

Порядок сложности задачи поиска в множестве слов вхождений подслова

Е. М. Перпер

Рассматриваемая в работе задача состоит в следующем: пусть дано множество слов; требуется для произвольного подслова найти все вхождения этого подслова в слова из этого множества. В данной работе приведена нижняя оценка времени работы алгоритмов, позволяющих осуществлять такой поиск, а также получен порядок объёма памяти для алгоритмов, осуществляющих поиск за минимальное по порядку время.

Ключевые слова: поиск вхождения подслов, нижняя оценка, верхняя оценка.

1. Введение

Необходимость поиска вхождений какого-либо подслова в слова из большого массива слов часто возникает на практике. Например, при пользовании словарём, когда нужно найти все вхождения какого-либо корня в содержащие его слова. Алгоритм, состоящий из перебора всех слов из словаря и проверки каждого из них с целью обнаружения нужного подслова, очевидно, займёт большое время.

Существует много алгоритмов, решающих похожую задачу — задачу поиска вхождений произвольного подслова («образца») в одно заранее выбранное слово («текст»). Среди самых быстрых алгоритмов (тех, в которых поиск осуществляется за время, пропорциональное сумме длины образца и количества вхождений образца в текст) — алгоритмы, использующие суффиксные деревья или суффиксные массивы со словарями рангов [1]. Каждый из них требует $O(n \log n)$ памяти в модели RAM (МПД, «машина произвольного доступа»), где n — длина текста. Идея суффиксного дерева была предложена П. Вейнером в 1973 году [2], затем это дерево неоднократно

улучшалось, главным образом, с целью уменьшения требуемой памяти (см., например, [1, 3]).

На основе суффиксного дерева может быть построено обобщённое суффиксное дерево (см., например, [4]), позволяющее искать образец в массиве слов за время, пропорциональное сумме длины образца и количества вхождений образца в массив слов. Это дерево требует $O(pn \log(pn))$ памяти, если количество слов в массиве равно p , а длина каждого слова в массиве равна n . В данной работе предложен (в рамках информационно-графовой модели данных, см. [5, 6]) алгоритм, фактически представляющий собой модификацию обобщённого суффиксного дерева. Он требует $O(pn)$ памяти. Это связано с тем, что в модели RAM длина машинного слова полагается равной логарифму числа символов в массиве слов, то есть $\log(pn)$, тогда как в информационно-графовой модели она неявно полагается равной 1. Показано, что время работы предложенного алгоритма превышает нижнюю оценку не более чем в три раза, а объём памяти, нужной для этого алгоритма, превышает нижнюю оценку не более чем в $2k + 11$ раз, где k — число букв в алфавите (k считается константой).

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

2. Основные понятия и формулировка результатов

Для произвольного натурального числа k обозначим $N_k = \{1, 2, \dots, k\}$. Пусть $W_{k,n} = \cup_{s=1}^n N_k^s \{k+1\}^{n-s}$ — множество слов длины не более n над алфавитом N_k . Слово длины s представлено набором длины n , первые s элементов которого — из множества N_k , и если $s < n$, то каждый из оставшихся $n - s$ элементов равен $k + 1$.

Для каждого слова $w \in W_{k,n}$ его длину будем обозначать через $l(w)$; i -й буквой слова $w \in W_{k,n}$, где $i \in \{1, \dots, n\}$, назовём i -й элемент соответствующего слову w набора. В частности, если $i > l(w)$, то i -я буква слова w равна $k + 1$. Будем обозначать i -ю букву слова w через $w[i]$. Через $w[a..b]$, где $a, b \in \mathbb{N}$, $1 \leq a \leq b \leq l(w)$, будем обозначать такое слово $v \in W_{k,n}$, что $l(v) = b - a + 1$ и $v[i] = w[a + i - 1]$ для всех натуральных i , не превышающих $l(v)$. Будем называть слово $w[a..b]$ подсловом слова w , начинающимся с его a -й буквы и заканчивающимся его b -й буквой. Будем говорить, что слова $w \in W_{k,n}$ и

$v \in W_{k,n}$ равны или совпадают (и писать $w = v$), если $l(v) = l(w)$ и $v[1] = w[1], v[2] = w[2], \dots, v[l(v)] = w[l(w)]$ (очевидно, это условие эквивалентно условию $v[1] = w[1], v[2] = w[2], \dots, v[n] = w[n]$). Для слова w его подслово $w[1..q], q \in \mathbb{N}, q \leq l(w)$, будем называть префиксом (длины q) слова w , а подслово $w[l(w) - q + 1..l(w)], q \in \mathbb{N}, q \leq l(w)$ — суффиксом (длины q) слова w . Если слово v — префикс слова w и $v \neq w$, то будем говорить, что v — собственный префикс слова w . Аналогично, если v — суффикс слова w и $v \neq w$, то будем говорить, что v — собственный суффикс слова w . Скажем, что слово w лексикографически следует за словом v , а слово v лексикографически предшествует слову w , если существует такое натуральное число i , не превышающее n , что $v[i] < w[i]$ и для всякого натурального числа j , меньшего i , j -я буква слова v равна j -й букве слова w . Будем записывать это следующим образом: $v < w$ (либо $w > v$). Конкатенацию слов v_1 и v_2 , то есть слово w такое, что $w[1..l(v_1)] = v_1, w[l(v_1) + 1..l(v_1) + l(v_2)] = v_2$, будем обозначать как $v_1 v_2$.

Пару (w, i) , где $w \in W_{k,n}$ и $i \in N_{l(w)}$, назовём вхождением в слово w . Множество всех вхождений в слова из некоторого множества M будем обозначать как $Oc(M)$.

Рассмотрим $X = W_{k,n}$ — множество запросов, то есть слов длины не более n над алфавитом N_k . Рассмотрим также $Y = N_k^n$ — множество записей (каждая запись является словом длины n над алфавитом N_k).

Введём бинарное отношение ρ , которое позволит устанавливать, когда вхождение $(v, i) \in Oc(Y)$ удовлетворяет запросу $x \in X$ (отношение поиска):

$$x \rho (v, i) \Leftrightarrow x = v[i..l(x) + i - 1].$$

Будем рассматривать задачу информационного поиска (ЗИП) $I = \langle X, V, \rho \rangle$, где $V = \{v_1, v_2, \dots, v_p\}, V \subseteq Y$. 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 вхождений $\omega \in Oc(V)$, для которых выполнено $x \rho \omega$. Назовём эту задачу *задачей поиска вхождений подслова*. Множество V в дальнейшем будем называть *библиотекой*. Так как X и ρ фиксированы, задача поиска вхождений подслова полностью определяется библиотекой. Похожая задача рассматривалась в [7, 8], но она состояла в перечислении не вхождений, а слов, содержащих эти вхождения.

Введём понятие информационного графа (ИГ) в соответствии с [5, 6], с той разницей, что там, где в [5, 6] рассматривалось множество Y , мы будем рассматривать множество $Oc(Y)$. В формальном определении понятия ИГ используются два описанных выше множества X и $Oc(Y)$, а также множество F предикатов, заданных на множестве X (предикаты — это функции, которые могут принимать только два значения: 0 или 1), и 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 некоторое вхождение из множества $Oc(Y)$. 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 графа, то через ψ_β обозначим полустепень исхода вершины β (то есть число исходящих из этой вершины рёбер). Каждой точке переключения β сопоставим некий элемент множества 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 элемент множества F . Это соответствие назовем нагрузкой предикатных ребер. Полученный нагруженный граф 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 вхождений, 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 . Тем самым понятие ИГ полностью определено.

Назовём каждую цепь, по которой запрос x проходит из корня в какой-либо лист, существенной цепью ИГ, соответствующей запросу x .

Скажем, что ИГ U решает ЗИП $I = \{X, V, \rho\}$, если для любого запроса $x \in X$ ответ ИГ U на этот запрос содержит все те и только те вхождения из $Oc(V)$, которые удовлетворяют запросу x , то есть $J_U(x) = \{\omega \in Oc(V) : x\rho\omega\} \stackrel{def}{=} J_I(x)$.

Обозначим $J_V = \{J_I(x) : x \in X\}$ — множество ответов, соответствующих библиотеке V . Назовём ответом любое множество, являющееся элементом J_V . То есть, ответ — это множество всех вхождений, удовлетворяющих одному и тому же запросу. Если задача I зафиксирована, ответ на запрос x будем обозначать как $J(x)$.

Определим понятие сложности информационного графа на запросе. Сделаем это в более общем виде, чем в [7, 8]: будем считать, что время вычисления функции f от запроса x зависит не только от f , но и от x . Обозначим через $Pr(U, x)$ множество всех предикатных вершин ИГ U , в которые проходит запрос x , а через $Sw(U, x)$ — множество всех переключательных вершин ИГ U , в которые проходит запрос x . Обозначим через $E(\beta)$ множество всех рёбер, выходящих из вершины β . Тогда определим сложность вычисления ИГ U на запросе x следующим образом:

$$T(U, x) = \sum_{\beta \in Sw(U, x)} t(g_\beta, x) + \sum_{\beta \in Pr(U, x)} \sum_{e \in E(\beta)} t(f_e, x),$$

где g_β — переключатель, сопоставленный вершине β , f_e — предикат, сопоставленный ребру e , $t(f, x)$ — время вычисления функции f от запроса x . $T(U, x)$ характеризует время обработки запроса x .

Объемом $Q(U)$ ИГ U назовем число рёбер в ИГ U . $Q(U)$ соответствует объёму памяти, используемой информационным графом.

В качестве базового множества \mathcal{F} будем рассматривать $\langle F, G \rangle$, $F = \{f(x)\}$, $G = G_1 \cup G_2$, где $G_1 = \{g_i(x), i \in N_n\}$, $G_2 = \{h_{v,i}(x), v \in W_{k,n} \setminus N_k^n, i \in N_{l(v)}\}$. Эти функции определяются следующим образом:

$$\begin{aligned}
 f(x) &\equiv 1; \\
 g_i(x) &= x[i]; \\
 h_{v,i}(x) &= \begin{cases} 1, \text{ если } i > l(x) \text{ или } x[i..l(x)] \text{ — собственный префикс } v, \\ 2, \text{ если } i \leq l(x) \text{ и } v \text{ — префикс } x[i..l(x)], \\ 3, \text{ во всех остальных случаях.} \end{cases} \quad (1)
 \end{aligned}$$

С помощью предиката, тождественно равного 1, моделируется перечисление ответа на запрос. Вычисление этого предиката соответствует переходу по ссылке (например, к вхождению из ответа), в то время как вычисление любого переключателя $g_i(x)$ включает в себя выделение i -й буквы слова x и переход по ссылке. По этой причине сложность вычисления предиката, тождественно равного 1, считается положительной, но меньшей, чем сложность вычисления переключателя $g_i(x)$. Будем считать, что сложность вычисления любого переключателя $g_i(x)$ равна 1, а сложность вычисления предиката $f(x)$ равна t , где $0 < t < 1$. Алгоритм вычисления переключателя $h_{v,i}(x)$ более сложен. Функция на языке C , реализующая этот алгоритм, может выглядеть, например, так:

```

int h(v,i,x)
{
    int j=0;
    do
    {
        if(x[j+i]!=v[j+1])
        {
            if(x[j+i]==0) return 1;
            return 3;
        }
        j++;
    }
    while(v[j+1]!=0);
    return 2;
}

```

Будем считать, что сложность каждого шага цикла равна 2 (так как производится ровно 2 сравнения), а сложность перехода по любой из ссылок 1, 2 и 3 равна 1. Тогда

$$t(h_{v,i}, x) = \begin{cases} 1 + 2 * \max(l(x) - i + 2, 1), & \text{если } h_{v,i}(x) = 1 \\ 1 + 2 * l(v), & \text{если } h_{v,i}(x) = 2, \\ 3 + 2 * \max_{j: x[i..i+j-1]=v[1..j]} j, & \text{если } h_{v,i}(x) = 3. \end{cases} \quad (2)$$

Введём следующие обозначения: $\mathcal{U}(\mathcal{I}, \mathcal{F})$ — множество всех ИГ над базовым множеством \mathcal{F} , решающих ЗИП $I = \langle X, V, \rho \rangle$; $e(x) = \min(l(x) + 1, n)$; $R(I, \mathcal{F}, x) = e(x) + t \cdot (|J_I(x)| - 1)$; $Q(p, n) = \max_{I: I = \langle X, V, \rho \rangle, |V|=p} \min_{U \in \mathcal{U}(\mathcal{I}, \mathcal{F})} Q(U)$; $T(I, x) = \min_{U \in \mathcal{U}(\mathcal{I}, \mathcal{F})} T(U, x)$.

Теорема 1. При $k \geq 2$ для каждой задачи $I = \langle X, V, \rho \rangle$, для любых $U \in \mathcal{U}(\mathcal{I}, \mathcal{F})$ и $x \in X$ выполняется следующее условие: если $|J_I(x)| \geq 1$, то $T(U, x) \geq R(I, \mathcal{F}, x)$.

Обозначим через \mathcal{U}_{fast} множество таких графов $U \in \mathcal{U}(\mathcal{I}, \mathcal{F})$, что для каждого из них при любых x из X выполняется условие $T(U, x) \leq 3R(I, \mathcal{F}, x)$.

Пусть $Q_{fast}(p, n) = \max_{I: I = \langle X, V, \rho \rangle, |V|=p} \min_{U \in \mathcal{U}_{fast}} Q(U)$.

Теорема 2. Для любых натуральных n и p выполнено $pn \leq Q(p, n) \leq Q_{fast}(p, n) \leq (2k + 1)pn$.

3. Доказательство теоремы 1

Доказательство теоремы, аналогичной теореме 1, приведено в [8]. Однако, есть и существенные отличия, связанные с тем, что в [8] осуществлялся поиск слов, а не вхождений в них; кроме того, в базовом множестве отсутствовало семейство функций $h_{v,i}(x)$. Поэтому приведём здесь доказательство теоремы 1.

Доказательство. Рассмотрим произвольную библиотеку $V \subseteq N_k^n$. Рассмотрим произвольный ИГ $U \in \mathcal{U}(\mathcal{I}, \mathcal{F})$, где $I = \langle X, V, \rho \rangle$. Пусть C — какая-либо цепь, по которой запрос проходит из корня в лист с вхождением (w, j) , удовлетворяющим запросу (обозначим этот лист γ), причём ни одна вершина этой цепи, кроме γ , не является листом. Обозначим через C' цепь, полученную из C удалением вершины γ и входящего в неё ребра. Рассмотрим произвольное $i \in \mathbb{N}$,

$1 \leq i \leq e(x)$. Пусть всем переключательным вершинам цепи C' сопоставлены функции, не зависящие от $x[i]$ существенно. Заменим $x[i]$ на 1, если $j+i-1 \leq n$ и $w[j+i-1] \neq 1$, и на 2 в противном случае. Обозначим получившийся запрос как x' . Этот запрос также будет проходить в лист с записью (w, j) . Мы получим, что либо $j + l(x') - 1 > n$, либо $w[j+i-1] \neq x'[i]$, то есть в любом случае вхождение (w, j) не удовлетворяет запросу x' . Имеем противоречие с тем, что граф U решает задачу поиска вхождений под слова. Получим, что для любого $i \in \mathbb{N}$, $1 \leq i \leq e(x)$ найдётся вершина цепи, не совпадающая с γ и такая, что ей сопоставлен переключатель, существенно зависящий от $x[i]$. Это либо переключатель $g_i(x)$, либо переключатель $h_{v,r}(x)$, для которого $l(v) \geq i - r$ и $x[r..i - 1] = v[1..i - r]$. В последнем случае в процессе вычисления $h_{v,r}(x)$ $x[i]$ сравнивается с $v[i - r + 1]$, после чего происходит ещё не менее одного сравнения. Таким образом, минимальный вклад, вносимый обработкой буквы $x[i]$ в сложность вычисления ИГ U на запросе x , равен 1. Значит, суммарный вклад в $T(U, x)$ переключателей, сопоставленных вершинам цепи C' — не меньше $e(x)$.

Пусть $|J_I(x)| > 1$. Для каждого отличного от γ листа, в который проходит запрос x , выберем ребро, по которому x входит в этот лист. Так как никакому листу не может быть сопоставлено более одного вхождения, все эти рёбра разные, причём если некоторые из этих рёбер исходят из одной и той же вершины, то это предикатные рёбра. Кроме того, ни одно из этих рёбер не принадлежит C . Значит, для прохода запроса по таким рёбрам будет вычислено не менее $|J_I(x)| - 1$ функций, отличных от переключателей, сопоставленных вершинам цепи C' . Так как $t < 1$, получим, что $T(U, x) \geq e(x) + t \cdot (|J_I(x)| - 1) = R(I, \mathcal{F}, x)$.

4. Доказательство теоремы 2

Введём ещё одно определение: будем говорить, что вершина α графа схемно достижима из вершины β , если существует ориентированная цепь из β в α .

Прежде чем перейти к непосредственному доказательству теоремы 2, докажем три вспомогательные леммы.

Лемма 1. Пусть $I = \langle X, V, \rho \rangle$ — задача поиска вхождения под слова, причём V состоит из r слов. Тогда число различных ответов, соответствующих V , не превосходит $2rp$, и найдётся ИГ U'

без листьев (то есть вершин, которым сопоставлены вхождения) над базовым множеством \mathcal{F} , обладающий следующими свойствами:

- 1) $Q(U') < 2(k + 4)pn$;
- 2) Для каждого запроса $x \in X$ выполнено $T(U', x) \leq 3e(x)$;
- 3) для каждого ответа J , соответствующего V , найдётся вершина $\kappa(J)$, из которой не выходит ни одно ребро и в которую попадают все те и только те запросы x , ответ на которые совпадает с J .

Доказательство. Рассмотрим все вхождения из $Os(V)$. Для каждого вхождения (v_i, j) рассмотрим $x = v_i[j..n]$ (напомним, что каждое слово длины s представляется набором длины n , первые s элементов которого — буквы этого слова, а каждый из оставшихся $n - s$ элементов равен $k + 1$). Выберем $e(x) + 1$ вершин, пронумеруем их от 1 до $e(x) + 1$, а затем для каждого $r \in \{1, \dots, e(x)\}$ проведём из вершины под номером r в вершину под номером $r + 1$ ребро, присвоив этому ребру номер $x[r]$. Вершину с номером $e(x) + 1$ дополнительно назовём $\lambda(x)$.

Теперь отождествим все вершины, имеющие номер 1. Получившуюся вершину объявим корнем. Далее последовательно для каждого натурального r от 1 до $n - 1$ сделаем следующее. Рассмотрим каждую вершину, имеющую номер r . Отождествим все рёбра с одинаковым номером, выходящие из этой вершины, и вершины, в которые входят эти рёбра (по построению, все отождествлённые вершины будут иметь номер $r + 1$). Заметим, что по построению из вершины с номером n не могут выходить два или более ребра с одним и тем же номером, так как это означало бы наличие в библиотеке одинаковых слов. Пример полученного графа см. на рис. 1.

Полученный граф является ориентированным деревом, так как в корень не входит ни одного ребра, а в каждую вершину, кроме корня, входит одно ребро. Каждой вершине α , в которую ведёт ребро с номером из N_k , можно сопоставить слово $x(\alpha)$, являющееся конкатенацией номеров рёбер, по которым можно пройти из корня в эту вершину. По построению графа, никаким двум различным вершинам не будет сопоставлено одно и то же слово (в противном случае эти вершины ранее должны были быть отождествлены). Кроме того, по построению графа, каждое слово $x(\alpha)$ является префиксом какого-либо слова $v_i[j..n]$, то есть подсловом какого-либо слова из V , и обратно, каждое подслово какого-либо слова из V является $x(\alpha)$ для какой-либо вершины α .

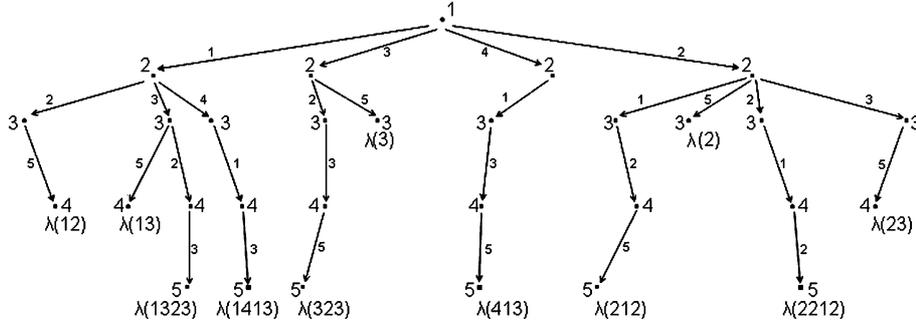


Рис. 1. Граф, построенный в лемме 1 на этапе отождествления вершин для $k = 4, n = 4, V = \{1323, 1413, 2212\}$.

Сопоставим каждой вершине с номером i , где $i \in N_n$, переключатель $g_i(x)$. Получим, что в каждую вершину α , в которую ведёт ребро с номером из N_k , пройдут все те и только те запросы, префикс которых совпадает с $x(\alpha)$; в вершину, в которую из некоторой вершины α ведёт ребро с номером $k + 1$, пройдёт только запрос $x(\alpha)$. Заметим, что концевыми вершинами дерева являются все те и только те вершины, которые обозначены $\lambda(v_i[j..n])$ для некоторых i и j , причём в каждую концевую вершину $\lambda(x)$ проходит только запрос x . Для каждой концевой вершины $\lambda(x)$ обозначим через $M(x)$ множество всех таких вхождений (v_i, j) , что $x = v_i[j..n]$. Очевидно, множества $M(x)$, соответствующие разным концевым вершинам, не пересекаются. При этом, из каждой вершины α схемно достижимы все те и только те вершины $\lambda(x)$, для которых $x(\alpha)$ является префиксом слова x . Так как ответ $J(x(\alpha))$ состоит из всех тех и только тех вхождений (v_i, j) , для которых $x(\alpha)$ является префиксом слова $v_i[j..n]$, то этот ответ представляет собой объединение множеств $M(x)$, соответствующих всем концевым вершинам, схемно достижимым из α .

Оценим число различных непустых ответов, соответствующих библиотеке. Каждому такому ответу J можно сопоставить вершину α , для которой $J(x(\alpha)) = J$, и либо α — концевая, либо из неё выходит не менее двух рёбер. Действительно, выберем среди вершин β , для которых $J(x(\beta)) = J$, ту, номер которой максимален, и обозначим её через α . Предположим, что из неё выходит ровно одно ребро. Пусть оно входит в некоторую вершину γ . Тогда $J(x(\gamma)) = J$, так как множества концевых вершин, схемно достижимых из α и γ , сов-

падают. При этом номер вершины γ больше номера вершины α , что противоречит выбору вершины α . Следовательно, число различных непустых ответов не превышает суммы числа концевых вершин дерева и числа его внутренних вершин, имеющих полустепень исхода не менее двух. Эта сумма, в свою очередь, не превышает $2pn - 1$, так как число концевых вершин дерева по построению не превышает pn . С учётом пустого ответа, библиотеке соответствует не более $2pn$ ответов.

Добавим в граф ещё одну вершину, назовём её $\varkappa(\emptyset)$.

Для каждой пары (i_1, i_2) , $i_1, i_2 \in N_{n+1}$, $2 \leq i_1 < i_2$, рассмотрим каждую цепочку вершин, имеющих номера i_1, \dots, i_2 , обладающую следующими свойствами: а) из каждой вершины этой цепочки, за исключением вершины с номером i_2 , выходит только одно ребро, номер этого ребра не равен $k + 1$, и это ребро входит в вершину, которая также принадлежит этой цепочке; б) добавить к этой цепочке какие-либо вершины так, чтобы свойство а) сохранилось, невозможно. Обозначим каждую принадлежащую цепочке вершину через α_r , где r — номер этой вершины. Заметим, что $J(x(\alpha_{i_1})) = J(x(\alpha_{i_1+1})) = \dots = J(x(\alpha_{i_2}))$, так как из всех α_r , $r \in \{i_1, i_1 + 1, \dots, i_2\}$ схемно достижимы одни и те же концевые вершины.

Обозначим $J(x(\alpha_{i_1}))$ как J . Рассмотрим произвольный запрос x , прошедший в вершину α_{i_1} , то есть $x[1..i_1 - 1] = x(\alpha_{i_1})$. Возможно несколько случаев:

- 1) x проходит в вершину α_{i_2} , то есть слово $x(\alpha_{i_2})$ является префиксом слова x ;
- 2) x является собственным префиксом слова $x(\alpha_{i_2})$, тогда, так как $x = x(\alpha_i)$ для некоторого $i \in \{i_1, \dots, i_2 - 1\}$, ответом на запрос x является J ;
- 3) x не проходит в вершину α_{i_2} и не совпадает с $x(\alpha_i)$ ни для какого $i \in \{i_1, \dots, i_2 - 1\}$. Тогда по построению графа получаем, что $J(x) = \emptyset$.

Удалим теперь вершины $\alpha_{i_1+1}, \dots, \alpha_{i_2-1}$ и выходящие из них рёбра, а ребро, выходящее из α_{i_1} , направим в α_{i_2} , и присвоим ему номер 2. Если вершина $\lambda(x(\alpha_{i_2}))$ существует (это либо α_{i_2} , либо в неё из α_{i_2} входит ребро номер $k + 1$), дополнительно назовём её $\varkappa(J)$, иначе выберем новую вершину, назовём её $\varkappa(J)$ и проведём в неё из α_{i_2} ребро с номером $k + 1$. Проведём в $\varkappa(J)$ из α_{i_1} ребро с номером 1. Кроме того, проведём из α_{i_1} в $\varkappa(\emptyset)$ ребро номер 3. Сопоставим вершине α_{i_1}

переключатель h_{v,i_1} , где $v = x(\alpha_{i_2})[i_1..i_2 - 1]$. Получим, что в случае 1) x по-прежнему проходит в вершину α_{i_2} , а если x совпадает с $x(\alpha_{i_2})$ — то и в $\varkappa(J)$; в случае 2) x проходит в вершину $\varkappa(J)$, а в случае 3) — в вершину $\varkappa(\emptyset)$.

Пусть все цепочки рассмотрены. Все вершины $\lambda(x)$, не обозначенные дополнительно как $\varkappa(J)$ для некоторого ответа J , обозначим дополнительно как $\varkappa(J(x))$. Из каждой вершины α , которой сопоставлен переключатель $g_i(x), i \in \{2, \dots, n\}$, и из которой не выходит ребро под номером $k+1$, выпустим это ребро, и направим его в новую вершину, которую назовём $\varkappa(J(x(\alpha)))$. Получим, что каждый запрос x , имеющий непустой ответ $J(x)$, пройдёт в вершину $\varkappa(J(x))$, и не пройдёт ни в какую вершину $\varkappa(J(x'))$, где $J(x) \neq J(x')$.

Выпустим из каждой вершины, которой сопоставлен переключатель $g_i(x), i \in \{1, \dots, n\}$, несколько рёбер, чтобы из вершины выходило всего $k+1$ рёбер. Пронумеруем эти рёбра так, чтобы из вершины выходило ребро с каждым номером из $\{1, \dots, k+1\}$. Направим все новые рёбра в вершину $\varkappa(\emptyset)$. Кроме того, направим в эту вершину из корня ребро номер $k+1$.

Построенный граф не имеет ориентированных циклов, так как в каждую вершину, имеющую ненулевую полустепень исхода, входит не более одного ребра, причём в корень не входит ни одно ребро. Так как при построении ИГ не использовались предикаты, каждый запрос проходит ровно в одну из вершин графа, из которой не выходит ни одно ребро. Каждая такая вершина названа $\varkappa(J)$ для некоторого ответа J , и уже показано, что если J непуст, то в $\varkappa(J)$ проходят все запросы, имеющие ответ J , и только они. Получим, что все запросы, ответ на которые пуст (и только они), проходят в $\varkappa(\emptyset)$. Таким образом, третье утверждение леммы доказано. Пример построенного графа см. на рис. 2.

Обозначим построенный ИГ U' . Легко заметить, что при обработке каждого запроса x каждая его буква обрабатывается не более одного раза, и если $l(x) < n$, то ещё не более одного раза обрабатывается буква $x[l(x)+1]$. Каждая буква обрабатывается либо в шаге цикла функции $h_{v,i}(x)$, либо в функции $g_i(x)$, то есть сложность обработки буквы не превышает 3. Поэтому время обработки запроса, то есть время, за которое запрос попадёт из корня в некоторую вершину $\varkappa(J)$, не превышает $3e(x)$. Тем самым, второе утверждение леммы доказано.

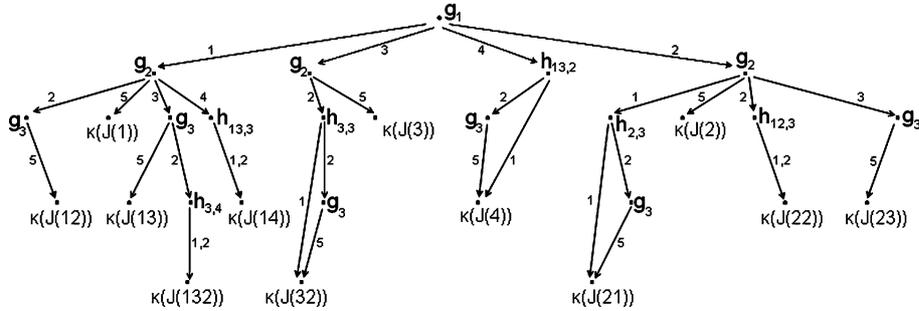


Рис. 2. Граф, построенный по лемме 1 для $k = 4, n = 4, V = \{1323, 1413, 2212\}$. Рёбра, ведущие в вершину $\varkappa(\emptyset)$, и сама эта вершина на рисунке не приведены.

Оценим объём графа U' . Заметим, что в U' из каждой вершины α , кроме корня и вершин с нулевой полустепенью исхода, выходит хотя бы одно ребро, входящее в вершину $\varkappa(J(x(\alpha)))$. С другой стороны, в каждую вершину $\varkappa(J), J \neq \emptyset$ входят рёбра из не более чем двух вершин, причём если в $\varkappa(J)$ входят рёбра из двух вершин, то одной из этих вершин сопоставлен переключатель из множества G_1 , а другой — переключатель из множества G_2 . Сопоставив непустому ответу J все вершины, из которых в $\varkappa(J)$ ведёт ребро, получим, что из этих вершин выходит суммарно не более $k + 4$ рёбер. Все вершины графа, которые мы не можем таким образом сопоставить какому-либо непустому ответу — корень и вершины с нулевой полустепенью исхода. Следовательно, $Q(U') \leq (k + 4)(2pn - 1) + k + 1 < (k + 4)2pn$. Лемма доказана.

Лемма 2. Пусть $I = \langle X, V, \rho \rangle$ — задача поиска вхождений под- слова, библиотеке V из p слов соответствует s различных отве- тов и ИГ U' без листьев над базовым множеством \mathcal{F} обладает тем свойством, что для каждого ответа J найдётся вершина $\varkappa(J)$, из которой не выходит ни одно ребро и в которую попадают все те и только те запросы x , ответ на которые совпадает с J . Тогда граф U' можно, добавив не более $pn + s - 1$ рёбер, достроить до ИГ U над базовым множеством \mathcal{F} , решающего задачу поиска подстроки, при- чём время обработки запроса x после попадания в вершину $\varkappa(J(x))$ не будет превышать $2(|J(x)| - 1)t$.

Доказательство. Для каждого слова x , для которого ответ $J(x)$ непуст, сделаем следующее. Если $l(x) < n$, то для каждого $r \in N_k$, для которого $0 < |J(xr)| < |J(x)|$, выпустим из $\varkappa(J(x))$ в $\varkappa(J(xr))$ ребро с предикатом $f(x)$, если это ребро ещё не проведено. Если найдётся хотя бы одно вхождение (v_i, j) такое, что $x = v_i[j..n]$, то одно из этих вхождений сопоставим вершине $\varkappa(J(x))$, а остальные — каким-либо новым вершинам, и в каждую из этих новых вершин проведём из $\varkappa(J(x))$ ребро с предикатом $f(x)$.

Построенный граф назовём графом U .

Покажем, что подграф U_1 графа U , состоящий из всех добавленных вершин и рёбер, а также всех вершин $\varkappa(J)$, кроме $\varkappa(\emptyset)$, представляет собой ориентированный лес. Пусть в некоторую вершину $\varkappa(J)$ ведёт рёбро из вершины $\varkappa(J_1)$ и из вершины $\varkappa(J_2)$, $J_1 \neq J_2$. Это значит, что найдутся такие запросы x_1 и x_2 , а также буквы i_1 и i_2 , что $J_1 = J(x_1)$, $J_2 = J(x_2)$ и $J = J(x_1i_1) = J(x_2i_2)$. Рассмотрим какое-либо вхождение (v_i, j) из J . Получим, что как x_1i_1 , так и x_2i_2 являются префиксами слова $v_i[j..n]$. Не ограничивая общности, будем считать, что x_1i_1 — префикс x_2i_2 . Если $x_1i_1 = x_2i_2$, то $x_1 = x_2$, откуда $J_1 = J_2$, что противоречит определению J_1 и J_2 . Следовательно, x_1i_1 является префиксом x_2 . Очевидно, $|J(x_1i_1)| \geq |J(x_2)|$, то есть $|J| \geq |J(x_2)|$, что противоречит построению графа. Мы получили, что в каждую вершину $\varkappa(J)$ ведёт не более одного ребра. В каждую добавленную вершину мы (по построению) провели ровно одно ребро. Кроме того, в этом графе нет циклов, так как все добавленные вершины являются концевыми, и если из некоторой вершины $\varkappa(J(x_1))$ идёт ребро в какую-либо вершину $\varkappa(J(x_1r))$, то по построению $|J(x_1r)| < |J(x_1)|$. Итак, граф является ациклическим, и в каждую его вершину входит не более одного ребра, поэтому этот граф — ориентированный лес.

Везде далее в список вершин, в которые запрос пройдёт из $\varkappa(J(x))$, будем включать и саму вершину $\varkappa(J(x))$. Покажем, что запрос x пройдёт из $\varkappa(J(x))$ во все те и только те листья, которым сопоставлены удовлетворяющие этому запросу вхождения. Для запросов, которым не удовлетворяет ни одно вхождение, это очевидно, так как вершине $\varkappa(\emptyset)$ не сопоставлено ни одно вхождение, и ни одно ребро не выходит из этой вершины. Докажем теперь это для всех запросов, которым удовлетворяет хотя бы одно вхождение. Проведём доказательство по индукции по длине запроса x .

Замечание 1. Так как граф U_1 — ориентированный лес, причём всем его рёбрам сопоставлены предикаты, тождественно равные 1, то любые два запроса x_1 и x_2 , прошедшие в какую-либо вершину $\varkappa(J)$, пройдут из неё во все вершины, схемно достижимые из $\varkappa(J)$.

Замечание 2. Запросу x удовлетворяет каждое вхождение (v_i, j) , для которого $x = v_i[j..n]$, а также каждое вхождение, которое удовлетворяет запросу x_i , где $i \in \{1, \dots, k\}$. Никакие другие вхождения запросу x не удовлетворяют.

База индукции: $l(x) = n$. Так как слову x может удовлетворять лишь вхождение $(x, 1)$, то это вхождение при построении U было приписано вершине $\varkappa(J(x))$. Кроме того, из $\varkappa(J(x))$ не были проведены никакие рёбра, то есть запрос не пройдёт из листа $\varkappa(J(x))$ ни в какие другие листья.

Пусть мы уже доказали для всех x длины не менее $q+1$. Докажем для всех x длины q , где $q \geq 1$. Возможно два случая:

1) Найдётся слово x' такое, что $l(x') > q$ и $J(x') = J(x)$. Тогда по предположению индукции запрос x' пройдёт из $\varkappa(J(x'))$ во все те и только те листья, которым сопоставлены удовлетворяющие этому запросу вхождения. Учитывая замечание 1 и то, что $J(x) = J(x')$, получаем, что предположение индукции верно и для x .

2) Для каждого запроса x' длины, превышающей q , $J(x) \neq J(x')$. Отсюда, в частности, следует, что для любого r из N_k выполнено $|J(x)| > |J(xr)|$. При построении из вершины $\varkappa(J(x))$ были проведены рёбра с предикатами, тождественно равными 1, ко всем вершинам $\varkappa(J(xr))$, для которых $0 < |J(xr)| < |J(x)|$. Кроме того, из $\varkappa(J(x))$ были проведены рёбра с теми же предикатами ко всем, кроме одной, вершинам с записями (v_i, j) , для которых $x = v_i[j..n]$. Единственная такая вершина, к которой не было проведено ребро — сама $\varkappa(J(x))$. По предположению индукции, запрос xr пройдёт из $\varkappa(J(xr))$ во все те и только те листья, которым сопоставлены удовлетворяющие этому запросу вхождения. Учитывая замечание 1, получим, что запрос x пройдёт во все листья, которым сопоставлены вхождения, удовлетворяющие запросам xr , и во все листья, которым сопоставлены записи (v_i, j) , для которых $x = v_i[j..n]$; ни в какие другие листья запрос x не пройдёт. Учитывая замечание 2, получаем, что для x верно предположение индукции.

Итак, мы доказали, что U решает задачу поиска вхождений под слова.

Вычислим число рёбер в графе U_1 . Так как U_1 — лес, это число равно числу вершин за вычетом числа компонент связности. В этом графе $s - 1$ либо s вершин $\varkappa(J)$, соответствующих непустым ответам J (в зависимости от того, есть ли среди s ответов, соответствующих библиотеке, пустой ответ). Кроме этих вершин, в U_1 есть новые вершины, каждой из которых сопоставлено некоторое вхождение. Одно и то же вхождение (v_i, j) не может быть сопоставлено более чем одной вершине, так как для него единственным образом определяется вершина $\varkappa(J(v_i[j..n]))$, которой либо сопоставлено это вхождение, либо из которой идёт ребро в новую вершину с этим вхождением. Следовательно, новых вершин не больше, чем различных вхождений, соответствующих библиотеке, то есть pn . Итак, мы получили, что в U_1 не более чем $pn + s - 1$ рёбер.

Оценим сложность обработки запроса x после попадания запроса в вершину $\varkappa(J(x))$, когда $J(x) \neq \emptyset$. Так как граф U_1 — ориентированный лес, то все вершины, в которые запрос пройдёт из $\varkappa(J(x))$, и выходящие из них рёбра образуют ориентированное дерево, корнем которого является $\varkappa(J(x))$. Назовём это дерево $U_1(J(x))$. Покажем, что каждая вершина этого дерева, которой не сопоставлено никакое вхождение, является внутренней, причём из неё выходит не менее двух рёбер. Действительно, по построению это может быть только некоторая вершина $\varkappa(J)$, причём ответ J непуст, так как в графе U_1 нет вершин, из которых в $\varkappa(\emptyset)$ идёт ребро. Пусть x' имеет максимальную длину среди всех запросов, ответом на которые является J . Получим, что нет ни одного соответствующего библиотеке вхождения (v_i, j) , для которого выполнено равенство $x' = v_i[j..n]$, так как одно из таких вхождений было бы сопоставлено $\varkappa(J)$. Следовательно, согласно замечанию 2, J является объединением нескольких непустых ответов $\varkappa(J(x'r))$, где $r \in N_k$. По определению x' ни один из этих ответов не совпадает с J , поэтому таких ответов не менее двух. По построению из $\varkappa(J)$ в каждую из вершин $\varkappa(J(x'r))$, где $i \in N_k$ и ответ $J(x'r)$ непуст, идёт ребро. Итак, из $\varkappa(J)$ выходит не менее двух рёбер. Так как в ориентированном дереве, в котором полустепень исхода каждой внутренней вершины не меньше двух, количество внутренних вершин меньше количества концевых вершин, то в дереве $U_1(J(x))$ вершин, которым не сопоставлено никакое вхождение, меньше, чем вершин, которым сопоставлено какое-либо вхождение. Так как U решает задачу поиска вхождений подслова и каждое соответствующее

библиотеке вхождение сопоставлено ровно одной вершине, в дереве $U_1(J(x))$ вхождение сопоставлено $|J(x)|$ вершинам, а всего в нём не более $2|J(x)| - 1$ вершин и $2|J(x)| - 2$ рёбер. Итак, общая сложность обработки запроса x после его попадания в $\varkappa(J(x))$ не превосходит $2(|J(x)| - 1)t$.

Лемма 3. Пусть $I = \langle X, V, \rho \rangle$, где $|V| = p$ — задача поиска вхождений подслово. Тогда найдётся ИГ $U \in \mathcal{U}(\mathcal{I}, \mathcal{F})$ такой, что $Q(U) \leq (2k + 11)pn$ и для каждого $x \in X$ выполнено $T(U, x) \leq 3R(I, \mathcal{F}, x)$.

Доказательство. Согласно лемме 1, количество ответов, соответствующих V , не превышает $2pn$. Рассмотрим граф U' , построенный по лемме 1. По лемме 2 его можно достроить до ИГ U , решающего задачу $I = \langle X, V, \rho \rangle$. Объём полученного графа будет равен сумме объёма графа U' и количества добавленных по лемме 2 рёбер, то есть $Q(U) \leq (k + 4)2pn + pn + 2pn - 1 < (2k + 11)pn$.

Время обработки произвольного запроса x графом U складывается из времени прохождения запроса из корня в вершину $\varkappa(J(x))$ и времени обработки запроса x после попадания в эту вершину, то есть $T(U, x) \leq 3e(x) + 2(|J(x)| - 1)t \leq 3R(I, \mathcal{F}, x)$. Лемма доказана.

Приведём пример графа, построенного по 3. Пусть $k = 4, n = 4, V = \{1323, 1413, 2212\}$. На рис. 3 изображён ИГ, решающий задачу $I = \langle X, V, \rho \rangle$.

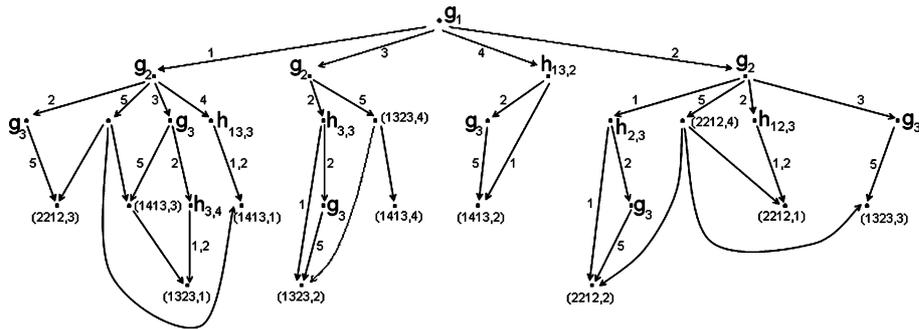


Рис. 3. Граф, решающий задачу для $k = 4, n = 4, V = \{1323, 1413, 2212\}$. Рёбра, ведущие в вершину $\varkappa(\emptyset)$, и сама эта вершина на рисунке не приведены. Все рёбра, у которых на рисунке нет номера, соответствуют предикату $f(x) \equiv 1$.

Перейдём к непосредственному доказательству теоремы 2.

Рассмотрим произвольный ИГ, решающий задачу $I = \langle X, V, \rho \rangle$, где $|V| = p$. Для каждого вхождения (v_i, j) найдётся запрос, которому это вхождение удовлетворяет, например, таковым будет запрос $v_i[j..n]$. Поэтому каждое вхождение (v_i, j) должно быть сопоставлено какому-либо листу, и в этот лист должна вести из корня ориентированная цепь рёбер. Так как листу должно быть сопоставлено ровно одно вхождение, всего рёбер в ИГ должно быть не меньше, чем различных вхождений, то есть pn . Итак, $Q(p, n) \geq pn$. Непосредственно из определений $Q(p, n)$ и $Q_{fast}(p, n)$ следует, что $Q(p, n) \leq Q_{fast}(p, n)$. Согласно лемме 3, найдётся ИГ $U \in \mathcal{U}_{fast}$ такой, что $Q(U) \leq (2k+1)pn$. Из этого следует, что $Q_{fast}(p, n) \leq (2k+1)pn$. Теорема доказана.

Список литературы

- [1] Navarro G., Mäkinen V. Compressed fulltext indexes // ACM Computing Surveys. — 2007. 39 (1). Article 2.
- [2] Weiner P. Linear pattern matching algorithm // Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory. — 1973. — P. 1–11.
- [3] Kurtz S. Reducing the Space Requirement of Suffix Trees // Softw. Pract. Exper. — 1999. 29 (13). — P. 1149–1171.
- [4] Gusfield D. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. — New York: Cambridge University Press, 1997. — P. 116.
- [5] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. — М.: Физматлит, 2002.
- [6] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С. Введение в теорию интеллектуальных систем. — М.: Изд. отд. ф-та ВМиК МГУ, 2006. — С. 94–117.
- [7] Перпер Е. М. О сложности поиска подстроки // Интеллектуальные системы. — 2012. Т. 16, вып. 1–4. — С. 299–320.
- [8] Перпер Е. М. Нижние оценки временной и объёмной сложности задачи поиска подслово // Дискрет. матем. — 2014. 26: 2. — С. 58–70.