УДК 510.5

А.В. Пруцков

РЕШЕНИЕ ЗАДАЧ ОБРАЩЕНИЯ И УДВОЕНИЯ С ПОМОЩЬЮ ЛИНЕЙНЫХ НОРМАЛЬНЫХ АЛГОРИТМОВ

Линейные нормальные алгоритмы являются модификацией нормальных алгоритмов Маркова и отличаются от них возможностью реализации линейного вычислительного процесса. Цель данной статьи состоит в решении классических задач теории нормальных алгоритмов Маркова – задач обращения и удвоения с линейной трудоемкостью. Для достижения данной цели необходимо разработать линейные нормальные алгоритмы решения этих задач. В статье показано, что даже при наложении ограничений предложенные алгоритмы остаются линейными по трудоемкости. Также разработаны линейные нормальные алгоритмы сравнения слов и сведения нормальных алгоритмов Маркова к линейным нормальным алгоритмам на основе алгоритмов обращения и удвоения.

Нормальные алгоритмы Маркова; линейные нормальные алгоритмы; обращающий и удваивающий алгоритмы.

A.V. Prutzkow

THE SOLUTION OF TASKS OF INVERSION AND DUPLICATION BY LINEAR NORMAL ALGORYTHMES

Linear normal algorythmes is a modification of Markov's normal algorythmes. and differ from it by possibility of linear computing process realization. The aim of this article is solving of classic tasks of the theory of Markov's normal algorythmes for linear time. It's necessary to develop linear normal algorythmes of solution of classic tasks of Markov's normal algorythmes for achieving this aim. In this article it demonstrates algorythmes are still linear even when limitation imposition. It offers linear normal algorythmes of words comparing and Markov's normal algorythmes to linear normal algorythmes adduction.

Markov's normal algorythmes; linear normal algorythmes; inversion and duplication algorythmes.

Введение. Нормальные алгоритмы Маркова являются одной из алгоритмических моделей. В теории нормальных алгоритмов Маркова существуют две классические задачи – задачи обращения и удвоения, которые являются линейными. Однако предложенные А.А. Марковым и Н.М. Нагорным нормальные алгоритмы решения этих задач имеют квадратичную трудоемкость [1]. Работы других авторов [2–3] также не предлагают линейного решения. Автором статьи была предложена модификация нормальных алгоритмов Маркова, названная линейными нормальными алгоритмами [4].

Цель данной статьи заключается в том, чтобы показать, что даже при введении ограничений возможна разработка линейных нормальных алгоритмов решения задач обращения и удвоения с линейной трудоемкостью, а также разработка алгоритмов решения более сложных задач на основе предложенных алгоритмов.

Схема линейного нормального алгоритма состоит из строк, каждая из которых снабжена меткой (левая часть строки до двоеточия), после которой следуют подстановки. Процесс применения линейного нормального алгоритма к строке S заключается в следующем. Схема просматривается сверху вниз, начиная с первой строки. Если метка является подстрокой строки S, то происходит применение подстановок, соответствующих данной метке. Каждая подстановка применяется один раз. После этого схема просматривается вновь, начиная с первой строки. Как и в нормальных алгоритмах Маркова, подстановка заключается в замене в строке S первого вхождения слева подстроки из левой части подстановки на подстроку из правой части подстановки.

Подстановки строки, помеченной пустым символом ∅, выполняются для любой строки S.

Алгоритм заканчивает выполнение в трех случаях: 1) ни одна метка P не встречается в строке S; 2) левая часть подстановки, которая соответствует метке, не встречается в строке S; 3) выполнена заключительная подстановка, т.е. подстановка, заканчивающаяся символом «•».

Введем обозначения:

∅ – пустой символ;

$Z_1, Z_2, \ldots, Z_n$ – буквы алфавита Z;

m – длина обрабатываемой строки;

→ λ+ – добавление символа λ к началу строки;

→ +λ – добавление символа λ к концу строки;

□ – конец примера.

Задача обращения. Задача обращения строки заключается в записи символов строки в обратном порядке. Например, «*abc*» – «*cba*».

Запишем схему линейного нормального алгоритма обращения А с использованием одного вспомогательного символа. Схема состоит из следующих подстановок:

1) $Z_i\lambda$: $Z_i\lambda \to \lambda; \to +Z_i$; i = 1, 2, …, n;

2) λ: $\lambda \to \varnothing$ •;

3) ∅: → +λ.

Указатель λ устанавливается в конец строки (подстановка 3). Каждый символ строки, который находится слева от указателя λ, удаляется, а затем добавляется в конец строки (подстановка 1). Таким образом, символы строки удаляются перед указателем λ и появляются справа от него в обратном порядке. Условием окончания работы алгоритма является отсутствие символов слева от указателя λ (подстановка 2). При выполнении данного условия указатель λ удаляется из строки.

Пример 1. Рассмотрим работу линейного нормального алгоритма обращения А на примере строки 123 по шагам.

0.		123
1.	3) ∅: → +λ	123λ
2–3.	1) 3λ: 3λ → λ; → +3	12λ3
4–5.	1) 2λ: 2λ → λ; → +2	1λ32
6–7.	1) 1λ: 1λ → λ; → +1	λ321
8.	2) λ: λ → ∅ •	321

На последнем шаге алгоритма получена строка 321 – обращение строки 123. □

Операция установки символа λ в конец строки

→ +λ

отсутствует в теории нормальных алгоритмов Маркова и является частью предлагаемой модификации. Составим линейный нормальный алгоритм обращения, который не использует операцию установки символа в конец строки.

Линейный нормальный алгоритм обращения Б, удовлетворяющий данному условию, включает следующие подстановки в своей схеме:

1) $Z_i\lambda$: $Z_i\lambda \to \lambda; \gamma \to Z_i\gamma;$ $i = 1, 2, \ldots, n$

2) λ: $\lambda \to \varnothing; \gamma \to \varnothing$ •

3) γZ_i: $\gamma Z_i \to Z_i\gamma;$ $i = 1, 2, \ldots, n$

4) γ: $\gamma \to \lambda\gamma$

5) $\varnothing$: $\to \gamma+$

Алгоритм обращения Б устанавливает указатель γ в начало строки (подстановка 5). Указатель γ двигается вправо по строке (подстановка 4). Как только указатель γ достигает конца строки, к нему добавляется указатель λ (подстановка 4). Далее алгоритм перемещает символы перед указателем λ в конец строки, обозначенный указателем γ (подстановка 1). После того как все символы обращены, алгоритм завершает работу, удаляя указатели λ и γ (подстановка 2).

Пример 2. Рассмотрим работу линейного нормального алгоритма обращения Б на примере строки 123 по шагам.

0.		123
1.	5) $\varnothing: \to \gamma+$	γ123
2.	3) $\gamma 1: \gamma 1 \to 1\gamma$	1γ23
3.	3) $\gamma 2: \gamma 2 \to 2\gamma$	12γ3
4.	3) $\gamma 3: \gamma 3 \to 3\gamma$	123γ
5.	4) $\gamma: \gamma \to \lambda\gamma$	123$\lambda\gamma$
6–7.	1) $3\lambda: 3\lambda \to \lambda; \gamma \to 3\gamma$	12λ3γ
8–9.	1) $2\lambda: 2\lambda \to \lambda; \gamma \to 2\gamma$	1λ32γ
10–11.	1) $1\lambda: 1\lambda \to \lambda; \gamma \to 1\gamma$	λ321γ
12–13.	2) $\lambda: \lambda \to \varnothing; \gamma \to \varnothing$ •	321

В результате работы алгоритма получена строка 321, которая является обращением строки 123. □

Для решения задачи обращения использовался второй вспомогательный символ γ, обозначающий конец строки. Чтобы не использовать символ γ, воспользуемся приемом, предложенным в работе [1], и заменим символ γ удвоенным символом $\lambda\lambda$.

Линейный нормальный алгоритм обращения В, использующий один вспомогательный символ, имеет схему со следующими подстановками:

1) $\lambda\lambda\lambda$: $\lambda\lambda\lambda \to \varnothing$ •

2) $Z_i\lambda\lambda$: $Z_i\lambda\lambda \to \lambda\lambda; \lambda \to Z_i\lambda;$ $i = 1, 2, \ldots, n$

3) λZ_i: $\lambda Z_i \to Z_i\lambda;$ $i = 1, 2, \ldots, n$

4) λ: $\lambda \to \lambda\lambda; \to \lambda+$

5) $\varnothing$: $\to \lambda+$

Работа алгоритма обращения В начинается с установки указателя λ в начало строки (подстановка 5). Указатель λ двигается вправо по строке (подстановка 3) и, достигнув ее конца, добавляет один символ λ в конец строки, а второй символ

λ – в начало строки (подстановка 4). Указатель λ в начале строки обозначает начало обращаемой строки. Удвоенный указатель λλ в конце строки двигается влево, удаляя символ перед собой; удаляемый символ добавляется перед указателем λ (подстановка 2). Достижение указателем λλ указателя λ означает, что обращение строки завершено, и указатели удаляются из строки (подстановка 1).

Пример 3. Рассмотрим работу линейного нормального алгоритма обращения В на примере строки 123 по шагам.

0.		123
1.	5) ∅: → λ+	λ1̲23
2.	3) λ1: λ1 → 1λ	1λ2̲3
3.	3) λ2: λ2 → 2λ	12λ̲3
4.	3) λ3: λ3 → 3λ	123λ̲
5–6.	4) λ: λ → λλ; → λ+	λ123λ̲λ̲
7–8.	2) 3λλ: 3λλ → λλ; λ → 3λ	3λ12λ̲λ̲
9–10.	2) 2λλ: 2λλ → λλ; λ → 2λ	32λ1λ̲λ̲
11-12.	2) 1λλ: 1λλ → λλ; λ → 1λ	321λ̲λ̲λ̲
13.	1) λλλ: λλλ → ∅ •	321

В результате работы алгоритма получено обращение строки 123. □

Все разработанные алгоритмы обращения имеют линейную трудоемкость (см. таблицу).

Алгоритмы обращения	Трудоемкость	Количество подстановок алгоритма
Алгоритм обращения А	2m + 2	2n + 2
Алгоритмы обращения Б и В	3m + 4	3n + 4
Алгоритм обращения А.А. Маркова и Н.М. Нагорного [1]	$m^2/2 + 5m/2 + 1$	$n^2 + n + 4$

Используем алгоритмы обращения для решения задачи сравнения двух слов x и y в строке S, разделенных символом 0. При этом потребуем, чтобы в случае равенства слов была результирующая строка 'равны', а в случае неравенства – 'не равны':

$$f(x,y)=\begin{cases}\text{'равны'}, & x=y,\\ \text{'не равны'}, & x\neq y.\end{cases}$$

Алгоритм сравнения слов работает следующим образом. Пусть строка содержит два слова baa и baa, разделенных символом 0:

baa0baa.

Обратим первое слово:

aab0baa.

Если символы слева и справа от символа 0 одинаковые, то алгоритм удаляет их из строки. Если символы слева и справа от символа 0 различны, то и слова различны.

Линейный нормальный алгоритм сравнения слов включает следующие подстановки:

1) λZ_i: $\lambda Z_i \to \lambda; \to Z_i+;$ $i = 1, 2, \ldots, n$

2) $\lambda 0$: $\lambda 0 \to \varphi 0$

3) $Z_i\varphi 0 Z_i$: $Z_i\varphi 0 Z_i \to \varphi 0;$ $i = 1, 2, \ldots, n$

4) $\varphi 0 Z_i$: $\varphi 0 Z_i \to \varnothing; \to \gamma+;$ $i = 1, 2, \ldots, n$

5) $Z_i\varphi 0$: $Z_i\varphi 0 \to \varnothing; \to \gamma+;$ $i = 1, 2, \ldots, n$

6) $\varphi 0$: $\varphi 0 \to$ 'равны' •

7) γZ_i: $\gamma Z_i \to \gamma;$ $i = 1, 2, \ldots, n$

8) γ: $\gamma \to$ 'не равны' •

9) $\varnothing$: $\to \lambda+$

Алгоритм состоит из двух этапов. На первом этапе обращается первое слово в строке (подстановки 1, 2 и 9). За основу взят алгоритм обращения А. На втором этапе одинаковые символы слева и справа от нуля в случае их равенства удаляются из строки (подстановка 3). Если слова равны между собой, то результатом работы алгоритма будет строка 'равны' (подстановка 6). Если слова не равны друг другу, то при сравнении символов возможны 3 случая: 1) сравниваемые символы не равны; 2) первое слово закончилось, а второе слово – нет; 3) второе слово закончилось, а первое слово – нет. Первый и второй случаи обрабатываются подстановкой 4, а третий – подстановкой 5. Однако во всех трех случаях все оставшиеся символы слов удаляются (подстановка 7), а результатом работы будет строка 'не равны' (подстановка 8).

Трудоемкость данного алгоритма сравнения слов T_{Cp} вычисляется по формуле

$$T_{Cp} = 3n_1 + n_2 - k + sg(n_1 + n_2 - 2k) + 3,$$

где n_1 – длина первого слова, n_2 – длина второго слова; k – количество одинаковых символов с начала слов подряд; $2n_1 + 2$ – трудоемкость обращения первого слова; sg(x) – функция сигнум [3]:

$$sg(x)=\begin{cases}0, x=0,\\ 1, x>0.\end{cases}$$

В случае неравенства слов выполняется на одну подстановку больше (подстановки 4–5), чем при равенстве слов. Поэтому необходима коррекция формулы с помощью функции сигнум sg(x). В случае равенства слов слагаемые $(n_2 - k)$ и $sg(n_1 + n_2 - 2k)$ равны нулю.

Задача удвоения. Задача удвоения строки заключается в приписывании слева или справа от строки ее копии. Например, «*abc*» – «*abcabc*».

Схема линейного нормального алгоритма А решения задачи удвоения строки алфавита Z = {a, b} будет состоять из следующих подстановок:

1) αa: $\alpha a \to a\alpha; \beta \to a\beta$

2) αb: $\alpha b \to b\alpha; \beta \to b\beta$

3) α: $\alpha \to \varnothing; \beta \to \varnothing$ •

4) $\varnothing$: $\to \beta\alpha+$

В конец строки добавляются указатели α и β (подстановка 4). Работа алгоритма заключается в перемещении указателя α влево и добавлении (копировании) символа перед указателем α в конец строки, обозначенный указателем β (подстановки 1–2). После того как указатель α достигнет начала строки, алгоритм заканчивает свое выполнение (подстановка 3).

Пример 4. Рассмотрим работу линейного нормального алгоритма удвоения А на примере той же строки baa по шагам.

0.		baa
1.	4) ∅: → βα+	β<u>α</u>baa
2–3.	2) αb: αb → bα; β → bβ	bβb<u>αa</u>
4–5.	1) αa: αa → aα; β → aβ	baβba<u>αa</u>
6–7.	1) αa: αa → aα; β → aβ	baaβbaa<u>α</u>
8–9.	3) α: α → ∅; β → ∅ •	baabaa

На девятом шаге алгоритма получена удвоенная строка baabaa. □

Построим линейный нормальный алгоритм удвоения с одним вспомогательным символом. Для этого воспользуемся тем же приемом замены символа β на удвоенный символ αα.

Схема линейного нормального алгоритма удвоения Б будет включать следующие подстановки:

1) ααa: ααa → aαα; α → aα

2) ααb: ααb → bαα; α → bα

3) α: α → ∅; αα → ∅ •

4) ∅: → ααα+

Пример 5. Рассмотрим работу линейного нормального алгоритма удвоения Б на примере строки baa по шагам.

0.		baa
1.	4) ∅: → ααα+	α<u>αα</u>baa
2–3.	2) ααb: ααb → bαα; α → bα	bαb<u>ααa</u>a
4–5.	1) ααa: ααa → aαα; α → aα	baαba<u>ααa</u>
6–7.	1) ααa: ααa → aαα; α → aα	baa<u>α</u>baaαα
8–9.	3) α: α → ∅; αα → ∅ •	baabaa

В результате работы алгоритма получена строка baabaa. □

Разработанные алгоритмы А и Б решают задачу удвоения с линейной трудоемкостью 2m + 3, тогда как нормальный алгоритм Маркова и Нагорного решает задачу удвоения с квадратичной трудоемкостью $m^2/2 + 5m/2 + 2$.

На основе линейного нормального алгоритма удвоения опишем линейный нормальный алгоритм сведения нормального алгоритма Маркова к линейному нормальному алгоритму. Алгоритм сведения перерабатывает схему нормального алгоритма Маркова в схему линейного нормального алгоритма, используя правила сведения алгоритмов, изложенные в работе [4].

Алгоритм сведения использует в качестве входных и выходных данных нормальную схему, записанную в виде строки.

Для записи нормальной схемы в виде строки подстановки записываются последовательно. Также используются следующие обозначения:

– признак конца подстановки;

↑ – разделитель левой и правой частей подстановки (→);

× – признак завершения алгоритма (•);

Λ – пустой символ (∅);

⊕ – признак добавления символа к строке слева или справа (+);

| – разделитель метки и подстановки в линейных нормальных алгоритмах (:).

Пусть множество $H = \{H_1, H_2, \ldots, H_k\}$ – множество символов, использующихся в левых и правых частях подстановок сводимого нормального алгоритма, множество $G = \{\#, \uparrow, \times, \oplus, |\}$ – множество элементов подстановок, а множество $J = \{\varepsilon, \varphi\}$ – множество вспомогательных символов алгоритма сведения. При этом множество H не имеет общих элементов с множествами G и J:

$$H \cap G = \varnothing \text{ и } H \cap J = \varnothing,$$

а множество Z является подмножеством множества H:

$$Z \subseteq H.$$

Пример 6. Запишем в виде строки линейный нормальный алгоритм, который перемещает первый символ в конец строки и содержит следующие подстановки.

1) $\alpha Z_i Z_j \rightarrow Z_j \alpha Z_i$; $\quad i, j = 1, 2, \ldots, n$

2) $\alpha Z_i \rightarrow Z_i \bullet$

3) $\rightarrow \alpha +$

Схема алгоритма в виде строки будет иметь следующий вид:

$\alpha Z_i Z_j \uparrow Z_j \alpha Z_i \# \alpha Z_i \uparrow Z_i \times \# \uparrow \alpha \oplus \#$.

Сведем данный нормальный алгоритм к линейному нормальному алгоритму

1) $\alpha Z_i Z_j$: $\quad \alpha Z_i Z_j \rightarrow Z_j \alpha Z_i$; $\quad i, j = 1, 2, \ldots, n$

2) αZ_i: $\quad \alpha Z_i \rightarrow Z_i \bullet$

3) ∅: $\quad \rightarrow \alpha +$

и запишем полученную схему в виде строки

$\alpha Z_i Z_j | \alpha Z_i Z_j \uparrow Z_j \alpha Z_i \# \alpha Z_i | \alpha Z_i \uparrow Z_i \times \# \Lambda | \uparrow \alpha \oplus \#$. □

Задачей алгоритма сведения будет удвоение левой части каждой подстановки. Алгоритм сведения нормального алгоритма Маркова к линейному нормальному алгоритму включает следующие подстановки:

1) $H_i\varepsilon$: $\quad H_i\varepsilon \rightarrow \varepsilon H_i$; $| \rightarrow |H_i$; $\quad i = 1, 2, \ldots, k$

2) $\#\varepsilon$: $\quad \#\varepsilon \rightarrow \varphi\#$

3) $\varepsilon|$: $\quad \varepsilon| \rightarrow \Lambda| \bullet$

4) ε: $\quad \varepsilon \rightarrow \varnothing \bullet$

5) $H_i\varphi$: $\quad H_i\varphi \rightarrow \varphi H_i$; $\quad i = 1, 2, \ldots, k$

6) $\times\varphi$: $\quad \times\varphi \rightarrow \varphi\times$

7) $\oplus\varphi$: $\quad \oplus\varphi \rightarrow \varphi\oplus$

8) $\#\uparrow\varphi$: $\quad \#\uparrow\varphi \rightarrow \varphi\#\Lambda|\uparrow$

9) $\uparrow\varphi$: $\quad \uparrow\varphi \rightarrow \varepsilon|\uparrow$

10) ∅: $\quad \rightarrow +\varepsilon$

В алгоритме используются 2 вспомогательных указателя: ε и φ. Указатель ε удваивает левые части подстановок (подстановки 1–2). Указатель φ обрабатывает правые части подстановок (подстановки 5–7). Подстановка 8 обрабатывает подстановки с пустой левой частью. При переходе из правой части подстановки в левую часть (подстановка 9) и при переходе между подстановками (подстановка 2) указатели чередуются. Алгоритм начинает работу с установки указателя ε в конец строки (подстановка 10) и заканчивает работу, когда этот указатель достигнет начала строки (подстановки 3–4).

Трудоемкость алгоритма сведения $T_{Свед}$ вычисляется по следующей формуле:

$$T_{Свед} = 2L + R + 2K – P + 2,$$

где L – количество символов в левой части подстановки (до стрелки); R – количество символов в правой части подстановки (после стрелки); K – количество подстановок; 2K – количество стрелок в подстановках ↑ и признаков конца подстановки #; P – количество подстановок с пустой левой частью (кроме первой) (подстановка 8); + 2 – установка и удаление указателей (подстановки 3–4 и 10).

Заключение. В статье показано, что даже при наложении ограничений линейные нормальные алгоритмы решают задачи обращения и удвоения с линейной трудоемкостью.

На основе алгоритмов обращения и удвоения разработаны линейные нормальные алгоритмы сравнения слов и сведения алгоритмов, оценена их трудоемкость.

Предложенная модификация линейных нормальных алгоритмов может использоваться для описания алгоритмов и обучения основам теории алгоритмов. Например, с помощью линейных нормальных алгоритмов были записаны алгоритмы преобразований числительных [4].

Для изучения нормальных алгоритмов Маркова и линейных нормальных алгоритмов в рамках курса «Математическая логика и теория алгоритмов» разработана программная система проверки знаний, позволяющая динамически генерировать задания обучаемым.

Направлением дальнейшей модернизации нормальных алгоритмов Маркова является замена подстановки другими операциями.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Марков А.А., Нагорный Н.М.* Теория алгорифмов. – М.: Наука, 1984. – 432 с.
2. *Нагорный Н.М.* Некоторые обобщения понятия нормального алгорифма // Тр. матем. ин-та АН СССР им. В.А. Стеклова, 52. – М.-Л.: Изд-во АН СССР, 1958. – С. 66-74.
3. *Цветков И.А.* Обращающий самопополняемый слева алгорифм в алфавите с одной дополнительной буквой // Математическое и программное обеспечение вычислительных систем: Межвуз. сб. науч. тр. / Под ред. А.Н. Пылькина. – М.: Горячая линия-Телеком, 2008. – С. 4-9.
4. *Пруцков А.В.* Линейная модификация нормальных алгоритмов Маркова // Информационные технологии в процессе подготовки современного специалиста: Межвуз. сб. статей. – Липецк, 2010. – Вып. 13. – С. 166-174.
5. *Кузнецов О.П., Адельсон-Вельский Г.М.* Дискретная математика для инженера. – М.: Энергоатомиздат, 1988. – 480 с.
6. *Пруцков А.В.* Обработка числительных естественных языков с помощью формальных грамматик и нормальных алгоритмов Маркова // Вестник Рязанского государственного радиотехнического университета. – Рязань, 2009. – Вып. 28. – С. 49-55.

Статью рекомендовал к опубликованию д.т.н., профессор В.Н. Ручкин.

Пруцков Александр Викторович – ГОУ ВПО «Рязанский государственный радиотехнический университет»; e-mail: ethiadmail@gmail.com; 390005, г. Рязань, ул. Гагарина, 59/1; тел.: +79106326176; к.т.н.; доцент.

Prutzkow Alexander Viktorovich – The Ryazan State Radio Engineering University; e-mail: ethiadmail@gmail.com; 59/1, Gagarin street, Ryazan, 390005, Russia; phone: +79106326176; cand. of tng. sc.; associate professor.