МГТУ им. Баумана

Лабораторная работа №7

По курсу: "Анализ алгоритмов"

Поиск подстроки в строке

Работу выполнила: Лаврова Анастасия, ИУ7-55Б

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

Москва, 2019

Оглавление

Введение

Целью данной лабораторной работы является изучение алгоритмов поиска подстроки в строке, в частности, алгоритма Кнута-Морриса-Пратта и алгоритма Бойера-Мура.
Задачи лабораторной работы: реализовать алгоритмы Кнута-Морриса-Пратта и Бойера-Мура.

1 | Аналитическая часть

Поиск подстроки в строке — одна из простейших задач поиска информации. Поиск подстроки в строке применяется в виде встроенной функции в текстовых редакторах, СУБД, поисковых машинах, языках программирования и т. п.

Пусть дана некоторая строка T (текст) и подстрока W (шаблон). Задача поиска подстроки сводится к поиску вхождения шаблона W в указанной строке T. Строго задача формулируется следующим образом: пусть задан массив T из N элементов и массив W из M элементов, $0 < M \leqslant N$. Если алгоритм поиска подстроки обнаруживает вхождение W в T, то возвращается индекс, указывающий на первое совпадение подстроки со строкой.

1.1 Стандартный алгоритм

Простейшим алгоритмом является примитивный алгоритм. Рассмотрим псевдокод этого алгоритма:

1) I = 1, J = 1
2) Сравнение T[I] с W[J]
3) Совпадение: J = J + 1, I = I +1
4) Несовпадение: J = 1, I = I + 1
5) Если J = M, то подстрока найдена
6) Если I + M > N, то подстрока отсутствует

1.2 Алгоритм Кнута-Морриса-Пратта

Идея алгоритма заключается в том, что при каждом несовпадении T[I] и W[J] мы сдвигаемся не на единицу, а на J, так как меньшие сдвиги не приведут к полному совпадению. Однако, этот алгоритм поиска дает выигрыш только тогда, когда несовпадению предшествовало некоторое число совпадений, иначе алгоритм аналогичен примитивному. Так как совпадения встречаются реже, чем несовпадения, выигрыш в большинстве случаев незначителен.
Алгоритм Кнута-Морриса-Пратта основан на принципе конечного автомата.В этом алгоритме состояния помечаются символами, совпадение с которыми должно в данный момент произойти. Из каждого состояния имеется два перехода: один соответствует успешному сравнению, другой — несовпадению. Успешное сравнение переводит нас в следующий узел автомата, а в случае несовпадения мы попадаемв предыдущий узел, отвечающий образцу.

Рассмотрим нахождение в строке "abeccacbadbabbad"подстроки "abbad"и построим следующий автомат (рис 1.1), где состояния маркируются ожидаемыми символами:

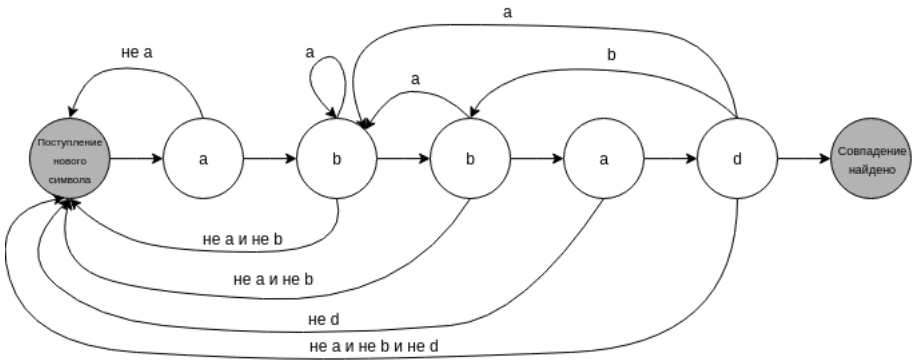

Рис. 1.1: Конечный автомат в алгоритме Кнута-Морриса-Пратта

1.3 Алгоритм Бойера-Мура

Алгоритм Бойера-Мура из трех алгоритмов считается наиболее быстрым. В среднем он делает сравнений меньше, чем N. Данный алгоритм считается стандартным для поиска на странице браузера или в текстовых редакторах.
Преимущество этого алгоритма в том, что ценой некоторого количества предварительных вычислений над шаблоном (но не над строкой, в которой ведётся поиск) шаблон сравнивается с исходным текстом не во всех позициях — часть проверок пропускаются как заведомо не дающие результата. Идея БМ-поиска – сравнение символов начинается с конца образца, а не с начала, то есть сравнение отдельных символов происходит справа налево. Затем с помощью некоторой эвристической процедуры вычисляется величина сдвига вправо s. И снова производится сравнение символов, начиная с конца образца.

Рассмотрим нахождение в строке "abeccacbadbabbad"подстроки "abbad"и построим следующий автомат (рис 1.1), где состояния маркируются ожидаемыми символами:

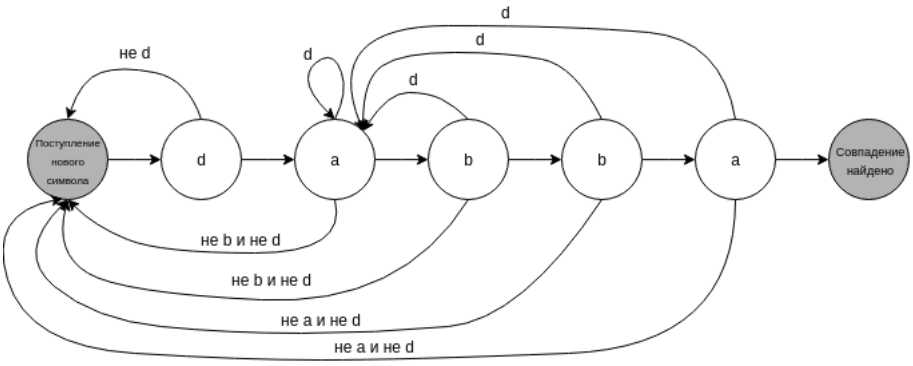

Рис. 1.2: Конечный автомат в алгоритме Бойера-Мура

2 | Конструкторская часть

Требования к вводу:
Программе на вход подается две строки: текст и шаблон
Требования к программе:
Программа должна находить первое вхождение шаблона в текст и его индекс (индексация строк начинается с нуля)

2.1 Разработка реализации алгоритмов

На рис. 2.1 представлена схема алгоритма Кнута-Морриса-Пратта:
На рис. 2.2 представлена схема алгоритма Бойера-Мура:

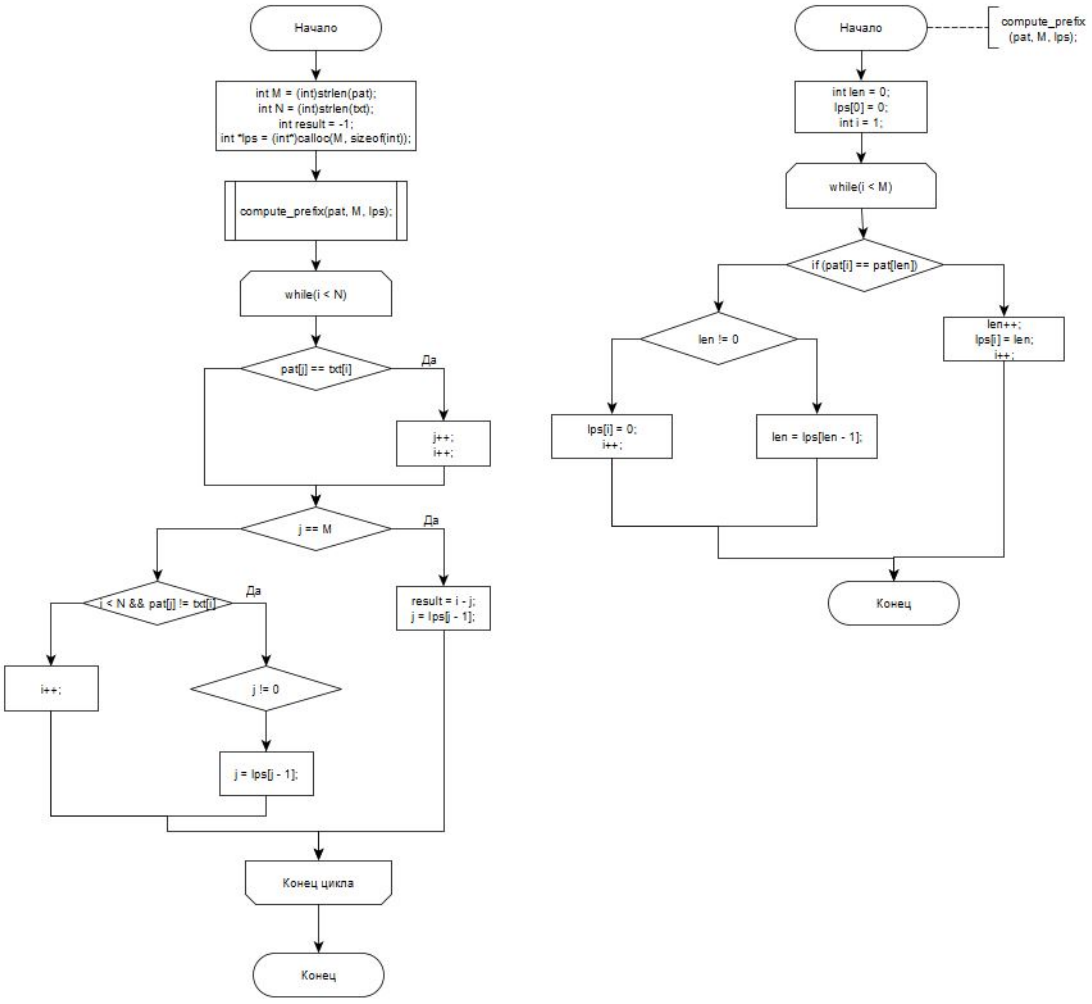

Рис. 2.1: Схема алгоритма Кнута-Морриса-Пратта

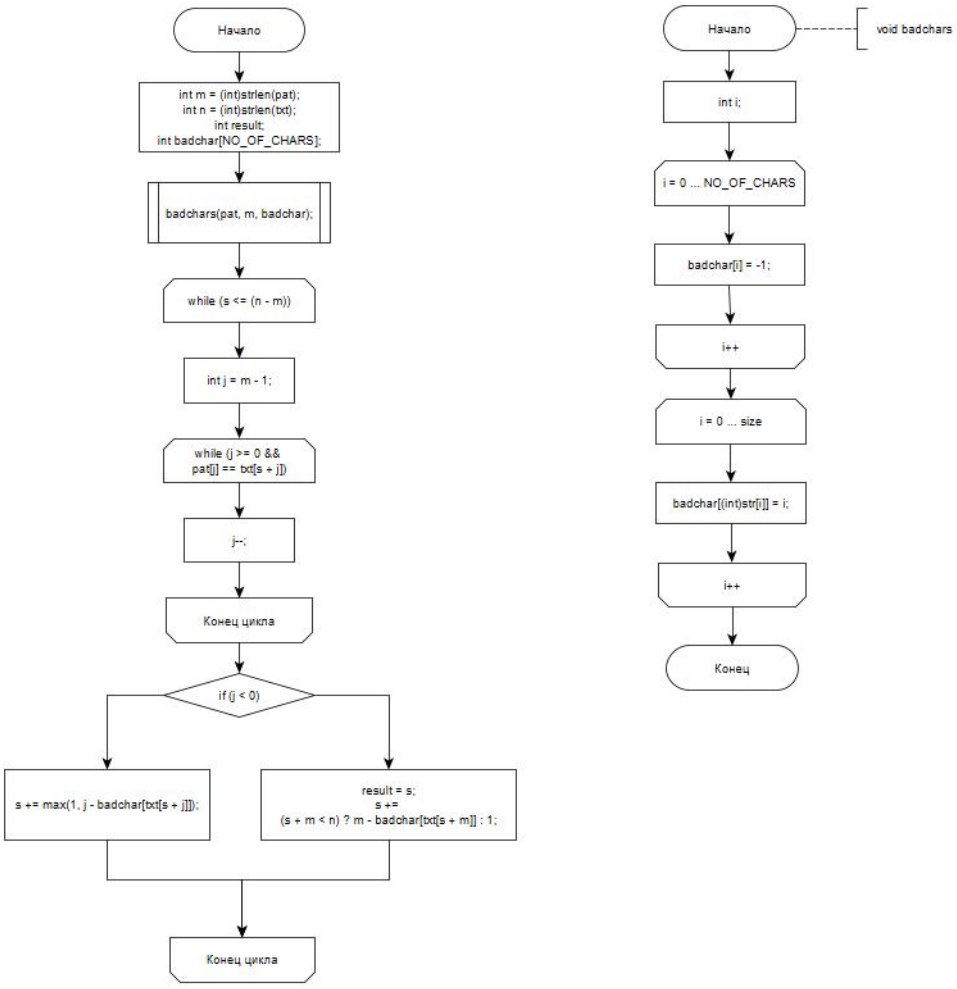

Рис. 2.2: Схема алгоритма Бойера-Мура

3 | Технологическая часть

3.1 Выбор ЯП

Для реализации программ я выбрала язык программирования C++, так имею большой опыт работы с ним. Среда разработки - Visual Studio.

В листинге 3.1 рассмотрен алгоритм Кнута-Морриса-Пратта, а в листинге 3.2 представлен алгоритм Бойера-Мура.

Листинг 3.1: Реализация алгоритма Кнута-Морриса-Пратта

```
int kmp_search(char* pat, char* txt)
{
  int M = (int)strlen(pat);
  int N = (int)strlen(txt);

  int result = -1;

  int *lps = (int*)calloc(M, sizeof(int));

  compute_prefix(pat, M, lps);

  int i = 0;
  int j = 0;
  while (i < N)
  {
    if (pat[j] == txt[i])
    {
      j++;
      i++;
```

```
20      }
21
22      if (j == M)
23      {
24        result = i - j;
25        j = lps[j - 1];
26      }
27
28      else if (i < N && pat[j] != txt[i])
29      {
30        if (j != 0)
31          j = lps[j - 1];
32        else
33          i++;
34      }
35    }
36    return result;
37  }
38
39  void compute_prefix(char* pat, int M, int* lps)
40  {
41
42    int len = 0;
43    lps[0] = 0;
44
45    int i = 1;
46    while (i < M)
47    {
48      if (pat[i] == pat[len])
49      {
50        len++;
51        lps[i] = len;
52        i++;
53      }
54      else
55      {
56        if (len != 0)
57        {
58          len = lps[len - 1];
59        }
```

```
60        else
61        {
62          lps[i] = 0;
63          i++;
64        }
65      }
66    }
67  }
```

Листинг 3.2: Реализация алгоритма Бойера-Мура

```
1  int bm_search(char *txt, char *pat)
2  {
3    int m = (int)strlen(pat);
4    int n = (int)strlen(txt);
5
6    int result;
7
8    int badchar[NO_OF_CHARS];
9
10   badchars(pat, m, badchar);
11
12   int s = 0;
13   while (s <= (n - m))
14   {
15     int j = m - 1;
16
17     while (j >= 0 && pat[j] == txt[s + j])
18       j--;
19
20     if (j < 0)
21     {
22       result = s;
23
24       s += (s + m < n) ? m - badchar[txt[s + m]] : 1;
25
26     }
27     else
28       s += max(1, j - badchar[txt[s + j]]);
29   }
30
```

```
31    return result;
32  }
33
34  void badchars(char *str, int size,
35    int badchar[NO_OF_CHARS])
36  {
37    int i;
38
39    for (i = 0; i < NO_OF_CHARS; i++)
40      badchar[i] = -1;
41
42    for (i = 0; i < size; i++)
43      badchar[(int)str[i]] = i;
44  }
```

4 | Исследовательская часть

4.1 Тесты

Далее приведены примеры работы программы (таблица 4.1):

Таблица 4.1: Набор тестовых данных

Текст	Шаблон	Ожидаемый индекс
“there they are “	“they “	6
“there they are “	“there “	0
“there they are “	“are “	11
“there they are “	“there they are “	0

На рис. 4.1 представлен пример работы программы:

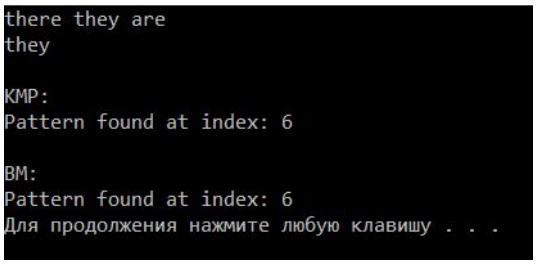

Рис. 4.1: Схема алгоритма Кнута-Морриса-Пратта

4.2 Пример работы алгоритма Кнута-Морриса-Пратта

Пусть у нас есть алфавит из пяти символов: a, b, c, d, e и мы хотим найти вхождение образца “abbad” в строке “abeccacbadbabbad”.

Таблица преффиксов будет выглядеть так.

a	b	b	a	d
0	0	0	1	0

Начало поиска.

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
a	b	b	a	d											
		a	b	b	a	d									
			a	b	b	a	d								
				a	b	b	a	d							
					a	b	b	a	d						
						a	b	b	a	d					
							a	b	b	a	d				
								a	b	b	a	d			
									a	b	b	a	d		
										a	b	b	a	d	
											a	b	b	a	d

Совпадение найдено.

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
		a	b	b	a	d									

4.3 Пример работы алгоритма Бойера-Мура

Пусть у нас есть алфавит из пяти символов: a, b, c, d, e и мы хотим найти вхождение образца “abbad” в строке “abeccacbadbabbad”.

Таблица смещений будет выглядеть так.

a	b	c	d	e
1	2	5	5	5

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
a	b	b	a	d											

Начало поиска.

Последний символ образца не совпадает с наложенным символом строки. Сдвигаем образец вправо на 5 позиций.

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
					a	b	b	a	d						

Второй символ не совпадает, сдвигаем на 5 символов вправо.

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
										a	b	b	a	d	

Последний символ не совпадает, сдвигаем на один символ вправо.

a	b	e	c	c	a	c	b	a	d	b	a	b	b	a	d
											a	b	b	a	d

Совпадение найдено.

4.4 Вывод

В данном разделе были приведены примеры работы программы.

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.