Нормальные алгорифмы

В. Н. Брагилевский

6 августа 2019 г.

1. Алфавит, слова, конкатенация слов, подслова и вхождения

Алфавит — это конечное множество знаков, называемых *буквами*. Например, алфавитом является множество $\Sigma = \{a, b\}$, это двухбуквенный алфавит. Конечная цепочка следующих друг за другом букв называется *словом*. Множество всех слов алфавита Σ обозначается символом Σ^*. Среди всех возможных слов выделим *пустое* слово, не содержащее никаких букв, и обозначим его символом ε ($\varepsilon \in \Sigma^*$). Слово из Σ^* называется также словом в алфавите Σ. Будем далее обозначать слова прописными латинскими буквами.

Конкатенацией слов P и Q называется слово, обозначаемое PQ и содержащее сначала цепочку букв слова P, а затем цепочку букв слова Q. Например, конкатенацией слов $abab$ и $aaabbb$ в алфавите $\Sigma = \{a, b\}$ является слово $ababaaabbb$.

Слово P называется *подсловом* слова Q, если Q является конкатенацией слов X, P и Y (то есть, если $Q = XPY$), где X и Y (называемые, соответственно, префиксом и суффиксом), возможно, пустые. Например, слово ab является подсловом слова $aabb$, причём $X = a$, $Y = b$. Другой пример: слово 111 является подсловом слова 11111, причём здесь пара X и Y может определяться различными способами. Наконец, само слово «слово» является подсловом слова «подслово», которое в свою очередь иногда называется *надсловом* слова «слово». Вообще, всякое слово является как своим подсловом, так и надсловом. Пустое слово является подсловом любого слова.

Подслово может *входить* в слово неоднократно. Например, имеется три *вхождения* слова ab в слово $ababab$. Каждое такое вхождение однозначно определяется парой, состоящей из префикса и суффикса (X, Y). Так, в приведённом примере три вхождения определяются парами: 1) $(\varepsilon, abab)$; 2) (ab, ab); 3) $(abab, \varepsilon)$. Вхождение непустого слова называется первым, если соответствующее слово не является подсловом префикса X. Например, первым вхождением слова ab в слово $ababab$ является вхождение, определяемое парой $X = \varepsilon$, $Y = abab$. Вхождения пустого слова в слово P определяются всеми возможными разбиениями слова P вида $P = XY$. Первым вхождением пустого слова считается вхождение с пустым префиксом (то есть вхождение в начало слова).

2. Марковские подстановки и их применение

Пара слов, обозначаемая $P \longrightarrow Q$, называется *марковской подстановкой*. Говорят, что подстановка *применима* к слову M, если левая часть подстановки (P) является подсловом слова M. Если подстановка $P \longrightarrow Q$ применима к слову M, то её *применением* называется *замена первого вхождения слова P в слово M на слово Q*.

В качестве примера рассмотрим марковскую подстановку в алфавите $\Sigma = \{a, b\}$:

$$ab \longrightarrow a$$

Эта подстановка применима к словам ab, $abab$, $aabb$, $aabaaabb$ и не применима к словам aa, ε, ba, $bbbbb$. В результате применения подстановки к словам из первого списка получим следующее (обратите внимание на использование двойной стрелки для обозначения применения подстановки к слову, первое вхождение подчёркнуто):

$$\begin{aligned} \underline{ab} &\Longrightarrow a \\ \underline{ab}ab &\Longrightarrow aab \\ a\underline{ab}b &\Longrightarrow aab \\ a\underline{ab}aaabb &\Longrightarrow aaaaabb \end{aligned}$$

В левой и правой частях подстановки может находиться пустое слово. В первом случае её смысл заключается во вставке правой части подстановки в начало слова, а во втором — удаление первого вхождения левой части подстановки. Следует заметить, что эти действия соответствуют нашим определениям: первое вхождение пустого слова является вхождением в начало слова, поэтому речь в этом случае идёт о вставке в начало, а замена на пустое слово является в точности удалением. Например, применение подстановки $\varepsilon \longrightarrow a$ к слову ba приводит к преобразованию $ba \Longrightarrow aba$, а применение подстановки $a \Longrightarrow \varepsilon$ к тому же слову — к преобразованию $ba \Longrightarrow b$. Подстановка вида $\varepsilon \longrightarrow P$ применима к любому слову, в том числе пустому, например, $\varepsilon \Longrightarrow P$. Здесь следует обратить внимание на различие между подстановкой (одинарная стрелка) и её действием на слово (двойная стрелка).

Далее нам потребуется особый вид подстановок, называемых *заключительными* или *финальными*. Будем обозначать их одинарной стрелкой с точкой:

$$P \longrightarrow. Q$$

Смысл этого названия станет понятен позднее.

3. Схемы нормальных алгорифмов

Схемой нормального алгорифма называется конечный упорядоченный набор марковских подстановок, некоторые из которых могут быть заключительными:

$$\begin{cases} P_1 & \longrightarrow (.)\ Q_1 \\ P_2 & \longrightarrow (.)\ Q_2 \\ & \dots \\ P_n & \longrightarrow (.)\ Q_n \end{cases}$$

Однократным применением схемы алгорифма к заданному слову называется применение первой (по порядку следования в схеме) применимой к нему марковской подстановки. Если ни одна из подстановок схемы к слову не применима, то и схема в целом является неприменимой к данному слову.

Нормальным алгорифмом называется последовательность однократных применений схемы к заданному слову, завершаемая в одном из двух случаев:

1) схема оказывается неприменимой к слову;
2) применена заключительная подстановка.

Во всех последующих примерах будем считать, что схемы определены для алфавита $\Sigma = \{a, b\}$. Рассмотрим пример применения схемы

$$\begin{cases} ab \longrightarrow \varepsilon \\ a \longrightarrow b \end{cases}$$

к слову $aababab$ (подчёркнуто первое вхождение левой части применимой подстановки):

$$a\underline{ab}abab \Longrightarrow a\underline{ab}ab \Longrightarrow a\underline{ab} \Longrightarrow \underline{a} \Longrightarrow b$$

Первые три шага нормального алгорифма состоят из применения первой подстановки, а на четвёртом шаге применяется вторая подстановка (первая уже неприменима). Четвёртый шаг оказывается последним, поскольку после него ни одна из подстановок неприменима (первый случай завершения нормального алгорифма).

Схема может состоять из единственной подстановки:

$$\begin{cases} a \longrightarrow b \end{cases}$$

Ясно, что соответствующий нормальный алгорифм заменяет все вхождения символа a на символ b, после чего завершается.

Наличие в схеме заключительной подстановки приводит к завершению нормального алгорифма сразу после её применения, например, применение схемы

$$\begin{cases} b \longrightarrow . \ a \\ a \longrightarrow b \end{cases}$$

к слову aaa приводит к следующим шагам нормального алгорифма:

$$\underline{a}aa \Longrightarrow \underline{b}aa \Longrightarrow aaa$$

На первом шаге первая подстановка неприменима (в слове отсутствует буква b), поэтому применяется вторая. В результате её действия буква b появляется, поэтому на втором шаге алгорифма применяется первая подстановка. Поскольку она является заключительной, действие алгорифма после её применения завершается.

Нормальный алгорифм может вообще не завершаться. Так, применение схемы

$$\begin{cases} a \longrightarrow b \\ b \longrightarrow a \end{cases}$$

к любому непустому слову приводит к следующему: сначала все буквы a заменяются на b, после чего начинается бесконечное «мигание» первой буквы слова, она заменяется то на a, то на b.

Слово в процессе преобразования нормальным алгорифмом может даже неограниченно увеличиваться, таково, к примеру, действие схемы

$$\begin{cases} \varepsilon \longrightarrow a \end{cases}$$

Соответствующий нормальный алгорифм будет бесконечно долго вставлять букву a в начало слова (эта ситуация будет иметь место, даже если начать с пустого слова).

4. Примеры нормальных алгорифмов

Рассмотрим простые примеры нормальных алгорифмов. Следующая схема предназначена для удаления всех символов входного слова в алфавите $\Sigma = \{a, b\}$:

$$\begin{cases} a \longrightarrow \varepsilon \\ b \longrightarrow \varepsilon \end{cases}$$

Сначала удаляются все символы a, затем все символы b. Для удаления одного любого символа следует обе подстановки сделать заключительными:

$$\begin{cases} a \longrightarrow. \varepsilon \\ b \longrightarrow. \varepsilon \end{cases}$$

Такой алгорифм оказывается «предвзятым»: если в слове есть символ a, то именно он и удаляется. Символ b удалится, только если символов a в слове нет. Удаляются первые вхождения соответствующих символов.

Рассмотрим односимвольный алфавит $\Sigma = \{|\}$, посредством которого представим натуральное число в унарной записи (пустое слово соответствует нулю). Схема нормального алгорифма, увеличивающего заданное число на единицу, выглядит следующим образом:

$$\begin{cases} \varepsilon \longrightarrow. | \end{cases}$$

Например: $|| \Longrightarrow |||$.

Вычитание единицы (считаем, что $0 - 1 = 0$):

$$\begin{cases} | \longrightarrow. \varepsilon \end{cases}$$

Более сложная задача: проверим, делится ли число, записанное в унарной системе счисления, на три. Для этого будем стирать по три цифры за один шаг алгорифма, после чего проверим, что осталось. Результатом преобразования будет символ $|$, если число делится на 3 и пустое слово в противном случае. Схема алгорифма имеет следующий вид:

$$\begin{cases} ||| \longrightarrow \varepsilon \\ || \longrightarrow. \varepsilon \\ | \longrightarrow. \varepsilon \\ \varepsilon \longrightarrow. | \end{cases}$$

Примеры: $||||||| \Longrightarrow |||| \Longrightarrow | \Longrightarrow \varepsilon$ (7 не делится на 3), $||||||||| \Longrightarrow |||||| \Longrightarrow ||| \Longrightarrow \varepsilon \Longrightarrow |$ (9 делится на 3).

5. Алгорифмы в расширениях алфавита

Нормальный алгорифм задаётся схемой, которая в свою очередь состоит из подстановок, левая и правая части которых содержат буквы некоторого алфавита. Тот же алфавит используется для записи слов, к которым нормальные алгорифмы применяются, и, соответственно, для результатов применения. Иногда символов, уже имеющихся в алфавите, недостаточно, то есть в процессе преобразования слова оказываются необходимы другие (вспомогательные) символы.

Алфавит Σ' называется *расширением* алфавита Σ, если $\Sigma' \supset \Sigma$. Говорят, что нормальный алгорифм действует *над* алфавитом Σ, если он задан в некотором расширении алфавита Σ (то есть в схеме алгорифма могут встречаться символы из расширения Σ). При этом важно, что и входное слово, и окончательный результат преобразования (то есть слово, получаемое после завершения алгорифма), содержат символы только из основного алфавита Σ, но не из его расширения. В ситуации, когда расширение алфавита не требуется, используют предлог *в*, то есть говорят «алгорифм действует в алфавите Σ».

Возможность расширить алфавит вспомогательными символами иногда позволяет существенно упростить решение задачи, а иногда делает невозможное без этого решение возможным. В качестве примера разработаем схему алгорифма, который удаляет первую букву непустого входного слова в алфавите $\Sigma = \{a, b\}$, какой бы она не была. Для этого вставим в начало слова вспомогательный символ $*$, а затем удалим его вместе со следующей за ним буквой заключительной подстановкой (таких подстановок в схеме должно быть две по числу букв в алфавите). При разработке схемы следует внимательно относиться к порядку следования подстановок, учитывая, что применяется всегда первая из применимых к слову подстановок. Используя эти соображения, получаем следующую схему:

$$\begin{cases} *a \longrightarrow . \ \varepsilon \\ *b \longrightarrow . \ \varepsilon \\ \varepsilon \longrightarrow * \end{cases}$$

Вспомогательный символ фиксирует во входном слове начальную позицию. Без его использования отличить первую букву от всех последующих невозможно. Заметим, что на первом шаге нормального алгорифма всегда будет применяться последняя подстановка схемы, а на втором — первая или вторая (заключительная):

$$abab \Longrightarrow *abab \Longrightarrow bab$$
$$baba \Longrightarrow *baba \Longrightarrow aba$$

Если применить эту схему к пустому слову, то мы получим зацикливание (первые две подстановки останутся после первого шага неприменимыми, а третья не предусматривает остановки):

$$\varepsilon \Longrightarrow * \Longrightarrow ** \Longrightarrow *** \Longrightarrow \ldots$$

Устраним зацикливание, добавив подстановку $* \longrightarrow . \ \varepsilon$, удаляющую одинокую $*$:

$$\begin{cases} *a \longrightarrow . \ \varepsilon \\ *b \longrightarrow . \ \varepsilon \\ * \longrightarrow . \ \varepsilon \\ \varepsilon \longrightarrow * \end{cases}$$

Теперь поведение на пустом слове следующее: $\varepsilon \Longrightarrow * \Longrightarrow \varepsilon$. По-прежнему применение заключительных подстановок становится возможным только после применения последней подстановки, вставляющей вспомогательный символ $*$ в начало слова.

Полезно рассмотреть ещё один пример нормального алгорифма, но уже над алфавитом $\Sigma = \{0, 1\}$. Предположим, что входное слово задаёт натуральное число в двоичной форме, и требуется увеличить его на единицу. Действие алгорифма будет состоять из двух этапов: сначала найдём конец слова (для этого вставленный в начало слова вспомогательный символ будет последовательно перемещаться в его конец), а затем выполним прибавление единицы с переносом единичного разряда (перенос будет реализован движением в обратном направлении), если это окажется необходимым. Схема этого алгорифма имеет вид (подстановки здесь помечены):

$$\begin{cases} 0b \longrightarrow .\ 1 & \text{(а)} \\ 1b \longrightarrow b0 & \text{(б)} \\ b \longrightarrow .\ 1 & \text{(в)} \\ a0 \longrightarrow 0a & \text{(г)} \\ a1 \longrightarrow 1a & \text{(д)} \\ 0a \longrightarrow 0b & \text{(е)} \\ 1a \longrightarrow 1b & \text{(ё)} \\ \varepsilon \longrightarrow a & \text{(ж)} \end{cases}$$

Первый этап реализуется применением подстановки (ж) и следующей за ней последовательностью применений подстановок (г) и (д). В результате первого этапа вспомогательный символ a оказывается в конце слова. Затем начинается второй этап: подстановки (е) или (ё) заменяют a на b и либо срабатывают заключительные подстановки (а) или (в), либо запускается перенос единичного разряда подстановкой (б). Рассмотрим действие алгорифма на примерах:

$$\varepsilon \overset{\text{(ж)}}{\Longrightarrow} a \overset{\text{(ж)}}{\Longrightarrow} aa \overset{\text{(ж)}}{\Longrightarrow} aaa \overset{\text{(ж)}}{\Longrightarrow} \ldots$$

$$0 \overset{\text{(ж)}}{\Longrightarrow} a0 \overset{\text{(г)}}{\Longrightarrow} 0a \overset{\text{(е)}}{\Longrightarrow} 0b \overset{\text{(а)}}{\Longrightarrow} 1$$

$$1 \overset{\text{(ж)}}{\Longrightarrow} a1 \overset{\text{(д)}}{\Longrightarrow} 1a \overset{\text{(ё)}}{\Longrightarrow} 1b \overset{\text{(б)}}{\Longrightarrow} b0 \overset{\text{(в)}}{\Longrightarrow} 10$$

$$11 \overset{\text{(ж)}}{\Longrightarrow} a11 \overset{\text{(д)}}{\Longrightarrow} 1a1 \overset{\text{(д)}}{\Longrightarrow} 11a \overset{\text{(ё)}}{\Longrightarrow} 11b \overset{\text{(б)}}{\Longrightarrow} 1b0 \overset{\text{(б)}}{\Longrightarrow} b00 \overset{\text{(в)}}{\Longrightarrow} 100$$

После достижения первым вспомогательным символом конца слова действуют следующие соображения. Если число заканчивается на цифру 0, то она заменяется на 1 и вычисление завершается. Если число заканчивается на цифру 1, то она заменяется на 0, а затем запускается перенос разряда. Разработанный алгорифм к пустому слову неприменим. Эту схему нетрудно обобщить на десятичную систему счисления: для этого достаточно добавить подстановки, дублирующие подстановки (а), (г)-(ё) для остальных цифр 2–9 и учесть, что роль 1, как разряда, для которого возникает перенос, теперь будет играть цифра 9.