Journal of Siberian Federal University. Engineering & Technologies, 2017, 10(1), 126-135
~ ~ ~

УДК 681.3.06

Combined String Searching Algorithm

Roman Yu. Tsarev*,
Elena A. Tsareva and Alexey S. Chernigovskiy
Siberian Federal University
79 Svobodny, Krasnoyarsk, 660041, Russia

Received 17.12.2016, received in revised form 29.01.2016, accepted 17.01.2017

The string search problem is a classical problem of data processing. Despite a number of existing algorithms for solving this problem, the work in this direction continues. The algorithm proposed in the article develops the theoretical basis of the string search problem by combining the algorithms of two different classes, i.e. forward and backward string searching algorithms, namely, Knuth-Morris-Pratt algorithm and Bower-Moore algorithm. The paper provides the analysis of the proposed combined algorithm and comparison of its results with the basic algorithms, confirming the efficacy of the combined string search algorithm.

Keywords: pattern, search, data processing, combined algorithm.

Citation: Tsarev R.Yu., Tsareva E.A., Chernigovskiy A.S. Combined string searching algorithm, J. Sib. Fed. Univ. Eng. technol., 2017, 10(1), 126-135. DOI: 10.17516/1999-494X-2017-10-1-126-135.

Комбинированный алгоритм поиска образа в строке

Р.Ю. Царев, Е.А. Царева, А.С. Черниговский
Сибирский федеральный университет
Россия, 660041, Красноярск, пр. Свободный, 79

Проблема поиска образа в строке является классической задачей обработки данных. Несмотря на ряд существующих алгоритмов решения задачи, работа в этом направлении продолжается. Предложенный алгоритм развивает теоретические основы задачи поиска образа в строке, комбинируя алгоритмы двух разных классов с прямым и обратным проходом образа, а именно алгоритмы Кнута-Морриса-Пратта и Боуера-Мура. В статье приведен анализ работы предложенного комбинированного алгоритма и сравнение результатов его работы с базовыми алгоритмами, подтверждающее эффективность комбинированного алгоритма поиска образа в строке.

Ключевые слова: образ, поиск, обработка данных, комбинированный алгоритм.

* Corresponding author E-mail address: tsarev.sfu@mail.ru

Введение

Поиск образа в строке – одна из простых, но, тем не менее, крайне важных задач. Об этом говорит широкая область применения результатов ее решения: текстовые редакторы, обозреватели, средства анализа и распознания речи, а также извлечения знаний, архиваторы данных и пр. [1, 2].

За последние тридцать лет были предложены десятки различных алгоритмов, позволяющих решать задачи поиска с теми или иными особенностями. М. Ахмед с коллегами разработал алгоритм поиска образа в строке, который в отличие от других алгоритмов сравнения строк никогда не выполняет более n сравнений символов при обработке текста длины n [3]. Т. Лерок дал ряд алгоритмов, основанных на хешировании q-грамм [4]. К. Фредрикссон и С. Грабовский разработали параллельный алгоритм, развивающий алгоритм двоичного поиска строки [5, 6]. Они предложили новый подход для алгоритма поиска строки, позволяющий получить оптимальное время поиска для заданного расстояния Хэмминга [6]. Л. Хи, Б. Фанг и Дж. Суи применили понятие окна, размер которого равен длине образа. Экспериментальное сравнение с существующими алгоритмами показало его преимущества и эффективность [7]. А. Худаиб и др. в [8] представили алгоритм двух скользящих окон. Данный алгоритм использует два скользящих окона, длина каждого из них равна образу. Оба окна движутся одновременно с двух концов строки до тех пор, пока не произойдет совпадение образа со строкой или оба окна не достигнут середины строки.

Неугасающий интерес к этой проблемной области подтверждает важность и актуальность задачи поиска образа в строке. В данной статье предложен алгоритм, основанный на двух широко известных и хорошо себя зарекомендовавших алгоритмах поиска образа в строке. Подобное объединение позволяет повысить эффективность решения поставленной задачи.

1. Теоретические основы комбинированного алгоритма

Предлагаемый комбинированный алгоритм поиска образа в строке основан на двух алгоритмах. Один из них разработан Кнутом, Моррисом и Праттом, второй – Боуером и Муром. Эти алгоритмы относятся к двум довольно большим подклассам алгоритмов поиска образа в строке. При сравнении символов образа и строки алгоритм Кнута-Морриса-Пратта проходит образ от начала к концу, а алгоритм Боуера-Мура – от конца к началу. В работе обоих этих алгоритмов можно выделить два этапа:

1. Формирование таблицы d, используемой при сдвиге образа по строке.

2. Непосредственно поиск образа в строке.

Рассмотрим детально работу этих алгоритмов поиска образа в строке. Изложение будет сопровождаться примерами и пояснениями с использованием нотации языка программирования Си.

1.1. Принципы работы алгоритма Кнута-Морриса-Пратта

Данный алгоритм был разработан Д. Кнутом, В. Праттом и независимо от них Дж. Моррисом в 1974 г. Однако опубликован он был совместно только в 1977 г. [9]. Этот алгоритм основывается на том соображении, что после частичного совпадения начальной части образа с соответствующими символами строки мы обладаем некоторой информацией, полученной

на основе самого образа, которая позволит продвинуться по строке не на единицу, как при простом поиске, а дальше. Для этого алгоритм Кнута-Морриса-Пратта (или КМП-алгоритм) использует при сдвиге образа таблицу *d*, которая формируется еще до начала поиска образа в строке. Поскольку таблица *d*, формируемая согласно КМП-алгоритму, будет также использоваться в комбинированном алгоритме, обозначим ее как d_{KMP}.

Итак, таблица d_{KMP} для алгоритма поиска образа в строке Кнута-Морриса-Пратта формируется на основе образа и содержит значения, которые в дальнейшем будут использованы при вычислении величины сдвига образа. Размер данной таблицы равен длине образа, которую можно при программировании на языке Си определить при помощи функции *int strlen*(*char* *) библиотеки <*string.h*>. Таким образом, таблица d_{KMP} фактически является одномерным массивом, состоящим из числа элементов, равного количеству символов в образе.

Первый элемент массива d_{KMP} всегда равен минус единице.

Для остальных символов образа значение элементов таблицы d_{KMP} вычисляется следующим образом: значение $d_{KMP}[j]$, соответствующее *j*-му символу образа, равно максимальному числу символов, непосредственно предшествующих данному символу, совпадающих с началом образа. При этом если рассматриваемому символу предшествует *k* символов, то во внимание принимаются только *k*–1 предшествующих символов.

Рассмотрим формирование таблицы d_{KMP} на примере образа “*barbarian*”. Так, на рис. 1 пятому символу образа “*a*” непосредственно предшествует один символ “*b*”, что совпадает с символом “*b*” в самом начале образа. Оба символа “*b*” выделены полужирным шрифтом. Поскольку количество совпавших символов равно единице, это значение получает элемент таблицы, соответствующий пятому символу образа “*a*”. Шестому символу образа “*r*” непосредственно предшествуют два символа “*ba*”, что совпадает с символом “*ba*” в начале образа. Поскольку символу “*r*” предшествуют два символа, совпадающие с двумя первыми символами образа, то соответствующий элемент таблицы d_{KMP} получает значение два. Никаким другим символам образа “*barbarian*” не предшествуют символы, совпадающие с началом образа, поэтому соответствующие элементы таблицы d_{KMP} равны нулю.

Таким образом, значения элементов таблицы d_{KMP} не зависят от строки, в которой будет происходить поиск образа. Для определения значения элементов таблицы d_{KMP} необходимо найти самую длинную последовательность символов образа, непосредственно предшествующих рассматриваемому символу, которая совпадает полностью с началом образа. Так как значения элементов таблицы d_{KMP} зависят только от образа, необходимо их вычислить перед началом поиска.

Вторым этапом работы КМП алгоритма является сравнение символов образа и строки и вычисление сдвига образа в случае их несовпадения. Символы образа рассматриваются слева направо, т. е. от начала к концу образа. При несовпадении символов образа и строки образ сдвигается вправо по строке на следующую величину:

b	a	r	**b**	a	r	i	a	n
–1	0	0	0	1	2	0	0	0

Рис. 1. Образ и таблица d_{KMP}

$$j - d_{KMP}[j],$$

где j – это индекс текущего символа в образе, а $d_{KMP}[j]$ – значение таблицы d_{KMP}, соответствующее этому символу.

Приведем пример поиска в строке образа "*barbarian*". На рис. 2 показан процесс работы КМП-алгоритма, сравниваемые символы подчеркнуты.

Обратите внимание: при каждом несовпадении символов образ сдвигается на все пройденное расстояние, поскольку меньшие сдвиги не могут привести к полному совпадению.

Что касается эффективности алгоритма поиска образа в строке Кнута-Морриса-Пратта, то его разработчики показывают, что требуется $n + m$ сравнений символов, что значительно лучше, чем $n * m$ сравнений при простом поиске (здесь n – длина строки, m – длина образа, $0 < m < n$) [1, 9]. При этом указатель сканирования строки i никогда не возвращается назад, в то время как при простом поиске после несовпадения просмотр всегда начинается с первого символа образа и поэтому может включать символы строки, которые уже просматривались ранее.

Однако поиск образа в строке с использованием алгоритма Кнута-Морриса-Пратта дает выигрыш только в том случае, когда несовпадению символов из образа и строки предшествовало некоторое число совпадений. Если при сравнении образа и строки первые же символы различны, то образ сдвигается только на один символ. Продвижение образа более чем на единицу происходит лишь в этом случае, когда происходит совпадение нескольких символов образа и строки.

1.2. Принципы работы алгоритма Боуера-Мура

В 1977 г. Р.С. Боуер и Дж.С. Мур предложили алгоритм, который не только улучшает обработку самого плохого случая, но и дает выигрыш в общем случае [10]. Алгоритм поиска образа в строке Боуера-Мура (или БМ-алгоритм) основывается на сравнениях символов с конца образа, а не с начала. Скорость в алгоритме Боуера-Мура достигается за счет того, что удается пропускать те части текста, которые заведомо не участвуют в успешном сопоставлении.

Как и при работе КМП-алгоритма, перед началом поиска на основе образа создается таблица d, используемая в дальнейшем при смещении образа по строке. Так как при работе комбинированного алгоритма данная таблица также будет использоваться, обозначим ее d_{BM}.

Первоначально всем элементам таблицы d_{BM} присваивается значение, равное длине образа.

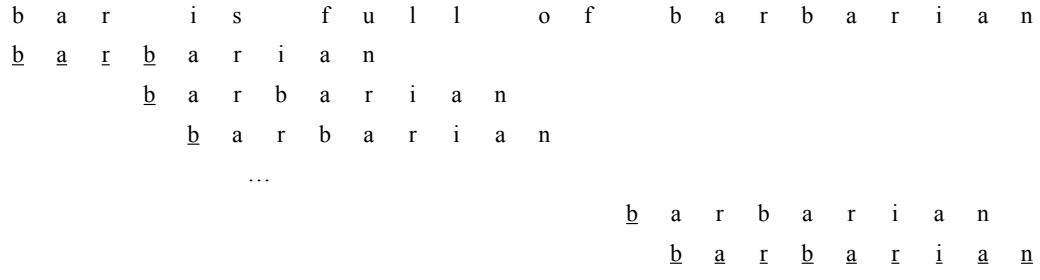

Рис. 2. Процесс работы алгоритма поиска образа в строке Кнута-Морриса-Пратта

Следующим шагом является присвоение каждому элементу таблицы d_{BM} значения, равного удаленности соответствующего символа образа от конца образа. На рис. 3 показан образ и удаленность каждого из его символов от конца образа. Однако если образ содержит несколько одинаковых символов, то элементу таблицы d_{BM}, соответствующему данному символу, присваивается значение, равное удаленности от конца образа самого правого из одинаковых символов [1]. Так, например, образ "*barbarian*" содержит три символа "*a*", удаленность от конца образа самого правого из них равна единице. В связи с этим элементам таблицы d_{BM}, соответствующим остальным символам "*a*" в образе, присваиваем значение, равное единице (рис. 3). Аналогично для всех символов "*r*" значения элементов таблицы d_{BM} принимаем равным трем, а для всех символов "*b*" – пяти.

При программной реализации алгоритма поиска образа в строке Боуера-Мура, как и комбинированного алгоритма, предлагается для создания таблицы d_{BM} использовать таблицу кодов символов ASCII (табл. 1).

Таблица ASCII содержит 256 символов, поэтому таблицу d_{BM} можно объявить как одномерный целочисленный массив, состоящий из 256 элементов: *int d*[256].

Любой печатный или служебный символ имеет свой код. Например, код символа "*a*" равен 97, код символа "*r*" – 114, а код пробела – 32. Можно отметить, что коды кириллических символов лежат в диапазоне от 192 до 255.

Рассмотрим особенности программной реализации таблицы d_{BM} на примере образа "*barbarian*". Поскольку данный образ содержит девять символов, то его длина равна девяти. Соответственно, на этапе инициализации всем элементам таблицы d_{BM} присваивается значение девять:

$$d[0] = d[1] = \ldots = d[254] = d[255] = 9.$$

Далее происходит присваивание элементам таблицы d_{BM} соответствующих значений, равных расстоянию от рассматриваемого символа до конца образа. При этом индекс элемента таблицы d_{BM}, который получает новое значение, определяется кодом ASCII рассматриваемого символа.

Так, код ASCII символа "*a*" образа равен 97. Соответствующее ему значение элементов таблицы d_{BM}, как это уже было показано на рис. 3, равно единице. Таким образом,

$$d[97] = 1.$$

Так же на языке программирования Си можно записать

$$d['a'] = 1.$$

Можно отметить, что при программной реализации элемент массива *d* с индексом девяносто семь соответствует всем символам "*a*" в образе "*barbarian*". И, таким образом, для всех

b	**a**	r	b	**a**	r	i	**a**	n
5	**1**	3	5	**1**	3	2	**1**	9

Рис. 3. Образ и таблица d_{BM}

Таблица 1. Таблица ASCII. Печатные символы

Код	Символ	Код	Символ	Код	Символ	Код	Символ
32	**пробел**	56	**8**	80	**P**	104	**h**
33	**!**	57	**9**	81	**Q**	105	**i**
34	**“**	58	**:**	82	**R**	106	**j**
35	**#**	59	**;**	83	**S**	107	**k**
36	**$**	60	**<**	84	**T**	108	**l**
37	**%**	61	**=**	85	**U**	109	**m**
38	**&**	62	**>**	86	**V**	110	**n**
39	**‘**	63	**?**	87	**W**	111	**o**
40	**(**	64	**@**	88	**X**	112	**p**
41	**)**	65	**A**	89	**Y**	113	**q**
42	*	66	**B**	90	**Z**	114	**r**
43	+	67	**C**	91	**[**	115	**s**
44	**,**	68	**D**	92	\	116	**t**
45	**-**	69	**E**	93	**]**	117	**u**
46	**.**	70	**F**	94	^	118	**v**
47	/	71	**G**	95	_	119	**w**
48	**0**	72	**H**	96	`	120	**x**
49	**1**	73	**I**	97	**a**	121	**y**
50	**2**	74	**J**	98	**b**	122	**z**
51	**3**	75	**K**	99	**c**	123	{
52	**4**	76	**L**	100	**d**	124	|
53	**5**	77	**M**	101	**e**	125	}
54	**6**	78	**N**	102	**f**	126	~
55	7	79	**O**	103	**g**		

символов “*a*” образа всем соответствующим элементам таблицы d_{BM} при такой записи присваивается значение, равное единице (рис. 3).

Аналогичным образом изменяются значения элементов таблицы d_{BM}, соответствующие символам образа “*i*”, “*r*”, “*b*”.

Вторым этапом работы БМ-алгоритма является собственно сам поиск образа в строке. При сравнении образа со строкой образ продвигается по строке слева направо. Однако сравнения символов образа и строки выполняются, двигаясь по образу справа налево.

Сравнение образа со строкой происходит до тех пор, пока: 1) не будет рассмотрен весь образ, что говорит о том, что соответствие между образом и некоторой частью строки найдено; 2) не закончится строка, что значит, что вхождений, соответствующих образу, в строке нет; 3) не произойдет несовпадения символов образа и строки, что вызывает сдвиг образа на несколько символов вправо и продолжение процесса поиска.

В том случае, если произошло несовпадение символов, смещение образа по строке определяется значением элемента таблицы d_{BM}. Однако индексом данного элемента является код ASCII символа *строки*. Подчеркнем, что, несмотря на то что массив d_{BM} формируется на основе

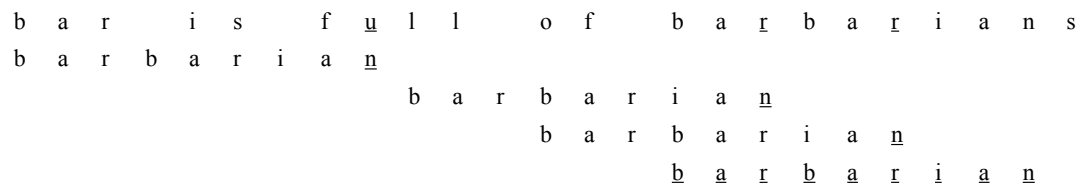

Рис. 4. Процесс работы алгоритма поиска образа в строке Боуера-Мура

образа, смещение определяется несовпавшим символом из строки. На рис. 4 показан пример работы БМ-алгоритма, сравниваемые символы подчеркнуты.

На первой итерации произошло несовпадение символа образа “*n*” и символа строки “*u*”. Напомним, что значения всех элементов массива d_{BM} для всех возможных символов на этапе инициализации приравнивается длине образа, т.е. девяти. Поскольку символ “*u*” в образе не встречается, то значение элемента *d*[‘*u*’] при дальнейшем формировании таблицы не менялось и осталось равным девяти. Поэтому образ смещается вправо на девять символов. Если бы произошло совпадение этих двух символов, то далее рассматривался бы предпоследний символ образа и соответствующий ему символ строки и т. д. При сравнивании образа и строки происходит несовпадение символов “*n*” и “*r*”. Опять же, определяя смещение по символу строки (*d*[‘*r*’]), получаем значение, равное трем. Образ сдвигается на три символа вправо. Так образ постепенно сдвигается по строке до тех пор, пока образ в строке не будет найден или не кончится строка.

Оценка эффективности БМ-алгоритма показывает, что практически всегда алгоритм требует значительно меньше *n* сравнений. В самых же благоприятных обстоятельствах, когда последний символ образа всегда попадает на несовпадающий символ строки, число сравнений равно *n* / *m* [1, 10].

1.3. Комбинация алгоритмов поиска образа в строке Кнута-Морриса-Пратта и Боуера-Мура

На основе изложенных алгоритмов поиска образа в строке Кнута-Морриса-Пратта и Боуера-Мура по созданию таблиц для определения сдвига и непосредственно поиску образа в строке был предложен следующий комбинированный алгоритм.

Шаг 1. Создание таблицы d_{KMP} согласно алгоритму поиска образа в строке Кнута-Морриса-Пратта.

Шаг 2. Создание таблицы d_{BM} согласно алгоритму поиска образа в строке Боуера-Мура.

Шаг 3. Полагаем начальное значение индекса *i*, соответствующее положению образа относительно строки.

Шаг 4. Полагаем начальные значения индексов j_{KMP} и j_{BM}, указывающих на начало и конец образа соответственно.

Шаг 5. Сравниваем символ образа с индексом j_{KMP} и соответствующий символ строки, сравниваем символ образа j_{BM} и соответствующий символ строки. Если хотя бы при одном сравнении символы не совпали, то переход на шаг 11.

Шаг 6. Если j_{KMP} меньше j_{BM}, то переход на шаг 9.

Шаг 7. Вывод сообщения о том, что образ совпал с частью строки.

Шаг 8. Переход на шаг 15.

Шаг 9. Увеличиваем индекс j_{KMP} на единицу (т.е. переходим к находящемуся правее символу), уменьшаем индекс j_{BM} на единицу (т.е. переходим к находящемуся левее символу).

Шаг 10. Переход на шаг 5.

Шаг 11. Выбираем большее смещение из двух: $j_{KMP} - d_{KMP}[j_{KMP}]$ и $d_{BM}[i$ + длина образа – $j_{BM}]$.

Шаг 12. Смещаем образ вправо относительно строки, увеличивая значение индекса i на смещение, определенное на шаге 11.

Шаг 13. Если сумма i и длины образа меньше длины строки, то переход на шаг 4.

Шаг 14. Вывод сообщения о том, что искомый образ не найден.

Шаг 15. Остановка.

Комбинированный алгоритм позволяет найти совпадение образа с частью строки при меньшем количестве сдвигов благодаря тому, что на шаге 11 выбирается больший сдвиг из двух, которые могут быть получены от алгоритмов поиска образа в строке Кнута-Морриса-Пратта и Боуера-Мура. При этом сами эти алгоритмы гарантируют, что, насколько бы большой ни был сдвиг, никакое совпадение образа с частью строки пропущено не будет.

2. Анализ результатов работы комбинированного алгоритма

Комбинированный алгоритм поиска образа в строке, основанный на алгоритмах Кнута-Морриса-Пратта и Боуера-Мура, был реализован в рамках компьютерной программы InfoSearch. Данная программа позволяет провести анализ работы алгоритмов Боуера-Мура, Кнута-Морриса-Пратта и комбинированного алгоритма. На рис. 5 представлен интерфейс программы.

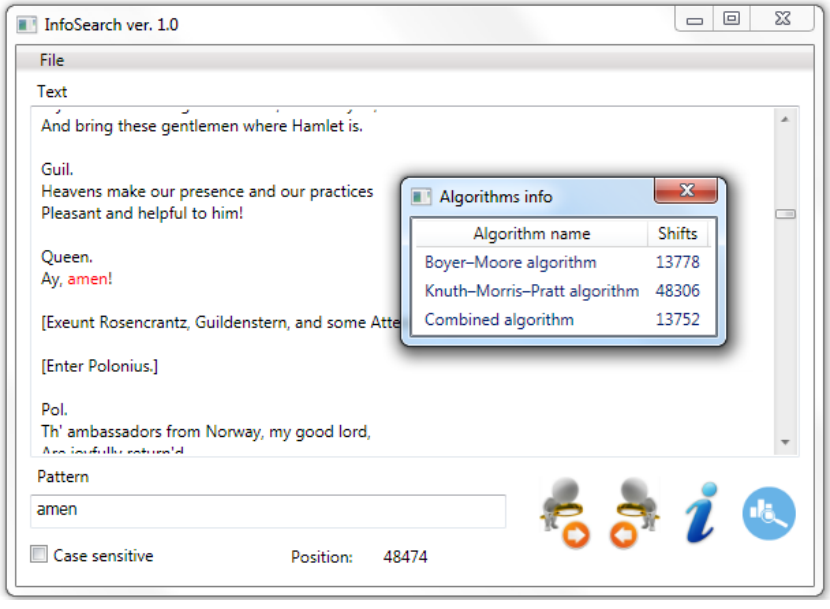

Рис. 5. Интерфейс программы, реализующей комбинированный алгоритм поиска образа в строке

Таблица 2. Результаты работы алгоритмов поиска образа в строке

Образ	Позиция	Количество смещений		
		КМП алгоритм	БМ алгоритм	Комбинированный алгоритм
amen	48474	48306	13778	13752
antique (1-е вхождение)	68539	67756	11676	11667
antique (2-е вхождение)	179186	177135	30677	30646
cozenage	166058	165546	24932	24921
habit (1-е вхождение)	24743	24519	5738	5729
habit (2-е вхождение)	29215	28949	6762	6751
habit (3-е вхождение)	113830	112669	26005	25973
habit (4-е вхождение)	171586	169831	39245	39204
herb	136155	133534	37481	37423
marble	30208	30025	5985	5948
marvel	18318	18193	3624	3592
matron	111378	110925	22596	22588
theme (1-е вхождение)	14063	13587	3171	3161
theme (2-е вхождение)	161372	156067	36456	36390
thieves	138812	135466	23527	23480
sea-fight	165436	164562	23227	23177
stone (1-е вхождение)	129310	128472	31382	31341
stone (2-е вхождение)	140499	139599	34100	34057

Анализ результатов работы рассмотренных алгоритмов поиска образа в строке может быть проиллюстрирован на примере трагедии Уильяма Шекспира «Гамлет». В табл. 2 представлены результаты работы алгоритмов при поиске указанных в первой колонке слов. Данные слова были выбраны в качестве искомых образов случайным образом.

Выполненный анализ показывает, что комбинированному алгоритму, для того чтобы найти вхождение образа в строку, требуется в несколько раз меньше сдвигов, чем алгоритму Кнута-Морриса-Пратта.

Комбинированный алгоритм также показывает лучшие результаты по количеству требуемых сдвигов по сравнению с алгоритмом Боуера-Мура, хотя здесь превосходство выражено не столь сильно, как превосходство комбинированного алгоритма над алгоритмом Кнута-Морриса-Пратта.

Заключение

В статье представлено решение чрезвычайно важной для компьютерного анализа текста задачи – поиск образа в строке. Предложенное решение данной задачи развивает теоретические основы компьютерных методов анализа языковых данных и служит решению практических задач компьютерной лингвистики.

Разработанный комбинированный алгоритм поиска образа в строке объединил в себе достоинства двух известных алгоритмов – алгоритмов поиска образа в строке Кнута-Морриса-Пратта и Боуера-Мура. Комбинированный алгоритм отличается более высокой эффективностью по сравнению с исходными алгоритмами за счет больших сдвигов при несовпадении символов строки и образа, что увеличивает скорость поиска вхождения искомого образа в строке. Комбинированный алгоритм поиска образа в строке может быть одинаково успешно применен для поиска в текстах как на английском, так и русском языках.

Благодарность

Авторы выражают благодарность доценту Ahmad Hassanat, Mu'tah University, Jordan за консультации при изучении алгоритмов поиска образа в строке.

Список литературы

[1] Wirth N. *Algorithms and Data Structures*. Prentice Hall, *NJ*, 1985.

[2] Faro S., Lecroq T. The exact online string matching problem: A review of the most recent results, *ACM Computing Surveys*, 2013, 45(2).

[3] Ahmed M., Kaykobad M., Chowdhury R.A. A new string matching algorithm, *International Journal of Computer Mathematics*, 2003, 80(7), 825–834.

[4] Lecroq T. Fast exact string matching algorithms, *Information Processing Letters*, 2007, 102(6), 229-235.

[5] Baeza-Yates R.A., Gonnet G.H. A new approach to text searching, *Communications of the ACM,* 1992, 35(10), 74–82.

[6] Fredriksson K., Grabowski S. Practical and optimal string matching, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2005, 3772 LNCS, 376-387.

[7] He L., Fang B., Sui J. The wide window string matching algorithm, *Theoretical Computer Science*, 2005, 332(1-3), 391-404.

[8] Hudaib A., Al-Khalid R., Suleiman D., Itriq M., Al-Anani A. A fast pattern matching algorithm with Two Sliding Windows (TSW), *Journal of Computer Science*, 2008, 4(5), 393-401.

[9] Knuth D., Morris J.H., Pratt V. Fast pattern matching in strings, *SIAM Journal on Computing*, 1977, 6(2), 323–350.

[10] Boyer R.S., Moore J.S. A fast string searching algorithm, *Communications of the ACM,* 1977, 20(10), 762–772.