Опубликовано в материалах 2-й межвузовской научной конференции по проблемам информатики СПИСОК-2011, с. 363-365.

Д. А. Паращенко, А. С. Станкевич
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Обработка строк на основе суффиксных автоматов

В настоящее время для решения большого числа строковых задач применяются суффиксные деревья [1]. При этом все известные алгоритмы построения суффиксного дерева за линейное время [1] достаточно сложны для понимания и реализации.

В настоящей работе разработан достаточно простой алгоритм построения суффиксного дерева за линейное время, содержащий в качестве одного из своих этапов построение суффиксного автомата. Таким образом, помимо алгоритмов Вайнера [2], Мак-Крейта [3] и Укконена [4] предложен еще один алгоритм построения суффиксного дерева за линейное время. Кроме этого в работе проведено сравнение времени их работы, а также сложности их реализации.

Поясним, как появилась идея использовать суффиксный автомат для решения рассматриваемой задачи. Ввиду того, что суффиксные деревья и суффиксные автоматы являются родственными структурами данных (суффиксный автомат является минимизированным суффиксным бором [5], а суффиксное дерево – сжатым суффиксным бором [5]), то автор посчитал целесообразным для решения строковых задач вместо суффиксных деревьев использовать суффиксные

автоматы. Одним из преимуществ этого подхода является тот факт, что суффиксный автомат является более простой структурой данных [6], а алгоритм его построения значительно проще в реализации, чем алгоритм построения суффиксного дерева.

Разработанный автором алгоритм построения суффиксного дерева за линейное время немного уступает другим алгоритмам построения суффиксного дерева (Укконена и Мак-Крейта) как по количеству используемой дополнительной памяти, так и по времени работы. Однако он реализуется намного проще и быстрее упомянутых выше алгоритмов. Это позволяет применять его в случаях, когда требуется быстро написать правильно работающий код, затратив на это минимум усилий, например, на соревнованиях ACM ICPC.

В работе также предложен метод, позволяющий избавиться от использования в алгоритмах суффиксных деревьев. Суть этого метода состоит в изменении алгоритма таким образом, чтобы вместо операций над суффиксным деревом в нем использовались операции над суффиксным автоматом. В связи с тем, что суффиксный автомат является более простой в использовании структурой данных по сравнению с суффиксным деревом, применение данного метода позволяет в большинстве случаях ускорить работу алгоритма.

Опишем суть упомянутого метода. Для этого введем некоторые определения и приведем ряд их свойств.

Определение. Правым контекстом [5] строки y в строке w называется множество всех таких строк x, что строка xy является суффиксом строки w.

Определение. *Множеством представителей* правого контекста C в строке w будем называть множество строк, правый контекст которых в строке w совпадает с C.

Лемма 1. Множество представителей правого контекста состоит из суффиксов наибольшего представителя, длина которых не меньше длины наименьшего представителя.

Лемма 2. Для любого правого контекста *C* и числа *len* существует не более одного слова *y* длины *len* с правым контекстом *C*.

Заметим, что можно говорить о представителях состояния суффиксного автомата, подразумевая под этим представителей соответствующего этому состоянию правого контекста.

Лемма 3. Множество всех строк, приводящих автомат из начального состояния в некоторое состояние *s*, совпадает с множеством представителей правого контекста этого состояния.

Состояние суффиксного автомата, в которое ведет суффиксная ссылка из состояния *s*, обозначим за *suffix(s)*, а длину наибольшего представителя состояния *s* – за $repr_{max}(s)$.

Теорема 1. Пусть непустое слово *x* является некоторым представителем состояния s_1 суффиксного автомата. Пусть s_2 – состояние, в которое придет суффиксный автомат, приняв на вход слово $x[2..|x|]$. Тогда:

- если $|x| = repr_{max}(suffix(s_1)) + 1$, то s_2 совпадает с $suffix(s_1)$;
- если $|x| > repr_{max}(suffix(s_1)) + 1$, то s_2 совпадает с s_1.

Для эмуляции суффиксного дерева необходимо установить связь между его состояниями и состояниями суффиксного автомата.

Теорема 2. Каждой вершине суффиксного дерева можно поставить в соответствие пару <*состояние суффиксного автомата, длина представителя этого состояния*>.

Следует понимать, что некоторым состояниям суффиксного автомата не будет соответствовать ни одна вершина суффиксного дерева. Указанные состояния обладают двумя свойствами: не являются конечными и при этом имеют лишь один исходящий переход. Все остальные состояния суффиксного автомата будем называть *вилками*.

Построив структуру, позволяющую эффективно переходить по переходам суффиксного автомата до ближайшей вилки, а также вычислив длины наибольших представителей состояний можно эффективно реализовывать на автомате все операции суффиксного дерева:

- получать корневую вершину дерева;
- определять, является ли некоторая вершина суффиксного дерева конечной;
- получать суффиксную ссылку из данной вершины (см. теорему 1);
- находить исходящую из данной вершины дугу с заданным первым символом дуговой метки.

На практике, значительная часть задач требует далеко не всех операций, предоставляемых суффиксным деревом. Это позволяет при переносе алгоритма на суффиксный автомат обойтись без вычисления длин наибольших представителей состояний, а в некоторых случаях – и без вычисления вилок.

Для демонстрации эффективности предложенного метода в работе рассмотрена задача о нахождении наибольшего общего префикса двух суффиксов [7]. Основной причиной выбора этой задачи является то, что к ней можно свести большое число других задач на строках. Установлено, что применение метода позволяет значительно сократить время фазы инициализации алгоритма для решения рассмотренной задачи.

В дальнейшем планируется выяснить, что выгоднее: сразу строить сжатый суффиксный автомат, или строить, а затем сжимать суффиксный автомат. Также планируется произвести сравнение количества памяти, требуемой для хранения суффиксного дерева, и памяти, требуемой для хранения суффиксного автомата с поддержкой всех либо лишь части операций суффиксного дерева.

Источники

1. *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология. СПб.: Невский диалект; БХВ-Петербург, 2003.
2. *Weiner P.* Linear pattern matching algorithms / Proc. of the 14th IEEE Symp. on Swithing and Automata Theory. 1973, pp.1-11.
3. *McCreight E. M.* A space-economical suffix tree construction alorithm // J. ACM. 1976. Vol 23, pp. 262-272.
4. *Ukkonen E.* Online construction of suffix-trees // Algoritmica. 1995. Vol. 14, pp. 249-260.
5. *Lothaire M.* Applied Combinatorics on Words // Encyclopedia of Mathematics and its Applications, 2005. Vol. 90. Cambridge University Press, Cambridge.
6. *Хопкрофт Дж., Мортвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
7. *Bender M., Farach-Colton M.* The LCA Problem Revisited / LATIN 2000, pp. 88-94.