587	1920
40	817	674	2297
50	604	1114	2318
60	692	841	2091
70	1112	914	2492
80	880	988	2560
90	987	1415	2235
100	1120	1380	2515
110	1219	1265	2718
120	1583	1630	2706
130	1315	1708	2845
140	1450	1444	2469
150	1536	1551	2541
160	1909	1902	2982
170	1681	2016	2625
180	1801	1779	3431
190	2481	2208	3555
200	2036	2210	4056

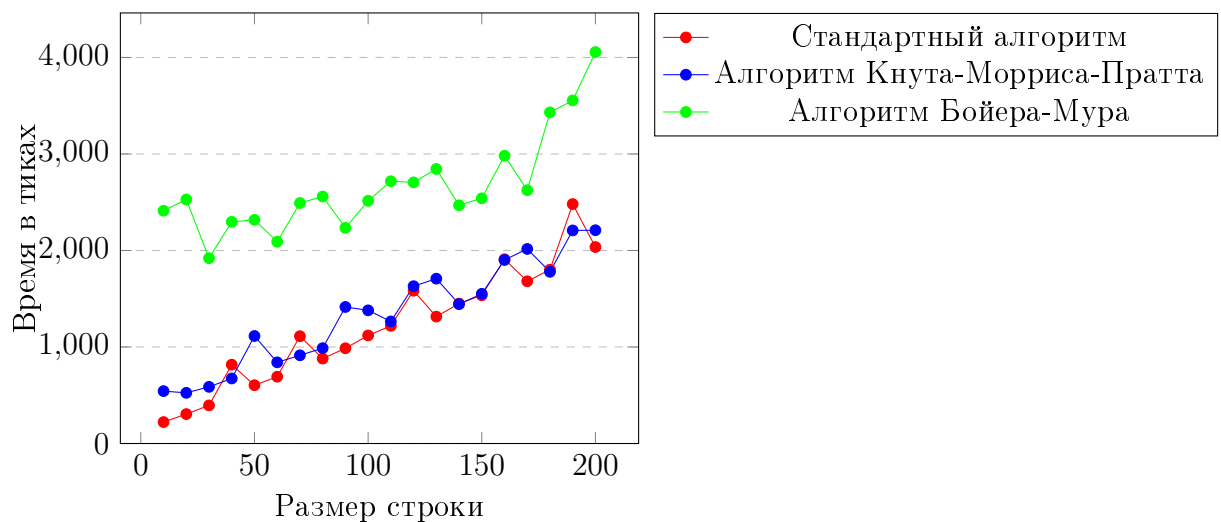


Рис. 4.2: График времени работы алгоритмов поиска подстроки в строке

Эксперименты показали, что наиболее эффективным алгоритмом из трех рассмотренных оказался стандартный алгоритм, а самым неэффективным оказался алгоритм Бойера-Мура.

Заключение

В ходе выполнения данной лабораторной работы были изучены два алгоритма для поиск подстроки в строке - Кнута-Морриса-Пратта и Бойера-Мура. Во время разработки программного обеспечения были получены практические навыки реализации указанных алгоритмов.

Литература

- [1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход
- [2] Окулов С.М. Алгоритмы обработки строк. – М.: БИНОМ. Лаборатория знаний, 2009.
- [3] <https://cppreference.com/> [Электронный ресурс]