УДК 519.6

К.В. ШАХБАЗЯН, Ю.Г. ШУКУРЯН

ОБ ОПТИМИЗАЦИОННЫХ ПРОБЛЕМАХ ВКЛЮЧЕНИЯ ТРЕКОВ

Ключевые слова: *теория треков, обработка строк, оптимизация, проблема включения.*

ВВЕДЕНИЕ

Основными структурами, в которых хранится и передается информация, являются строки. Поэтому эффективная обработка строк с целью извлечения из них полезной информации стала предметом изучения целой ветви компьютерной науки (computer science) — стрингологии (stringology).

Часто строки обладают следующим важным свойством: некоторые пары соседних букв в строке могут быть переставлены без потери информации. В этом случае исходная строка может считаться эквивалентной строке, полученной при перестановке букв. Известным примером являются коды программ, где соседние команды могут быть переставлены при известных условиях. Это свойство эквивалентности широко используется при компактации кодов. Поэтому имеет смысл рассматривать задачи извлечения информации из строк с учетом такого рода эквивалентностей. Эту возможность предоставляет теория треков.

Понятие трека [1] было предложено в качестве модели параллельных процессов в вычислительных системах, и теория треков является удобным аппаратом для анализа параллельных процессов [1–3]. Решение задач сравнения треков важно там, где важно изучение строк с коммутирующими буквами, например в обработке сообщений, кодировании, искусственном интеллекте.

Трек есть класс эквивалентности строк в алфавите Σ, порождаемой заданным на алфавите Σ симметричным отношением I независимости (коммутативности) букв. Две строки считаются эквивалентными, если одна из них может быть получена из другой путем последовательных перестановок соседних коммутирующих букв. Множество треков образует свободный частично-коммутативный моноид $M(\Sigma, D)$, где $D = (\Sigma \times \Sigma) \setminus I$ — отношение зависимости на алфавите Σ.

Пусть $T = \{t_1, \ldots, t_k\}$ — конечное множество треков, принадлежащих моноиду $M(\Sigma, D)$. Важными характеристиками взаимной близости треков из T являются наибольший подтрек, общий для всех $t \in T$, и наименьший общий надтрек, т.е. трек,

содержащий в себе все треки $t \in T$ в качестве подтреков. (Трек s называется подтреком трека t, если существуют строки $y \in t$, $x \in s$ такие, что x есть подпоследовательность y.) В этом случае t называется надтреком трека s. Соответственно кратчайший общий неподтрек и длиннейший общий ненадтрек являются характеристиками различия между треками множества T. Если заданы два множества треков: U и S, то важной характеристикой этой пары множеств является кратчайший трек, различающий эти множества, а именно он либо является подтреком каждого трека из U, но не является подтреком каждого трека из S, либо, наоборот, не является подтреком каждого трека из U, но является подтреком каждого трека из S.

В настоящей статье рассматриваются следующие оптимизационные проблемы для конечного множества треков T. Найти:

1) наибольший общий подтрек,
2) кратчайший общий неподтрек,
3) кратчайший общий надтрек,
4) наибольший общий ненадтрек.

Цель статьи — доказательство полиномиальной сложности решения этих проблем для конечных множеств треков. Может показаться, что это NP-полные задачи даже для множеств T, состоящих из двух треков. Простейшее решение указанных задач можно представить себе так (например, задача 1): перебрать все пары строк (y_1, y_2), представляющих треки множества $T = \{t_1, t_2\}$. Для каждой пары известным методом [10] найти наибольшую общую подпоследовательность (т.е. трек, представленный этой подпоследовательностью), а затем из найденных подпоследовательностей выбрать наибольшую (аналогичные рассуждения для задач 2–4).

Для конечного $|T|$ предлагаются алгоритмы полиномиальной сложности решения этих задач. Заметим, что полученные оценки сложности существенно зависят от структуры алфавита зависимости, поскольку в них входит параметр m — число клик в минимальном покрытии кликами графа алфавита зависимости (Σ, D).

Упомянутые проблемы широко известны в формулировке для строк: 1) задача о длиннейшей общей для множества строк подпоследовательности, 2) задача о кратчайшей общей для множества строк неподпоследовательности, 3) задача о кратчайшей общей надпоследовательности, 4) задача о длиннейшей общей ненадпоследовательности.

Эти задачи для множеств строк рассматривались многими авторами [4–19]. Для случая фиксированной строки Baeza-Yates [4] ввел ациклический граф подпоследовательностей — DASG, т.е. минимальный частичный автомат, определяемый заданной строкой и распознающий язык подпоследовательностей этой строки. С использованием автомата DASG [8–11] были решены многие задачи сравнения множеств строк. Перечисленные задачи даже для строк являются NP-трудными в их общей постановке [12, 18, 19].

При решении задач, поставленных для треков, используем способ, предложенный Baeza-Yates и Crochemore [10] для строк. В основе этого способа лежит минимальный ациклический автомат $A(t)$, определяемый треком t, аналогичный DASG и распознающий язык подтреков трека t. Автомат $A(t)$ может быть построен за время $O(m|\Sigma|n^{m+1})$, где n — длина трека t, m — число клик в минимальном покрытии кликами графа алфавита зависимости (Σ, D).

Автомат $A(t)$ служит основой для полиномиальных алгоритмов решения следующих задач.

1. Найти общий наибольший подтрек для множества треков $T = \{t_1, \ldots, t_k\}$.

2. Найти общий кратчайший неподтрек для множества треков $T = \{t_1, \ldots, t_k\}$.

Автомат $A(t)$ используется также для нахождения кратчайшего различающего трека.

Задачи 3 и 4, относящиеся к надтрекам и ненадтрекам, решены построением графов, среди путей которых есть пути, несущие искомые решения.

Примером области, в которой возникает задача о надтреках, является компактация (оптимизация) кодов. В частности, при выполнении процедуральной абстракции [22], используемой при компактации кодов для процессоров с преди-

катными инструкциями (типа процессоров ARM, ориентированных на использование в портативных устройствах), происходит поиск наименьшей общей надстроки инструкций для множества кодов-строк. Однако естественно рассматривать коды как строки (инструкций), принадлежащие треку, алфавит независимости которого порождается коммутирующими инструкциями. В этом случае поиск общей надстроки может быть заменен поиском надтрека, что может привести к лучшей компактации.

Структура статьи такова: после предварительных сведений (разд. 1) в разд. 2 предлагается алгоритм построения по заданной строке $y \in t$ минимального частичного ациклического автомата $A(t)$, распознающего подтреки трека t и аналогичного графу DASG для строк.

Разд. 3 посвящен алгоритмам нахождения наибольшего общего подтрека, кратчайшего общего неподтрека и наименьшего различающего подтрека. С этой целью по набору строк $y_1 \in t_1, \ldots, y_k \in t_k$ — представителей треков множества $T = \{t_1, \ldots, t_k\}$ строится граф $G(T)$, в котором пути, исходящие из корня, несут слова, представляющие подтреки, общие для подмножеств треков множества T. Сложность предлагаемых алгоритмов составляет $O(m|\Sigma|n^{km+1})$.

В разд. 4 решаются задачи о наименьшем общем надтреке и наибольшем общем ненадтреке. Для решения первой задачи строится граф $B(t_1, t_2)$, каждый путь которого, начинающийся в корне, несет слово, представляющее надтрек для пары префиксов $p_1 \in \mathrm{Pref}(t_1)$, $p_2 \in \mathrm{Pref}(t_2)$. На основании этого задача сводится к нахождению слова, несомого кратчайшим путем в этом графе. Сложность предлагаемого алгоритма составляет $O(m|\Sigma|n^{2m+1})$. При построении использованы оценка числа префиксов трека и структура графа префиксов, описанные в [13] .

Для решения второй задачи (при условии существования решения) строится граф $G(T)$, в котором пути, начинающиеся в корне, несут все слова, представляющие ненадтреки для T. Тем самым задача сводится к нахождению слова, несомого длиннейшим путем в этом графе. Сложность алгоритма составляет $O(mk|\Sigma|^v)$.

1. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

Ниже используются следующие обозначения.

Пусть $y = y_1 \ldots y_n$ — строка в произвольном конечном алфавите Σ; $|y| = n$ — длина строки y. Множество $J(y) = \{1, \ldots, n\}$ будем называть множеством позиций строки y. Число $j \in J(y)$ назовем позицией буквы a в строке y, если $y_j = a$. Число позиций буквы a в строке y обозначим $|y|_a$, т.е. $|y|_a = |\{j \mid y_j = a,\ j \le n\}|$. Позиция $j \in J(y)$ называется k-й позицией буквы a в строке y, если $y_j = a$ и префикс $y_1 \ldots y_j$ строки y содержит k позиций буквы a, т.е. $|y_1 \ldots y_j|_a = k$. Обозначим k-ю позицию буквы a в строке y через $\mathrm{pos}(y, a, k)$. Позиция $\mathrm{pos}(y, a, k)$ определена только для $k \le |y|_a$.

Пусть Σ — конечный алфавит, $D \subset \Sigma \times \Sigma$ — рефлексивное и симметричное отношение зависимости, $I = (\Sigma \times \Sigma) \setminus D$ — отношение независимости или перестановочности. Отношение I индуцирует отношение эквивалентности $\approx$ на Σ^*. Две строки $x, y \in \Sigma^*$ являются эквивалентными относительно $\approx$, если существует последовательность $z_1, \ldots, z_n$ строк таких, что $x = z_1$, $y = z_n$ и для всех $i (1 \le i < n)$ существуют строки z_i', z_i'' и буквы a_i, b_i, удовлетворяющие условиям

$$\begin{cases} z_i = z_i' a_i b_i z_i'', \\ z_{i+1} = z_i' b_i a_i z_i'', \text{ где } (a_i, b_i) \in I, \end{cases}$$

т.е. две строки эквивалентны по отношению $\approx$ тогда и только тогда, когда одна строка может быть получена из другой перестановкой соседних независимых букв. Тогда множество $M = M(\Sigma, D)$ классов эквивалентных строк из Σ^* по отношению $\approx$ называется трековым моноидом, его элементы называются треками. На элементах M определено умножение — конкатенация. Трек t обозначается $[x]$ для любой представляющей строки $x \in t$. Длина трека t есть длина любого его представления $x \in t$ и обозначается $|t|$.

Для заданных двух треков p, $t \in M(\Sigma, D)$ считают, что p есть префикс t, если $t = pq$, где $q \in M(\Sigma, D)$. Обозначим $\mathrm{Pref}(t)$ множество префиксов трека t.

Пусть $(\Sigma_1, \ldots, \Sigma_m)$ — покрытие алфавита зависимости (Σ, D) кликами, т.е. семейство подмножеств Σ таких, что

$$\bigcup_{i=1}^{m} \Sigma_i = \Sigma,\ \Sigma_i \times \Sigma_i \subset D\ (i = 1, 2, \ldots, m),\ (a, b) \in D \Leftrightarrow \exists i\colon\ a,\ b \in \Sigma_i.$$

Далее везде будем считать $\Sigma_1, \ldots, \Sigma_m$ наименьшим покрытием кликами алфавита зависимости (Σ, D). Тогда m — число клик в наименьшем покрытии. Заметим, что $m = \alpha$, где α — размер наибольшей клики в графе независимости (Σ, I) алфавита моноида.

Любой трек $t \in M(\Sigma, D)$ может быть представлен m-кой строк [1], которую обозначим $\pi(t) = \{\pi_1(t), \ldots . \pi_m(t)\}$. Здесь $\pi_i(t) \in \Sigma_i^*$ — проекция строки $y \in t$ на Σ_i. Обозначим $r_i(t)$ длину проекции $\pi_i(t)$. Произведение длин всех проекций, увеличенных на единицу, обозначим $r(t) = \prod_{i=1}^{m} (r_i(t) + 1)$.

Для каждого заданного $t \in M(\Sigma, D)$ любой префикс $p \in \mathrm{Pref}(t)$ может быть представлен m-й целых чисел $\underline{p} = (r_1(p), \ldots, r_m(p))$ [13, 21]. Вычисление $\underline{p[a]}$ по заданному $\underline{p}$ требует $O(m)$ шагов.

Определение 1. Пусть $s, t \in M(\Sigma, D)$. Будем считать, что трек s является подтреком трека t, если существуют строки $x \in s$, $y \in t$ такие, что строка x является подпоследовательностью строки y. В этом случае будем писать $s \subset t$, а также считать, что трек t является надтреком трека s. ■

В [20, 21] приведено следующее эквивалентное определение отношения $s \subset t$.

Определение 1′. Пусть $s, t \in M(\Sigma, D)$. Трек s вложен в трек t (s является подтреком трека t), если существуют треки $t_0, \ldots, t_l$, $s_1, \ldots, s_l \in M(\Sigma, D)$ такие, что $t = t_0 s_1 t_1 \ldots s_l t_l$ и $s = s_1 \ldots s_l$. ■

Далее будем использовать термин «подтрек» и «надтрек» по ассоциации с «подпоследовательностью» и «надпоследовательностью». Существенно, что $s \subset t$ тогда и только тогда, когда для любой строки $y \in t$ существует строка $x \in s$ такая, что x есть подпоследовательность строки y. Однако обратное неверно, и из $s \subset t$ не следует, что для каждой строки $x \in s$ существует строка $y \in t$ такая, что x есть подпоследовательность строки y. Приведем пример.

Пример 1. Рассмотрим моноид $M(\Sigma, D)$, где $\Sigma = \{a, b, c\}$, $D = \{(a, b), (b, c)\}$. Пусть $t = [abc]$. Трек t состоит из единственной строки $y = abc$. Если $s = [ac]$, то строка $ca \in s$, но ca не является подпоследовательностью строки $y = abc$. ■

2. АВТОМАТ ПОДТРЕКОВ $A(t)$

Утверждение 1. Для каждого трека $t \in M(\Sigma, D)$ существует M-автомат $A(t)$, распознающий язык $L_{A(t)} = \{x \mid [x] \subset t\}$, т.е. язык подтреков трека t такой, что:

а) $A(t)$ — частичный ациклический автомат, число состояний $A(t)$ не превосходит $r(t) = \prod_{i=1}^{m} (r_i(t) + 1)$;

б) оценка $r(t)$ числа состояний автомата $A(t)$ достижима при $D = \varnothing$, т.е. при $m = |\Sigma|$, когда все буквы алфавита коммутируют;

в) при $m = 1$, т.е. при $I = \varnothing$, ациклический автомат $A(t)$ совпадает с автоматом DASG для строки $y \in t$; заметим, что в этом случае каждый трек $t \in M(\Sigma, D)$ содержит единственную строку;

г) по любой строке $y \in t$ можно построить автомат $A(t)$ за время $O(m|\Sigma||t|^{m+1}) = O(\alpha|\Sigma||t|^{\alpha+1})$, где α — размер наибольшей клики в графе независимости (Σ, I) алфавита моноида.

Доказательство. Пусть трек t задан строкой $y = y_1 \ldots y_{|y|} \in \Sigma^*$. Для произвольной строки $x = x_1 \ldots x_{|x|}$ выполняется $[x] \subset [y]$ в моноиде $M(\Sigma, D)$ тогда и толь-

ко тогда, когда существует вложение $\varphi_x : J(x) = \{1, \ldots, |x|\} \to J(y) = \{1, \ldots, |y|\}$ множества позиций строки x в множество позиций строки y, сохраняющее букву и порядок:

$$x_i = y_{\varphi_x(i)}, \ (x_i, x_j) \in D, \ i, j \in J(x), \ i < j \Rightarrow \varphi_x(i) < \varphi_x(j). \tag{1}$$

Условия (1) не определяют однозначно вложение φ_x. Далее будем считать, что вложение φ_x конкретизировано следующими рекуррентными соотношениями.

1. Для пустого слова e имеем $\varphi_e = \varnothing$.

2. Если построено вложение φ_x для строки $x = x_1 \ldots x_{|x|}$, то для любого $a \in \Sigma$ вложение φ_{xa} для строки $xa = x_1 \ldots x_{|x|+1}$ определяется как

$$\begin{cases} \varphi_{xa}(i) = \varphi_x(i), \text{ если } i \le |x|, \\ \varphi_{xa}(|x|+1) = \min\{i \mid y_i = a, \ i \notin F(\varphi_x)\}, \end{cases} \tag{2}$$

где $F(\varphi_x)$ — множество запрещенных для $\varphi_{xa}(|x|+1)$ позиций строки y, определяемое равенством

$$F(\varphi_x) = \varphi_x(J(x)) \cup \{i \mid \exists\, j > i\colon \ j \in \varphi_x(J(x)), \ (y_i, y_j) \in D\}.$$

Множество $F(\varphi_x)$, кроме позиций $\varphi_x(J(x))$, содержит позиции букв, зависящих от букв множества $\varphi_x(J(x))$ и расположенных в строке y следующим образом: если $i \in F(\varphi_x)$, то либо $i \in \varphi_x(J(x))$, либо существует позиция $j \in J(y)$ такая, что $i < j$, $j \in \varphi_x(J(x))$ и y_i зависит от y_j.

Очевидно, что если для строк x и y вложения φ_x, построенного по правилу (2), не существует, то не существует никакого другого вложения, удовлетворяющего условию (1). Если известно множество $F(\varphi_x)$, то

$$F(\varphi_{xa}) = F(\varphi_x) \cup \{j \mid j \le \varphi_{xa}(|x|+1), \ (a, y_j) \in D\} =$$

$$= F(\varphi_x) \cup \{j \mid j \le \min\{i \mid y_i = a, \ i \notin F(\varphi_x)\}, \ (a, y_j) \in D\}. \tag{3}$$

Поставим в соответствие вложению φ_x: $J(x) \to J(y)$ набор целых чисел $P(\varphi_x) = (P_1(\varphi_x), \ldots, P_m(\varphi_x))$, который определим рекуррентно как

$$P(\varphi_s) = (0, \ldots, 0) = P_0,$$

$$P_i(\varphi_{xa}) = \begin{cases} P_i(\varphi_x), \text{ если } a \notin \Sigma_i, \\ \operatorname{pos}(\pi_i(y), a, k_{\max} + 1), \text{ если } a \in \Sigma_i, \end{cases} \quad i \in \{1, \ldots, m\}, \tag{4}$$

где число $k_{\max}(P(\varphi_x))$ — номер максимального вхождения буквы a в строках $\{\pi_i(y), \ i = 1, \ldots, m\}$ в позиции, не превосходящей соответствующего $P_i(\varphi_x)$:

$$k_{\max}(P(\varphi_x)) = \max_{i=1,\ldots,m} \{\max\{k \mid \operatorname{pos}(\pi_i(y), a, k) \le P_i(\varphi_x)\}\}.$$

Если $k_{\max}(P(\varphi_x)) + 1 > |y|$, то набор $P_i(\varphi_{xa})$ не определен. Легко видеть, что согласно (3) существует взаимно однозначное соответствие между $P_i(\varphi_x)$ и $F(\varphi_x)$, определяемое формулой

$$\operatorname{pos}(y, a, k) \in F(\varphi_x) \leftrightarrow \exists i \in \{1, \ldots, m\}\colon \ \operatorname{pos}(\pi_i(y), a, k) \le P_i(\varphi_x).$$

Поэтому можно считать, что набор $P(\varphi_x)$ представляет множество позиций $F(\varphi_x)$.

Далее определим автомат $A(t) = (\mathcal{P}_t, \Sigma, \delta_t, P_0, \mathcal{P}_t)$. Множеством состояний автомата $A(t)$ является $\mathcal{P}_t = \{P = (P_1, \ldots, P_m) \mid P_i \le |\pi_i(y)| = r_i, \ P_i \in N\}$. Переходы автомата $A(t)$ суть правила (4), согласно которым выполняется переход $P(\varphi_x) \xrightarrow{a} P'(\varphi_{xa})$. Начальное состояние есть $P_0 = (0, \ldots, 0)$, каждое достижимое $P \in \mathcal{P}_t$ является финальным состоянием автомата $A(t)$. Очевидно: $P_0 \xrightarrow{a} P(\varphi_x)$ тогда и только тогда, когда $[x] \subset [y]$.

Заметим, что если $P \xrightarrow{a} P'$ и $P = (P_1, \ldots, P_m)$, $P' = (P'_1, \ldots, P'_m)$, то существует $i \in \{1, \ldots, m\}$, для которого $P_i < P'_i$. Отсюда следует ацикличность графа переходов автомата $A(t)$. Легко увидеть, что $\delta_t(P, ab) = \delta_t(P, ba)$ для каждого $P \in \mathcal{P}_t$, откуда следует, что $A(t)$ есть M-автомат.

1. Докажем теперь минимальность автомата $A(t)=(\mathcal{P}_t,\Sigma,\delta_t,P_0,\mathcal{P}_t)$. Необходимо показать, что если $P, P'\in\mathcal{P}_t$ и $P\neq P'$, то автомат $A_P=(\mathcal{P}_t,\Sigma,\delta_t,P,\mathcal{P}_t)$ не эквивалентен автомату $A_{P'}=(\mathcal{P}_t,\Sigma,\delta_t,P',\mathcal{P}_t)$. Для каждой буквы $a\in\Sigma$ и каждого набора $P=(P_1,\ldots,P_m)\in\mathcal{P}_t$ определим число $k(a,P)=\max\limits_i\{|\pi_{i1}\ldots\pi_{iP_i}|_a \mid i\in\Sigma_i\}$, $a\in\Sigma$, т.е. максимальное число позиций буквы a в префиксах проекций $\pi_i(t)$ длины P_i. Если $P\neq P'$, найдется буква $a\in\Sigma$ такая, что $k(a,P)\neq k(a,P')$.

Предположим $k(a,P)<(a,P')$. Тогда строка $a^{|y|_a-k(a,P)}$ допускается автоматом A_P и не допускается автоматом $A_{P'}$. Минимальность автомата $A(t)$ доказана.

2. Очевидно, что для числа состояний автомата $A(t)$ верна оценка $|\mathcal{P}_t|\le r(t)$. Если $D=\varnothing$, т.е. $m=\alpha=|\Sigma|$, и клики покрытия содержат по одной букве, то достижимо каждое состояние $(P_1,\ldots,P_m)$, где для $\Sigma_i=\{a\}$ $P_i\le|t|_a$.

3. Если D — полный граф, $a=m=1$, то каждый трек $t\in M(\Sigma,D)$ содержит единственную строку y. В этом случае число состояний множества $\mathcal{P}_t$, где $[y]=t$, есть $|t|+1=|y|+1$ и автомат $A(t)$ совпадает с автоматом DASG [4].

4. Вычисление каждого перехода имеет временную сложность $O(m|t|)=O(a|t|)$, поскольку требует просмотра всех проекций π_i. Следовательно, построение графа переходов $A(t)$ имеет сложность $O(m|\Sigma||t|^{m+1})=O(\alpha|\Sigma||t|^{\alpha+1})$.

Утверждение 1 доказано. ■

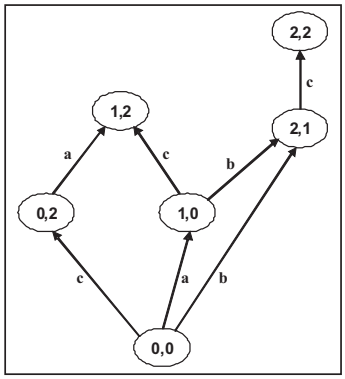
Пример 2. Рассмотрим моноид $M(\Sigma,D)$, где $\Sigma=\{a,b,c\}$, $D=\{(a,b),(b,c)\}$, трек $t=[aabcbc]$. Тогда $\Sigma_1=\{a,b\}$, $\Sigma_2=\{b,c\}$ — клики покрытия (Σ,D), $\pi_1(t)=aabb$, $\pi_2(t)=bcbc$ — проекции t на клики Σ_1,Σ_2.

Определим, вложены ли в t треки $s_1=[cac]$, $s_2=[cacb]$.

Имеем

$\varphi_b=\varnothing$	$F(\varphi_b)=\varnothing$	$P(\varphi_b)=(0,0)$	
$\varphi_c=\{(1,4)\}$	$F(\varphi_c)=\{3,4\}$	$P(\varphi_c)=(0,2)$	$[c]\subset t$
$\varphi_{ca}=\{(1,4),(2,1)\}$	$F(\varphi_{ca})=\{1,3,4\}$	$P(\varphi_{ca})=(1,2)$	$[ca]\subset t$
$\varphi_{cac}=\{(1,4),(2,1),(3,6)\}$	$F(\varphi_{cac})=\{1,3,4,5,6\}$	$P(\varphi_{cac})=(1,4)$	$s_1=[cac]\subset t$
φ_{cacb} не определено,	$F(\varphi_{cacb})$ не определено, следовательно,		$s_2=[cacb]\not\subset t$.

Легко увидеть, что $(0,0)\xrightarrow{c}(0,2)\xrightarrow{a}(1,2)\xrightarrow{c}(1,4)$. ■



Šuń. 1. Γšąō ļåšåðīāīā ąāņīģąņą $A([abc])$

Пример 3. Рассмотрим моноид $M(\Sigma,D)$: $\Sigma=\{a,b,c\}$, $D=\{(a,b),(b,c)\}$. Построим автомат $A(t)$ для $t=[abc]$. На рис. 1 изображен граф переходов автомата $A(t)$. В вершинах записаны достижимые состояния из $\mathcal{P}_t$ для $y=abc$. ■

3. ЗАДАЧИ О ПОДТРЕКАХ И НЕПОДТРЕКАХ ДЛЯ МНОЖЕСТВА $T\subseteq M(\Sigma,D)$

Пусть задано множество треков $T=\{t_1,\ldots,t_k\}\subseteq M(\Sigma,D)$. Предположим $e\notin T$, $t_i\not\subset t_j$ при $i\neq j$, $i,j\in\{1,\ldots,k\}$.

Введем в рассмотрение граф подтреков для множества T, который послужит основой алгоритмов сравнения треков конечного множества T.

Построим ориентированный граф $G(T)=(Q,U,q_0,\mu)$, где Q — вершины, U — дуги графа $G(t)$, q_0 — его корень, μ — функция, метящая дуги $G(T)$.

Используя обозначения, введенные в разд. 2 для автомата $A(t)$, положим

$$Q=(\mathcal{P}_{t_1}\cup\{\#\})\times\ldots\times(\mathcal{P}_{t_k}\cup\{\#\}),\ q_T^0=(P_0\times P_0\times\ldots\times P_0).$$

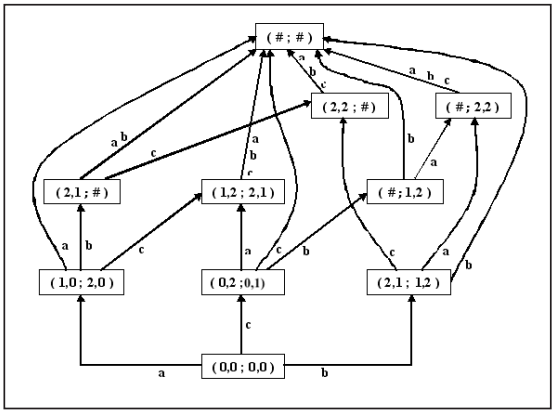
Далее, пусть $q=(q_1,\ldots,q_k)$, $q'=(q'_1,\ldots,q'_k)\in Q$. Пара $(q,q')\in U$ и ее метка $\mu(q,q')=a$ тогда и только тогда, когда для всех $i=1,\ldots,k$ имеет место

$q'_i=\delta_{t_i}(a,q_i)$, если переход $q'_i=\delta_{t_i}(a,q_i)$ определен в автомате $A(t_i)$;

$q'_i=\#$, если этот переход не определен или если $q_i=\#$.

Итак, граф $G(T)$ построен. Он имеет $O(n^{mk})$ вершин, где $n=\max_i\{|t_i|\}$, k — число треков множества T, m — число клик в наименьшем покрытии кликами графа зависимости алфавита Σ. Ацикличность и минимальность автомата $A(T)$ с графом переходов $G(T)$ очевидна. Сложность построения одного перехода составляет $O(kmn)$, откуда следует полиномиальная сложность $O(km|\Sigma|n^{mk+1})$ построения графа $G(T)$.

Пример 4. Пусть $\Sigma=\{a,b,c\}$, $D=\{(a,b),(b,c)\}$, $t_1=[abc]$, $t_2=[cba]$. Рассмотрим проекции $\pi_1(t_1)=ab$, $\pi_2(t_1)=bc$, $\pi_1(t_2)=ba$, $\pi_2(t_2)=cb$ на клики $\Sigma_1=\{a,b\}$, $\Sigma_2=\{b,c\}$. Пусть также $T=\{t_1,t_2\}$ — множество, состоящее из двух треков. Построим граф переходов автомата $G(t)$ (рис. 2). ■



Šćń. 2. Ćšąō ļåšåõīāīā ąāņīğąņą $G(T)$

Построенный граф $G(T)$ можно использовать для решения разных задач поиска подтреков и неподтреков множества T.

Пусть в графе $G(T)$ путь, несущий строку x, начинается в корне q_0 и заканчивается в вершине $q=(q_1,\ldots,q_k)$. Определим подмножество $K(q)=\{i\mid q_i\neq\#,\ i=1,\ldots,k\}$. Тогда $[x]\subset t_i$ для всех $i\in K(q)$ и $[x]\not\subset t_i$ для всех $i\notin K(q)$. Следовательно, x является общим подтреком для t_i при $i\in K(q)$ и общим неподтреком для всех t_i при $i\notin K(q)$.

Отсюда вытекает решение следующих задач по построенному графу $G(T)$.

Задача о наибольшем общем подтреке. Задано конечное множество треков $T=(t_1,\ldots,t_k)\subseteq M(\Sigma,D)$. Каждый трек t_i задан строкой $y_i\in t_i$. Требуется найти в $M(\Sigma,D)$ подтрек наибольшей длины, общий для всех треков множества T.

Задача решается путем построения графа $G(T)$ и поиска в нем длиннейшего среди путей, ведущих к вершинам $q=(q_1,\ldots,q_k)$, которые не содержат символов $\#$. Так как существует полиномиальный алгоритм для поиска такого пути, то задача о наибольшем общем подтреке может быть решена за время $O(m|\Sigma|n^{mk+1})$. ■

Задача о кратчайшем общем неподтреке. Задано конечное множество треков: $T=(t_1,\ldots,t_k)\subseteq M(\Sigma,D)$. Каждый трек t_i задан строкой $y_i\in t_i$. Требуется найти в $M(\Sigma,D)$ кратчайший трек, не являющийся подтреком для всех треков множества T.

Задача решается путем поиска в графе $G(T)$ кратчайшего среди путей, ведущих к вершине $q=(\#,\ldots,\#)$. Следовательно, задача о кратчайшем общем неподтреке может быть решена за время $O(m|\Sigma|n^{mk+1})$, т.е. за время построения графа $G(T)$ и время поиска в нем нужного кратчайшего пути.

Задача о кратчайшем различающем треке. Пусть заданы два множества треков: $U=\{u_1,\ldots,u_k\}$ и $S=\{s_1,\ldots,s_l\}$, $u_i,s_j\in M(\Sigma,D)$, $i=1,\ldots,k$, $j=1,\ldots,l$.

Определение 4. Треком, различающим множества треков U и S, называется такой трек $v\in M(\Sigma,D)$, что либо v — общий подтрек для треков множества U, но не является подтреком для каждого трека множества S, либо, наоборот, v является общим подтреком для треков множества S, но не является подтреком каждого трека множества U.

Положим $T=\{u_1,\ldots,u_k,s_1,\ldots,s_l\}=U\cup S$. Задача нахождения кратчайшего различающего трека решается путем поиска по графу $G(T)$ кратчайшего пути, ведущего из q_0 в вершину $q=(q_1,\ldots,q_{k+l})$ такую, что либо

$$q_1=\#,\ldots,\ q_k=\#,\ \text{но}\ q_{k+1}\neq\#,\ldots,\ q_{k+l}\neq\#,$$

либо

$$q_1\neq\#,\ldots,\ q_k\neq\#,\ \text{но}\ q_{k+1}=\#,\ldots,\ q_{k+l}=\#.$$

Следовательно, задача о кратчайшем различающем треке может быть решена за время $O(m(k+l)|\Sigma|n^{m(k+l)+1})$, т.е. за время построения $G(T)$ и время поиска в нем кратчайшего пути к вершине описанного вида. ■

Пример 5. На графе $G(T)$, построенном на рис. 2 (пример 4), можно проиллюстрировать решение перечисленных задач. Пусть $t_1=[abc]$, $t_2=[cba]$. Наибольшим общим подтреком для $T=\{t_1,t_2\}$ является $[ac]$. Кратчайшие общие неподтреки для T суть $[aa]$, $[bb]$, $[cc]$. Если $U=\{t_1\}$, $S=\{t_2\}$, то кратчайшие треки, различающие U и S, суть $[ab]$, $[cb]$. ■

4. ЗАДАЧИ О НАДТРЕКАХ И НЕНАДТРЕКАХ ДЛЯ МНОЖЕСТВА $T\subseteq M(\Sigma,D)$

Пусть задано множество треков $T=\{t_1,\ldots,t_k\}\subseteq M(\Sigma,D)$. Предположим $e\notin T$, $t_i\not\subset t_j$ при $i\neq j$, $i,j\in\{1,\ldots,k\}$.

Определение 5. Пусть $t_1,t_2\in M(\Sigma,D)$. Общим надтреком для треков t_1,t_2 будем называть трек $t\in M(\Sigma,D)$, если $t_1\subset t$, $t_2\subset t$. ■

Задача о наименьшем общем надтреке. Задача о наименьшем общем надтреке заключается в нахождении трека наименьшей длины, который является общим надтреком для t_1,t_2. Каждый трек t_i задан строкой $y_i\in t_i$.

Обозначим $l(p_1,p_2)$ длину общего наименьшего надтрека для $p_1\in\mathrm{Pref}(t_1)$ и $p_2\in\mathrm{Pref}(t_2)$. Рассмотрим множество троек:

$$\Phi(p_1,p_2)=\{\langle p_1',p_2',a\rangle\mid\ a\in\Sigma,\ ((p_1=p_1'[a])\wedge(p_2=p_2'[a]))\vee$$

$$\vee((p_1=p_1')(p_2=p_2'[a]))\vee((p_1=p_1'[a])(p_2=p_2'))\},$$

где $p_1'\in\mathrm{Pref}(t_1)$ и $p_2'\in\mathrm{Pref}(t_2)$.

Очевидно, что в множестве $\Phi(p_1,p_2)$ существует такая тройка $\langle p_1',p_2',a\rangle$, что $l(p_1,p_2)=l(p_1',p_2')+1$. Следовательно, рекуррентные формулы для вычисления $l(p_1,p_2)$ имеют вид $l(p_1,p_2)=1+\min\{l(p_1',p_2')\mid\langle p_1',p_2',a\rangle\in\Phi(p_1,p_2)\}$, $l(e,e)=0$, где e — пустой трек.

Утверждение 2. Существует частичный ациклический автомат $B(t_1,t_2)$, распознающий конечный язык $L_{B(t_1,t_2)}\subseteq\{x\mid t_1,t_2\subset[x]\}$, который содержит все кратчайшие надтреки треков t_1 и t_2. Автомат $B(t_1,t_2)$ имеет $O(n^{2a})=O(n^{2m})$ состояний, где $n=\max\{|t_1|,|t_2|\}$. Этот автомат может быть построен за $O(m|\Sigma|n^{2m+1})$ шагов.

Доказательство. Построим автомат $B(t_1,t_2)=(\mathrm{Pref}(t_1)\times\mathrm{Pref}(t_2),(e,e),(t_1,t_2),\delta)$, где граф переходов определяется так: если $(p_1,p_2)\in\mathrm{Pref}(t_1)\times\mathrm{Pref}(t_2)$, то из вершины (p_1',p_2') существует переход $\delta(a,(p_1',p_2'))=(p_1,p_2)$ при $\langle p_1',p'_2,a\rangle\in\Phi(p_1,p_2)$. Очевидно, что каждый путь из вершины (e,e) в вершину (p_1,p_2) несет строку x такую, что $p_1,p_2\subset[x]$, т.е. $[x]$ — общий надтрек треков p_1 и p_2. Как показано в [13], $|\mathrm{Pref}(t)|=O(n^m)$. Воспользуемся этим результатом для оценки сложности построения автомата $B(t_1,t_2)$.

Сложность построения автомата $B(t_1,t_2)$ составляет $O(m|\Sigma|n^{2m+1})$, где $n=\max\{|t_1|,|t_2|\}$, так как $|\mathrm{Pref}(t_1)|\leq|t_1|^m$, $|\mathrm{Pref}(t_2)|\leq|t_2|^m$, а сложность построения одного перехода составляет $O(mn)$.

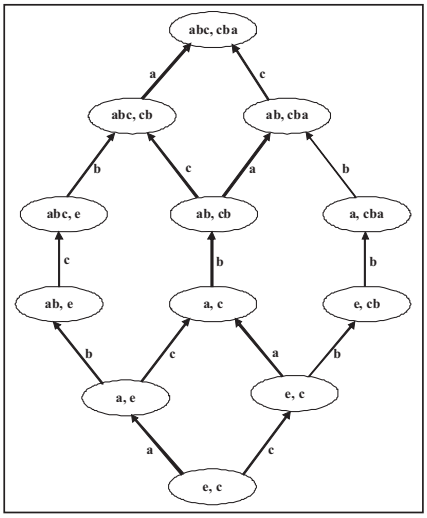
Задача о нахождении наименьшего общего надтрека сводится к построению автомата $B(t_1, t_2)$ и нахождению в его графе переходов кратчайшего пути, ведущего к вершине (t_1, t_2). ■

Задача легко обобщается на конечное множество треков $T = \{t_1, \dots, t_k\}$.

Пример 6. Рассмотрим моноид $M(\Sigma, D)$, где $\Sigma = \{a, b, c\}$, $D = \{(a, b), (b, c)\}$, и треки $t_1 = [abc]$, $t_2 = [cba]$. Граф переходов автомата $B(t_1, t_2)$ изображен на рис. 3. В каждой вершине графа указаны соответствующие префиксы треков t_1 и t_2. Кратчайшим надтреком для t_1 и t_2 является трек $t = [acbac]$. ■

Задача о наибольшем общем ненадтреке. Задача о наибольшем общем ненадтреке для $T = \{t_1, \dots, t_k\} \subseteq M(\Sigma, D)$ заключается в построении трека t наибольшей длины такого, что $t_i \not\subset t$ для всех $i = 1, \dots, k$.

В [18] показано, что задача о нахождении наибольшей общей ненадпоследовательности для множества строк L в алфавите A имеет решение тогда и только тогда, когда для каждой буквы $a \in A$ строка $a^{\nu_a} \in L$ при некотором $\nu_a > 1$. Легко увидеть, что это условие является необходимым и достаточным и для существования наибольшего общего ненадтрека.



Šćń. 3. Ćšąō ļåšåõīāīā ąāņīġąņą $B(t_1, t_2)$

Утверждение 3. Пусть $T = \{t_1, \dots, t_k\} \subseteq M(\Sigma, D)$ и для каждой буквы $a \in \Sigma$ существует число $\nu_a > 1$ такое, что трек $[a^{\nu_a}] \in T$. Тогда задача о построении наибольшего общего ненадтрека сводится к задаче нахождения длиннейшего пути в ациклическом графе $G(T)$ с числом вершин, не превосходящим $|\Sigma|^{\nu}$, где $\nu = \Sigma_{a \in \Sigma}(\nu_{a-1})$. Построение графа $G(T)$ требует $O(mk|\Sigma^{\nu}|)$ шагов.

Доказательство. Определим граф $G(T) = (V, E, \mu)$ следующим образом.

1. Вершины графа $G(T)$ суть наборы $P = (\underline{p_1}, \dots, \underline{p_k})$, где $p_i \in \mathrm{Pref}(t_i)$, $i = 1, \dots, k$.

2. Вершина $P_0 = (0, \dots, 0) \in V$.

3. Если вершина $P = (\underline{p_1}, \dots, \underline{p_k}) \in V$, то переход $P \xrightarrow{a} P' = (\underline{p'_1}, \dots, \underline{p'_k})$ определен тогда и только тогда, когда для всех $i = 1, \dots, k$ выполнено равенство

$$p'_i = \begin{cases} p_i, \text{ если } p_i[a] \notin \mathrm{Pref}(t_i), \\ p_i[a], \text{ если } t_i \neq p_i[a] \notin \mathrm{Pref}(t_i). \end{cases}$$

При этом $P' \in V$, дуга $(P, P') \in E$ и помечена буквой a.

Ацикличность графа $G(T)$ очевидна.

Рассмотрим произвольный путь $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \dots \xrightarrow{a_l} P_l$. Трек $[a_1 \dots a_l]$ не содержит более ν_{a-1} экземпляров буквы a для каждой буквы $a \in \Sigma$. Трек $[a_1 \dots a_l]$ является общим ненадтреком для множества T, поскольку содержит лишь собственные префиксы треков T. Граф $G(T)$ содержит все пути, несущие треки с указанными свойствами. Очевидно, что длиннейший путь в $G(T)$ несет наибольший общий ненадтрек множества T.

Пример 7. Рассмотрим моноид $M(\Sigma, D)$, где $\Sigma = \{a, b, c\}$, $D = \{(a, b), (b, c)\}$, и множество треков $T = \{[abc], [cba], [aa], [bb], [cc]\}$.

Длиннейшими ненадтреками для T являются треки $[cab], [bca]$.

ЗАКЛЮЧЕНИЕ

Построенный минимальный детерминированный ациклический автомат $A(t)$ для трека $t \in M(\Sigma, D)$ допускает язык подтреков трека t. Автомат использован для решения задач поиска общего наибольшего подтрека, общего кратчайшего неподтрека для треков конечного множества T и для задачи поиска кратчайшего подтрека, различающего множества. Эти задачи могут быть решены за полиномиальное время при ограниченном $|T|$. Решаются также задачи о построении наименьшего общего надтрека и наибольшего общего ненадтрека для треков множества T. Получены алгоритмы полиномиальной сложности при конечном $|T|$.

СПИСОК ЛИТЕРАТУРЫ

1. Dikert V., Rozenberg G. The book of traces // Handbook of Formal Languages — **3**. — Beyond Words. — N.Y.: Springer-Verlag, 1997. — 285 p.
2. Letichevsky A.A. On the equivalence of automata over semigroup // Theoretic Cybernetics. — 1970 — **6.** — P. 3–71 (in Russian).
3. Mazurkiewicz A. Concurrent program schemes and their interpretations // DAIMI Rep. PB. — Aarhus: Aarhus University. — 1977. — **78**. — P. 76–92.
4. Baeza-Yates R.A. Searching sequences // Theoretical Computer Science. — 1991. — **78**, N 2. — P. 363–376.
5. Fraser C.B. Subsequences and supersequences of strings. Ph.D. Thesis // University of Glasgow, UK, 1955. — P. 1–121.
6. Crochemore M., Hancart C., Lecroq T. Algorithms on strings // Cambridge University Press, 2007. — 392 p.
7. Crochemore M., Rytter W. Jewels of stringology // World Scientific Publiching Co. Rtc. Ltd., 2002. — 309 p.
8. Crochemore M., Melchar B., Tronicek Z. Directed acyclic subsequence graph: overview // J. of Discrete Algorithms. — 2003. — **1**, N 3. — P. 255–280.
9. Crochemore M., Giambruno L. On-line construction of a small automaton for a finite set of words // The Prague Stringology Conference, 2009. — P. 39–51.
10. Crochemore M., Tronicek Z. Directed aciclic subsequence graph for multiple texts // Technical Report. — IGM-99. — Institut Gaspard-Monge, 1999. — P. 1–93.
11. Melchar B., Polcar T. The longest common subsequence problem. A finite automata approach // LNCS. — 2003. — **2759.** — P. 93–106.
12. Tronicek Z. Problems related to subsequences and supersequences // Strong Processing and Information Retreaval Symposium, 1999. — P. 199–205.
13. Avellone A., Goldwurm M. Analysis of algorithms for the recognition of rational and context-free trace languages // Theoretical Informatics and Applications. — 1998. — **32**, N 2. — P. 17–65.
14. Boasson L., Cegielski P., Guessarian I., Matiyasevich Y. Window-accumulated subsequence problem is linear // Annals of Pure and Applied Logic. — 2001. — **113**, N 7. — P. 36–45.
15. Guessarian I., Cegielski P. Tree inclusions in windows and slices // CSIT Conference, 2000. — P. 47–50.
16. Cegielski P., Guessarian I., Matiyasevich Y. Multiple serial episode matching // CSIT Conference, 2005. — P. 26–38.
17. Ji X., Bailey J., Dong G. Mining minimal distinguishing subsequences pattern with gap constraints // ICDM, 2005. — P. 194–201.
18. Rubinov A.R., Timkovsky V.G. String noninclusion optimization problems // SIAM J. Discrete Math. — 1998. — **11**, N 3. — P. 456–467.
19. Middendorf M. The shortest nonsubsequence problem is NP-complete // Theoret. Comput. Sci. — 1993. — **108**. — P. 365–369.
20. Shahbazyan K.V., Shoukourian Yu.H. Inclusion problems in trace monoids // CSIT Conference, 2009. — 78–92.
21. Shahbazyan K.V., Shoukourian Yu.H. On some matching problems in trace monoids // Patrick Cegielski, ed. / Studies in Weak Arithmetics, Lecture Notes, 196 — CSLI Public. — Stanford, 2010. — 213 p.
22. Cheung W., Evans W., Moses J. Predicated instructons for code compaction // Proceedings of the 7th International Workshop on Software and Compilers for Embedded Systems (SCOPES), LNCS. — 2003. — **2826**. — P. 17–32.

Поступила 14.05.2010