

О сложности поиска подстроки

Е. М. Перпер

В работе рассматривается задача поиска подслова во множестве слов. Эта задача состоит в следующем: пусть дано множество слов; надо для произвольного подслова найти все слова из этого множества, в которых это подслово содержится. В данной работе разработаны 3 алгоритма поиска, с помощью которых получены оценки функциональной сложности поиска.

Ключевые слова: слово, подслово, поиск, информационный граф, сложность, объём.

1. Введение

Важной задачей современной вычислительной лингвистики является обеспечение компьютера пониманием естественного языка. Для этого, в частности, компьютер должен уметь узнавать множество различных характеристик (морфологических, синтаксических, семантических) каждого слова из данного ему текста (см., например, [1]). Как правило, эти характеристики содержатся в специальных словарях. Однако не всегда слово содержится в словаре в той же форме, что и в тексте (например, если слово — имя существительное, то, возможно, в словаре оно будет содержаться лишь в именительном падеже). В связи с этим возникает задача нахождения всех слов в словаре, содержащих в себе заданное подслово. Очевидно, можно перебирать все слова из словаря и искать в каждом из них заданное подслово, однако это может занять очень много времени. Цель данной работы — найти такие алгоритмы поиска, которые позволяли бы находить все нужные слова за небольшое время.

Автор выражает благодарность профессору Эльяру Эльдаровичу Гасанову за научное руководство и помощь в научной работе.

2. Основные понятия и формулировка результатов

Обозначим через

$$W_n^k = \bigcup_{s=1}^n \{1, 2, \dots, k\}^s \quad (1)$$

множество слов длины не более n над алфавитом $\{1, 2, \dots, k\}$. Для каждого слова $w \in W_n^k$ его длину будем обозначать как $l(w)$. Через $w[a \dots b]$, где $1 \leq a \leq l(w)$, $1 \leq b \leq l(w)$, будем обозначать подслово слова w , начинающееся с его a -й буквы и заканчивающееся его b -й буквой, если $a \leq b$, и пустое слово иначе. Через $w[a]$, $1 \leq a \leq l(w)$, будем обозначать a -ю букву слова w . Будем говорить, что слова $w \in W_n^k$ и $v \in W_n^k$ равны или совпадают (и писать $w = v$), если $l(v) = l(w)$ и $v[1] = w[1], v[2] = w[2], \dots, v[l(v)] = w[l(w)]$. Будем считать, что любые два пустых слова совпадают. Для слова w его подслово $w[1 \dots q]$, $q \in \mathbb{N}$, $q \leq l(w)$, будем называть началом длины q слова w , а подслово $w[l(w) - q + 1 \dots l(w)]$, $q \in \mathbb{N}$, $q \leq l(w)$ — концом длины q слова w .

Скажем, что слово v лексикографически предшествует слову w , если выполняется одно из следующих двух условий: 1) $\exists i \in \mathbb{N}$, $i \leq \min(l(v), l(w))$: $v[i] < w[i]$ и $\forall j \in \mathbb{N}$, $j < i$: $v[j] = w[j]$; 2) $l(v) < l(w)$ и $\forall j \in \mathbb{N}$, $j \leq l(v)$: $v[j] = w[j]$. Будем записывать это следующим образом: $v < w$.

Конкатенацией слов v_1 и v_2 (будем обозначать её $v_1 + v_2$) назовём слово w такое, что $l(w) = l(v_1) + l(v_2)$, $w[1 \dots l(v_1)] = v_1$, $w[l(v_1) + 1 \dots l(w)] = v_2$.

Скажем, что слово v лексикографически предшествует слову w относительно r -й буквы, если $r \leq \min(l(v), l(w))$ и $(v[r \dots l(v)] + v[1 \dots r - 1]) < (w[r \dots l(w)] + w[1 \dots r - 1])$. Будем записывать это следующим образом: $v <_r w$.

Рассмотрим множество запросов

$$X = W_n^k \quad (2)$$

и множество записей

$$Y = W_n^k. \quad (3)$$

Введём бинарное отношение ρ , которое позволит устанавливать, когда запись $y \in Y$ удовлетворяет запросу $x \in X$ (отношение поиска):

$$x\rho y \Leftrightarrow \exists i \in \mathbb{N} : x = y[i \dots l(x) + i - 1], x \in X, y \in Y. \quad (4)$$

Будем рассматривать задачу информационного поиска $I = \langle X, V, \rho \rangle$, где

$$V = \{v_1, v_2, \dots, v_p\}, V \subseteq Y. \quad (5)$$

Назовём эту задачу задачей поиска подстроки. Множество V в дальнейшем будем называть библиотекой. Так как X и ρ фиксированы, задача поиска подстроки полностью определяется библиотекой.

Содержательно будем считать, что задача $I = \langle X, V, \rho \rangle$ состоит в перечислении для произвольно взятого запроса $x \in X$ всех тех и только тех записей из V , которые находятся в отношении ρ с запросом x , то есть удовлетворяют запросу x .

Скажем, что слово $v_2 \in V$ непосредственно следует за словом $v_1 \in V$ (или: v_2 — следующее за v_1 слово, или: слово v_1 непосредственно предшествует слову v_2) относительно r -й буквы, если $v_1 <_r v_2$, и при этом в библиотеке V нет слова v_3 такого, что $v_1 <_r v_3 <_r v_2$.

Задачу I будем рассматривать в рамках информационно-графовой модели данных (см., например, [2, 3]).

Введём понятие информационного графа (ИГ).

В формальном определении понятия ИГ используются 2 описанных выше множества X и Y , а также:

множество F предикатов, заданных на множестве X (предикаты — это функции, которые могут принимать только два значения: 0 или 1);

множество G переключателей, заданных на множестве X (переключатели — это функции, область значений которых является начальным отрезком натурального ряда).

Пару $\mathcal{F} = \langle F, G \rangle$ будем называть базовым множеством.

Рассмотрим произвольный ориентированный граф.

Выделим в нём одну вершину. Назовём её корнем.

Выделим в графе какие-либо другие вершины. Назовём их листьями. Сопоставим каждому листу некоторую запись из множества Y . Это соответствие назовем нагрузкой листьев.

Выделим в графе некоторые вершины (это могут быть в том числе корень и листья) и назовем их точками переключения. Если β —

вершина сети, то через ψ_β обозначим полустепень исхода вершины β (то есть число исходящих из этой вершины рёбер).

Каждой точке переключения β сопоставим некий символ из G . Это соответствие назовем нагрузкой точек переключения.

Для каждой точки переключения β ребрам, из нее исходящим, поставим во взаимно однозначное соответствие числа из множества $\{1, 2, \dots, \psi_\beta\}$. Эти ребра назовем переключательными, а это соответствие — нагрузкой переключательных ребер.

Ребра, не являющиеся переключательными, назовем предикатными.

Каждому предикатному ребру графа сопоставим некоторый символ из множества F . Это соответствие назовем нагрузкой предикатных ребер. Полученную нагруженную сеть назовем информационным графом над базовым множеством $\mathcal{F} = \langle F, G \rangle$.

Определим функционирование ИГ.

Скажем, что

предикатное ребро проводит запрос $x \in X$, если предикат, приписанный этому ребру, принимает значение 1 на запросе x ;

переключательное ребро, которому приписан номер r , проводит запрос $x \in X$, если переключатель, приписанный началу этого ребра, принимает значение r на запросе x ;

ориентированная цепь ребер проводит запрос $x \in X$, если каждое ребро цепи проводит запрос x ;

запрос $x \in X$ проходит в вершину β ИГ, если существует ориентированная цепь, ведущая из корня в вершину β , которая проводит запрос x ;

запись y , приписанная листу α , попадает (включается) в ответ ИГ на запрос $x \in X$, если запрос x проходит в лист α .

Ответом ИГ u на запрос x назовем множество записей, попавших в ответ ИГ на запрос x , и обозначим его $J_{u(x)}$. Эту функцию $J_{u(x)}$ будем считать результатом функционирования ИГ u и называть функцией ответа ИГ u .

Понятие ИГ полностью определено.

Скажем, что ИГ u решает ЗИП $I = \{X, V, \rho\}$, если для любого запроса $x \in X$ ответ на этот запрос содержит все те и только те записи из V , которые удовлетворяют запросу x , то есть $J_{u(x)} = \{y \in V : x\rho y\}$.

Обозначим как $\mathcal{U}(I, \mathcal{F})$ множество всех ИГ над базовым множеством \mathcal{F} , решающих ЗИП I .

Везде далее будем рассматривать задачу поиска подстроки, то есть задачу $I = \langle X, V, \rho \rangle$, где X , V и ρ определены соотношениями (1)–(5).

Алгоритмом построения информационного графа назовём отображение U , сопоставляющее библиотеке V определённый информационный граф u , решающий ЗИП $I = \langle X, V, \rho \rangle$ (будем также обозначать этот граф как $U(V)$). Это отображение мы называем алгоритмом, так как мы будем описывать его с помощью определённой последовательности шагов, выполнив которые, мы построим информационный граф.

Определим понятие сложности информационного графа на запросе. Пусть $N(f, u, x)$ — количество вычислений функции f при обработке в ИГ u запроса x , а $t(f, x)$ — отображение, ставящее каждой паре (f, x) , где f — функция, x — запрос, в соответствие некоторое положительное число (будем считать, что это число характеризует время вычисления функции f на запросе x). Сложностью вычисления ИГ u на запросе x назовём число $T(u, x) = \sum_{f \in F \cup G} N(f, u, x)t(f, x)$. Число

$T(u) = \max_{x \in X} T(u, x)$ будем называть сложностью ИГ u . Через $T(U, p)$ обозначим $\max_{|V|=p} T(U(V))$, то есть максимальную из сложностей всех информационных графов, построенных с помощью алгоритма U по библиотекам из p слов.

Объёмом $Q(u)$ ИГ u назовём число ребер в ИГ u . Через $Q(U, p)$ обозначим $\max_{|V|=p} Q(U(V))$, то есть максимальный из объёмов всех информационных графов, построенных с помощью алгоритма U по библиотекам из p слов.

Сложностью задачи I при базовом множестве \mathcal{F} и заданном объёме q назовём число $T(I, \mathcal{F}, q) = \min\{T(u) : u \in \mathcal{U}(I, \mathcal{F}), Q(u) \leq q\}$.

В качестве базового множества будем рассматривать пару $\mathcal{F} = \langle F, G \rangle$, где

$$F = F_1 \cup F_2, \quad G = G_1 \cup G_2 \cup G_3; \quad (6)$$

$$F_1 = \{f_a^1(w) : a \in \mathbb{Z}, a \geq 0\}; \quad (7)$$

$$F_2 = \{f_v^2(w) : v \in Y\}; \quad (8)$$

$$G_1 = \{g_h^1(w) : h \in \mathbb{Z}, h \geq 0\}; \quad (9)$$

$$G_2 = \{g_h^2(w) : h \in \mathbb{Z}, h \geq 0\}; \quad (10)$$

$$G_3 = \{g^3(w)\}; \quad (11)$$

$$f_a^1(w) = \begin{cases} 1, & \text{если } l(w) \leq a; \\ 0, & \text{если } l(w) > a; \end{cases} \quad (12)$$

$$f_v^2(w) = \begin{cases} 1, & \text{если } w = v[1 \dots l(w)]; \\ 0, & \text{если } w \neq v[1 \dots l(w)]; \end{cases} \quad (13)$$

$$g_h^1(m, w) = w[h + 1]; \quad (14)$$

$$g_h^2(w) = \begin{cases} 1, & \text{если } l(w) > h; \\ 2, & \text{если } l(w) \leq h; \end{cases} \quad (15)$$

$$g^3(w) = l(w). \quad (16)$$

Теорема 1. Пусть $I = \langle X, V, \rho \rangle$ — задача поиска подстроки, определённая соотношениями (1)–(5), где k — фиксированное натуральное число. Пусть \mathcal{F} — базовое множество, определяемое соотношениями (6)–(16), и пусть для функций (12)–(16) выполнено $t(f^1, x) = t(g^1, x) = t(g^2, x) = t(g^3, x) = 1$, $t(f^2, x) = l(x)$. Тогда для сложности $T(I, \mathcal{F}, q)$ при заданном объёме $q \geq n(p + 1)$ справедливы оценки:

$$n + d - 1 \leq T(I, \mathcal{F}, q) \leq \begin{cases} n + p(n + 1)^2/4, & \text{если } q \geq n(p + 1) \\ 2n + d, & \text{если } q \gtrsim pn^2k/2 \\ 1 + n + d, & \text{если } q \gtrsim pn^3k/6 \end{cases}$$

при $p \rightarrow \infty$, где $d = d(V)$ — наибольшее по всем возможным запросам $x \in X$ количество слов из библиотеки $V \subseteq Y$, удовлетворяющих запросу x .

Доказательство теоремы проводится с помощью алгоритмов U_1 , U_2 , U_3 построения информационных графов, решающих задачу I .

3. Алгоритмы U_1, U_2, U_3 построения ИГ

1. Алгоритм U_1 построения ИГ.

Пусть $V = \{v_1, v_2, \dots, v_p\}$.

С помощью шагов 1)–9) будем строить подграф, решающий задачу нахождения по запросу w для каждого $m \in \mathbb{N}$, $m \leq \max_{v \in V} l(v) - l(w) + 1$ всех слов v_i таких, что $v_i[m \dots l(w) - 1 + m] = w$, если одно такое слово, лексикографически предшествующее относительно m -й буквы всем словам v_i , уже найдено.

1) Для каждого слова $v_j, j = 1, \dots, p$ выберем $l(v_j)$ листьев и припишем каждому из этих листьев слово v_j . Назовём эти листья $\gamma_{j,1}, \gamma_{j,2}, \dots, \gamma_{j,l(v_j)}$.

2) Положим $r = 1$. Счётчик r будет отражать номер слова из библиотеки, которое мы в данный момент рассматриваем.

3) Положим $i = 1$. Счётчик i будет содержать номер рассматриваемой нами в данный момент буквы слова v_r .

4) Выпустим из листа $\gamma_{r,i}$ $n - i + 1$ рёбер, припишем им числа от 1 до $n - i + 1$. Сопоставим этому листу переключатель $g^3(w)$.

5) Положим $t = 1$. Счётчик t будет содержать длину рассматриваемого нами в данный момент подслова слова v_r .

6) Выходящее из листа $\gamma_{r,i}$ ребро под номером t направим в лист $\gamma_{j,m}$, которому сопоставлено слово $v_j \in V$ такое, что выполнены следующие условия:

а) $v_j[m \dots t + m - 1] = v_r[i \dots t + i - 1]$, $m \geq i$ и $v_j[s \dots t + s - 1] \neq v_r[i \dots t + i - 1]$, $s = 1, \dots, m - 1$;

б) если $m = i$, то слово v_j непосредственно следует за словом v_r относительно i -й буквы, если же $m > i$, то слово v_j предшествует относительно m -й буквы всем остальным словам, для которых выполнено условие а);

с) среди всех слов, удовлетворяющих условиям а) и б) при некотором m , слово v_j удовлетворяет условиям а) и б) при наименьшем m .

Если нет ни для одного слова из V , для которого при некотором m были бы выполнены условия а) и б), то направим выходящее из листа $\gamma_{r,i}$ ребро под номером t в лист, которому не приписано никакое слово (иногда такую вершину будем называть пустым листом).

7) Если $t < l(w) - i + 1$ (значит, не все подслова, начинающиеся с i -й буквы слова v_r , рассмотрены), увеличим t на 1 и перейдём к пункту 6.

8) Если $i < l(v_r)$, увеличим i на 1 и перейдём к пункту 4 (будем рассматривать следующую букву слова v_r).

9) Если $r < p$, увеличим r на 1 и перейдём к пункту 3 (будем рассматривать следующее слово из библиотеки).

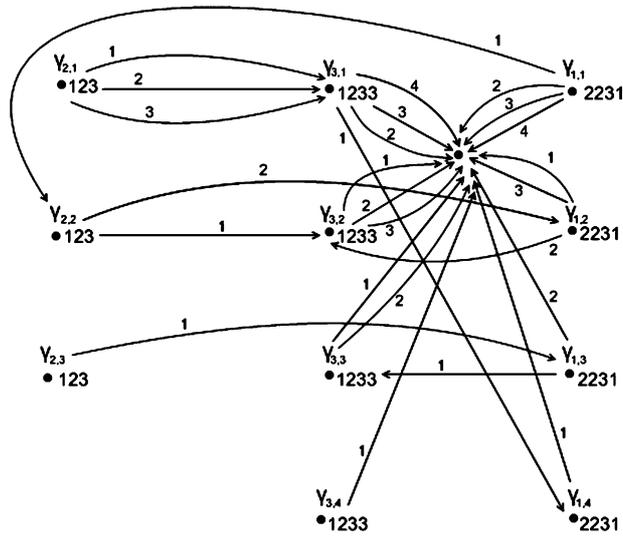


Рис. 1. Вид графа, построенного на этапах 1–9 алгоритма U_1 по библиотеке $\{2231, 123, 1233\}$. Каждой вершине сопоставлен переключатель $g^3(w)$.

С помощью шагов 10)–18) будем строить подграф, решающий задачу нахождения по запросу w для каждого $m \in \mathbb{N}$, $m \leq \max_{v \in V} l(v) - l(w) + 1$ слова v_i такого, что $v_i[m \dots l(w) - 1 + m] = w$ и v_i лексикографически предшествует относительно m -й буквы всем остальным словам $v \in V$ таким, что $v[m \dots l(w) - 1 + m] = w$.

10) Через $V' = \{v'_1, v'_2, \dots, v'_p\}$, $1 \leq i \leq p$ будем обозначать множество, состоящее из всех слов $v_j[a \dots b]$, где $j = 1, 2, \dots, p$; $a = 1, \dots, l(v_j)$; $b = a, \dots, l(v_j)$.

Пример: если $k = 3$, $V = \{2231, 123, 1233\}$, то $V' = \{2231, 223, 231, 22, 23, 31, 2, 3, 1, 123, 12, 1233, 233, 33\}$.

11) Выберем корень. Выпустим из него n рёбер, припишем им числа от 1 до n . Сопоставим корню переключатель $g_0^3(w)$.

12) Положим $r = 1$. Счётчик r будет отображать номер рассматриваемого нами в данный момент слова из множества V' . Будем

иметь в виду, что для слова $v'_i[a \dots b]$, $a = 1, \dots, l(v_j)$; $b = a, \dots, l(v_j)$ всегда найдётся такое число $m \in \mathbb{N}$, $m \leq n$ и такое слово $v_j \in V$, что $v'_i[a \dots b] = v_j[m \dots b - a + m]$ (это видно из определения множества V').

13) Если $r = 1$ либо $\forall s, 1 \leq s < r : l(v'_s) \neq l(v'_r)$, вершину, в которую ведёт $l(v'_r)$ -е исходящее из корня ребро, назовём $\beta_{r,0}$. Выпустим из этой вершины k рёбер, припишем им числа от 1 до k . Сопоставим вершине $\beta_{r,0}$ переключатель $g_0^1(w)$. Положим $q = 0, d = r$.

14) Если $\exists s, 1 \leq s < r : l(v'_s) = l(v'_r)$, рассмотрим слово $v'_d, d < r$ такое, что значение $q = q(v'_d, v'_r) = \max_{b < r} q(v'_b, v'_r)$, где число $q(v'_b, v'_r)$ задаётся условиями: $l(v'_b) = l(v'_r), v'_b[1 \dots q] = v'_r[1 \dots q]$ и $v'_b[q+1] \neq v'_r[q+1]$. Если таких слов несколько, то выберем из них то, у которого число d наименьшее. То есть среди слов $v'_b, b < r$ выбираются все слова длины $l(v'_r)$, имеющие наиболее длинное общее начало со словом v'_r , и из этих слов выбирается рассмотренное нами раньше остальных. Число q отражает длину общего начала.

Пример: если $k = 3, V = \{2231, 123, 1233\}$, то на шаге 14 в момент $r = 13$ мы имеем $v'_r = 233$. Найдём d и q . В моменты $r = 1, \dots, 12$ были рассмотрены 3 слова длины $l(v_13') = 3 : 223, 231, 123$. У слов $v_2 = 223$ и 233 совпадает первая буква; у слов $v_3 = 231$ и 233 — первая и вторая буквы; у слов $v_10 = 231$ и 233 не совпадают уже первые буквы. Поэтому имеем $d = 3, q = 2$.

15) Положим $h = q + 1$. Счётчик h будет указывать, сколько букв слова v'_r мы уже рассмотрели.

16) Если $h < l(v'_r)$ (значит, ещё не все буквы слова v'_r рассмотрены), вершину, в которую ведёт исходящее из вершины $\beta_{r,h-1}$ ($\beta_{d,q}$, если $h = q + 1$) ребро под номером $v'_r[h]$, назовём $\beta_{r,h}$. Выпустим из этой вершины k рёбер, припишем им числа от 1 до k . Сопоставим вершине $\beta_{r,h}$ переключатель $g_h^1(w)$. Увеличим h на 1 и снова выполним пункт 16.

17) (переход на этот пункт произойдёт, если $h = l(v'_r)$.) Исходящее из вершины $\beta_{r,h-1}$ ($\beta_{d,q}$, если $h = q + 1$) ребро под номером $v'_r[h]$ направим в лист $\gamma_{j,m}$, которому сопоставлено слово $v_j \in V$ такое, что для него выполнены 2 условия:

- а) $v_j[m \dots h + m - 1] = v'_r$ и для всех i , для которых $v_i[m \dots h + m - 1] = v'_r, i \neq j$, имеем $v_j <_m v_i$;
- б) $\forall i v_i[m' \dots h + m' - 1] \neq v'_r, m' = 1, \dots, m - 1$.

18) Если $r < p'$ (значит, не все слова из множества V' рассмотрены), увеличим r на 1 и перейдём к пункту 13. Если $r = p'$, то информационный граф, решающий ЗИП I , построен.

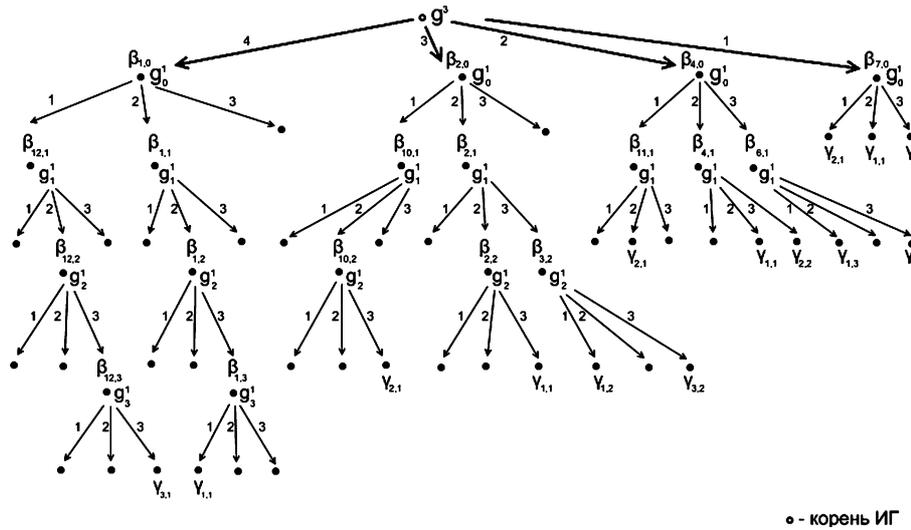


Рис. 2. Вид графа, построенного на этапах 10–18 алгоритма U_1 по библиотеке $\{2231, 123, 1233\}$. На рисунке ради удобства сделано так, что некоторые листья $\gamma_{i,j}$ встречаются несколько раз; на самом деле это один и тот же лист.

Покажем, что информационный граф, построенный по алгоритму U_1 , решает ЗИП I .

Пусть запрос представляет собой слово w . Заметим сначала, что если $l(w) > \max_{1 \leq i \leq p} l(v_i)$, то запрос пройдёт из корня по ребру с номером $l(w)$ в вершину, из которой не исходит ни одно ребро, и которой не приписано никакое слово. Значит, ответом ИГ на этот запрос будет пустое множество (как и должно быть, если данный ИГ решает ЗИП I). Если же $l(w) \leq \max_{1 \leq i \leq p} l(v_i)$, то запрос пройдёт в вершину $\beta_{i,0}$, где i — наименьшее число такое, что $l(w) = l(v'_i)$.

Пусть теперь запрос прошёл в некоторую вершину $\beta_{d,h}$. Если не существует такого r , что $l(v'_r) = l(v'_d)$, $v'_r[1 \dots h] = v'_d[1 \dots h]$, $w[h+1] = v'_r[h+1]$, то запрос пройдёт в вершину, из которой не исходит ни

одно ребро, и которой не приписано ни одно слово. Пусть такое r существует. Возможны 2 случая:

1) если $h < l(w) - 1$, запрос w пройдёт в вершину $\beta_{s,h+1}$ такую, что $l(v'_s) = l(v'_r)$, $v'_s[1 \dots h+1] = v'_r[1 \dots h+1]$, $s \leq r$;

2) если $h = l(w) - 1$, запрос w пройдёт в вершину $\gamma_{j,m}$, которой сопоставлено слово $v_j \in V$, удовлетворяющее условиям а) и б) пункта 17 алгоритма U_1 (с заменой v'_r на w). Как видно из этих условий, слово v_j будет удовлетворять запросу w

Как видно из этих условий, запрос пройдёт в лист, которому приписано некоторое слово, в том и только в том случае, если в множестве V' есть слово, совпадающее с запросом. Заметим теперь, что в библиотеке V есть хотя бы 1 слово, удовлетворяющее запросу w , тогда и только тогда, когда в V' найдётся слово, совпадающее с w . Отсюда следует, что если в V нет слов, удовлетворяющих запросу, то в ответ на запрос не будет выдано ни одно слово из V .

Пусть в библиотеке V есть слово, удовлетворяющее запросу w . Тогда в множестве V' (содержащей все под слова всех слов из V) найдётся слово $v'_i = w$. В этом случае запрос пройдёт в лист, которому, как видно из условий а) и б) пункта 17 алгоритма U_1 , приписано слово, удовлетворяющее запросу. Из этого листа запрос может последовательно пройти и в другие листья. Как видно из условий а), б) и с) пункта 6 алгоритма U_1 , для каждого удовлетворяющего запросу слова запрос пройдёт ровно в одну вершину, которой это слово приписано. Ни в какие другие вершины, за исключением пустого листа, запрос не пройдёт. Таким образом, ИГ, построенный по алгоритму U_1 , решает ЗИП I.

II. Алгоритм U_2 построения информационного графа.

Пусть $V = \{v_1, v_2, \dots, v_p\}$.

Шаги 1–9 полностью совпадают с шагами 1–9 алгоритма U_1 .

С помощью шагов 10)–23) будем строить подграф, решающий задачу нахождения по запросу w для каждого $m \in \mathbb{N}$, $m \leq \max_{v \in V} l(v) - l(w) + 1$ слова v_i такого, что $v_i[m \dots l(w) - 1 + m] = w$ и v_i лексикографически предшествует относительно m -й буквы всем остальным словам $v \in V$ таким, что $v[m \dots l(w) - 1 + m] = w$.

10) Через $V' = \{v'_1, v'_2, \dots, v'_p\}$ будем обозначать множество, состоящее из всех слов $v_j[a \dots l(v_j)]$, где $j = 1, 2, \dots, p$; $a = 1, \dots, l(v_j)$.

Пример: если $k = 3$, $V = \{2231, 123, 1233\}$, то $V' = \{2231, 231, 31, 1, 123, 23, 3, 1233, 233, 33\}$.

11) Выберем корень. Назовём его $\beta_{1,0}$. Выпустим из него k рёбер, припишем им числа от 1 до k . Сопоставим корню переключатель $g_0^1(w)$.

12) Положим $r = 1$. Счётчик r будет отображать номер рассматриваемого нами в данный момент слова из множества V' . Будем иметь в виду, что для слова $v'_i[1 \dots b]$, $b = 1, \dots, l(w)$ всегда найдётся такое число $m \in \mathbb{N}$, $m \leq n$ и такое слово $v_j \in V$, что $v'_i[1 \dots b] = v_j[m \dots b+m-1]$ (это видно из определения множества V').

13) Если $r = 1$, положим $q = 0$, иначе рассмотрим слово v'_d , $d < r$ такое, что значение $q = q(v'_d, v'_r) = \max_{b < r} q(v'_b, v'_r)$, где число $q(v'_b, v'_r)$ задаётся условием: $v'_b[1 \dots q] = v'_r[1 \dots q]$ и либо $l(v'_r) = q$, либо $l(v'_b) = q$, либо $v'_b[q+1] \neq v'_r[q+1]$. Если таких слов несколько, то выберем из них то, у которого число d наименьшее. То есть среди слов v'_b , $b < r$ выбираются все слова, имеющие наиболее длинное общее начало со словом v'_r , и из этих слов выбирается рассмотренное нами раньше остальных. Число q отражает длину общего начала.

Пример: если $k = 3$, $V = \{2231, 123, 1233\}$, то на шаге 13 в момент $r = 8$, мы имеем $v'_r = 1233$. Найдём d и q . У слова $v_5 = 123$ три первые буквы совпадают с тремя первыми буквами слова 1233 и длина слова v_5 равна 3; длина общего начала каждого из слов v'_j , $j < 8$, $j \neq 5$ и слова 1233 — не более 2. Поэтому имеем $d = 5$, $q = 3$.

14) Если $q = l(v'_r)$ (значит, подграф, соответствующий слову v'_r , уже построен), перейдём к пункту 23.

15) Положим $h = q + 1$. Счётчик h будет указывать, сколько букв слова v'_r мы уже рассмотрели.

16) Если $l(v'_d) > q$ (значит, либо $q = 0$, либо в построенном на данный момент графе есть вершина $\beta_{d,h-1}$), то вершину, являющуюся концом $v'_d[h]$ -го исходящего из вершины $\beta_{d,h-1}$ (из корня, если $q = 0$) ребра, назовём $\alpha_{r,h}$ и перейдём к пункту 20.

17) (переход на этот пункт осуществляется, если $l(v'_d) = q$) Вершину, являющуюся концом 1-го исходящего из вершины $\alpha_{d,h-1}$ ребра, назовём $\beta_{r,h-1}$.

18) Выпустим из вершины $\beta_{r,h-1}$ k рёбер, припишем им числа от 1 до k . Сопоставим данной вершине переключатель $g_q^1(w)$.

19) Вершину, являющуюся концом $v'_r[h]$ -го исходящего из вершины $\beta_{r,h-1}$ ребра, назовём $\alpha_{r,h}$.

20) Выпустим из вершины $\alpha_{r,h}$ 2 ребра, припишем им числа 1 и 2. Сопоставим данной вершине переключатель $g_h^2(w)$.

21) Исходящее из вершины $\alpha_{r,h}$ ребро под номером 2 направим в лист $\gamma_{j,m}$, которому сопоставлено слово $v_j \in V$ такое, что для него выполнены 2 условия:

а) $v_j[m \dots h + m - 1] = v'_r[1 \dots h]$ и для всех i , для которых $v_i[m \dots h + m - 1] = v'_r[1 \dots h]$, $i \neq j$, имеем $v_j <_m v_i$;

б) $\forall i v_i[m' \dots h + m' - 1] \neq v'_r[1 \dots h]$, $m' = 1, \dots, m - 1$.

22) Если $l(v'_r) > h$ (значит, ещё не все буквы слова v'_r рассмотрены), увеличим вершину, в которую ведёт исходящее из вершины $\alpha_{r,h}$ ребро под номером 1, назовём $\beta_{r,h}$, увеличим h на 1 и перейдём к пункту 19.

23) Если $r < p'$ (значит, не все слова из множества V' рассмотрены), увеличим r на 1 и перейдём к пункту 13. Если $r = p'$, то информационный граф, решающий ЗИП I , построен.

Покажем, что информационный граф, построенный по алгоритму U_2 , решает ЗИП I .

Пусть запрос прошёл в некоторую вершину $\beta_{d,h}$. Если не существует такого r , что $v'_r[1 \dots h] = v'_d[1 \dots h]$, $w[h + 1] = v'_r[h + 1]$, то запрос пройдёт в пустой лист.

Если же такое r существует, запрос пройдёт в вершину $\alpha_{s,h+1}$ такую, что $v'_s[1 \dots h + 1] = v'_r[1 \dots h + 1]$, $s \leq r$. Далее возможны 3 случая:

1) если $h + 1 < l(w)$, $h + 1 < l(v'_s)$, запрос w пройдёт в вершину $\beta_{s,h+1}$;

2) если $h + 1 < l(w)$, $h + 1 = l(v'_s)$, запрос w пройдёт в пустой лист;

3) если $h + 1 = l(w)$, $h + 1 < l(v'_s)$, запрос w пройдёт в лист $\gamma_{j,m}$, которому сопоставлено слово $v_j \in V$ такое, что для него выполнены условия а) и б) пункта 21 алгоритма U_2 .

Как видно из этих условий, запрос пройдёт в лист, которому приписано некоторое слово, в том и только в том случае, если в множестве V' есть слово v'_i такое, что $w = v'_i[1 \dots l(w)]$. Но такое слово v'_i в V' есть тогда и только тогда, когда в V есть хотя бы 1 слово, удовлетворяющее запросу w . Отсюда следует, что если в V нет слов, удовлетворяющих запросу, то в ответ на запрос не будет выдано ни одно слово из V .

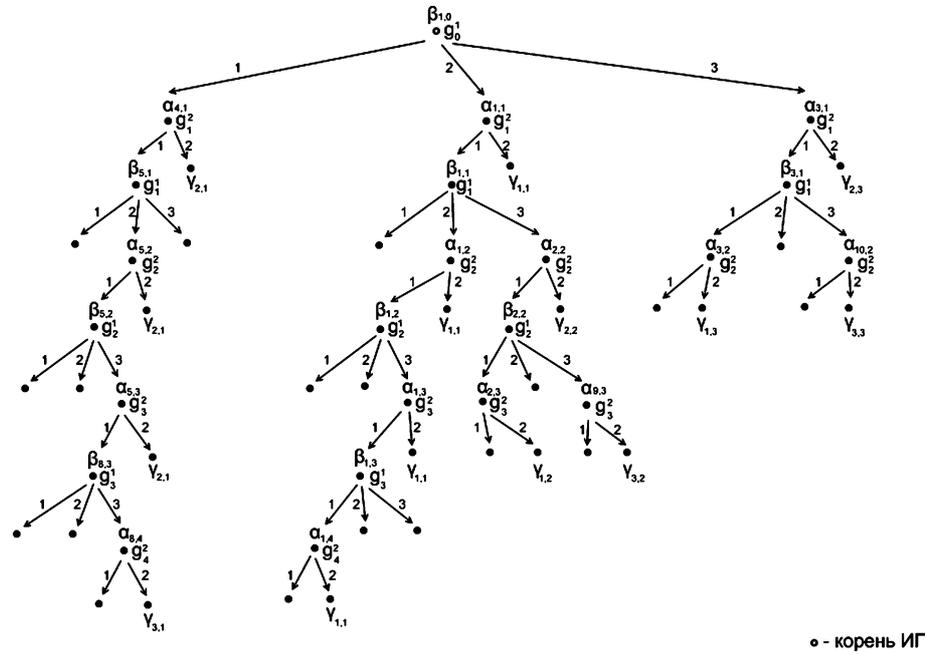


Рис. 3. вид графа, построенного на этапах 10–23 алгоритма U_2 по библиотеке {2231, 123, 1233}.

Пусть в библиотеке V есть слово, удовлетворяющее запросу w . Тогда в множестве V' найдётся слово v'_i такое, что $v'_i[1 \dots l(w)] = w$. В этом случае запрос пройдёт в лист, которому, как видно из условий а) и б) пункта 21 алгоритма U_2 , приписано слово, удовлетворяющее запросу. Из этого листа запрос может последовательно пройти и в другие листья. Как видно из условий а), б) и с) пункта 6 алгоритма U_1 , для каждого удовлетворяющего запросу слова запрос пройдёт ровно в одну вершину, которой это слово приписано. Ни в какие другие вершины, за исключением пустого листа, запрос не пройдёт. Таким образом, ИГ, построенный по алгоритму U_2 , решает ЗИП I .

III. Алгоритм U_3 построения информационного графа.

Пусть $V = \{v_1, v_2, \dots, v_p\}$.

1) Пусть $n' = \max_{1 \leq i \leq p} l(v_i)$. Выберем p листьев. Припишем каждому из этих листьев слово из библиотеки V так, чтобы каждое

слово из V было приписано к какому-нибудь листу. Для каждого $i, 1 \leq i \leq p$, лист, которому приписано слово v_i , назовём γ_i .

2) Выберем корень. Выпустим из него n' рёбер. Сопоставим им предикаты $f_1^1(w), f_2^1(w), \dots, f_{n'}^1(w)$. Вершину, в которую ведёт ребро с предикатом $f_i^1(w), i = 1, \dots, n'$, назовём β_i .

3) Положим $r = 1$. Счётчик r будет отражать номер слова из библиотеки, которое мы в данный момент рассматриваем.

4) Для каждого $j \leq l(v_r)$ выпустим из вершины β_j в вершину γ_r ребро. Сопоставим этому ребру предикат $f_{v_r[l(v_r)-j+1 \dots l(v_r)]}^2(w)$.

5) Если $r < p$ (значит, ещё не все слова из библиотеки V рассмотрены), увеличим r на 1 и вернёмся к пункту 3. Если $r = p$, то информационный граф, решающий ЗИП I , построен.

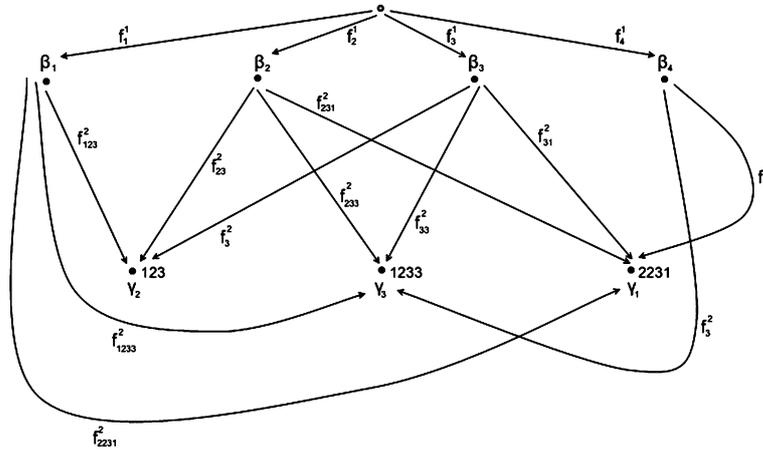


Рис. 4. вид графа, построенного с помощью алгоритма U_3 по библиотеке $\{2231, 123, 1233\}$.

Покажем, что информационный граф, построенный по алгоритму U_3 , решает ЗИП I . Заметим сначала, что если $l(w) > n' = \max_{1 \leq i \leq p} l(v_i)$, то запрос не пройдет ни по одному исходящему из корня ребру (так как результат вычисления каждого предиката $f_i^1(w), i = 1, \dots, n'$ окажется равен 0). Если же $l(w) \leq n'$, то запрос пройдет в вершины $\beta_{l(w)}, \dots, \beta_{n'}$. Из каждой вершины $\beta_i, l(w) \leq i \leq n'$ запрос попадет в вершину γ_r тогда и только тогда, когда $l(v_r) \geq i$ и $f_{v_r[l(v_r)-i+1 \dots l(v_r)]}^2(w) = 1$, то есть $v_r[l(v_r) - i + 1 \dots l(v_r) - i + l(w)] = w$.

Если обозначим $l(v_r) - i + 1$ как a , то получим, что слово v_r будет включено в ответ на запрос w тогда и только тогда, когда $\exists a, 1 \leq a \leq l(v_r) - l(w) + 1 : v_r[a \dots l(w) + a - 1] = w$, то есть когда v_r удовлетворяет запросу w . Итак, ИГ, построенный по алгоритму U_3 , решает ЗИП I .

4. Оценки сложности информационных графов, построенных по алгоритмам U_1, U_2, U_3

I. Оценка сложности ИГ, построенного по алгоритму U_1 .

Пусть ИГ u_1 построен с помощью алгоритма U_1 по библиотеке $V = \{v_1, \dots, v_p\}$. Если в библиотеке V нет слов, удовлетворяющих запросу w , то при обработке этого запроса произойдёт одно вычисление переключателя $g^3(w)$ и не более $\min(l(w), n)$ вычислений переключателей $g_h^1(w), 0 \leq h \leq \min(l(w), n) - 1$.

Если же в библиотеке V есть ровно $d > 0$ слов, удовлетворяющих запросу w , то при обработке этого запроса произойдёт одно вычисление переключателя $g^3(w)$, $l(w)$ вычислений переключателей $g_h^1(w), 0 \leq h \leq l(w)$ и d вычислений переключателя $g^3(w)$. Очевидно, запрос w будет обрабатываться тем дольше, чем больше числа $l(w)$ и d .

Таким образом, $T(u_1, x) \leq (d + 1)t(g^3, x) + l(w)t(g^1, x)$.

II. Оценка сложности ИГ, построенного по алгоритму U_2 .

Пусть ИГ u_2 построен с помощью алгоритма U_2 по библиотеке $V = \{v_1, \dots, v_p\}$. Если в библиотеке V нет слов, удовлетворяющих запросу w , то при обработке этого запроса произойдёт одно вычисление переключателя не более $\min(l(w), n)$ вычислений переключателей $g_h^1(w)$ и $g_{h+1}^2(w), 0 \leq h \leq \min(l(w), n) - 1$.

Если же в библиотеке V есть ровно $d > 0$ слов, удовлетворяющих запросу w , то при обработке этого запроса произойдёт $l(w)$ вычислений переключателей $g_h^1(w)$ и $g_{h+1}^2(w), 0 \leq h \leq l(w)$, а также d вычислений переключателя $g^3(w)$. Очевидно, запрос w будет обрабатываться тем дольше, чем больше числа $l(w)$ и d .

Таким образом, $T(u_2, x) \leq dt(g^3, x) + l(w)(t(g^1, x) + t(g^2, x))$.

III. Оценка сложности ИГ, построенного по алгоритму U_3 .

Пусть ИГ u_3 построен с помощью алгоритма U_3 по библиотеке $V = \{v_1, \dots, v_p\}$. Тогда при обработке запроса w будет вычислено $n' = \max_{1 \leq i \leq p} l(v_i)$ предикатов $f_a^1(w), 1 \leq a \leq n'$, сопоставленных исходящим из корня рёбрам. Кроме того, для каждого слова $v_i \in V$ такого, что $l(v_i) \geq l(w)$, будет вычислено $l(v_i) - l(w) + 1$ предикатов $f_{v_i[l(v_i)-j+1..l(v_i)]}^2(w), l(w) \leq j \leq l(v_i)$.

Таким образом, $T(u_3, x) \leq n't(f^1, x) + p(n' - l(w) + 1)t(f^2, x)$.

Напомним, что мы положили $t(f^1, x) = t(g^1, x) = t(g^2, x) = t(g^3, x) = 1, t(f^2, x) = l(x)$.

Мы получим:

$$\begin{aligned} T(u_1, x) &\leq 1 + l(w) + d, \\ T(u_2, x) &\leq 2l(w) + d, \\ T(u_3, x) &\leq n' + pl(w)(n' - l(w) + 1). \end{aligned}$$

Обозначим через d' максимальное по всем запросом число слов в библиотеке V , удовлетворяющих какому-либо запросу. Выберем такие значения $l(w), d, n'$, что сложность ИГ u_1, u_2, u_3 для них будет максимальной. Тогда

$$\begin{aligned} T(U_1, p) &\leq 1 + n + d', \\ T(U_2, p) &\leq 2n + d', \\ T(U_3, p) &\leq n + (n + 1)^2/4. \end{aligned}$$

5. Оценки объёмов информационных графов, построенных по алгоритмам U_1, U_2, U_3

I. Оценим $Q(U_2, p)$.

Пусть ИГ u_2 построен с помощью алгоритма U_2 по библиотеке $V = \{v_1, \dots, v_p\}$, и имеет наибольший объём среди всех ИГ, построенных с помощью алгоритма U_2 по библиотеке из p слов. Заметим, что в этом случае длина всех слов в V должна быть максимально возможной. Это означает, что если $p \leq k^n$, то $l(v_j) = n, j = 1, \dots, p$, а если $\sum_{i=s+1}^n k^i < p \leq \sum_{i=s}^n k^i, 1 \leq s \leq n - 1$, то в библиотеке V есть ровно k^i слов длины $i = s + 1, \dots, n$, а все остальные слова в библиотеке

имеют длину s . (Действительно, пусть это не так. Тогда рассмотрим библиотеку $V^* = \{v_1^*, \dots, v_p^*\}$ такую, что $v_j^* = v_j, j \neq i; l(v_i^*) > l(v_i), v_i^*[1 \dots l(v_i)] = v_i$. Рассмотрим граф, построенный на шагах 1–9 алгоритма U_2 по библиотеке V^* , и граф, построенный на шагах 1–9 того же алгоритма по библиотеке V . В первом графе из каждой вершины $\gamma_{i,j}, j = 1, \dots, l(v_i)$ исходит $l(v_i^*) - j + 1$ рёбер, а во втором — $l(v_i) - j + 1$ рёбер; из всех вершин $\gamma_{r,j}, j = 1, \dots, l(v_r), r \neq i$ в обоих графах исходит одно и то же количество рёбер. Значит, первый граф имеет больший объём, чем второй. Граф, построенный на шагах 10–23 алгоритма U_2 по библиотеке V^* , имеет, как нетрудно заметить, не меньший объём, чем граф, построенный на шагах 10–23 того же алгоритма по библиотеке V . Тогда граф, построенный по библиотеке V^* , имеет больший объём, чем граф, построенный по библиотеке V . Получаем противоречие).

Оценим сначала количество рёбер подграфа графа u_2 , построенного на шагах 1–9 алгоритма U_2 (обозначим это число как Q_1). Как легко заметить, этот подграф состоит из вершин $\gamma_{r,j}, r = 1, \dots, p, j = 1, \dots, n$ и выходящих из них рёбер, причём из каждой вершины $\gamma_{r,j}$ выходит $l(v_r) - j + 1$ рёбер. Пусть $\sum_{i=s+1}^n k^i < p \leq \sum_{i=s}^n k^i, 1 \leq s \leq n-1$.

Тогда

$$\begin{aligned} Q_1 &= \sum_{i=s+1}^n (k^i \sum_{j=1}^i (i-j+1)) + (p - \sum_{i=s+1}^n k^i) \sum_{j=1}^s (s-j+1) \leq \\ &\leq \sum_{i=s+1}^n (k^i \sum_{j=1}^n (n-j+1)) + (p - \sum_{i=s+1}^n k^i) \sum_{j=1}^n (n-j+1) = \\ &= p \sum_{j=1}^n (n-j+1) = pn(n+1)/2. \end{aligned}$$

Итак, $Q_1 \leq pn(n+1)/2$.

Если же $p \leq k^n$, $Q_1 = p \sum_{j=1}^n (j-n+1) = pn(n+1)/2$. Таким образом, при любом p $Q_1 \leq pn(n+1)/2$.

Оценим теперь количество рёбер подграфа графа u_2 , построенного на шагах 10–23 алгоритма U_2 (обозначим это число как Q_2).

Заметим, что если $p = k^n$, то строимое на шаге 10 множество $V' = \{v'_1, v'_2, \dots, v'_{p'}\}$, состоящее из всех слов $v_j[a \dots l(v_j)]$, где $j = 1, 2, \dots, p$; $a = 1, \dots, l(v_j)$, будет содержать абсолютно все слова из W_n^k . Поэтому при $p > k^n$ подграф, построенный на шагах 10–23, будет содержать то же количество рёбер, что и при $p = k^n$. Вычисляя Q_2 , будем считать, что $p \leq k^n$.

Подграф графа u_2 , построенный на шагах 10–23 алгоритма U_2 , состоит из различных вершин $\alpha_{i,j}$ и $\beta_{i,j-1}$ $1 \leq i \leq p'$, $1 \leq j \leq n$ и исходящих из них рёбер. Из каждой вершины $\alpha_{i,j}$ выходит 2 ребра, из вершины $\beta_{i,j} - k$ рёбер.

Обозначим число вершин $\alpha_{i,j}$ для фиксированного j как $N_\alpha(j)$, число вершин $\beta_{i,j}$ для фиксированного j как $N_\beta(j)$. Заметим, что $N_\beta(j) = N_\alpha(j)$, $1 \leq j \leq n-1$, и $N_\beta(0) = 1$. Число же вершин $\alpha_{i,j}$ для фиксированного $1 \leq j \leq n$ не может превышать, с одной стороны, числа слов из библиотеки V' , имеющих длину j , а с другой — увеличенного в k раз числа вершин $\beta_{i,j-1}$ (так как в каждую вершину $\alpha_{i,j}$ ведёт ребро из $\beta_{i',j-1}$). Так как не более $p(n-j+1)$ слов из V' имеет длину j , мы имеем $N_\alpha(j) \leq \min(p(n-j+1), kN_\beta(j-1))$. Пусть $r \in \mathbb{Z}$, $0 \leq r \leq n-1$ — такое число, что $k^r/(n-r+1) < p \leq k^{r+1}/(n-r)$. Получаем, что $N_\alpha(j) = N_\beta(j) \leq k^j$, $1 \leq j \leq r$; $N_\alpha(j) = N_\beta(j) \leq p(n-j+1)$, $r < j < n$; $N_\alpha(n) \leq p$. Тогда

$$\begin{aligned} Q_2 &\leq k + (k+2) \left(\sum_{j=1}^r k^j + p \sum_{j=r+1}^{n-1} (n-j+1) \right) + 2p = \\ &= k + (k+2)k \frac{k^r + k}{k-1} + \frac{p(k+2)(n-r+2)(n-r-1)}{2} + 2p. \end{aligned}$$

Итак, если $p \leq k^n$,

$$\begin{aligned} Q(U_2, p) = Q_1 + Q_2 &\leq k + (k+2)k \frac{k^r + k}{k-1} + \\ &+ \frac{1}{2}p((k+2)(n-r+2)(n-r-1) + n(n+1) + 4). \end{aligned}$$

Если же $p > k^n$,

$$Q(U_2, p) \leq k + (k+2)k \frac{k^{n-1} + k}{k-1} + 2k^n + \frac{pn(n+1)}{2}.$$

II. Оценим $Q(U_1, p)$.

Пусть ИГ u_1 построен с помощью алгоритма U_1 по библиотеке $V = \{v_1, \dots, v_p\}$, и имеет наибольший объём среди всех ИГ, построенных с помощью алгоритма U_1 по библиотеке из p слов. Как и для ИГ, построенного с помощью алгоритма U_2 , в этом случае длина всех слов в V должна быть максимально возможной. Доказывается это аналогично тому, как доказывалось для графа, построенного с помощью алгоритма U_2 .

Заметим, что граф, построенный на шагах 1–9 алгоритма U_1 , совпадает с графом, построенным на шагах 1–9 алгоритма U_2 по той же библиотеке. Поэтому количество рёбер подграфа графа u_1 , построенного на шагах 1–9 алгоритма U_1 (обозначим это число как Q_1) не превышает $pn(n+1)/2$.

Оценим теперь количество рёбер подграфа графа u_1 , построенного на шагах 10–18 алгоритма U_1 (обозначим это число как Q_2). Заметим, что если $p = k^n$, то строимое на шаге 10 множество $V' = \{v'_1, v'_2, \dots, v'_{p'}\}$, состоящее из всех подслов всех слов из V , будет содержать абсолютно все слова из W_n^k . Поэтому при $p > k^n$ подграф, построенный на шагах 10–18, будет содержать то же количество рёбер, что и при $p = k^n$. Вычисляя Q_2 , будем считать, что $p \leq k^n$.

Заметим, что подграф графа u_1 , построенный на шагах 10–18 алгоритма U_1 , состоит, помимо корня и выходящих из него рёбер, из вершин $\beta_{i,j-1}$, $1 \leq i \leq p'$, $1 \leq j \leq n$, и выходящих из них рёбер. Из каждой вершины $\beta_{i,j}$ выходит k рёбер. Число вершин $\beta_{i,j}$ для фиксированных j и $s = l(v'_i)$ обозначим $N(j, s)$ (при этом, как легко видеть, $0 \leq j \leq s-1$).

Число вершин $\beta_{i,j}$ для фиксированных $j > 0$ и $s = l(v'_i)$ не может превышать, с одной стороны, числа слов из библиотеки V' , имеющих длину s , а с другой — увеличенного в k раз числа вершин $\beta_{i,j-1}$ таких, что $s = l(v'_i)$ (ведь в каждую вершину $\beta_{i,j}$ ведёт ребро из вершины $\beta_{i,j-1}$ такой, что $l(v'_i) = l(v'_t)$). Так как не более $p(n-s+1)$ слов из V' имеет длину s , мы имеем $N(j, s) \leq \min_{p(n-s+1), kN(j-1, s)}$. Кроме того, $N(0, s) = 1$.

Пусть $p \leq k^{n-1}$ и $r \in \mathbb{N}$, $1 \leq r < n$ — такое число, что $k^{r-1}/(n-r+1) < p \leq k^r/(n-r)$. Тогда $N(j, s) \leq k^{j-1}$, $1 \leq j \leq r$, и $N(j, s) \leq p(n-s+1)$, $j > r$. Следовательно,

$$\begin{aligned}
 Q_2 &\leq n + k \left(\sum_{s=1}^r \sum_{j=0}^{s-1} k^j + \sum_{s=r+1}^n \sum_{j=0}^{r-1} k^j + p \sum_{s=r+1}^n (n-s+1)(s-r) \right) = \\
 &= k \frac{(n-r+1)k^{r+1} - (n-r)k^r - k}{(k-1)^2} - \frac{n}{(k-1)} + \\
 &\quad + \frac{kp(n-r)(n-r+1)(n-r+2)}{6},
 \end{aligned}$$

откуда

$$\begin{aligned}
 Q(U_2, p) = Q_1 + Q_2 &\leq k \frac{(n-r+1)k^{r+1} - (n-r)k^r - k}{(k-1)^2} + \\
 &\quad + \frac{kp(n-r-2)(n-r)(n-r+1)}{6} + \frac{pn(n+1)}{2}.
 \end{aligned}$$

Если же $p > k^{n-1}$, то

$$Q(U_2, p) \leq \frac{pn(n+1)}{2} + k \sum_{s=1}^n \sum_{j=0}^{s-1} k^j = \frac{pn(n+1)}{2} + k \frac{k^{n+1} - k}{(k-1)^2} - \frac{n}{k-1}.$$

III. Оценим $Q(U_3, p)$.

Пусть ИГ u_3 построен с помощью алгоритма U_3 по библиотеке $V = \{v_1, \dots, v_p\}$, и имеет наибольший объём среди всех ИГ, построенных с помощью алгоритма U_3 по библиотеке из p слов. Обозначим $\max_{1 \leq i \leq p} l(v_i)$ через n' . Легко видеть, что рёбра ИГ u_3 — это либо рёбра, входящие в вершины $\beta_i, 1 \leq i \leq n'$ (всего таких рёбер n'), либо рёбра, входящие в вершины $\gamma_r, 1 \leq r \leq p$. В каждую вершину γ_r входит ровно $l(v_r)$ рёбер. Таким образом, $Q(u_3) = n' + \sum_{r=1}^p l(v_r)$. Так как u_3 имеет наибольший объём среди всех ИГ, построенных с помощью алгоритма U_3 по библиотеке из p слов, то длина всех слов в V должна быть максимально возможной. Это означает, что если $p \leq k^n$, то $l(v_j) = n, j = 1, \dots, p$, а если $\sum_{i=s+1}^n k^i < p \leq \sum_{i=s}^n k^i, 1 \leq s \leq n-1$, то в библиотеке V есть ровно k^i слов длины $i = s+1, \dots, n$, а все остальные слова в библиотеке имеют длину s . Если $p \leq k^n, Q(u_3) = n + np = n(p+1)$. Если же $\sum_{i=s+1}^n k^i < p \leq \sum_{i=s}^n k^i, 1 \leq s \leq n-1$, то

$$\begin{aligned}
Q(u_3) &= n + \sum_{i=s+1}^n ik^i + (p - \sum_{i=s+1}^n k^i)s \leq \\
&\leq \sum_{i=s+1}^n nk^i + (p - \sum_{i=s+1}^n k^i)n = n + np = n(p+1).
\end{aligned}$$

Таким образом, $Q(U_3, p) \leq n(p+1)$.

6. Доказательство теоремы

Рассмотрим произвольный запрос $x = w$. Пусть этому запросу удовлетворяют $d > 0$ слов из библиотеки V . Для того, чтобы в ответ на запрос w включилась какая-нибудь запись $v \in V$, должно произойти не менее $l(w)$ проверок, удостоверяющих, что $v[m+i-1] = w[i]$, $i = 1, \dots, l(w)$. Для того, чтобы запрос прошёл во все остальные $d-1$ листьев, каждому из которых приписано слово, удовлетворяющее запросу, необходимо ещё не менее $d-1$ переходов по рёбрам (в лучшем случае эти переходы будут осуществляться после вычисления предиката 1). Итак, $n + d - 1 \leq T(I, \mathcal{F}, q)$.

Будем учитывать, что так как p не превышает количества всех слов длины n над алфавитом $\{1, \dots, k\}$, равного $k(k^n - 1)/(k - 1)$, и по условию теоремы $p > n^{k-1}$, то при $p \rightarrow \infty$ $n = O(\log p)$. Выпишем теперь полученные в параграфах 4 и 5 верхние оценки сложности и объёма информационных графов, построенных по алгоритмам U_1, U_2, U_3 : $Q(U_1, p) \lesssim pn^3k/6$, $T(U_1, p) \leq 1 + n + d$; $Q(U_2, p) \lesssim pn^2k/2$, $T(U_2, p) \leq 2n + d$; $Q(U_3, p) \leq n(p+1)$, $T(U_3, p) \leq n + p(n+1)^2/4$. Из этих соотношений непосредственно вытекает доказательство теоремы.

Список литературы

- [1] Сокирко А. Семантические словари в автоматической обработке текста (по материалам системы ДИАЛИНГ). <http://www.aot.ru/docs/sokirko>
- [2] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. М.: Физматлит, 2002.
- [3] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С. Введение в теорию интеллектуальных систем. М.: Изд-во ф-та ВМиК МГУ, 2006.