Анализ алгоритмов поиска подстроки в тексте с учетом их практического применения на основе экспериментов

В.А. Михайлов
Казанский (Приволжский) федеральный университет
Казань, Россия
v.mikhailov23@gmail.com

Т.А. Михайлова
Башкирский государственный университет
Уфа, Россия
t.a.mihailova@yandex.ru

Э.Н. Мифтахов
Башкирский государственный университет
Уфа, Россия
promif@mail.ru

С.А. Мустафина
Башкирский государственный университет
Уфа, Россия
mustafina_sa@mail.ru

***Аннотация*—В статье приведены результаты исследования детерминированных алгоритмов решения задачи поиска подстроки в строке – простейший последовательный алгоритм поиска, алгоритм Рабина-Карпа, алгоритм Кнута-Морриса-Пратта и алгоритм Бойера-Мура. Посредством проведения анализа асимптотических оценок сложности алгоритмов построен вероятностный алгоритм поиска и осуществлена его программная реализация. Для моделирования практического применения рассматриваемых алгоритмов проведены вычислительные эксперименты, результаты которых отражают способы применения каждого алгоритма.**

Ключевые слова— поиск подстроки в тексте, детерминированный алгоритм, вероятностный алгоритм поиска, оценка сложности, простейший последовательный алгоритм поиска, алгоритм Рабина-Карпа, алгоритм Кнута-Морриса-Пратта, алгоритм Бойера-Мура.

1. Введение

Одной из популярных задач поиска информации является поиск подстроки в тексте. Данный класс задач аналогичен поиску «иголки в стоге сена» и описывается следующим образом. Пусть задан текст t и шаблон (строка) p. Необходимо определить, входит ли некоторый заданный шаблон символов в исходный текст. В случае вхождения – найти массив номеров символов, начиная с которого p содержится в t, в противном случае – вывести пустой массив.

Описанная задача встречается на практике очень часто как функция во множестве часто используемых приложений, таких как текстовые редакторы, браузеры, поисковые системы, СУБД, системы определения плагиата, системы анализа хакерских атак и т.п.

Для решения данной постановки существует множество алгоритмов. Следовательно, актуальной является задача выбора алгоритма поиска подстроки в строке, для решения которой целесообразно провести их экспериментально-сравнительный анализ посредством программной реализации детерминированных и вероятностного алгоритмов поиска [1].

2. Оценка сложности

В рамках исследования были изучены детерминированные алгоритмы поиска подстроки:

- простейший последовательный алгоритм поиска;
- алгоритм Рабина-Карпа [2, 3];
- алгоритм Кнута-Морриса-Пратта [2, 3];
- алгоритм Бойера-Мура [4].

Первоначально в результате, приведенном в таблице 1, оценки сложности на модели однопроцессорной машины с произвольным доступом к памяти (как наиболее близкой к работе современных процессоров) был выбран и построен алгоритм на основе выполнения алгоритма Рабина-Карпа. Так как другие алгоритмы имели в качестве недостатка проблему плохой модифицируемости получения универсальной для разных шаблонов префикс-функции и таблицы хороших суффиксов, так как они полностью зависят от содержания текста шаблона.

Таблица I. Оценки временной сложности детерминированных алгоритмов поиска подстроки

Algorithm	**Временная сложность**		
	предварительных вычислений	***фазы сравнения***	***сравнения в худшем случае***
Простейший алгоритм поиска	0	O(m(n-m+1))	$O(n^2)$
Алгоритм Кнута-Морриса-Пратта	O(m)	O(n)	O(n+m)
Алгоритм Бойера-Мура	O(m + |∑|)	O(n)	O(n+m+|∑|)
Алгоритм Рабина-Карпа	O(m)	O(n)	O(m(n-m+1))

Также стоит отметить, что, хотя и алгоритму Рабина-Карпа свойственно возникновение коллизии хешей разных строк (что может иногда приводить к неверному результату), он решает проблему неверного ответа путем посимвольной перепроверки шаблона и рассматриваемого фрагмента текста, чьи хеши совпадают [4, 5]. Однако временная сложность в худшем случае может стать не лучше средней сложности простейшего алгоритма последовательного поиска. Поэтому в качестве основной идеи построения вероятностного алгоритма замена посимвольной проверки на повторный вызов алгоритма, используя другую хеш-функцию. Для упрощения был использован кольцевой полиномиальный

хеш, где предварительно случайным образом выбиралось полиномиальное основание. Это позволяет уменьшить временную сложность алгоритма в худшем случае до O(n+m).

3. Реализация алгоритмов и проведение эксперимента

В процессе исследования алгоритмы были программно реализованы на языке программирования Java и протестированы на ЭВМ со следующей конфигурацией:

- процессор Intel Core i5-9300h, 4x2.4 GHz;
- оперативная память DDR4 16GB 2666 MHz (двухканальный режим памяти).

При этом эксперименты были проведены повторно несколько раз, чтобы учесть зависимость результатов от характеристик и загрузки процессора фоновыми приложениями, чтения-записи информации в ОЗУ. В качестве конечного результата использовалось среднее полученных результатов выполнения соответствующего алгоритма.

Следует также отметить, что во всех экспериментах выполнялся поиск по всем позициям строк, поэтому эксперименты близки к наихудшим случаям, поскольку все алгоритмы должны проверять все возможные позиции подстрок на соответствие шаблонам.

Вычислительные эксперименты были проведены в форме поиска одного и нескольких шаблонов в неупорядоченной базе данных, содержащей более миллиарда строк, фрагмент которого приведён в таблице 2, и поиска одного и нескольких шаблонов в тексте книги – первый том произведения Толстого Л.Н. «Война и мир», содержащий 1274973 символа.

Таблица II. Фрагмент неупорядоченной базы данных

8536	22802	3212	55444	2016-12-02	17:52:32+03
672360	64356	139276	7254	2012-04-25	22:37:38+06
96995	32916	234170	1203	2004-12-31	05:54:51+04
658474	19236	22357	3379	2012-02-11	14:15:29+02

Результаты экспериментально-сравнительного анализов приведены в таблице 3 и 4. Они показали, что алгоритмы Бойера-Мура и Кнута-Морриса-Пратта имеют значительные преимущества в одиночном поиске и часто используются в повседневных приложениях (текстовые редакторы, поиск в странице браузера и т.д.), однако плохо модифицируются для параллельных и вероятностных вычислений, что даёт преимущество алгоритму Рабина-Карпа там, где необходима частая и параллельная обработка фрагментов в часто использующихся данных (системы антиплагиата, системы учёта отзывов, анализ хакерских атак). Простейший алгоритм последовательного поиска полезен для ознакомления и практически возможен в использовании поиска в документе небольшого объёма.

Таблица III. Среднее время выполнения экспериментов поиска в неупорядоченной базе данных

Алгоритм	**Среднее время выполнения (мс)**			
	Без учета предварительных вычислений		***Без учета предварительных вычислений***	
	1 шаблон	**1 шаблон**	**1 шаблон**	**1 шаблон**
Простейший алгоритм поиска	196	654	196	654
Алгоритм Кнута-Морриса-Пратта	132	350	154	467
Алгоритм Бойера-Мура	89	359	113	484
Алгоритм Рабина-Карпа	57	240	757	944
Вероятностный алгоритм	53	179	3653	3838

Таблица IV. Среднее время выполнения экспериментов поиска в книге

Алгоритм	**Среднее время выполнения (мс)**			
	Без учета предварительных вычислений		***С учетом предварительных вычислений***	
	1 шаблон	**5 шаблонов**	**1 шаблон**	**5 шаблонов**
Простейший алгоритм поиска	14	325	14	325
Алгоритм Кнута-Морриса-Пратта	~2.52	11	4	18
Алгоритм Бойера-Мура	7	25	9	33
Алгоритм Рабина-Карпа	~1.67	8	16	22
Вероятностный алгоритм	~1.62	7	67	72

4. Заключение

В результате можно сделать вывод, что каждый существующий алгоритм поиска подстроки имеет свои достоинства и недостатки, по данной причине и имеет свою наиболее подходящую прикладную область применения по сравнению с другими аналогами. Разные результаты алгоритмов в той или иной ситуации при близких асимптотических оценках временной сложности возникают вследствие опускания констант, которые могут оказать влияние на практическое время выполнения

Литература

[1] Гасфилд, Д. Строки, деревья и последовательности в алгоритмах: информатика и вычислительная биология / Д. Гасфилд. – СПб.: Невский Диалект, 2003. – 654 с.

[2] Кормен, Т.Х. Алгоритмы: построение и анализ / Т.Х. Кормен, Ч.Е. Лейзерсон, Р.Л. Ривест, К. Штайн. – М.: Вильямс, 2013. – 1328 с.

[3] Смит, Б. Методы и алгоритмы вычислений на строках / Б. Смит. – М.: ООО «И.Д. Вильямс», 2006. – 496 с.

[4] Rabin, M.O. Efficient randomized pattern-matching algorithms / M.O. Rabin, R.M. Karp // IBM Journal of Research and Development. – 1987. – Vol. 31(2). – P. 249-260.

[5] Rabin, M.O. Fingerprinting by random polynomials / M.O. Rabin // Tech Report TR-CSE-03-01. Center for Research in Computing Technology, Har-vard University. – 1981. – P. 1-14.

[6] Pachocki, J. Where to use and how not to use polynomial string hashing / J. Pachocki, J. Radoszewski // Olympiads in Informatics. – 2013. – Vol. 7. – P. 90-100.