Сложность поиска идентичных объектов в случайных базах данных

Н. С. Кучеренко

В работе рассматриваются алгоритмы решения задачи поиска идентичных объектов на интервале $(0, 1)$, использующие только операции сравнения. Показано, что для любой задачи из этого класса существует оптимальный по сложности алгоритм, схема которого имеет вид дерева. Рассмотрено поведение сложности оптимального алгоритма для двух естественных классов задач. В первом классе запросы к базе данных представляют собой случайную величину с произвольным непрерывным распределением, а сама база данных — равномерную сетку. Во втором классе база данных — случайный равномерно распределенный вектор, и запросы также распределены равномерно. Установлено, что в обоих классах сложность алгоритма оптимального поиска близка к сложности стандартного логарифмического поиска.

1. Введение

Теория хранения и поиска информации является важным разделом теории интеллектуальных систем. Одним из ключевых объектов этой теории является информационный граф (ИГ) [1] — управляющая система, которая позволяет рассматривать имеющиеся модели данных и задачи, связанные с ними, с более общих позиций.

Задача поиска идентичных объектов в том или ином виде встречается во всех информационных системах и базах данных, она состоит в поиске в информационном массиве объекта, идентичного объекту-запросу. В данной работе в рамках информационно-графовой модели данных рассматривается задача поиска идентичных объектов

(ЗПИО) на интервале $(0, 1)$. Практически такая задача возникает, когда на объектах информационного массива введен линейный порядок, алфавитный или числовой, и объекты рассортированы в соответствии с этим порядком.

Идея представлять алгоритмы поиска, использующие только операции сравнения, с помощью деревьев возникла в 50-е годы 20-го века. В 1959 году Э. Н. Гильберт и Э. Ф. Мур исследовали такие алгоритмы с точки зрения сложности, которая характеризует среднее время работы алгоритма, и показали, что можно построить оптимальный, в смысле введенной сложности, алгоритм поиска за $O(n^3)$ шагов (n — мощность информационного массива или базы данных), и привели оценки сложности такого алгоритма. В 1971 году Д. Э. Кнут показал, что построение этого алгоритма поиска можно улучшить до порядка $O(n^2)$ шагов. Дальнейшие упрощения методов построения этого алгоритма были произведены в 1977 А. М. Гарсия и М. Л. Вочем, их метод позволяет построить оптимальный алгоритм поиска за $O(n \cdot \log_2 n)$ шагов.

В данной работе алгоритмы, решающие ЗПИО и использующие только операции сравнения, исследуются с более общих позиций. В общем случае информационный граф, моделирующий алгоритм поиска, может иметь не древовидную структуру. Однако в работе показано, что для любой задачи поиска идентичных объектов на интервале $(0, 1)$ можно построить оптимальный ИГ, являющийся деревом.

С помощью информационного графа всегда можно реализовать логарифмический поиск, сложность которого для любой ЗПИО будет равна логарифму от мощности базы данных. В то же время можно привести пример задач, для которых сложность оптимального ИГ может быть как константная, так и совпадать со сложностью логарифмического поиска.

Автором проведено исследование поведения сложности оптимального ИГ для двух естественных классов задач. В первом классе база данных — равномерная сетка на интервале $(0, 1)$, а запрос произвольная случайная величина, которая имеет плотность. Для этого класса задач показано, что сложность оптимального ИГ с ростом мощности базы данных асимптотически ведет себя как логарифмический поиск.

Во втором классе, предполагается что база данных — случай-

ный вектор, компоненты которого равномерно распределены, запросы также равномерно распределены на интервале $(0,1)$. Показано, что в среднем на этом классе сложность оптимального ИГ незначительно отличается от сложности логарифмического поиска.

Результаты данной работы анонсированы в [2].

2. Основные понятия и результаты

В данной работе исследуются алгоритмы поиска, использующие только операции сравнения. Ввиду этого, будет использоваться упрощенная версия терминологии из книги [1].

Задача поиска идентичных объектов (ЗПИО) — это четверка $((0,1), V, \rho_=, f(x))$, где $(0,1)$ — множество запросов, $V = (y_1, y_2, \ldots, y_n)$ — конечный набор точек интервала $(0,1)$, элементы которого упорядочены по возрастанию. Множество V называется библиотекой, а элементы библиотеки — записями. $\rho_=$ — отношение равенства на множестве $(0,1) \times (0,1)$. Предполагается, что запрос $x \in (0,1)$ — случайная величина на интервале $(0,1)$, распределение которой задается с помощью функции плотности $f(x)$. Отношение $\rho_=$ задает функцию ответов $J(x)$, определенную на множестве запросов $(0,1)$, следующим образом

$$J(x) = \begin{cases} \{y_i\}, & \text{если } \exists i \in [1..n] : y_i \rho_= x, \\ \varnothing, & \text{иначе.} \end{cases}$$

Для реализации функции ответов используются операции сравнения вещественных чисел из интервала $(0,1)$, которые заданы в виде функции $g(x,y)$

$$\forall y, x \in (0,1) \quad g(y,x) = \begin{cases} 1, & \text{если } y < x, \\ 2, & \text{если } y = x, \\ 3, & \text{если } y > x. \end{cases}$$

Функция $g_a(x) = g(a,x)$, $a \in (0,1)$ называется переключателем. Определим базовое множество переключателей $G = \{g_a(x) | a \in (0,1)\}$. Над базовым множеством G строится *информационный граф* (ИГ), который хранит в себе схему вызова переключателей из G.

Информационный граф — это связный ориентированный конечный граф без кратных ребер и петель, вершины и ребра которого размечены следующим образом: выделены два подмножества вершин P и L, так что $P \cap L = \varnothing$. Вершины множества P называются *переключательными*, а вершины множества L — *листьями*. Ребра, исходящие из переключательных вершин, называются также *переключательными*. Среди переключательных вершин выделена одна вершина v_0, которая называется *корневой*. Переключательным вершинам сопоставлены переключатели из G. Переключатель $g_a(x)$, сопоставленный переключательной вершине v, называется ее нагрузкой и обозначается $[v]_P = g_a(x)$. Листьям сопоставляются записи библиотеки. Это сопоставление обозначается $[v]_L = y_i, v \in L$; y_i называется нагрузкой листа v. Переключательным ребрам сопоставляются элементы множества $\{\Lambda, 1, 2, 3\}$, при этом из переключательной вершины не выходит двух ребер с одинаковыми номерами. Если ребру (v_{i_1}, v_{i_2}) сопоставлен номер $i \in \{1, 2, 3\}$, то номер i называется нагрузкой ребра (v_{i_1}, v_{i_2}) и обозначается $[(v_{i_1}, v_{i_1})] = i$.

Определим *функционирование* информационного графа. Переключательное ребро (v_{i_1}, v_{i_2}), которому приписан номер i, проводит запрос $x \in X$, если переключатель, приписанный началу этого ребра, принимает значение i на запросе x, то есть если $[(v_{i_1}, v_{i_2})] = [v_{i_1}]_P(x)$. Не переключательное ребро не проводит запрос, также не проводит запрос переключательное ребро, которому был сопоставлен символ Λ. Ориентированная цепочка ребер проводит запрос $x \in X$, если каждое ребро цепочки проводит запрос x. Запрос $x \in X$ проходит в вершину v ИГ, если существует ориентированная цепочка, ведущая из корневой вершины в вершину v, которая проводит запрос x. Запись y, приписанная листу v', называется *результатом функционирования ИГ на запросе* x, если запрос x проходит в лист v'. *Результат функционирования ИГ* U — это функция $R_U(x)$, которая каждому запросу x сопоставляет множество всех записей, которые являются результатом функционирования ИГ на запросе x.

Информационный граф U *решает* ЗПИО $((0,1), V, \rho_=, f(x))$, если результат функционирования ИГ U совпадает с функцией ответов $J(x)$, то есть $R_U(x) = J(x), \forall x \in (0,1)$.

Если в информационном графе U удалить «нефункциональные»

элементы, а именно, не переключательные ребра и ребра, которым сопоставлен символ Λ, а затем оставить только одну компоненту связности, которой принадлежит корневая вершина, то результат функционирования $R_U(x)$ не изменится. Поэтому далее рассматриваются ИГ с точностью до этих операций удаления.

Сложностью ИГ U на запросе x называется число

$$T(U,x) = \sum_{v \in P} \varphi_v(x),$$

где P — множество переключательных вершин, а $\varphi_v(x)$ — предикат, истинный на x, если x проходит в вершину v, и ложный в противном случае. Предикат $\varphi_v(x)$ называется *функцией фильтра вершины v*. Величина $T(U,x)$ как функция от x является измеримой случайной величиной [1]. *Сложностью ИГ* $T(U)$ называется математическое ожидание величины $T(U,x)$, которое можно записать в виде

$$T(U) = \int_0^1 T(U,x) f(x) dx.$$

Обозначим через S_I множество ИГ, которые решают ЗПИО $I = ((0,1), V, \rho_=, f)$. Сложностью ЗПИО I называется величина $T(I) = \inf_{U \in S_I} T(U)$. Информационный граф, на котором достигается инфимум, называется *оптимальным* информационным графом (ОИГ).

ИГ называется древовидным, если

1) Основанием орграфа ИГ является дерево.
2) Полустепень захода вершины, выбранной корнем, равна нулю. Из корневой вершины остальные вершины орграфа достижимы.
3) Все вершины имеют полустепень исхода либо 0, либо 3. При этом, если из вершины выходит 3 дуги, то хотя бы одна из них заходит в висячую вершину.
4) У каждой внутренней вершины исходящие дуги пронумерованы таким образом, что дуга с номером 2 ведет в висячую вершину.
5) Все внутренние вершины являются переключательными. Если переключательной вершине v сопоставлен переключатель g_a, и

a равняется некоторой записи библиотеки y, то висячей вершине, в которую ведет дуга с номером 2, сопоставлена эта запись y.

Автором была исследована структура оптимальных ИГ для задачи поиска идентичных объектов на интервале $(0, 1)$.

Теорема 1. *Для любой задачи поиска идентичных объектов* $I = ((0,1), V, \rho_=, f), V = \{y_1, y_2, \ldots, y_n\}$ *существует оптимальный древовидный информационный граф, содержащий* n *переключательных вершин, которым взаимно однозначно сопоставлены* n *переключателей* $\{g_{y_1}, g_{y_2}, \ldots, g_{y_n}\}$.

Этот результат позволяет искать оптимальные информационные графы в конечном множестве древовидных ИГ D_I, которые отвечают условиям теоремы. Оптимальный ИГ из множества D_I можно построить с помощью алгоритма Гарсия-Вача, описанного в книге [3]. Этот алгоритм определяет структуру искомого ИГ, вычисляя порядка $n \cdot \log_2 n$ арифметических операций.

С помощью информационной графа всегда можно реализовать логарифмический поиск, который имеет сложность равную $]\log_2(n+1)[$, $n = |V|$. Сложность же оптимального ИГ может быть равна как константе, так и $]\log_2(n+1)[$, $n = |V|$, в зависимости от конкретной задачи. Поэтому представляет интерес изучение поведения сложности ИГ на классах задач.

В работе рассмотрен класс ЗПИО

$$\{I_n^f = ((0,1), V, \rho, f), V = \left(\frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}\right) | n \in \mathbb{N}\}.$$

Для этого класса задач верна теорема.

Теорема 2. *Для любой функции плотности распределения* $f(x)$ *при* $n \to \infty$ *выполнено* $T(I_n^f) \sim \log_2 n$.

Также рассмотрен класс задач

$$\Upsilon_n = \{I(V) = ((0,1), V, \rho_=, \chi_{(0,1)}) : V = (y_{(1)}, y_{(2)}, \ldots, y_{(n)})\},$$

где $y_{(1)}, y_{(2)}, \ldots, y_{(n)}$ — вариационный ряд, составленный из независимых равномерно распределенных на интервале $(0,1)$ случайных величин $y_1, y_2, \ldots, y_n$; $\chi_{(0,1)}(x) \equiv 1$ на интервале $(0,1)$. Если через $T_n(V)$ обозначить сложность ЗПИО $I(V)$ из класса Υ_n, то как функция от V величина $T_n(V)$ является случайной. Верна следующая теорема.

Теорема 3. *Пусть* $\mathbf{M}_V(T_n(V))$ — *математическое ожидание* $T_n(V)$, *тогда для любого* $n \in \mathbb{N}$ *выполнено*

$$]\log_2(n+1)[\geqslant \mathbf{M}_V(T_n(V)) \geqslant \log_2(n+1) - \frac{1-\gamma_{n+1}}{\ln 2},$$

где $\gamma_n = \sum_{i=1}^{n} \frac{1}{i} - \ln n$.

γ_n — сходящаяся последовательность, члены которой положительны. Предел этой последовательности называется постоянной Эйлера $\gamma = 0{,}57\ldots$

3. Доказательство теоремы 1

Порядок просмотра вершин в процессе функционирования ИГ на произвольном запросе x представляет из себя ормаршрут вида

$$\left(v_{i_0}, \left(v_{i_0}, v_{i_1}\right), v_{i_1}, \ldots, \left(v_{i_{l-1}}, v_{i_l}\right), v_{i_l}\right),$$

такой что

1) v_{i_0} — корневая вершина, вершины $v_{i_0}, v_{i_1}, \ldots, v_{i_{l-1}}$ — различные и являются переключательными;
2) ормаршрут проводит запрос x, то есть

$$[v_{i_j}]_P(x) = [(v_{i_j}, v_{i_{j+1}})], \quad j = 0, \ldots, l-1;$$

3) вершина v_{i_l} удовлетворяет одному из трех условий
   а) v_{i_l} — не является переключательной вершиной,
   б) v_{i_l} — переключательная вершина и $v_{i_l} \in \{v_{i_0}, \ldots, v_{i_{l-1}}\}$,
   в) v_{i_l} — переключательная вершина и не существует ребра, исходящего из этой вершины, который проводит запрос x.

Обозначим такой ормаршрут через m_x и будет говорить что запрос x проходит по маршруту m_x в процессе функционирования ИГ. *Длиной* ормаршрута $|m_x|$ назовем число различных переключательных вершин ормаршрута.

Два запроса назовем *эквивалентными*, если ормаршруты, по которым проходят эти запросы в процессе функционирования ИГ одинаковые.

Лемма 1. *Для любой ЗПИО* $I = ((0,1), V, \rho_=, f)$ *и для любого ИГ* $U \in S_I$ *можно построить ИГ* $U' \in D_I$, *такой что* $T(U') \leqslant T(U)$.

Доказательство. Если запрос x проходит по ребру $\left(v_{i_j}, v_{i_{j+1}}\right), j = 1, \ldots, k-1$, где $[v_{i_j}] = g_{a_j}(x), [\left(v_{i_j}, v_{i_{j+1}}\right)] = m_j$, то запрос удовлетворяет одному из трех неравенств вида

$$\begin{aligned} x < a_j, &\quad \text{если } m_j = 1, \\ x = a_j, &\quad \text{если } m_j = 2, \\ x > a_j, &\quad \text{если } m_j = 3. \end{aligned}$$

А если запрос x в процессе функционирования ИГ U проходит по всему ормаршруту

$$\left(v_{i_0}, \left(v_{i_0}, v_{i_1}\right), v_{i_1}, \ldots, \left(v_{i_{l-1}}, v_{i_l}\right), v_{i_l}\right),$$

то он удовлетворяет системе из l неравенств, задаваемых l ребрами этого ормаршрута. Обозначим эту систему через L_x При этом если запрос x' удовлетворяет системе L_x, то он тоже проходит поэтому ормаршруту в процессе функционирования ИГ U. Таким образом, класс эквивалентности запроса x — это решения системы L_x. Заметим, что решением таких систем может быть либо точка, либо интервал. Поскольку орграф ИГ конечный, то множество ормаршрутов, по которым проходят запросы в процессе его функционирования, конечно. Таким образом, классы эквивалентности на множестве запросов $(0,1)$ выглядят как чередование точек и интервалов

$$\{(0, z_1), \{z_1\}, (z_1, z_2), \{z_2\}, \ldots, (z_{r-1}, z_r), \{z_r\}, (z_r, 1)\}.$$

Обозначим через max_U длину самого длинного ормаршрута, по которому проходит запрос.

Для произвольного $l, 1 \leqslant l \leqslant max_U$ обозначим через X_l множество запросов, длина ормаршрута которых не меньше l. Заметим, что

$$X_1 = (0,1) \subseteq X_2 \subseteq \cdots \subseteq X_{max_U} \neq \varnothing.$$

При фиксированном $l, 1 \leqslant l \leqslant max_U$ два запроса x_1 и x_2 из множества X_l назовем (l)-подобными, если первые l вершин их ормаршрутов совпадают. Отношение (l)-подобия задает на множестве X_l отношение эквивалентности. Классы эквивалентности, на которые разбивается множество X_l назовем классами (l)-подобия.

Все запросы множества $X_1 = (0,1)$ (1)-подобны. Рассмотрим как изменяются классы (k)-подобия от $k = 1$ до $k = max_U$. Сопоставим классам (k)-подобия обозначение E_k^j, где значение нижнего индекса k будет означать, что класс E_k^j — есть класс (k)-подобия, а значение верхнего индекса j есть универсальный номер этого класса, такой что для всех классов (s)-подобия при всех значениях $s = 1, \ldots, max_U$ значение верхних индексов различные.

Рассмотрим классы (1)-подобия. Такой класс всего один, он содержит все запросы из множества $(0,1)$, обозначим его E_1^0. Дальнейшую нумерацию опишем индуктивно. Допустим, что просмотрены и пронумерованы классы (k)-подобия, $k = 1, \ldots, s$, занумеруем классы $(s+1)$-подобия.

Рассмотрим произвольный класс (s)-подобия E_s^j. Обозначим первые s вершин ормаршрутов, по которым проходят запросы из этого класса, через $(v_1, \ldots, v_s)$. Если v_s — не переключательная вершина, то запросы из класса E_s^j не содержатся во множестве X_{s+1}. Рассмотрим ситуацию, когда v_s — переключательная вершина с нагрузкой $g_a(x)$. Обозначим через A_{s+1}^{3j+i} все запросы из класса E_s^j на которых переключатель $g_a(x)$ принимает значение $i, i \in \{1,2,3\}$.

Рассмотрим следующие случаи

1) Множество E_s^j является интервалом (u,w), рассмотрим следующие подслучаи.
   а) Если $a \in (u,w)$, тогда справедливо

$$A_{s+1}^{3j+1} = (u,a), A_{s+1}^{3j+2} = \{a\}, A_{s+1}^{3i+3} = (a,w).$$

Если из вершины v_s исходит ребро с номером i, то A_{s+1}^{3j+i} является классом $(s+1)$-подобия. Обозначим этот класс через E_{s+1}^{3j+i}.

б) $a \notin (u, w)$, в этом случае только одно из множеств $A_{s+1}^{3j+i}, i = 1, 3$ не пустое. Если из вершины v_s исходит ребро с номером i, то $A_{s+1}^{3j+i} = (u, w)$ является классом $(s+1)$-подобия. Обозначим этот класс через E_{s+1}^{3j+1}.

2) Множество $E_s^j = \{u\}$. Возможны следующие подслучаи.

а) $a = u$, только множество A_{s+1}^{3i+2} не пустое и $A_{s+1}^{3i+2} = \{u\}$. И если из вершины v_s исходит ребро с номером 2 то множество A_{s+1}^{3j+2} является классом $(s+1)$-подобия, обозначим этот класс через E_{s+1}^{3j+1}.

б) $a \neq u$, тогда запрос u не содержится в множестве X_{s+1}.

Просмотрев таким образом все классы (s)-подобия мы просмотрим и пронумеруем все классы $(s+1)$-подобия. Процесс остановим на значении s равном max_U.

С помощью введенной нумерация можно построить древовидный ИГ U'', у которого длина самого длинного ормаршрута, по которому проходит запрос, равна max_U, и при любом $k = 1, \ldots, max_U$ множество запросов, у которых длина ормаршрута в ИГ U'' не меньше k, равно множеству запросов X_k. Множество X_k в ИГ U'' разбивается на классы (k)-подобия так же, как в ИГ U. При выполнении этих условий сложности ИГ U'' и ИГ U равны, так как любой запрос $x \in (0, 1)$ проходит в ИГ U и ИГ U'' по ормаршрутам одинаковой длины.

Опишем построение древовидного ИГ U'' с помощью индукции.

На первом шаге имеем орграф (V_1, E_1), где $V_1 = \{v_1^0\}$, v_1^0 — корневая вершина, множество ребер E_1 пустое. Вершине v_1^0 сопоставим класс (1)-подобия E_1^0, и считаем ее вершиной первого уровня.

Пусть построен орграф (V_s, E_s). Построим орграф (V_{s+1}, E_{s+1}).

Возьмем вершину v_s^i уровня s. Пусть ей сопоставлен класс (s)-подобия E_s^j. Среди классов $(s+1)$-подобия просмотрим классы с верхними индексами $3j+1, 3j+2, 3j+3$.

Рассмотрим следующие случаи, которые соответствуют случаям при нумерации

1) Множество E_s^j является интервалом (u, w).
   а) Если
   $$A_{s+1}^{3j+1} = (u, a), A_{s+1}^{3j+2} = \{a\}, A_{s+1}^{3i+3} = (a, w),$$
   и среди классов $(s+1)$-подобия есть класс E_{s+1}^{3j+i}, тогда ко множеству вершин добавляем вершину v_{s+1}^{3j+i}, ко множеству ребер ребро (v_s^j, v_{s+1}^{3j+i}). Вершине v_s^j ставим в соответствие переключатель $g_a(x)$, ребру (v_s^j, v_{s+1}^{3j+i}) номер i. Вершине v_{s+1}^{3j+i} сопоставим класс E_{s+1}^{3j+i}, вершину v_{s+1}^{3j+i} считаем вершиной $s+1$-го уровня.
   б) Если только одно из множеств $A_{s+1}^{3j+i}, i = 1, 3$ не пустое и среди классов $(s+1)$-подобия есть класс E_{s+1}^{3j+1}, то к множеству вершин добавляем вершину v_{s+1}^{3j+1}, ко множеству ребер ребро (v_s^j, v_{s+1}^{3j+1}).
   Вершине v_s^j ставим в соответствие переключатель $g_w(x)$, ребру (v_s^j, v_{s+1}^{3j+i}) номер 1. Вершине v_{s+1}^{3j+1} сопоставим класс E_{s+1}^{3j+1}, и считаем ее вершиной уровня $s+1$.
2) Множество $E_s^j = \{u\}$. Если множество A_{s+1}^{3i+2} не пустое и есть запись библиотеки y_i, равная u, то вершина v_s^j называется листовой и ей сопоставляется запись y_i.
3) Если случай не удовлетворяет ни одному из перечисленных условий, то никаких построений не производится.

Процесс построения заканчиваем при $s = max_U$. Заметим, что пункт 2 обеспечивает достаточное условие того, что построенный древовидный ИГ U'' решает ЗПИО I.

Далее описывается процесс изменения только что построенного древовидного ИГ, так чтобы переключательным вершинам были сопоставлены переключатели из множества

$$G(V) = \{g_{y_1}, g_{y_2}, \ldots, g_{y_n}\},$$

при этом ИГ остается древовидным и решает ЗПИО I.

Начинаем просматривать классы (k)-подобия с номера max_U, пока во множестве классов не найдем класс $\{y_i\}, y_i \in V$. Обозначим это

значение k через k_0. Если есть классы (k)-подобия, где $k > k_0$, то в ИГ удалим все вершины которым соответствовали эти классы при построении и исходящие из них ребра. Получившийся древовидный ИГ решает ЗПИО, поскольку вершины с листьями не удалялись.

Теперь будем просматривать переключательные вершины ИГ, начиная с уровня k_0, которым соответствует переключатель не из множества $G(V)$. Как только встречается такая вершина v, в ИГ делаются локальные преобразования. Предположим, что мы просмотрели и заменили переключатели на всех уровнях от k_0 до уровня $s+1$ включительно. Покажем, как просматривать и заменять переключатели на уровне s. Если вершина находится на уровне s и ей сопоставлен класс (s)-подобия вида $\{u\}$, то из вершины удаляем все исходящие ребра. А если $u \in V$, то вершине приписываем запись, равную u, иначе ничего вершине не приписываем.

Если вершина v находится на уровне s и ей сопоставлен класс (s)-подобия (u, w), то рассмотрим несколько случаев.

1) Если отцу вершины v сопоставлен класс $(s-1)$-подобия (u, w) и переключатель $g_b, b \notin (u, w)$ то сама вершина, как мы определили при построении, имеет верхний индекс вида $3 \cdot j + 1$. В этом случае все поддерево с корнем в вершине v переносится на на место ее отца, и соответственно меняется нумерация.
2) Обозначим отца вершины v через v'. Если вершине v' сопоставлен класс $(s-1)$-подобия (u, w) и переключатель $g_b, b \in (u, w)$, то здесь опять может быть два подслучая.
   а) Первый, когда переключатель g_a вершины v, такой что $a \notin (u, w)$. В этом случае заменяем вершину v на ее сына, в которого ведет ребро под номером 1. Так как сын с уровня $s+1$, то ему сопоставлен нужный переключатель.
   б) Второй подслучай, когда $a \in (u, w)$ надо разобрать подробнее. Здесь снова появляются подслучаи в зависимости от того, как расположены записи библиотеки в интервале (u, w).
      – В отрезке (u, w) нет точек библиотеки, тогда удаляем все поддерево с корнем в вершине v.
      – В отрезке (u, w) есть точки библиотеки и все они нахо-

дятся слева от a. В этом случае выберем ближайшую к a точку библиотеки из (u, w), обозначим ее y_j, заменим переключатель вершины g_a на переключатель g_{y_j}, удалим все ребра, кроме ребра под номером 1, и присоединим новое ребро с номером 2 обозначим его (v, v''), где v'' листовая вершина с записью y_j.
После этого преобразования запрос x, равный записи y_j, пройдет в листовую вершину с записью y_j, поэтому получившийся ИГ будет решать ЗПИО I. Сложность ИГ на запросах из множества $(u, y_j]$ не увеличилась, а на остальных запросах не изменилась. Таким образом сложность получившегося ИГ не увеличилась.

– В отрезке (u, w) есть точки библиотеки и все они находятся справа от a. В этом случае выберем ближайшую к a точку библиотеки из (u, w), обозначим ее y_j, заменим переключатель вершины g_a на переключатель g_{y_j}, удалим все ребра, кроме ребра под номером 3, и присоединим новое ребро с номером 2 обозначим его (v, v''), где v'' листовая вершина с записью y_j.
После этого преобразования запрос x, равный записи y_j, пройдет в листовую вершину с записью y_j, поэтому получившийся ИГ будет решать ЗПИО I. Сложность ИГ на запросах из множества $[y_j, w)$ не увеличилась, а на остальных запросах не изменилась. Таким образом сложность получившегося ИГ не увеличилась.

– В отрезке (u, w) есть точки библиотеки и они находятся как по правую сторону от a, так и по левую. Перед тем как делать изменения в ИГ нужно более тщательно изучить подразбиения (u, w) в классах $(k+1)$-подобия и ниже. Пусть самая ближняя к a слева запись библиотеки y_j, а справа — y_{j+1}. Надо выбрать на какой из двух переключателей g_{y_j} или $g_{y_{j+1}}$ заменить переключатель вершины v. Сравним длину ормаршрута запросов из интервала (y_j, a) и длину ормаршрута запросов из интервала (a, y_{j+1}). Первый ормаршрут обозначим через m_1, второй через m_2.

A. Если $|m_1| \leqslant |m_2|$, заменяем переключатель $g_a(x)$ вершины v на переключатель $g_{y_{j+1}}(x)$. Если из вершины v исходило ребро под номером 2, это ребро удаляем и вместо него добавляем новое ребро, обозначим его (v, v''), где v'' — листовая вершина с записью y_{j+1}. Рассмотрим получившийся ИГ. Запрос, равный записи y_{j+1}, проходит в лист с записью y_{j+1}, поэтому получившийся ИГ решает ЗПИО I. Сложность ИГ U'' при проделанных преобразованиях не изменилась на запросах из множества $(0,1) \setminus (a, y_{j+1}]$. Длины ормаршрутов запросов из множества $(a, y_{j+1}]$ не стали больше, поэтому сложность ИГ на этих запросах не увеличилась.

B. Если $|m_1| > |m_2|$, заменяем переключатель $g_a(x)$ вершины v на переключатель $g_{y_j}(x)$. Если из вершины v исходило ребро под номером 2, это ребро удаляем и вместо него добавляем новое ребро, обозначим его (v, v''), где v'' — листовая вершина с записью y_j. Рассмотрим получившийся ИГ. Запрос, равный записи y_j, проходит в лист с записью y_j, поэтому получившийся ИГ решает ЗПИО I. Сложность ИГ U'' при проделанных преобразованиях не изменилась на запросах из множества $(0,1) \setminus [y_j, a)$. Длины ормаршрутов запросов из множества $[y_j, a)$ не стали больше, поэтому сложность ИГ на этих запросах не увеличилась.

После просмотра всех уровней ИГ U'', мы получим древовидный ИГ U''', в котором всем переключательным вершинам сопоставлены переключатели вида $g_{y_i}, y_i \in V$. Древовидный ИГ U''' был построен так, что для любой не корневой вершины существует только один путь из корня в эту вершину, и по этот путь проводит некоторый запрос. Множество запросов, которые проходят по этому пути, может быть либо интервалом вида $(y_i, y_{i+s}), s \geqslant 1, y_i, y_{i+s} \in V$, либо точкой $y_i, y_i \in V$, так как переключательным вершинам этого пути сопоставлены только переключатели вида $g_{y_i}, y_i \in V$.

Преобразуем древовидный ИГ U''' в ИГ U', каждая переключательная вершина v которого удовлетворяет следующим свойствам

1) по пути из корневой вершины в переключательную вершину v проходит множество запросов вида $(y_i, y_{i+s}), s \geqslant 2$;
2) переключательной вершине v сопоставлен переключатель g_{y_j}, такой что $y_j \in (y_i, y_{i+s})$.

Переключательные вершины ИГ U''', которые не удовлетворяют этим свойствам, назовем *неправильными*. Корневая вершина не может являться неправильной по определению. Покажем как уменьшить число неправильных вершин на единицу.

Рассмотрим произвольную неправильную вершину v. Через E обозначим множество запросов, которые проходят по пути из корневой вершины в вершину v. Рассмотрим несколько случаев

1) Множество E имеет вид (y_i, y_{i+1}). В этом случае вершину v и все поддерево с корнем в этой вершине можно удалить. Сложность ИГ при этом не увеличиться. Ормаршруты запросов, которые равны записям библиотеки, не изменятся, значит получившийся ИГ будет решать ЗПИО I.
2) Множество E имеет вид $\{y_i\}$. В этом случае из вершины v удалим все исходящие ребра, объявим ее листовой и сопоставим ей запись y_i. Сложность ИГ не увеличится. Ормаршруты запросов, которые равны записям из множества $V \setminus y_i$, не изменяться, а запрос, равный записи y_i, пройдет в листовую вершину v с записью y_i, следовательно получившийся ИГ будет решать ЗПИО I.
3) Множество E имеет вид $(y_i, y_{i+s}), s \geqslant 2$, но переключатель g_{y_j} вершины v таков, что $y_j \notin (y_i, y_{i+s})$. В этом случае заменим вершину v ее отцом v', то есть сделаем следующие преобразования: удалим ребро (v, v') и вершину v', вместо вершины v' подсоединим все поддерево с корнем в вершине v и сопоставим вершине v переключатель вершины v'. Сложность получившегося ИГ не изменяться. Из ормаршрутов запросов, равных записям отрезка (y_i, y_{i+s}), удалится всего одно ребро (v, v'), при этом система которая будет определяться новыми ормаршрутами будет эквивалентна старой. Следовательно запросы из отрезка (y_i, y_{i+s})

пройдут в соответствующие им листовые вершины. Ормаршруты запросов, равных другим записям, не изменяться. Таким образом получившийся ИГ будет решать ЗПИО I.

Просмотрев таким образом все неправильные вершины ИГ U''', мы получим ИГ U', все переключательные вершины которого удовлетворяют требуемым свойствам. В ИГ U' переключатели сопоставленные разным переключательным вершинам разные. Поскольку в ИГ U' который решает ЗПИО I, запрос, равный записи библиотеки y_i должен обязательно пройти по пути из корневой вершины в переключательную вершину с переключателем g_{y_i}, множество переключателей сопоставленных всем переключательным вершинам ИГ U' должно содержать множество

$$G(V) = \{g_{y_1}, g_{y_2}, \ldots, g_{y_n}\}.$$

Значит в ИГ U' всего n переключательных вершин, которым которым взаимно однозначно сопоставлены переключатели из множества $G(V)$. ИГ U' искомый. Лемма доказана.

Доказательство теоремы 1. В силу леммы верно следующее тождество

$$\inf_{U \in S_I} T(U) = \inf_{U \in D_I} T(U).$$

Множество D_I конечно, поэтому инфимум достигается на ИГ из множества D_I. Теорема доказана.

4. Доказательство теоремы 2

Обозначим через p_i вероятность того, что запрос x принадлежит интервалу (y_i, y_{i+1}),

$$p_i = \mathbf{P}\left(x \in (y_i, y_{i+1})\right),$$

где $i = 0, \ldots, n$, $y_0 = 0$, $y_{n+1} = 1$.

Лемма 2. *Пусть ЗПИО $I = ((0,1), V, \rho_=, f)$, ИГ $U \in D_I$, тогда сложность ИГ U можно представить в виде линейной комбинации*

$$T(U) = \sum_{i=0}^{n} a_i \cdot p_i,$$

где $a_i \in \mathbb{N}$.

Доказательство. ИГ U принадлежит множеству D_I, поэтому он имеет n переключательных вершин с переключателями

$$\{g_{y_1}, g_{y_2}, \ldots, g_{y_n}\}, \text{ где } y_1, \ldots, y_n \text{ записи библиотеки } V.$$

Поскольку ИГ U решает ЗПИО I, классы эквивалентности запросов для ИГ U имеют вид

$$(0, y_1), \{y_1\}, (y_1, y_2), \{y_2\}, \ldots, (y_{n-1}, y_n), \{y_n\}, (y_n, 1).$$

Запросы из одного класса эквивалентности имеют одинаковую сложность, следовательно $T(U, x)$ как функция от x постоянна на классах эквивалентности и имеет вид

$$T(U, x) = \sum_{i=0}^{n} a_i \cdot \chi_{(y_i, y_{i+1})} + \sum_{i=1}^{n} b_i \cdot \chi_{\{y_i\}},$$

где $a_i, b_i \in \mathbb{N}$б a_i — сложность запросов из класса (y_i, y_{i+1}), b_i — сложность запросов из класса $\{y_i\}$, функция χ_M равна единице на подмножестве M интервала $(0, 1)$ и нулю на множестве $(0, 1) \setminus M$. Вычисляя сложность ИГ U, получим

$$T(U) = \int_0^1 T(U, x) f(x) dx = \sum_{i=0}^{n} a_i \cdot \int_0^1 \chi_{(y_i, y_{i+1})} f(x) dx = \sum_{i=0}^{n} a_i \cdot p_i.$$

Лемма доказана.

Пусть $I = ((0, 1), V, \rho, f)$, $V = \{y_1, \ldots, y_n\}$, $U \in D_I$, $T(U) = \sum_{i=0}^{n} a_i \cdot p_i, a_i \in \mathbb{N}$. Пусть $k \in \mathbb{N}, k > 1$. Обозначим через t_k число коэффициентов в линейной комбинации, значение которых меньше либо равно k.

Лемма 3. *Справедлива следующая оценка*

$$t_k \leqslant 2^k.$$

Доказательство. Скажем, что запись y_i проверена на уровне k_i, если соответствующая переключательная вершина с нагрузкой g_{y_i} находится на уровне k_i. ИГ $U \in D_I$, поэтому сложность ИГ U на всех запросах из интервала (y_i, y_{i+1}), $i = 1, \ldots, n-1$, одинакова. Обозначим эту сложность через a_i. Для нее справедливо равенство

$$a_i = \max(k_i, k_{i+1}).$$

Для интервалов вида $(0, y_1)$ и $(y_n, 1)$ верно $a_0 = k_1, a_n = k_n$. Пусть в графе U всего m записей проверено на уровне меньшем или равном k, тогда $t_k \leqslant m + 1$, поскольку на число интервалов больше $m + 1$ необходимо больше чем m записей, которые будут их концевыми точками. Значение $m + 1$ достигается в случае когда $(0, y_1), (y_1, y_2), \ldots, (y_{m-1}, y_m), (y_m, 1)$. За первые k уровней в ИГ U может быть проверена не более $2^k - 1$ запись, соответственно $t_k \leqslant 2^k$. Значение 2^k достигается в случае когда $|V| = 2^k - 1$ и ИГ U реализует бинарный поиск. Лемма доказана.

Упорядочим величины $p_0, p_1, \ldots, p_n$ по возрастанию. Обозначим этот вариационный ряд через $p_{(0)}, p_{(1)}, \ldots, p_{(n)}$.

Лемма 4. *Пусть* $I = ((0,1), V, \rho, f)$, $V = \{y_1, \ldots, y_n\}$, $U \in D_I$, *тогда*

$$T(U) > [(\log_2(n+1) - \log_2 \log_2(n+1))] \times \sum_{i=1}^{p^*} p_{(i)},$$

$$p^* = \left[(n+1)\left(1 - \frac{1}{\log_2(n+1)}\right)\right].$$

Доказательство. Положим для результатов предыдущей леммы

$$k = [(\log_2(n+1) - \log_2 \log_2(n+1))],$$

тогда

$$t_k \leqslant 2^k = 2^{[(\log_2(n+1) - \log_2 \log_2(n+1))]} \leqslant$$

$$\leqslant 2^{\log_2(n+1) - \log_2 \log_2(n+1)} = \frac{n+1}{\log_2(n+1)}.$$

Значит число коэффициентов, значение которых больше, чем k, равно

$$(n+1) - t_k \geqslant n+1 - \frac{n+1}{\log_2(n+1)} = (n+1)\left(1 - \frac{1}{\log_2(n+1)}\right).$$

Обозначим $p^* = [(n+1)\,(1-1/(log_2(n+1)))]$. Если допустить что самым большим коэффициентам соответствуют самые маленькие значения p_i, то получим нижнюю оценку для сложности ИГ U

$$T(U) = \sum_{i=0}^{n} a_i p_i = \sum_{a_i>k} a_i p_i + \sum_{a_i \leqslant k} a_i p_i > \sum_{a_i>k} k p_i > k \sum_{i=0}^{p^*} p_{(i)}.$$

Подставив значение k, получим

$$T(U) > [(\log_2(n+1) - \log_2\log_2(n+1))] \times \sum_{i=0}^{p^*} p_{(i)}.$$

Лемма доказана.

Доказательство теоремы 2. Рассмотрим задачу $I_n^f = ((0,1), V, \rho_=, f)$, $V = \{\frac{1}{n+1}, \ldots, \frac{n}{n+1}\}$. В силу предыдущей леммы для любого ИГ U из класса $D_{I_n^f}$ верно

$$T(U) > [(\log_2(n+1) - \log_2\log_2(n+1))] \times \sum_{i=0}^{p^*} p_{(i)}.$$

Поскольку в класса $D_{I_n^f}$ содержится оптимальный ИГ, для сложности задачи I_n^f верна та же нижняя оценка

$$T(I_n^f) > [(\log_2(n+1) - \log_2\log_2(n+1))] \times \sum_{i=0}^{p^*} p_{(i)}.$$

Заметим что в сумме число слагаемых $p^* = [(n+1)\,(1-1/\log_2(n+1))]$ стремится к $(n+1)$. Поскольку $p_i = \int_{y_i}^{y_{i+1}} f(x)dx$ и $y_i = \frac{i}{n+1}, i = 1, \ldots, n$ можно записать сумму в виде

$$\sum_{i=0}^{p^*} p_{(i)} = \int_A f(x)dx,$$

где множество A состоит из всех отрезков, которые входят в сумму. Каждый отрезок имеет меру $\frac{1}{n+1}$. Поэтому мера множества A равна

$$|A| = \frac{1}{n+1}(n+1)\left(1 - \frac{1}{log_2(n+1)}\right) = 1 - \frac{1}{\log_2(n+1)}.$$

В силу непрерывности интеграла Лебега $\int_A f(x)dx = 1 + f_1(n)$, где функция $f_1(n) = o(1)$ при $n \to \infty$.

$$[(\log_2(n+1) - \log_2\log_2(n+1))] \times \sum_{i=0}^{p^*} p_{(i)} \geqslant$$
$$\geqslant (\log_2(n+1) - \log_2\log_2(n+1) - 1) \cdot (1 + f_1(n)).$$

Раскрывая скобки в последнем выражении, получим

$$(\log_2(n+1) - \log_2\log_2(n+1) - 1) \cdot (1 + f_1(n)) = \log_2 n - f_2(n),$$

где функция $f_2(n)$ есть $o(\log_2 n)$ при $n \to \infty$. Получается, что сложность задачи I_n^f

$$T(I_n^f) > \log_2 n - f_2(n),$$

где функция $f_2(n)$ есть $o(\log_2 n)$ при $n \to \infty$.

С помощью ИГ всегда можно реализовать бинарный поиск, что дает верхнюю оценку сложности ЗПИО I_n^f

$$]\log_2(n+1)[\geqslant T(I_n^f).$$

Применяя получившиеся верхние и нижние оценки к классу ЗПИО $\{I_n^f, n \in \mathbb{N}\}$, получим что при $n \to \infty$

$$T(I_n^f) \sim \log_2 n.$$

Теорема доказана.

5. Доказательство теоремы 3

Определим функцию $H(x_0, \ldots, x_n) = \sum_{i=0}^{n} -x_i \cdot \log_2 x_i$, которая называется *энтропией*. Справедливо следующее утверждение

Утверждение 1. *Если ИГ* $U \in D_I$, *где* $I = ((0,1), V, \rho_=, f)$, *и сложность* $T(U)$ *представлена в виде линейной комбинации* $\sum_{i=0}^{n} a_i p_i$, *где* $a_i \in \mathbb{N}$, *то*

$$\sum_{i=0}^{n} 2^{-a_i} = 1.$$

Доказательство. Докажем утверждение индукцией по мощности библиотеки V.

Пусть $|V| = 1$. Множество ИГ D_I, которые решают ЗПИО I с одной записью в библиотеке, состоит из одного лишь ИГ U_I, содержащего одну переключательную вершину. Поэтому для этого ИГ $a_0 = 1$, $a_1 = 1$, и утверждение верно.

Пусть утверждение верно для всех ЗПИО с мощностью библиотеки V, равной k, покажем что утверждение верно для ЗПИО с мощностью библиотеки V, равной $k+1$. Рассмотрим произвольные ЗПИО I и ИГ $U \in D_I$. Обозначим элементы библиотеки $V = \{y_1, \dots y_n\}$. Рассмотрим для ИГ U классы (k)-подобия с самым большим номером k, равным k_0. Все такие классы имеют вид (y_i, y_{i+2}), выберем произвольный класс (y_j, y_{j+2}).

Если бы запись y_{j+1} не принадлежала библиотеке V, то, удалив из ИГ U поддерево с корнем в вершине с нагрузкой $g_{y_{j+1}}$, мы получим ИГ U', который решает задачу $I' = ((0,1), V \setminus \{y_{j+1}\}, \rho_=, f)$. Обозначим набор вероятностей задачи I' через $(p'_0, \dots, p'_{n-1})$, а соответствующие коэффициенты в линейной комбинации через $a'_0, \dots, a'_{n-1}$, тогда

$$a'_i = a_i,\ i = 0, \dots, j-1,\ \ a'_j = a_j - 1 = a_{j+1} - 1,$$
$$a'_i = a_{i+1},\ i = j+1, \dots, n-1.$$

К задаче I' и ИГ U' применимо предположение индукции, поэтому верно

$$\sum_{i=0}^{n-1} 2^{-a'_i} = 1.$$

Применим это к ИГ U

$$\sum_{i=0}^{n} 2^{-a_i} = \sum_{i=0}^{j-1} 2^{-a_i} + 2^{-a_j} + 2^{-a_{j+1}} + \sum_{i=j+2}^{n} 2^{-a_i} =$$
$$= \sum_{i=0}^{j-1} 2^{-a'_i} + 2^{-(a'_j+1)} + 2^{-(a'_j+1)} + \sum_{i=j+1}^{n-1} 2^{-a'_i} = \sum_{i=0}^{n-1} 2^{-a'_i} = 1.$$

Утверждение доказано.

Лемма 5. *Для любой ЗПИО $I = ((0,1), V, \rho_=, f)$, для любого ИГ $U \in D_I$ верно*

$$T(U) \geqslant H(p_0, \ldots, p_n),$$

где $p_i = \mathbf{P}\,(x \in (y_i, y_{i+1}))$, $i = 0, \ldots, n$, $y_0 = 0$, $y_{n+1} = 1$.

Доказательство. Запишем сложность ИГ U в виде

$$T(U) = \sum_{i=0}^{n} a_i p_i = -\sum_{i=0}^{n} p_i \log_2(2^{-a_i}).$$

Несложно показать что функция $h(x_0, \ldots, x_n) = -\sum_{i=0}^{n} p_i \log_2 x_i$, где x_i вещественные числа из отрезка $[0,1]$, такие что $\sum_{i=0}^{n} x_i = 1$, достигает своего минимума в точке $(p_0, \ldots, p_n)$. Таким образом,

$$T(U) = -\sum_{i=0}^{n} p_i \log_2(2^{-a_i}) \geqslant -\sum_{i=0}^{n} p_i \log_2 p_i = H(p_0, \ldots, p_n).$$

Лемма доказана.

Лемма 6.

$$M(H(p_0, \ldots, p_n)) = \frac{1}{\ln 2}\left(1 + \frac{1}{2} + \cdots + \frac{1}{n+1} - 1\right).$$

Доказательство. Поскольку на множестве запросов введена равномерная мера, то p_i — это длина интервала (y_i, y_{i+1}). Обозначим через $F_{p_i}(t) = \mathbf{P}\,(p_i < t), t \in (0,1)$, распределение величины p_i, а плотность этого распределения через $f_{p_i}(x)$.

Рассмотрим эквивалентную ситуацию, когда $n+1$ случайная точка выбирается на ориентированной окружности длины 1. Если окружность разрезать в $(n+1)$-ой точке, то получаем случайно выпавшие n точек на отрезке длины 1. В силу симметрии окружности распределение длин всех отрезков одинаково, то есть $F_{p_i}(t) = F_{p_1}$, $i = 2, \ldots, n$. Вычислим распределение длины первого интервала F_{p_1}. Вероятность того, что $\mathbf{P}\left(|(0, y_{(1)})| > t\right)$, равна вероятности того, что все точки попали в отрезок длины $(1-t)$, следовательно

$$\mathbf{P}\left(|(0, y_{(1)})| > t\right) = (1-t)^n.$$

Тогда функции плотности распределений $f_{p_i}(t)$ величин p_i равны

$$f_{p_i}(t) = f_{p_1} = (F_{p_1}(t))' = (1-(1-t)^n)' = n(1-t)^{n-1}.$$

С учетом этого посчитаем математическое ожидание энтропии

$$M(H(p_0, \ldots, p_n)) = M\left(-\sum_{i=0}^{n} p_i \log_2 p_i\right) =$$

$$= -\sum_{i=0}^{n} M(p_i \log_2 p_i) = -(n+1)\int_0^1 x \log_2 x \cdot n(1-x)^{n-1}\, dx.$$

Можно преобразовать последний интеграл к виду

$$-(n+1)\int_0^1 x \log_2 x \cdot n(1-x)^{n-1} dx =$$

$$= -n(n+1)\int_0^1 x \log_2 x \sum_{i=0}^{n-1} C_{n-1}^i (-1)^i x^i\, dx =$$

$$= -n(n+1)\sum_{i=0}^{n-1} C_{n-1}^i (-1)^i \int_0^1 x^{i+1} \log_2 x\, dx =$$

$$= \frac{-n(n+1)}{\ln 2}\sum_{i=0}^{n-1} C_{n-1}^i (-1)^i \int_0^1 x^{i+1} \ln x\, dx.$$

Интегрированием по частям получаем

$$\int_0^1 x^{i+1} \ln x\, dx = -\frac{1}{(i+2)^2}.$$

Подставим полученное значение в сумму и распишем биномиальный коэффициент

$$\frac{n(n+1)}{\ln 2}\sum_{i=0}^{n-1} C_{n-1}^{i}(-1)^i\frac{1}{(i+2)^2} = \\ = \frac{n(n+1)}{\ln 2}\sum_{i=0}^{n-1}(-1)^i\frac{(n-1)!}{i!(n-1-i)!}\frac{1}{(i+2)^2}.$$

Внесем коэффициент $n(n+1)$ под знак суммы и свернем биномиальный коэффициент

$$\frac{1}{\ln 2}\sum_{i=0}^{n-1}(-1)^i\frac{(n-1)!n(n+1)}{i!(i+1)(i+2)((n+1)-(i+2))!}\frac{(i+1)(i+2)}{(i+2)^2} = \\ = \frac{1}{\ln 2}\sum_{i=0}^{n-1}(-1)^i C_{i+2}^{n+1}\frac{i+1}{i+2}.$$

Сделаем замену $j = i+2$ и представим сумму в виде двух частей

$$\frac{1}{\ln 2}\sum_{i=2}^{n+1}(-1)^j C_j^{n+1}\frac{j-1}{j} = \frac{1}{\ln 2}\left(\sum_{i=2}^{n+1}(-1)^j C_j^{n+1} - \sum_{i=2}^{n+1}(-1)^j C_j^{n+1}\frac{1}{j}\right).$$

Внутри скобок прибавим и отнимем выражения равные $(-1)^0 C_0^{n+1}$, $(-1)^1 C_1^{n+1}$ и $(-1)^1 C_1^{n+1}$, получим

$$\frac{1}{\ln 2}\left(\sum_{i=0}^{n+1}(-1)^j C_j^{n+1} - \sum_{i=1}^{n+1}(-1)^j C_j^{n+1}\frac{1}{j} - 1 + (n+1) - (n+1)\right) = \\ = \frac{1}{\ln 2}\left(0 + \sum_{i=1}^{n+1}(-1)^{j+1} C_j^{n+1}\frac{1}{j} - 1\right).$$

Последнюю сумму можно записать в виде интеграла

$$\sum_{i=1}^{n+1}(-1)^{j+1} C_j^{n+1}\frac{1}{j} = \int_0^1 \frac{1-(1-x)^{n+1}}{x}dx$$

и вычислить, сделав замену $y = 1 - x$,

$$\int_0^1 \frac{1-(1-x)^{n+1}}{x}dx = \int_0^1 \frac{1-y^{n+1}}{1-y}dy = \int_0^1 \sum_{i=1}^{n} y^i dy = \sum_{i=1}^{n+1} \frac{1}{i}.$$

Собирая вместе все вычисленные значения получим

$$M(H(p_0,\ldots,p_n)) = \frac{\sum_{i=1}^{n+1} \frac{1}{i} - 1}{\ln 2}.$$

Лемма доказана.

Доказательство теоремы 3. В силу лемм этого параграфа

$$\mathbf{M}_V(T_n(V)) \geqslant \frac{\sum_{i=1}^{n+1} \frac{1}{i} - 1}{\ln 2}.$$

Известно что $\sum_{i=1}^{n} \frac{1}{i}$ можно представить в виде

$$\sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma_n,$$

где γ_n, $n = 1, 2, \ldots$ сходящаяся последовательность, члены которой положительны. Предел этой последовательности называется постоянной Эйлера $\gamma = 0{,}57\ldots$

Учитывая это, неравенство можно записать в виде

$$\mathbf{M}_V(T_n(V)) \geqslant \log_2(n+1) - \frac{1-\gamma_{n+1}}{\ln 2}.$$

Сложность задачи $T_n(V)$ всегда не больше $]\log_2(n+1)[$, поскольку такую сложность имеет ИГ, реализующий логарифмический поиск. Значит

$$]\log_2(n+1)[\geqslant \mathbf{M}_V(T_n(V)) \geqslant \log_2(n+1) - \frac{1-\gamma_{n+1}}{\ln 2}.$$

Теорема доказана.

6. Заключение

В данной работе было показано, что для двух широких классов задач поиска идентичных объектов оптимальное решение задачи по сложности близко к алгоритму логарифмического поиска. На практике это означает, что при необходимости построения «хорошего» ИГ для задач из таких классов можно не тратить вычислительные ресурсы на построение оптимального информационного графа, а пользоваться логарифмическим поиском. В работе содержатся оценки, которые позволяют оценить «потери» от такого решения.

Автором было проведено дальнейшее исследование по данной тематике. Для обобщения «второго» класса задач — задач, в которых база данных — случайный вектор с произвольным непрерывным распределением, а запросы распределены равномерно — также оказалось, что сложность оптимального ИГ асимптотически равна сложности ИГ логарифмического поиска. Данный результат анонсирован в [4].

Автор выражает благодарность своему научному руководителю профессору Гасанову Э. Э. за постановку задачи и внимание к работе.

Список литературы

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

[2] Кучеренко Н. С. О сложности поиска идентичных объектов для случайных баз данных // Материалы IX Международной конференции «Интеллектуальные системы и компьютерные науки». Т. 1. Ч. 2. С. 171. 2006.

[3] Knuth D. E. Optimum binary search trees // Acta Informatica 1971. T. 1. No 1. C. 14–25.

[4] Кучеренко Н. С. Оценки сложности поиска идентичных объектов для случайных баз данных // Материалы IX Международного семинара «Дискретная математика и ее приложения», посвященного 75-летию со дня рождения академика О. Б. Лупанова. С. 329–331. 2007.