Моделирование вычислительных задач информационного поиска

А. П. Пивоваров

Работа посвящена определению и сравнению двух моделей для описания алгоритмов решения вычислительных задач информационного поиска. Первая модель почти совпадает с понятием информационного графа [1], вторая получается использованием для вычислений автоматных функций. Исследование производится на примере модельной задачи — вычислении размера ответа задачи одномерного интервального поиска. Обосновано преимущество использования автоматных функций для вычислительных задач информационного поиска.

Ключевые слова: вычислительная задача информационного поиска, информационный граф, вычислительный информационный граф, одномерный интервальный поиск, моделирование поиска, характеристики алгоритмов поиска.

1. Введение

Формализации понятия задачи информационного поиска (ЗИП) может быть проведена многими способами. В работе [1] рассматриваются задачи поиска, в которых в ответ на запрос надо перечислить элементы базы данных, удовлетворяющие запросу — так называемые перечислительные задачи поиска. Для формального описания и оценки алгоритмов поиска в [1] используется понятие информационного графа (ИГ), использование которого показывает свою эффективность для оценки таких характеристик алгоритма поиска, как время поиска в среднем, время поиска в худшем случае и затраты памяти, необходимые для реализации алгоритма.

Кроме перечислительных задач поиска можно выделить и другие классы задач. Например, задачи поиска представителя, заключающиеся в том, чтобы найти хотя бы один элемент базы данных, удовлетворяющий запросу (либо ответить, что таких элементов в базе данных нет). Любая перечислительная задача информационного поиска имеет модификацию поиска представителя. В [2] рассматривается поиск представителя в двумерной задаче о метрической близости. Для описания алгоритмов поиска таких задач в [2] предлагается также использовать информационные графы.

В данной работе исследуется другой вид задач: вычислительные задачи информационного поиска (ВЗИП). В отличие от перечислительной задачи информационного поиска, смысл вычислительной задачи состоит не в том, чтобы перечислить записи из базы данных, удовлетворяющие запросу, но вычислить некоторую заданную функцию, в качестве аргумента которой передается множество таких записей. Например, задача может состоять в том, чтобы вычислить, сколько записей из библиотеки удовлетворяют запросу (при этом сами эти записи выдавать не требуется).

В общем случае информационный граф не может быть использован для описания алгоритмов решения ВЗИП, потому что его ответом на запрос всегда является некоторое множество записей, в то время как ответ на запрос ВЗИП может быть, например, целым числом или объектом какой-либо иной природы. Мы рассмотрим естественную модификацию ИГ, после проведения которой в качестве ответа на запрос может выступать элемент любого заданного множества. Эту естественную модификацию информационного графа мы назовем упрощенным вычислительным информационным графом (УВИГ). Будет также предложена альтернативная модификация ИГ, также подходящая для описания алгоритмов решения ВЗИП, но более сложная, чем УВИГ. Эта модификация будет названа вычислительным информационным графом (ВИГ). Будет проведено сравнение возможностей УВИГ и ВИГ на модельном примере — вычислении мощности ответа для задачи одномерного интервального поиска. В качестве основной характеристики решения в данной работе будет использоваться не время работы алгоритма (как это обычно делает-

ся), но объем памяти, необходимой для реализации алгоритма. Это связано с тем, что в ряде случаев с помощью УВИГ не удается адекватно описать алгоритмы поиска: сильно завышается необходимый объем памяти, требуемый для описания алгоритма. В частности, далее будет показано, что в случае вычисления мощности ответа для задачи одномерного интервально поиска, объем требуемой для решения памяти в терминах древовидных УВИГ должен расти квадратично от объема базы данных.

В конце работы делается вывод о целесообразности использования именно вычислительных информационных графов для описания алгоритмов решения вычислительных задач информационного поиска.

Автор выражает глубокую благодарность профессору Э. Э. Гасанову за постановку задачи и помощь в работе.

2. Основные понятия и формулировка результатов

Здесь и далее мы используем обозначения и понятия из [1, 2, 3].

Пусть X — множество запросов, Y — множество записей, а ρ — бинарное отношение на $X \times Y$ называемое отношением поиска. Будем называть тройку $S = \langle X, Y, \rho \rangle$ *типом задач информационного поиска*. Тройка $I = \langle X, V, \rho \rangle$, где V — конечное подмножество Y (называемое в дальнейшем библиотекой) называется *задачей информационного поиска* (ЗИП) типа $S = \langle X, Y, \rho \rangle$ (пишем $I \in S$). Фактически задача $I = \langle X, V, \rho \rangle$ состоит в том, чтобы для заданного запроса $x \in X$ перечислить те и только те записи V, которые состоят в отношении ρ с запросом x. Другими словами, задача — найти множество $\mathcal{J}_I(x) = \{y \in V : x \,\rho\, y\}$ для любого заданного $x \in X$. Алгоритм поиска должен находить эти записи и выдавать по одной на выход.

Данная модель хорошо подходит для описания именно такой ситуации, когда от алгоритма требуется перечислить все записи из библиотеки, удовлетворяющие поступившему запросу. Однако в ряде ситуаций требуется получить не сам ответ как множество записей, но результат вычисления некоторой функции на нем. Например, нам

требуется узнать количество записей, состоящих в отношении ρ с запросом x. Будем называть эти задачи вычислительными задачами информационного поиска (ВЗИП).

Такие задачи можно формализовать следующим образом. Тип задач представляет собой пятерку $\langle X, Y, Z, \rho, \xi\rangle$, где X — по-прежнему множество запросов, Y — множество всех возможных записей, Z представляет из себя множество ответов, ρ — отношение поиска, заданное на $X \times Y$. $\xi : 2^Y \to Z$ — функция ответа, определенная на 2^Y — множестве подмножеств Y (достаточно, чтобы ξ была определена для всех конечных подмножеств Y). Сама задача информационного поиска аналогично старому определению представляет из себя пятерку $\langle X, V, Z, \rho, \xi\rangle$, где V — конечное подмножество Y. Содержательно задача поиска будет пониматься в том, чтобы для заданного запроса $x \in X$ находить такой ответ $z \in Z$, что $z = \xi(\{y \in V : x\,\rho\,y\})$. Будем обозначать этот ответ как $z_I(x) = \xi(\{y \in V : x\,\rho\,y\})$ и называть *правильным ответом для задачи I на запрос x*.

Для описания алгоритмов решения перечислительных задач информационного поиска используется понятие информационного графа ([1, 3]). Сейчас мы дадим определение информационного графа (ИГ), вычислительного информационного графа (ВИГ) и упрощенного вычислительного информационного графа (УВИГ).

Сначала введем параллельно понятия ИГ, УВИГ и ВИГ.

В формальном определении понятия ИГ(соответственно, УВИГ и ВИГ) используются следующие множества:

- множество запросов X;
- *множество F одноместных предикатов*, заданных на множестве X;
- *множество G одноместных переключателей*, заданных на множестве X (*переключатели* — это функции, область значений которых является начальным отрезком натурального ряда);
- множество записей Y (только для ИГ).

Для определения УВИГ и ВИГ нам понадобится еще одно множество:

- множество ответов Z;

Для определения ВИГ нам кроме того потребуются следующие множества:

- множество состояний ВИГ M с выделенным элементом $m_0 \in M$, называемым *начальным состоянием*, кроме того зафиксируем *функцию вычисления ответа по состоянию* $\sigma : M \to Z$;
- *множество* H *функций изменения состояния вида* $h : M \to M$, *такое, что любые две функции из* H *коммутируют относительно операции суперпозиции.*

Пару $\mathcal{F} = \langle F, G \rangle$ (тройку $\mathcal{F} = \langle F, G, Z \rangle$ для УВИГ, шестерку $\mathcal{F} = \langle F, G, H, M, m_0, \sigma \rangle$ для ВИГ) будем называть *базовым множеством.*

Определение понятия ИГ (УВИГ, ВИГ) разбивается на два шага. На первом шаге раскрывается структурная часть этого понятия, на втором — функциональная.

Определение ИГ (УВИГ, ВИГ) с точки зрения его структуры.

Пусть нам дана ориентированная многополюсная сеть.

Выделим в ней один полюс и назовем его *корнем*, а остальные полюса назовем *листьями.*

Выделим в сети некоторые вершины и назовем их *точками переключения* (полюса могут быть точками переключения).

Если β — вершина сети, то через ψ_β обозначим *полустепень исхода* вершины β.

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 $\{\overline{1, \psi_\beta}\}$. Эти ребра назовем *переключательными*, а это соответствие — *нагрузкой переключательных ребер.*

Ребра, не являющиеся переключательными, назовем *предикатными.*

Каждому предикатному ребру сети сопоставим некоторый символ из множества F. Это соответствие назовем *нагрузкой предикатных ребер.*

Сопоставим каждому листу сети некоторый элемент из множества Y (Z для УВИГ, H для ВИГ). Это соответствие назовем *нагрузкой листьев*.

Полученную нагруженную сеть назовем *информационным графом (упрощенным вычислительным информационным графом, вычислительным информационным графом)* над базовым множеством $\mathcal{F} = \langle F, G\rangle$ ($\mathcal{F} = \langle F, G, Z\rangle$, $\mathcal{F} = \langle F, G, H, M, m_0, \sigma\rangle$).

Определение функционирования ИГ (УВИГ, ВИГ).

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 n, проводит запрос $x \in X$, если переключатель, приписанный началу этого ребра, принимает значение n на запросе x; ориентированная цепочка ребер проводит запрос $x \in X$, если каждое ребро цепочки проводит запрос x; запрос $x \in X$ проходит в вершину β ИГ (УВИГ, ВИГ), если существует ориентированная цепочка, ведущая из корня в вершину β, которая проводит запрос x.

Сначала рассмотрим случай ИГ. Запись y, приписанная листу α, попадает 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 $\mathcal{J}_U(x)$. Эту функцию $\mathcal{J}_U(x) : X \to 2^Y$ будем считать результатом функционирования ИГ U и называть *функцией ответа ИГ U*.

Рассмотрим случай УВИГ. Элемент $z \in Z$, приписанный листу α будем считать ответом УВИГ на запрос $x \in X$, если запрос x проходит в лист α. Этот ответ z обозначим как $z_U(x)$ и будем называть *функцией ответа УВИГ U*. В случае, если запрос x проходит в два листа α_1 и α_2, которым приписаны ответы z_1 и z_2 соответственно и при этом $z_1 \neq z_2$, то считаем, что $z_U(x)$ не определено. Также $z_U(x)$ не определено в случае, если x не проходит ни в один лист УВИГ U.

Рассмотрим случай ВИГ. Пусть $\alpha_1, \ldots, \alpha_p$ — это все листья, в которые проходит запрос x. Пусть этим листьям приписаны функции $h_1, \ldots, h_p$ соответственно. Назовем состояние $m_U(x) = h_1 \circ \cdots \circ h_p(m_0)$ *финальным состоянием* ВИГ U на запро-

се x. Если запрос x не проходит ни в один лист, положим $m_U(x) = m_0$. Будем называть $z_U(x) = \sigma(m_U(x))$ *функцией ответа ВИГ* U.

Понятия ИГ, УВИГ и ВИГ полностью определены.

Пусть β — некоторая вершина ИГ (УВИГ, ВИГ). Тогда обозначим $\varphi_\beta(x)$ предикат на X, такой что $\varphi_\beta(x) = 1$ для тех и только тех запросов x, которые проходят в вершину β. Предикат $\varphi_\beta(x)$ называется функцией фильтра вершины β.

Будем говорить, что ИГ U решает ЗИП $I = \langle X, V, \rho\rangle$ если для любого запроса $x \in X$ имеем $\mathcal{J}_U(x) = \mathcal{J}_I(x)$, где $\mathcal{J}_I(x) = \{y \in V \colon x\,\rho\,y\}$.

Аналогично будем говорить, что УВИГ U (или ВИГ U) решает ВЗИП $I = \langle X, V, Z, \rho, \xi\rangle$ если для любого запроса $x \in X$ имеем $z_U(x) = z_I(x)$, где $z_I(x) = \xi\,(\{y \in V \colon x\,\rho\,y\})$.

Каждому ИГ (УВИГ, ВИГ) U поставим в соответствие следующую процедуру поиска. Входом процедуры является запрос x. Выход процедуры — множество записей $\mathcal{J}$ (элемент $z \in Z$ или сигнал о том, что ответ не определен для УВИГ, элемент $z \in Z$ для ВИГ) называемое *ответом процедуры на запрос* x. Процедура может оставлять пометки на вершинах ИГ (УВИГ, ВИГ). Кроме того, используется промежуточное множество вершин A. УВИГ имеет переменную для хранения ответа, которая может либо быть пустой, либо содержать элемент из множества Z. В начале процедуры эта переменная пуста. В случае ВИГ дополнительно имеется состояние ВИГ $m \in M$. В начале m устанавливается равным m_0.

Начнем описание самой процедуры. Корень ИГ (УВИГ, ВИГ) U помечается и добавляется в множество A. Процедура рассматривает элементы A по одному до тех пор пока A не пусто. Для каждой рассматриваемой вершины β множества A, процедура делает следующее:

- Если вершина β является листом, то
  - для ИГ запись, приписанная вершине β добавляется к множеству ответов $\mathcal{J}$;
  - для УВИГ рассматриваем ответ z, приписанный вершине β. Если в данный момент переменная хранения ответа пуста или содержит тот же самый ответ z, то присваиваем ей ответ z. В остальных случаях процедуру поиска

ответа останавливаем и выдаем сигнал о том, что ответ не определен;

– для ВИГ рассматривается функция изменения состояния $h \in H$, приписанная листу β. Эта функция применяется к текущему состоянию m, то есть мы устанавливаем $h(m)$ в качестве текущего состояния ВИГ;

- Если β — переключательная вершина, то процедура вычисляет значение переключателя $g(x)$, приписанного вершине β. Пусть $g(x) = n$, число n приписано некоторому переключательному ребру (β, γ) и вершина γ не помечена. Тогда вершина γ помечается и добавляется к A;
- Если β — не переключательная вершина, то рассматриваются все ребра, исходящие из β. Для каждого ребра (β, γ), исходящего из β процедура вычисляет предикат $f(x)$, приписанный предикатному ребру (β, γ). Если $f(x) = 1$ и вершина γ не помечена, то вершина γ помечается и добавляется к A;
- Вершина β исключается из множества A.

Легко видеть, что для ИГ U ответ приведенной процедуры $\mathcal{J}$ на любой запрос x равен $\mathcal{J}_U(x)$.

Аналогично, для УВИГ U если для некоторого $x \in X$ значение $z_U(x)$ определено, то к концу выполнения процедуры это значение будет содержаться в переменной хранения ответа. Его мы и выдадим как результат выполнения процедуры. Если же $z_U(x)$ не определено, то в процессе обработки произойдет выход с сигналом о том, что ответ не определен.

В свою очередь, ВИГ U к моменту завершения описанных действий будет находиться в состоянии m, которое как не трудно заметить равно $m_U(x)$. После этого процедура вычисляет $\sigma(m)$. Это значение и является выходом процедуры. Очевидно, ответ z данной процедуры равен $z_U(x)$ для любого запроса x.

Сложностью ИГ (УВИГ, ВИГ) U на запросе x назовем число $T(U, x)$, равное сумме числа переключателей и предикатов, вычисленных в процессе обработки запроса x. Эту величину также называют временем работы U на запросе x. *Верхней сложностью*

(или временем работы в худшем случае) называют величину $\hat{T}(U) = \max_{x \in X} T(U, x)$.

Объемом $Q(U)$ ИГ (УВИГ, ВИГ) U назовем число ребер в ИГ (ВИГ) U. Эта величина характеризует объем памяти, требуемый для реализации алгоритма поиска, задаваемого ИГ (УВИГ, ВИГ) U.

В рамках данной работы основным оцениваемым параметром будет именно объем графа. Введем следующие обозначения. Пусть задано некоторое базовое множество для УВИГ $\mathcal{F} = \langle F, G, Z \rangle$. Пусть задана некоторая ВЗИП I. Обозначим $\mathcal{U}^{uvig}(I, \mathcal{F})$ множество всех УВИГ над базовым множеством $\mathcal{F}$, которые решают задачу I. Обозначим $\mathcal{U}^{uvig}_{\mathcal{D}}(I, \mathcal{F})$ множество всех древовидных УВИГ над базовым множеством $\mathcal{F}$, которые решают задачу I. *Сложностью задачи* I *при базовом множестве* $\mathcal{F}$ назовем число

$$Q^{uvig}(I, \mathcal{F}) = \min\{Q(U) \colon U \in \mathcal{U}^{uvig}(I, \mathcal{F})\}.$$

Число

$$Q^{uvig}_{\mathcal{D}}(I, \mathcal{F}) = \min\{Q(U) \colon U \in \mathcal{U}^{uvig}_{\mathcal{D}}(I, \mathcal{F})\}$$

назовем *древовидной сложностью задачи* I *при базовом множестве* $\mathcal{F}$.

Если k — натуральное число, S — тип вычислительных задач информационного поиска, то обозначим

$$\mathcal{I}(k, S) = \{I = \langle X, V, Z, \rho, \xi \rangle \in S \colon |V| = k\}.$$

Будем исследовать функцию Шеннона, характеризующую сложность (либо древовидную сложность) класса ВЗИП $\mathcal{I}(k, S)$:

$$\begin{aligned} \mathcal{Q}^{uvig}(k, S, \mathcal{F}) &= \sup_{I \in \mathcal{I}(k,S)} Q^{uvig}(I, \mathcal{F}), \\ \mathcal{Q}^{uvig}_{\mathcal{D}}(k, S, \mathcal{F}) &= \sup_{I \in \mathcal{I}(k,S)} Q^{uvig}_{\mathcal{D}}(I, \mathcal{F}). \end{aligned}$$

Аналогичные функции $\mathcal{Q}^{vig}(k, S, \mathcal{F})$ и $\mathcal{Q}^{vig}_{\mathcal{D}}(k, S, \mathcal{F})$ введем и для ВИГ — с той лишь разницей, что использоваться должно базовое множество для ВИГ $\mathcal{F} = \langle F, G, H, M, m_0, \sigma \rangle$.

Определим модельную задачу, для которой получены основные результаты данной работы. Введем тип вычислительных задач информационного поиска S^{card}_{int1}, заданный следующими соотношениями:

$$S^{card}_{int1} = \langle X_{int1}, Y_{int1}, \mathbb{Z}, \rho_{int1}, \xi_{int1} \rangle, \text{ где} \tag{1}$$

$$X_{int1} = \{(x_1, x_2) \in [0,1]^2 \colon x_1 \leqslant x_2\}, \tag{2}$$

$$Y_{int1} = [0,1], \tag{3}$$

$$\forall (x_1, x_2) \in X_{int1}, \forall y \in Y_{int1} \colon (x_1, x_2)\, \rho_{int1}\, y \Leftrightarrow x_1 \leqslant y \leqslant x_2, \tag{4}$$

$$\forall W \subseteq Y_{int1} \colon \xi_{int1}(W) = |W|. \tag{5}$$

Данная ВЗИП является модификацией перечислительной задачи одномерного интервального поиска в смысле вычисления мощности ответа. Содержательно перечислительная задача состоит в следующем. Задана библиотека — конечное подмножество отрезка $[0,1]$. Задача состоит в том, чтобы для любого запроса $x = (x_1, x_2)$ находить те элементы библиотеки, которые попали в отрезок $[x_1, x_2]$. Данная задача рассматривается в [4]. Наша вычислительная модификация состоит в том, чтобы найти не сами элементы базы данных, лежащие в указанном отрезке-запросе, но лишь количество таких элементов.

Для такой задачи можно предложить следующее неформальное решение. Для произвольного запроса $x = (x_1, x_2)$ мы решаем задачу поиска мощности ответа для задачи о доминировании (то есть поиска записей библиотеки, не превышающих значения запроса) дважды: первый раз используя в качестве запроса x_2 и второй раз используя в качестве запроса x_1, а затем вычитаем второй ответ из первого (когда используем в качестве запроса x_1, будем искать количество записей строго меньших, чем значение запроса). Если использовать, например, бинарный поиск для решения каждой задачи о доминировании, нам потребуется память порядка k (где k — количество записей в библиотеке). Таким образом, можно ожидать,что суммарное количество требуемой для алгоритма памяти должно быть $O(k)$.

Рассмотрим множество предикатов F_{int1} и переключателей G_{int1}, заданные следующими условиями:

$$F_{int1} = \left\{ f^{id}(x) \equiv 1 \right\}, \tag{6}$$

$$G_{int1} = G_1 \cup G_2, \tag{7}$$

$$G_1 = \left\{ g^1_{\leqslant,a}(x) = \begin{cases} 1, \text{ если } x_1 \leqslant a, \\ 2, \text{ если } x_1 > a \end{cases}, \text{ где } a \in \mathbb{R} \right\}, \tag{8}$$

$$G_2 = \left\{ g^2_{<,a}(x) = \begin{cases} 1, \text{ если } x_2 < a, \\ 2, \text{ если } x_2 \geqslant a \end{cases}, \text{ где } a \in \mathbb{R} \right\}, \tag{9}$$

Рассмотрим также множество M_{int1}, выделенный его элемент $m_0 \in M_{int1}$, набор функций H_{int1} (из M_{int1} в M_{int1}) и функцию $\sigma_{int1}\colon M_{int1} \to \mathbb{Z}$:

$$M_{int1} = \mathbb{Z},\ m_0 = 0, \tag{10}$$

$$H_{int1} = \{h_a : \mathbb{Z} \to \mathbb{Z}\colon a \in \mathbb{Z}\}, \tag{11}$$

$$\forall a \in \mathbb{Z},\ \forall m \in \mathbb{Z}\colon h_a(m) = m + a, \tag{12}$$

$$\forall m \in \mathbb{Z}\colon \sigma_{int1}(m) = m. \tag{13}$$

Определим два базовых множества $\mathcal{F}^1_{int1}$ и $\mathcal{F}^2_{int1}$, над которыми мы будем строить УВИГ и ВИГ соответственно:

$$\mathcal{F}^1_{int1} = \langle F_{int1}, G_{int1}, \mathbb{Z}\rangle, \tag{14}$$

$$\mathcal{F}^2_{int1} = \langle F_{int1}, G_{int1}, H_{int1}, M_{int1}, m_0, \sigma_{int1}\rangle. \tag{15}$$

Теперь мы готовы сформулировать основные результаты данной работы.

Теорема 1. *Рассмотрим тип вычислительных задач информационного поиска S^{card}_{int1}, определяемый соотношениями (1)–(5). Рассмотрим базовое множество $\mathcal{F}^1_{int1}$, определенное соотношением (14). Для любого натурального числа k справедливо:*

$$\frac{k(k+1)}{2} \leqslant \mathcal{Q}^{uvig}_{\mathcal{D}}(k, S_{int1}, \mathcal{F}^1_{int1}) \leqslant k^2 + 3k,$$
$$4k - 4 \leqslant \mathcal{Q}^{uvig}(k, S_{int1}, \mathcal{F}^1_{int1}) \leqslant 6k\log_2 k + 4k + 2\log_2 k + 2.$$

Следствие 1. *В условиях теоремы 1 выполнено:*

$$\mathcal{Q}^{uvig}_{\mathcal{D}}(k, S_{int1}, \mathcal{F}^1_{int1}) \asymp k^2 \ \ (при\ k \to \infty),$$
$$k \lesssim \mathcal{Q}^{uvig}(k, S_{int1}, \mathcal{F}^1_{int1}) \lesssim k \log_2 k \ \ (при\ k \to \infty).$$

Теорема 2. *Рассмотрим тип вычислительных задач информационного поиска S^{card}_{int1}, определяемый соотношениями (1)–(5). Рассмотрим базовое множество $\mathcal{F}^2_{int1}$, определенное соотношением (15). Для любого натурального числа k справедливо:*

$$4k - 4 \leqslant \mathcal{Q}^{vig}(k, S_{int1}, \mathcal{F}^2_{int1}) \leqslant \mathcal{Q}^{vig}_{\mathcal{D}}(k, S_{int1}, \mathcal{F}^2_{int1}) \leqslant 4k + 2.$$

Следствие 2. *В условиях теоремы 2 выполнено:*

$$\mathcal{Q}^{vig}(k, S_{int1}, \mathcal{F}^2_{int1}) \sim \mathcal{Q}^{vig}_{\mathcal{D}}(k, S_{int1}, \mathcal{F}^2_{int1}) \sim 4k \ \ (при\ k \to \infty).$$

3. Случай древовидных графов

Лемма 1. *Рассмотрим ВЗИП $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1}\rangle \in S^{card}_{int1}$, где S_{int1} определяется соотношениями (1)–(5), $V \subseteq [0,1]$, $|V| < \infty$. Рассмотрим базовое множество $\mathcal{F}^1_{int1}$, определенное соотношением (14). Пусть D — древовидный УВИГ над базовым множеством $\mathcal{F}^1_{int1}$ и D решает задачу I. Тогда имеет место следующая оценка:*

$$Q(D) \geqslant \frac{|V|(|V|+1)}{2}.$$

Доказательство. Пусть $V = \{y_1, \ldots, y_k\}$, где $0 \leqslant y_1 < \cdots < y_k \leqslant 1$, $k = |V|$. Пусть $x_{ij} = (y_i, y_j)$ для всех $i, j \in \mathbb{Z}$ таких что $k \geqslant j \geqslant i \geqslant 1$. Тогда множество записей библиотеки, состоящих в отношении поиска с запросом x_{ij} есть $\{y \in V \colon x_{ij}\, \rho_{int1}\, y\} = \{y \in V \colon y_i \leqslant y \leqslant y_j\} = \{y_i, \ldots, y_j\}$ и $\xi_{int1}(\{y \in V \colon x_{ij}\, \rho_{int1}\, y\}) = j - i + 1$.

По предположению D решает задачу I, поэтому для запроса x_{ij} существует путь, ведущий из корня D к листу, которому приписан ответ $j - i + 1$, такой что этот путь проводит запрос x_{ij}. Обозначим последнее ребро этого пути как c_{ij}. Покажем, что все c_{ij} различны.

Предположим, что существуют два различных запроса x_{ij} и x_{ab}, такие что $c_{ij} = c_{ab}$. Тогда оба запроса проходят в лист α, которому приписано значение $j - i + 1$. Одновременно, этому же листу должно быть приписано значение $b - a + 1$. Таким образом, $j - i + 1 = b - a + 1$ и $j - i = b - a$. Если $a = i$, то $b = j$ и $x_{ij} = x_{ab}$. Поэтому $a \neq i$ и без потери общности можно считать, что $a > i$. Рассмотрим запрос x_{ib} и путь, ведущий из корня D к листу α. Запросы x_{ij} и x_{ab} проходят по этому пути. Так как проводимость каждого ребра этого пути есть представляет из себя предикат, сравнивающий либо первую, либо вторую координату запроса с константой (строгим или нестрогим образом), то запрос x_{ib} также проходит по этому пути. Но $a > i \Rightarrow b - a \neq b - i \Rightarrow b - a + 1 \neq b - i + 1$ и для запроса x_{ib} граф выдает неверный ответ $b - a + 1$. Мы получили противоречие, так как по предположению граф D решает задачу I. Поэтому все c_{ij} различны, а так как их $\frac{|V|(|V|+1)}{2}$, то утверждение леммы доказано.

Лемма 2. *Рассмотрим ВЗИП* $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1} \rangle \in S^{card}_{int1}$, *где* S_{int1} *определяется соотношениями (1)–(5),* $V \subseteq [0, 1]$, $|V| < \infty$. *Рассмотрим базовое множество* $\mathcal{F}^2_{int1}$, *определенное соотношением (15). Существует древовидный ВИГ* D *над базовым множеством* $\mathcal{F}^2_{int1}$, *такой что* D *решает ВЗИП* I *и имеет следующие характеристики:*

$$\hat{T}(D) \leqslant 2\,]\log_2(|V| + 1)[+2,$$
$$Q(D) = 4|V| + 2.$$

Доказательство. Сконструируем граф, удовлетворяющий условиям леммы. Для этого сначала сконструируем ВИГ D_+ над базовым множеством $\mathcal{F}^2_{int1}$ следующим образом. Рассмотрим сбалансированное дерево с $k + 1$ листом (где $k = |V|$). Данное дерево содержит ровно $2k$ ребер, а самый длинный путь из корня к листу состоит из $]\log_2(k+1)[$ ребер. Припишем функции h_i $(i = 0, \ldots, k)$ листьям дерева начиная с самого левого листа и заканчивая самым правым. Пусть $V = \{y_1, \ldots, y_k\}$ и $y_1 < \cdots < y_k$. Для каждой внутренней вершины α дерева сделаем следующее. Обозначим $R(\alpha)$ множество всех листьев, лежащих в правом поддереве вершины α. Рассмотрим множество ин-

дексов $I(\alpha) = \{i\colon h_i$ приписана некоторой $\beta \in R(\alpha)\}$. Так как вершина α — внутренняя, то $I(\alpha)$ не пусто. Пусть i_α — наименьший элемент $I(\alpha)$. Объявим α переключательной вершиной и припишем ей переключатель $g^2_{<,y_i}$. Припишем 1 левому ребру, выходящему из α и 2 правому ребру. Обозначим полученный граф как D_+.

Так как самый длинный путь из корня к листу в D_+ состоит из $]\log_2(k+1)[$ ребер, то для любого запроса вычисляется не более $]\log_2(k+1)[$ переключателей. Таким образом $\hat{T}(D_+) \leqslant]\log_2(k+1)[$.

Используя D_+, построим D_- следующим образом. Возьмем копию D_+ заменим в ней каждый переключатель $g^2_{<,y_i}$ соответствующим ему переключателем $g^1_{\leqslant,y_i}$. Также заменим каждую функцию h_i, приписанную некоторому листу на h_{-i}.

Наконец, сконструируем требуемый ВИГ D. Возьмем вершину и объявим ее корнем графа. Выпустим из нее два предикатных ребра, каждому из которых приписан предикат f_{id}. Обозначим концы этих ребер как v_+ и v_-. Выпустим граф D_+ из v_+ и D_- из v_-. Требуемый граф D сконструирован.

Посмотрим, как D обрабатывает запрос $x = (x_1, x_2)$. В подграфе D_+ запрос проходит в вершину, которой приписана функция h_i, где i — максимальное число, такое что выполнено неравенство $y_i \leqslant x_2$ (или 0 если таких записей в библиотеке нет). Таким образом, i равно количеству записей из V, которые не превосходят x_2. В подграфе D_- запрос проходит в вершину, которой приписана функция h_{-j}, где j — максимальное число, такое что $y_j < x_1$ (или 0 если таких записей нет). Таким образом, j равно числу записей из V, которые строго меньше, чем x_1. Состояние $m_D(x) = h_i \circ h_{-j}(0) = i - j$ является финальным состоянием D после обработки запроса x. А так как $\sigma(m) = m$, то ответом D на запрос x является число $i-j$. Но $i-j$ в точности равно числу записей V, которые не больше, чем x_2 и не меньше, чем x_1, то есть числу записей из V, состоящих в отношении поиска с запросом x. Получаем, что D решает ВЗИП I.

Так как $Q(D_+) = Q(D_-) = 2k$, имеем $Q(D) = 4k+2$. А из неравенств $\hat{T}(D_+) \leqslant]\log_2(k+1)[$ и $\hat{T}(D_-) \leqslant]\log_2(k+1)[$ следует, что $\hat{T}(D) \leqslant 2\,]\log_2(k+1)[+2$ (добавляются два предиката, исходящие из корня D). Лемма полностью доказана.

Лемма 3. *Рассмотрим ВЗИП* $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1}\rangle \in S^{card}_{int1}$, *где* S_{int1} *определяется соотношениями (1)–(5),* $V \subseteq [0,1]$, $0 < |V| < \infty$. *Рассмотрим базовое множество* $\mathcal{F}^2_{int1}$, *определенное соотношением (15). Существует древовидный УВИГ* D *над базовым множеством* $\mathcal{F}^1_{int1}$, *такой что* D *решает ВЗИП* I *и имеет следующие характеристики:*

$$\hat{T}(D) \leqslant 2\,]\log_2(|V|+1)[,$$
$$Q(D) = |V|^2 + 3|V|.$$

Доказательство. Пусть $V = \{y_1, \ldots, y_k\}$, где $0 \leqslant y_1 < \cdots < y_k \leqslant 1$, $k = |V|$. Сначала построим бинарное дерево D_+, как делали это в лемме 2. Из каждого листа α_i $(0 \leqslant i \leqslant |V|)$ этого дерева (которому в лемме 2 мы приписывали функцию изменения состояния h_i), кроме листа α_0, выпустим по бинарному дереву аналогичному D_- из леммы 2, но построенному не по всей базе данных V, а по ее подмножеству $V_i = \{y_t \colon 1 \leqslant t \leqslant i\}$. Листья этого дерева обозначим α_{ij} $(0 \leqslant j \leqslant i)$. При этом в качестве α_{ij} мы берем именно тот лист, которому в лемме 2 мы бы приписывали h_{-j}. После этого листу α_0 приписываем ответ 0, а каждому из листьев α_{ij} $(0 \leqslant j \leqslant i \leqslant |V|)$ припишем ответ $i - j$. Полученный граф и будет искомым графом D. Доказательство того, что D решает I аналогично тому, как это делается в лемме 2. Осталось оценить характеристики графа D.

Оценим объем полученного графа. Количество ребер в дереве D_+ в точности равно $2k$, а в каждом дереве, выпущенном из вершины α_i $(1 \leqslant i \leqslant |V|)$ в точности $2|V_i| = 2i$ ребер. Сложим все вместе и получим $Q(D) = 2k + \sum_{i=1}^{k} 2i = 2k + k(k+1) = k^2 + 3k$.

Оценим время работы в худшем случае. Так как самый длинный путь из корня к листу в D состоит из не более чем $2]\log_2(k+1)[$ ребер, то для любого запроса вычисляется не более $2]\log_2(k+1)[$ переключателей. Таким образом $\hat{T}(D) \leqslant 2\,]\log_2(k+1)[$, что завершает доказательство леммы.

4. Случай произвольных графов

Лемма 4. *Рассмотрим ВЗИП* $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1} \rangle \in S^{card}_{int1}$, *где* S_{int1} *определяется соотношениями (1)–(5),* $V \subseteq [0,1]$, $|V| < \infty$. *Рассмотрим базовое множество* $\mathcal{F}^1_{int1}$, *определенное соотношением (14). Пусть* U — *некоторый УВИГ над базовым множеством* $\mathcal{F}^1_{int1}$ *и* U *решает задачу* I. *Тогда имеет место следующая оценка:*

$$Q(U) \geqslant 4|V| - 4.$$

Доказательство. Пусть $V = \{y_1, \ldots, y_k\}$, где $0 \leqslant y_1 < \cdots < y_k \leqslant 1$, $k = |V|$. Рассмотрим все числа $a \in [0,1]$, такие что переключатель $g^2_{<,a}$ приписан некоторой вершине U. Обозначим множество таких чисел A_2. Множество A_2 — конечно, так как в U конечное число вершин. Рассмотрим запись y_i $(2 \leqslant i \leqslant k)$. Так как $y_i > y_1 \geqslant 0$, то $y_i > 0$. Предположим, что $y_i \notin A_2$. Возьмем такое ε, что $0 < \varepsilon \leqslant y_i$ и $[y_i - \varepsilon, y_i] \cap A_2 = \varnothing$. Такое ε существует, так как A_2 конечно. Рассмотрим два запроса $x = (0, y_i)$ и $x' = (0, y_i - \varepsilon)$. Для этих запросов выполнено $z_I(x') < i = z_I(x)$, следовательно $z_I(x) \neq z_I(x')$. Но каждое ребро проводящее запрос x, также проводит запрос x', и наоборот. Следовательно, и ответы на эти запросы граф U выдаст одинаковые. Но по условию граф U решает задачу I. Мы пришли к противоречию. Следовательно, $y_i \in A_2$, а значит в U есть переключательная вершина, которой приписан переключатель $g^2_{<,y_i}$. Аналогичным образом, доказывается, что для любого i в пределах от 1 до $k-1$ включительно в графе U есть вершина, которой приписан переключатель $g^1_{\leqslant,y_i}$. Итого, мы выделили в U как минимум $2k-2$ различных переключательных вершин. Из каждой из них выходит по 2 ребра и все эти ребра различны. Таким образом, U содержит не менее $4k-4$ ребер. Лемма доказана.

Лемма 5. *Рассмотрим ВЗИП* $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1} \rangle \in S^{card}_{int1}$, *где* S_{int1} *определяется соотношениями (1)–(5),* $V \subseteq [0,1]$, $|V| < \infty$. *Рассмотрим базовое множество* $\mathcal{F}^2_{int1}$, *определенное соотношением (15). Пусть* U — *некоторый ВИГ над базовым множеством* $\mathcal{F}^2_{int1}$ *и* U *решает задачу* I. *Тогда имеет место следующая оценка:*

$$Q(U) \geqslant 4|V| - 4.$$

Доказательство. Доказательство аналогично доказательству леммы 4.

Лемма 6. *Рассмотрим ВЗИП* $I = \langle X_{int1}, V, \mathbb{Z}, \rho_{int1}, \xi_{int1}\rangle \in S^{card}_{int1}$, *где* S_{int1} *определяется соотношениями (1)–(5),* $V \subseteq [0,1]$, $0 < |V| < \infty$. *Рассмотрим базовое множество* $\mathcal{F}^1_{int1}$, *определенное соотношением (14). Тогда существует такой* U — *УВИГ над базовым множеством* $\mathcal{F}^1_{int1}$, *что* U *решает задачу* I *и имеют место следующие оценки:*

$$Q(U) \leqslant 6|V|\log_2|V| + 4|V| + 2\log_2|V| + 2,$$
$$\hat{T}(U) \leqslant \log_2^2|V| + 6\log_2|V| + 5.$$

Доказательство. Пусть $V = \{y_1, \ldots, y_k\}$, где $0 \leqslant y_1 < \cdots < y_k \leqslant 1$. Пусть $M =]\log_2(|V|+1)[$. Из этого сразу следует, что

$$2^M \geqslant |V| + 1, \tag{16}$$
$$2^{M-1} < |V| + 1. \tag{17}$$

Из (17) следует

$$2^{M-1} \leqslant |V|. \tag{18}$$

Пусть $x = (x_1, x_2) \in X_{int1}$ Введем также обозначения $z^+_I(x)$ и $z^-_I(x)$ согласно следующим формулам:

$$z^+_I(x) = |\{y \in V : y \leqslant x_2\}|, \tag{19}$$
$$z^-_I(x) = |\{y \in V : y < x_1\}|. \tag{20}$$

Очевидно, что для ответа задачи I на запрос x справедливо соотношение

$$z_I(x) = z^+_I(x) - z^-_I(x). \tag{21}$$

Для предъявления требуемого графа U построим сначала последовательность УВИГ U_i, где i пробегает от 0 до M. Для графов U_i будем требовать выполнения следующих условий:

1) Каждый граф U_i, кроме U_M имеет ровно 2^i концевые вершины (то есть вершины, из которых не выходит ни одно ребро). Последний граф U_M имеет $|V|+1$ концевую вершину;
2) Пусть $\{\alpha_j^i\}_{j=0}^{\min(2^i-1,|V|)}$ — концевые вершины графа U_i. Тогда функции фильтров вершин α_j^i имеют следующий вид:

$$\varphi_{\alpha_j^i}(x) = \begin{cases} 1, \text{ если } z_I(x) = j \ (mod\ 2^i), \\ 0, \text{ иначе.} \end{cases} \tag{22}$$

Это означает, что запрос x проходит в вершину α_j^i тогда и только тогда, когда $z_I(x) = j \ (mod\ 2^i)$.

Графы U_i будем строить по индукции. В качестве U_0 возьмем граф, состоящий из одного только корня — он же будет и единственной концевой вершиной U_0, то есть α_0^0. Пусть уже построен граф U_i, $0 \leqslant i \leqslant M-1$. Покажем, как с его помощью построить граф U_{i+1}. Будем перебирать по очереди все концевые вершины U_i. Пусть текущая концевая вершина есть α_j^i, где $0 \leqslant j \leqslant 2^i - 1$. Рассмотрим подмножество библиотеки $V_j^i = \{y_t \colon j < t \leqslant |V|,\ t = j \ (mod\ 2^i)\}$. Далее возможны два случая.

1) Пусть $V_j^i = \varnothing$. Это значит, что $j + 2^i > |V|$. Такая ситуация может иметь место только в случае $i = M-1$. Если запрос x прошел в вершину α_j^i, то по предположению индукции $z_I(x) = j \ (mod\ 2^i)$. Но тогда $z_I(x) = j + m \cdot 2^i$, где m — некоторое неотрицательное целое число. Из $j + 2^i > |V|$ и того, что $z_I(x) \leqslant |V|$ следует, что $m = 0$. Таким образом из того, что запрос x прошел в вершину α_j^i следует $z_I(x) = j$. Имеем следующую цепочку утверждений. Запрос x проходит в вершину α_j^i $\Rightarrow$ $z_I(x) = j \Rightarrow z_I(x) = j \ (mod\ 2^{i+1}) \Rightarrow z_I(x) = j \ (mod\ 2^i) \Rightarrow$ Запрос x проходит в вершину α_j^i. Из этой цепочки следует, что запрос x проходит в вершину α_j^i тогда и только тогда, когда $z_I(x) = j \ (mod\ 2^{i+1})$. Получаем, что функция фильтра вершины α_j^i в точности такая, какая должна быть у вершины α_j^{i+1} согласно условию (22). Объявляем вершину α_j^i одновременно вершиной α_j^{i+1}.

2) Пусть $V_j^i \neq \varnothing$. Построим сбалансированное бинарное дерево с $|V_j^i|+1$ вершиной. Занумеруем его концевые вершины слева направо числами от 0 до $|V_j^i|$. Каждой внутренней вершине α рассматриваемого дерева припишем переключатель $g^2_{<,y_{j+2^i s(\alpha)}}$, где $s(\alpha)$ — наименьший номер, соответствующий некоторой концевой вершине из правого поддерева рассматриваемой внутренней вершины α. Левому ребро, ведущему из α припишем номер 1, а правому номер 2. После этого склеим все концевые вершины дерева с четными номерами между собой и обозначим полученную вершину как β^i_{0j}. Затем склеим все концевые вершины с нечетными номерами и обозначим полученную вершину как β^i_{1j}. Легко убедиться, что если запрос прошел в вершину α^i_j, то он проходит далее в β^i_{cj} $(c = 0, 1)$ в том и только том случае, когда выполнено следующее равенство:

$$\left[\frac{z_I^+(x) - j}{2^i}\right] = c \ (mod\ 2). \tag{23}$$

Рассмотрим еще одно подмножество библиотеки $W_i = \{y_t \colon 1 \leqslant t \leqslant |V|,\ t = 0\ (mod\ 2^i)\}$. Из неравенства $i \leqslant M-1$ и определения M следует, что $W_i \neq \varnothing$. Построим бинарное дерево аналогично построенному выше, используя вместо V_j^i множество W_i и приписывая внутренним вершинам дерева вместо переключателя $g^2_{<,y_{j+2^i s(\alpha)}}$ переключатель $g^1_{\leqslant,y_{j+2^i s(\alpha)}}$. У полученного графа склеим все вершины с четными номерами в одну, а с нечетными — в другую. По одной копии полученного графа выпустим из каждой вершины β^i_{cj} $(c \in \{0,1\})$. Концевые вершины графа, выпущенного из β^i_{cj} назовем γ^i_{0cj} и γ^i_{1cj} соответственно (вершина, полученная склеиванием концевых вершин с четными номерами будет называться γ^i_{0cj}). При этом запрос x, прошедший в α^i_j, проходит далее в вершину γ^i_{dcj} $(d, c \in \{0,1\})$ тогда и только тогда, когда выполнено равенство (23) и кроме того выполнено следующее равенство:

$$\left[\frac{z_I^-(x)}{2^i}\right] = d \ (mod\ 2). \tag{24}$$

Пусть запрос x проходит в γ^i_{dcj}. Распишем подробней полученные условия. Равенство (23) означает, что

$$\left[\frac{z_I^+(x)-j}{2^i}\right] = c + 2l \tag{25}$$

для некоторого целого l. Отсюда получаем

$$c + 2l \leqslant \frac{z_I^+(x)-j}{2^i} < c + 2l + 1. \tag{26}$$

Точно так же для некоторого целого m имеет место

$$d + 2m \leqslant \frac{z_I^-(x)}{2^i} < d + 2m + 1. \tag{27}$$

Вычитая равенство (27) из (26) и используя (21) получим:

$$c - d + 2(l-m) - 1 < \frac{z_I(x)-j}{2^i} < c - d + 2(l-m) + 1. \tag{28}$$

Но из того, что запрос x проходит в α^i_j следует, что $z_I(x) = j \ (mod\ 2^i)$, то есть $\frac{z_I(x)-j}{2^i}$ есть целое число. Числа $c-d+2(l-m)-1$ и $c-d+2(l-m)+1$ тоже целые, а значит $\frac{z_I(x)-j}{2^i} = c-d+2(l-m)$. Получаем, что $z_I(x) = j + (c-d)2^i \ (mod\ 2^{i+1})$. Таким образом, если запрос x проходит в вершину γ^i_{dcj} и $d = c$, то $z_I(x) = j \ (mod\ 2^{i+1})$. Если же запрос x проходит в вершину γ^i_{dcj} и $d \neq c$, то $z_I(x) = j + 2^i \ (mod\ 2^{i+1})$. Склеим вершины γ^i_{00j} и γ^i_{11j} между собой и обозначим полученную вершину как α^{i+1}_j. Вершины γ^i_{01j} и γ^i_{10j} тоже склеим и обозначим полученную вершину как $\alpha^{i+1}_{j+2^i}$. Мы уже доказали, что если запрос x проходит в вершину α^{i+1}_j, то $z_I(x) = j \ (mod\ 2^{i+1})$ (и аналогичное утверждение для $\alpha^{i+1}_{j+2^i}$). Наоборот, если $z_I(x) = j \ (mod\ 2^{i+1})$, то тем более $z_I(x) = j \ (mod\ 2^i)$ и по предположению индукции запрос x проходит в вершину α^i_j. Но граф, выпущенный из α^i_j содержит только переключатели, следовательно запрос x должен дойти до одной из двух концевых вершин α^{i+1}_j

или $\alpha^{i+1}_{j+2^i}$. Пройти во вторую вершину он не может, так как тогда было бы выполнено $z_I(x) = j + 2^i \ (mod\ 2^{i+1})$, что неверно. Значит, запрос проходит в вершину α^{i+1}_j. Аналогично если $z_I(x) = j + 2^i \ (mod\ 2^{i+1})$, то запрос x проходит в вершину $\alpha^{i+1}_{j+2^i}$. Таким образом, функции фильтров новых вершин α^{i+1}_j и $\alpha^{i+1}_{j+2^i}$ удовлетворяют условию (22).

Пройдя по всем концевым вершинам α^i_j графа U_i, получим граф U_{i+1}, удовлетворяющий условиям индукции.

Возьмем теперь граф U_M и каждой его концевой вершине α^M_j $(0 \leqslant j \leqslant |V|)$ припишем ответ j. Покажем, что полученный УВИГ решает задачу I. По построению графа U_M имеем: запрос x проходит в вершину α^M_j тогда и только тогда, когда $z_I(x) = j \ (mod\ 2^M)$. Но из (16) следует, что $|V| < 2^M$. Поэтому условие $z_I(x) = j \ (mod\ 2^M)$ равносильно тому, что $z_I(x) = j$.

Осталось оценить параметры $Q(U_M)$ и $\hat{T}(U_M)$.

Оценим $Q(U_M)$ — количество ребер в построенном графе. Посчитаем, сколько ребер мы добавили к графу U_i, чтобы достроить его до U_{i+1} $(0 \leqslant i \leqslant M-1)$. В графе U_i в точности 2^i концевых вершин. Каждую концевую вершину α^i_j мы либо не трогали (в случае, если $|V^i_j| = 0$), либо выпустили по бинарному дереву с $|V^i_j| + 1$ концевой вершиной (дальнейшее склеивание некоторых концевых вершин не влияет на количество ребер). В таком дереве ровно $2|V^i_j|$ ребер. После склеивания концевых вершин этого дерева мы получили две концевые вершины и из каждой выпустили еще по одному бинарному дереву с $|W_i| + 1$ концевой вершиной в каждом. Тем самым мы добавили еще два раза по $2|W_i|$ ребер к нашему графу. В итоге получаем, что при переходе от U_i к U_{i+1} было добавлено не более $\sum_{j=0}^{2^i-1}(2|V^i_j| + 4|W_i|)$ ребер. Так как при фиксированном i все V^i_j не пересекаются между собой, и кроме того элементы y_t при $1 \leqslant t \leqslant 2^i - 1$ не принадлежат ни одному из V^i_j, то $\sum_{j=0}^{2^i-1} |V^i_j| \leqslant |V| - 2^i + 1$. По построению $|W_i| = \left[\frac{|V|}{2^i}\right] \leqslant \frac{|V|}{2^i}$. Итого $\sum_{j=0}^{2^i-1}(2|V^i_j| + 4|W_i|) = 2\sum_{j=0}^{2^i-1} |V^i_j| + 2^i(4|W_i|) \leqslant 2(|V| - 2^i + 1) + 2^i(4\frac{|V|}{2^i}) = 6|V| - 2^{i+1} + 2$.

Начальный граф U_0 не содержал ни одного ребра. Используя только что полученную оценку сверху на количество ребер, которое нужно добавить для перехода от U_i к U_{i+1}, получаем $Q(U_M) \leqslant \sum_{i=0}^{M-1}(6|V| - 2^{i+1} + 2) \leqslant 6M|V| - (2^{M+1} - 2) + 2M$. Из (16) получаем $2 - 2^{M+1} \leqslant -2|V|$. Подставляя это неравенство в полученную выше оценку на $Q(U_M)$, получаем $Q(U_M) \leqslant 2M(3|V| + 1) - 2|V|$. Неравенство (18) можно переписать в следующем виде:

$$M \leqslant 1 + \log_2 |V|. \tag{29}$$

В итоге $Q(U_M) \leqslant 2(1+\log_2 |V|)(3|V|+1)-2|V| = 6|V|\log_2 |V|+4|V|+2\log_2 |V| + 2$.

Оценим $\hat{T}(U_M)$. Проследим, как УВИГ U_M обрабатывает произвольный запрос x. Сначала x проходит из корня графа в одну из концевых вершин графа U_1, затем из нее до одной из концевых вершин графа U_2 и так далее до некоторой концевой вершины U_M. Оценим сверху количество операций, которые вычисляются в процессе перехода запроса x из концевой вершины графа U_i в концевую вершину U_{i+1} $(0 \leqslant i \leqslant M-1)$. Пусть концевая вершина U_i, в которую проходит запрос x есть α_j^i. Количество вычисленных на исследуемом этапе операций не превышает тогда величины $\left]\log_2(|V_j^i| + 1)\right[+ \left]\log_2(|W_i| + 1)\right[$. Легко видеть, что $\max_j |V_j^i|$ достигается при $j = 0$, и кроме того $V_0^i = W_i$. Получаем, что $\left]\log_2(|V_j^i| + 1)\right[+ \left]\log_2(|W_i| + 1)\right[\leqslant 2\left]\log_2(|W_i| + 1)\right[$. Оценим теперь время работы U_M в худшем случае: $\hat{T}(U_M) \leqslant \sum_{i=0}^{M-1} 2\left]\log_2(|W_i| + 1)\right[\leqslant \sum_{i=0}^{M-1} 2\left]\log_2(\frac{|V|}{2^i} + 1)\right[$. Но по (18) при $0 \leqslant i \leqslant M-1$ выполнено $|V| \geqslant 2^i$. Таким образом, получаем $\hat{T}(U_M) \leqslant \sum_{i=0}^{M-1} 2\left]\log_2(2\frac{|V|}{2^i})\right[< \sum_{i=0}^{M-1} 2(\log_2(2\frac{|V|}{2^i}) + 1) \leqslant 2\sum_{i=0}^{M-1}(\log_2 |V| - i + 2) \leqslant 2(M\log_2 |V| + 2M - \sum_{i=0}^{M-1} i) = 2M(\log_2 |V| + 2) - M(M-1) = M(2\log_2 |V| - M + 5)$. Из (16) следует, что $M > \log_2 |V|$, а значит $2\log_2 |V| - M < \log_2 |V|$. Продолжая предыдущую оценку, имеем $\hat{T}(U_M) < M(\log_2 |V| + 5)$. Воспользовавшись неравенством (29), получаем окончательную оценку $\hat{T}(U_M) < (1 + \log_2 |V|)(\log_2 |V| + 5) = \log_2^2 |V| + 6\log_2 |V| + 5$. Лемма доказана.

5. Доказательство основных результатов и выводы

Доказательство теоремы 1. Оценки для $\mathcal{Q}_{\mathcal{D}}^{uvig}(k, S_{int1}, \mathcal{F}_{int1}^{1})$ являются следствиями из лемм 1 и 3. Оценки для $\mathcal{Q}^{uvig}(k, S_{int1}, \mathcal{F}_{int1}^{1})$ следуют из лемм 4 и 6.

Доказательство теоремы 2. Требуемые оценка следуют из лемм 2 и 5.

Для рассматриваемой модельной задачи — вычисления мощности ответа задачи одномерного интервального поиска. Существует простой неформальный алгоритм решения, который может быть без каких-либо затруднений формализован в терминах ВИГ, что и производится в ходе доказательства леммы 2. Однако, реализовать этот алгоритм в терминах УВИГ с сохранением линейного (по отношению к размеру библиотеки) роста объема графа не получается. Теорема 2 утверждает, что объем древовидного УВИГ, решающего нашу задачу должен расти не менее, чем квадратично по отношению к размеру библиотеки. Для произвольных УВИГ, впрочем, нижнюю оценку объема, превышающую линейную функцию так и не удалось получить. Точно неизвестно, можно ли построить УВИГ, объемы которых растут линейно по отношению к размеру библиотеки. Так или иначе, переход от древовидных графов к произвольным дает выигрыш с точки зрения объема графа. Но даже если выяснится, что УВИГ с линейно растущим объемом можно построить, использование ВИГ для описания алгоритмов решения ВЗИП представляется более предпочтительным, так как формализация простого алгоритма решения модельной задачи в терминах УВИГ требует квадратичного объема памяти, что не является адекватной характеристикой объема памяти, требуемого для реализации данного алгоритма.

В большинстве случаев алгоритмы решения ВЗИП должны хранить результаты промежуточных вычислений в процессе поиска ответа. УВИГ не имеет памяти и вынужден хранить эти результаты в своей структуре, что ведет к увеличению объема графа. ВИГ имеет память, в которой может хранить результаты промежуточных вычис-

лений. Именно поэтому ВИГ может описывать алгоритмы решения ВЗИП более адекватно.

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. М.: ФИЗМАТЛИТ, 2002.

[2] Пивоваров А. П. Поиск представителя в задаче о метрической близости // Интеллектуальные системы. Т. 12, вып. 1–4. 2008. С. 333–350.

[3] Gasanov E. E. On Functional Complexity of Two-dimensional Manhattan Metrics Closeness Problem // Emerging Database Research In East Europe. Proceedings of the pre-conference workshop of VLDB. 2003. P. 51–56.

[4] Gasanov E. E. On a one-dimensional interval search problem // Discrete Math. Appl. Vol. 5. No. 2. 1995. P. 117–135.