УДК 004.89
DOI: 10.17277/vestnik.2017.01.pp.048-060

К ВОПРОСУ ПОСТРОЕНИЯ ОПТИМАЛЬНЫХ ПО ВРЕМЕНИ ИНФОРМАЦИОННО-ПОИСКОВЫХ ГРАФОВ

Д. В. Поляков[1], А. И. Попов[2], С. А. Дузькрятченко[1], Е. Н. Лепёшкин[1]

кафедра «Информационные системы и защита информации» (1), dimadress@yandex.ru; отдел электронного обучения (2), ФГБОУ ВО ТГТУ», г. Тамбов, Россия

Ключевые слова: ассоциативный массив; интервальный поиск; информационно-поисковый граф; оптимизационная задача; хеш-таблица.

Аннотация: Предложен подход к усовершенствованию константного в худшем случае алгоритма поиска идентичных объектов. Введено определение гибридного информационно-поискового графа и поставлена оптимизационная задача его построения. Рассмотрены подходы к решению поставленной задачи оптимизации.

Введение

Бурное развитие информационных технологий во многом сформировало картину научно-технического прогресса конца XX, начала XXI вв. Рост числа процессорных устройств и объединение их одну сеть привело к аккумулированию человечеством огромных объемов данных. По некоторым прогнозам [1, 2] к 2020 г. число устройств в Интернете достигнет 50 млрд с учетом неоднородного распространения информационных технологий в мире, в среднем по 10 устройств на человека.

Очевидно, что такой объем процессорных устройств не могут обеспечить смартфоны и планшеты. Действительно, спрос на данный товар достиг некоторого состояния насыщения, при котором покупка новых устройств обычно происходит в целях приобретения более современного или статусного устройства. Таким образом, данный процесс оказывает влияние не столько на рост числа устройств в сети, сколько на их обновление.

Становятся популярными продукты, связанные с микроконтроллерами. На фоне удешевления технологий и роста процессорной мощности микросхем рядовым потребителям и малому бизнесу стали доступны простые и дешевые решения на базе объединения микропроцессорных устройств. Например, элементы технологии «умный дом» становятся доступны рядовым покупателям, а компании теперь могут себе позволить отслеживать с помощью датчиков не только параметры высокотехнологичного производства, но и практически любые показатели различных бизнес-процессов.

Транспортная логистика может быть оптимизирована и застрахована от злоумышленных действий недобросовестных сотрудников путем внедрения механизмов контроля транспортных средств на основе системы датчиков, снабженных

микроконтроллерами, и позволяет в реальном времени записывать изменения координат, скорости, топлива, веса груза и многое другое.

Вместе с тем подобный рост использования и востребованность компьютерных технологий формирует спрос на высокопроизводительные системы работы с большими объемами данных. Для обработки последних создано огромное количество алгоритмов, многие из которых базируются на крайне ресурсоемких методах искусственного интеллекта. Они позволяют оптимизировать логистику, выявить факт слива топлива или кражи груза.

Другим классическим примером обработки больших объемов данных на основе интеллектуальных алгоритмов является широко применяемый в торговых сетях всего мира анализ поведения покупателей на основе отслеживания использования скидочных, накопительных и кредитных карт. Данный анализ давно зарекомендовал себя в качестве важнейшей составляющей формирования маркетинговой политики компаний [3].

Однако для высокопроизводительной работы ресурсоемких алгоритмов на больших объемах данных требуется быстрое решение информационно-поисковых задач (**ИПЗ**) [4] – выборки данных из общего массива, отвечающих тем или иным условиям.

Подходы к хранению данных при решении ИПЗ

Алгоритмы, решающие ИПЗ, в настоящее время реализуются на ассоциативных массивах – структурах данных, специально приспособленных для хранения элементов таким образом, чтобы обеспечить высокую скорость информационного поиска. Данные структуры целесообразно разделять на два типа: связные деревья и хеш-функции. *Связные деревья* представляют собой граф, каждая вершина которого содержит некоторое значение, а левые и правые поддеревья элементы меньше и больше рассмотренного значения соответственно. Поиск по данному дереву включает сравнение искомого значения с элементом в вершине и последующем поиске в левом или правом поддеревьях.

Поиск будет наиболее быстрым в том случае, если дерево сбалансировано [4, 5], то есть для каждого узла дерева верно, что в левом и правом поддереве одинаковое или отличающееся на единицу число элементов. Использующиеся на сегодняшний день ассоциативные массивы на основе бинарных деревьев, такие как рандомизированное дерево [5, 6], АВР-дерево [5, 6], красно-черное дерево [5] отличаются только способом поддержания сбалансированности в той или иной мере при добавлении и удалении элементов ассоциативного массива.

Идея ассоциативных массивов на основе *хеш-функций* базируется на построении сюръективного отображения из множества хранимых объектов на адресное пространство, в котором эти объекты хранятся. Для решения поисковой задачи достаточно отобразить поисковый запрос на адресное пространство и проверить находится ли в этом месте соответствующий запросу объект.

Можно показать [5], что даже при поиске в ассоциативном массиве, основанном на сбалансированном дереве, каждая проверка поискового запроса в узле приводит к аналогичной подзадаче на числе элементов в среднем вдвое меньше первоначального. Время поиска объекта в данном массиве пропорционально логарифму числа хранимых значений. Вместе с тем, поиск в ассоциативном массиве на основе хеш-функции (хеш-таблицы) выполняется за некоторое константное время, не изменяемое при росте числа входных данных, что является крайне важным при работе с большими объемами информации.

Однако практика показала, что преимущество хеш-таблиц не является самоочевидным. Во первых, операция сравнения, используемая в ассоциативных массивах на основе бинарных деревьев, работает гораздо быстрее группы операций, формализующих хеш-функцию. Во вторых, крайне сложно с ассоциативными массивами такого рода осуществлять операции добавления элемента. Они могут привести к коллизиям – ситуациям, когда хеш-функция отображает новый объект в уже занятое адресное пространство. Использующиеся на сегодняшний день хеш-таблицы отличаются друг от друга алгоритмами построения отображений на адресное пространство и способами решения коллизий. В роли последних выступают алгоритмы линейного зондирования, двойного хеширования, универсальное хеширование, хеширование с цепочками [5, 6] и др.

Вместе с тем эффективность большинства из рассмотренных методов зависит от качества хеш-функции. Так, если работа данного отображения приводит к большому числу коллизий, то в результате многократного использования операции добавления в ассоциативный массив, среднее время поиска может деградировать до линейного. Это означает, что время решения ИПЗ растет пропорционально числу хранимых элементов, что соответствует решению той же задачи на неспециализированных структурах данных (индексных массивах и линейных списках).

Важно отметить, что хеш-таблицы с хорошими показателями требуют больших объемов памяти. Так, например, универсальное хеширование, практически не подверженное снижению скорости информационного поиска при появлении даже большого числа коллизий [5, 6], требует огромных дополнительных объемов памяти [5]. Высокие скоростные характеристики решения поисковых задач и отсутствие необходимости в дополнительной памяти показывают так называемые совершенные хеш-таблицы [6, 7]. Вместе с тем, построенные на их основе ассоциативные массивы не поддерживают операции добавления элементов.

Другой важной характеристикой хеш-функций является их монотонность. Только монотонные хеш-функции лежат в основе ассоциативных массивов, способных осуществлять высокоскоростной интервальный поиск [4]. Данный тип поисковой задачи основан на выявлении хранимых объектов, входящих в интервал-запрос. Интервальный поиск все чаще требуется при решении задач обработки больших объемов данных [4].

Без ограничения общности положим, что хранимые данные представляют собой некоторые числовые значения из заданного диапазона. В работах [4, 8] предложен константный в худшем случае алгоритм (**КХСА**) решения ИПЗ. Хеш-функция в рамках проектирования данного алгоритма является монотонной, а значит, построенный на ее основе ассоциативный массив вполне может иметь высокие скоростные показатели при решении задачи интервального поиска (**ЗИП**) всех значений, находящихся в некотором диапазоне, выступающем запросом. Вместе с тем, в [8] показано, что КХСА для хранения n элементов требует в среднем объем памяти, пропорциональный n^2, что существенно ограничивает использование эффективного алгоритма при работе с большими объемами данных.

Важно отметить, что все рассматриваемые ассоциативные массивы эффективны только при размещении хранимых объектов в RAM-виде компьютерной памяти, позволяющей осуществлять доступ к любому байту этой памяти по ее адресу за константное время. Данное свойство зачастую присуще именно оперативной памяти, поэтому при работе с большими объемами данных с помощью ассоциативных массивов ограничения по памяти являются критическими. Действительно, виртуальная память на сегодняшний день характеризуется куда большими ограничениями, чем ее аналог на твердотельных накопителях.

Ассоциативный массив с константным в худшем случае алгоритмом информационного поиска

Рассмотрим более подробно хеш-функцию, позволяющую реализовать КХСА поиска объектов [8].

Пусть $X = \{x_0, x_1, ..., x_n\}$ – множество хранимых элементов, $|X| = n+1$. Без ограничения общности положим, что все x_i, $i = \overline{0,n}$ имеют числовую природу и $x_0 \le x_1 \le x_2 \le ... \le x_n$.

Найдем величину

$$\Delta = \min_{i=0,n-1} (x_{i+1} - x_i). \quad (1)$$

Введем в рассмотрение арифметическую прогрессию y_0, y_1, ..., y_m,... с $y_0 = x_0$ и разностью Δ. Остановимся на первых m членах данной прогрессии, где

$$m = \left\lceil \frac{x_n - x_0}{\Delta} \right\rceil. \quad (2)$$

Несложно показать [9], что $(\forall\, x \in X)\ \left(\exists !\, i = \overline{0, m-1} \mid x \in [x_i, x_{i+1})\right)$. Обозначим данное i, соответствующее некоторому x, $i(x)$. Тогда, как показано в [4, 8, 9], $i(x)$ находится по формуле

$$i(x) = \left\lceil \frac{x - x_0}{\Delta} \right\rceil. \quad (3)$$

Кроме того в приведенных работах [4, 8, 9] отмечено, что $(\forall\, x_1, x_2 \in X)$ $\left(i(x_1) \ne i(x_2)\right)$ и означает инъективность отображения $i : X \to \left\{k \mid k = \overline{0, m-1}\right\}$.

На основе данных рассуждений для хранения множества X целесообразно создать массив из m элементов. Обозначим данный массив A, а его i-ю ячейку – $A[i]$. Будем хранить произвольный элемент $x \in X$ в ячейке $A[i(x)]$, где $i(x)$ вычисляется по формуле (3). В силу инъективности данного отображения в каждую ячейку попадет только один хранимый элемент. При решении задачи о поиске идентичного объекта – проверки, есть ли соответствующий объект в ассоциативном массиве, – достаточно вычислить по формуле (3) индекс ячейки, в которой хранился бы данный объект и проверить, есть ли он там. Геометрический смысл ассоциативного массива на основе (1) – (3) представлен на рис. 1.

Одним из основных недостатков рассмотренного подхода к организации ассоциативного массива является большой объем дополнительной памяти, то есть пустых ячеек в массиве A. По теоретическим оценкам [8] m возрастает с ростом числа хранимых элементов n пропорционально n^2, в то время как очевидно, что для хранения n объектов используется только n ячеек массива A.

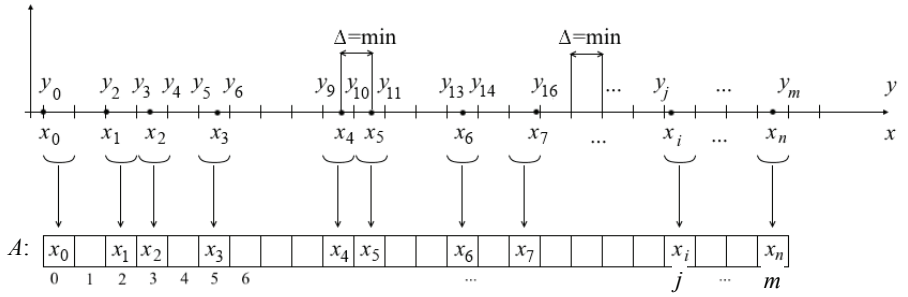

Рис. 1. Геометрический смысл ассоциативного массива на основе (1) – (3)

Усовершенствование ассоциативного массива с константным в худшем случае алгоритмом информационного поиска

В работе [9] предложено усовершенствование посредством отображения ассоциативного массива с КХСА информационного поиска, позволяющее снизить объем дополнительной неиспользуемой памяти. В основе такого усовершенствования лежит работа не с множеством X, а с его отображением на основе некоторой непрерывной строго возрастающей биекции f. Таким образом, после выбора биекции формулы (1) – (3) примут вид:

$$\Delta = \min_{i=0,n-1} \left(f\left(x_{i+1}\right) - f\left(x_i\right) \right); \tag{4}$$

$$m = \left\lceil \frac{f\left(x_n\right) - f\left(x_0\right)}{\Delta} \right\rceil; \tag{5}$$

$$i\left(x\right) = \left\lceil \frac{f\left(x\right) - f\left(x_0\right)}{\Delta} \right\rceil. \tag{6}$$

Отметим, что в качестве биекции можно взять $f(x) = x$. Данное отображение задает частный случай, при котором (4) – (6) принимают вид (1) – (3) и, соответственно, никак не повлияет на объемы памяти, необходимые ассоциативному массиву данного вида (рис. 2).

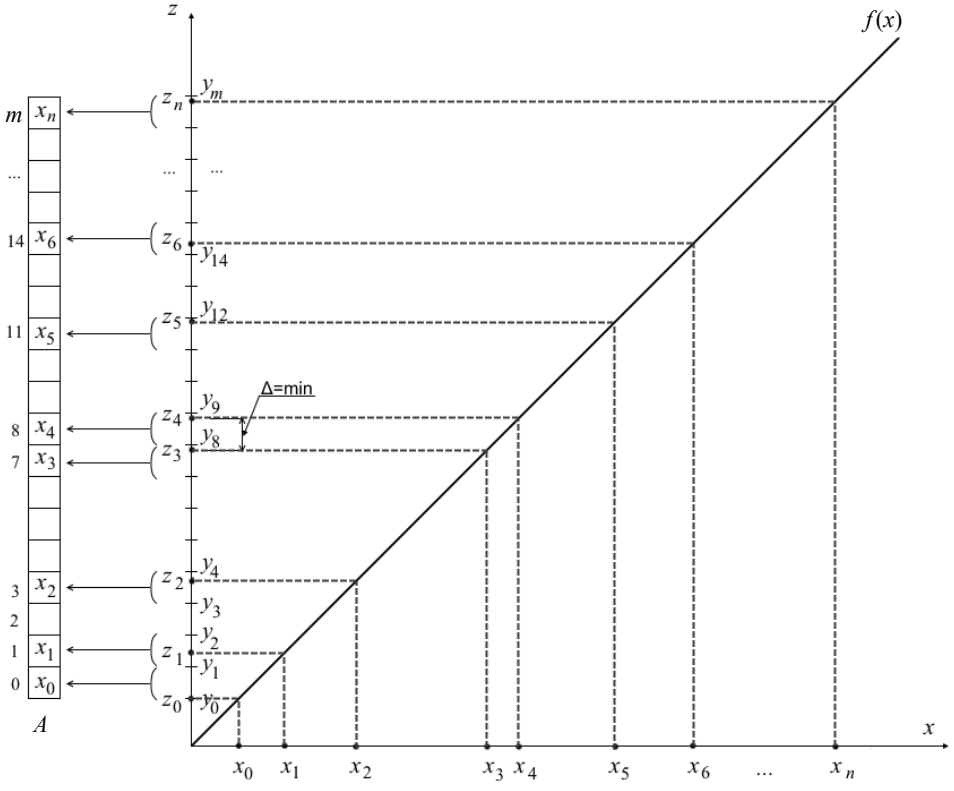

Рис. 2. Использование в (4) – (6) в качестве отображения $f(x) = x$

Вместе с тем, если график выбранного отображения будет иметь другой, более удачный (с точки зрения расположения на числовой прямой хранимых объектов) вид (рис. 3), то это приведет к существенному сокращению требуемой для хранения объектов памяти.

Отметим, что выбрать отображение $f(x)$, таким образом, чтобы число ячеек массива m стало равным количеству хранимых элементов n можно всегда, например, с помощью полиномиальной интерполяции [9]. Единственное верхнее ограничение на степень интерполяционного многочлена определяется числом хранимых элементов – n, и с ростом степени многочлена растет и сложность его вычисления.

Бинарные деревья обладают сложностью поиска идентичного объекта пропорциональной $\log(n)$ и обеспечивают высокую скорость элементарных операций сравнения в отличие от операций сложения и умножения. Использование в рамках предложенного подхода отображения, формализованного многочленом со степенью пропорциональной даже логарифму от числа хранимых элементов, нецелесообразно.

Оотображение, формализованное посредством формул (4) – (6) и последующего обращения к некоторому элементу массива, удобно представить в виде вершины дерева, подобного бинарному, но имеющему более двух потомков. Последние в предлагаемой модели формализованы листьями – ячейками массива A. Пример такого вырожденного дерева представлен на рис. 4.

Из рисунка 4 видно, что преобразование (4) – (6) отображает все точки из некоторых полуинтервалов в одни и те же ячейки массива. В отличие от преобразования (1) – (3) данные полуинтервалы имеют разную длину, характеризуемую видом f.

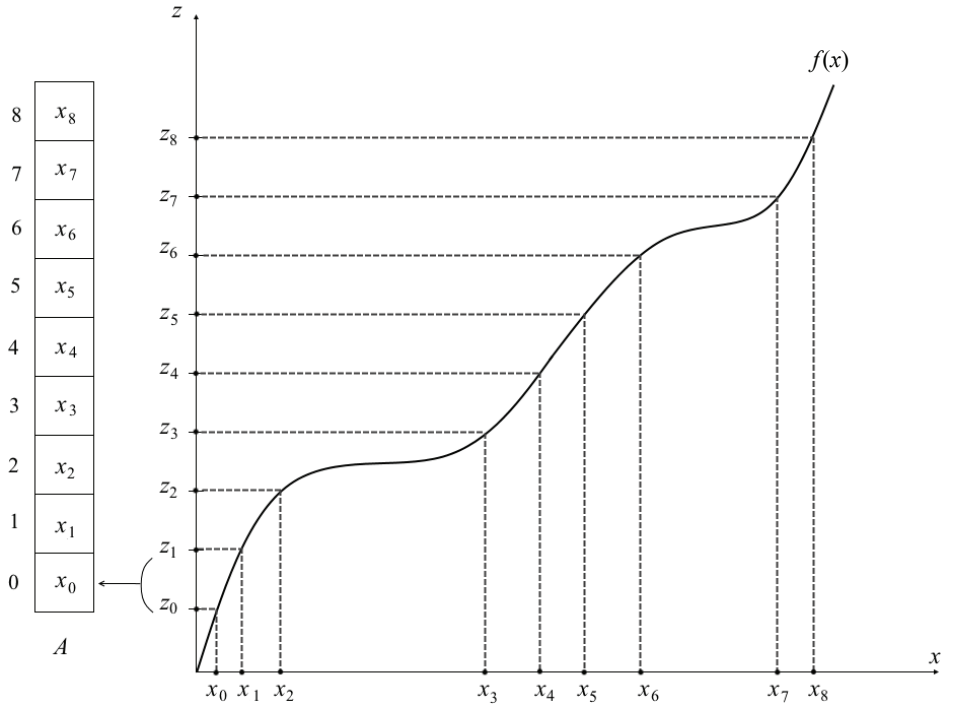

Рис. 3. Результат удачного выбора отображения $f(x)$

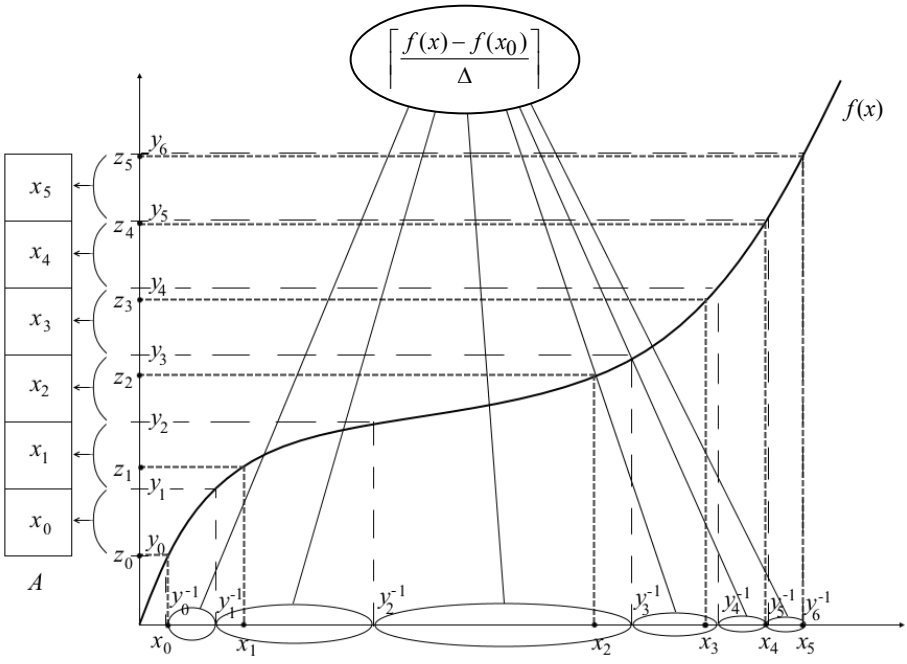

Рис. 4. Представление предлагаемого ассоциативного массива в виде дерева с единичной глубиной

На практике, в качестве отображения f удобно взять кусочно-непрерывную функцию. Действительно, если разделить хранимые объекты на относительно однородные подмножества, во многих случаях степень многочленов, формализующих f, резко понижается (рис. 5).

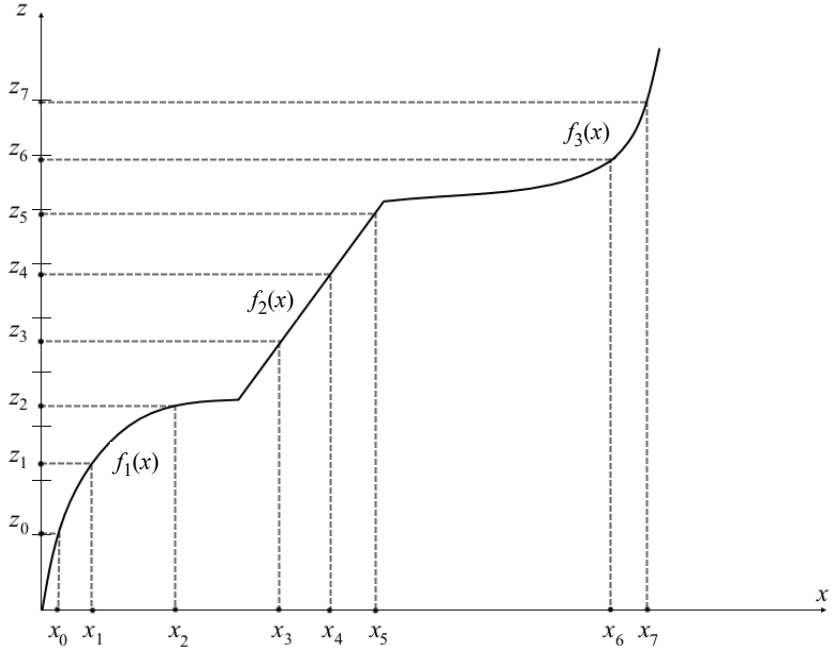

Рис. 5. Использование кусочно-непрерывной функции в качестве отображения f

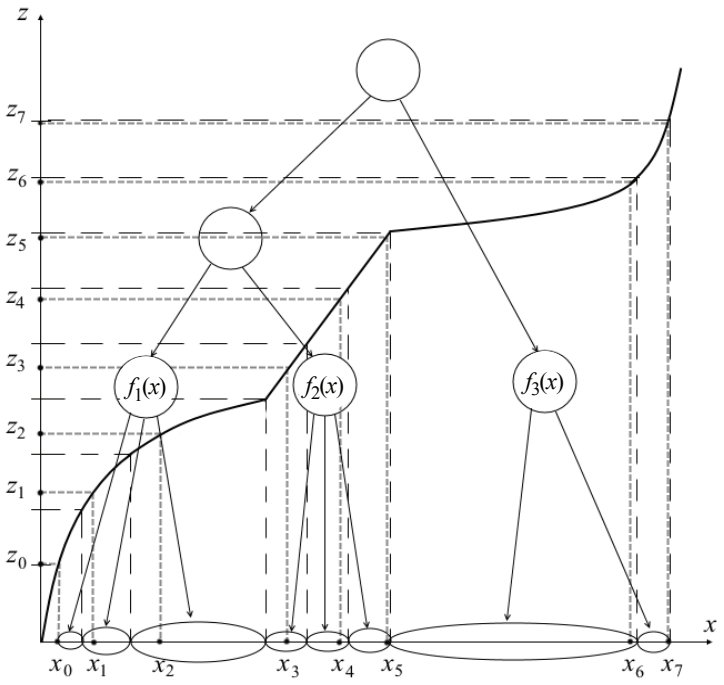

Рис. 6. Пример графа, формализующего преобразование (4) – (6) при условии кусочно-заданной функции *f*

Вычисление данной функции аналогично бинарному поиску, осуществляемому в целях нахождения полуинтервала, поиск на котором уже осуществляется на основе (4) – (6) с использованием многочлена, задающего на нем *f*.

Пример графа, соответствующего кусочно-заданной *f* (см. рис. 5) представлен на рис. 6. Выражение (6) ставит в соответствие всем точкам некоторого полуинтервала определенное число – индекс в диапазоне от 0 до $m-1$. Длина каждого такого полуинтервала, по сути, задается функцией *f*. Формула (4) гарантирует, что в данный полуинтервал попадет не более одного хранимого объекта.

Отказавшись от ограничения на величину Δ, задаваемого (4), можно рассматривать (6) как некоторый классификатор, разбирающий хранимые элементы на группы из неравных интервалов и сводящий задачу поиска идентичного объекта к аналогичной на одном из подынтервалов. Данная классификация, с одной стороны, позволяет за одну операцию на вершине в несколько раз снизить размерность задачи информационного поиска. С другой стороны, очевидно, что операция (6) требует больше времени, чем операции сравнения в вершинах бинарного дерева. Выбор наилучшего подхода зависит от конкретного вида множества хранимых элементов.

Постановка оптимизационной задачи построения гибридного информационно-поискового графа

Гибридный информационно-поисковый граф (**ГИПГ**) – структура данных, формализующая работу ассоциативного массива на основе предлагаемого в (4) – (6) подхода, где в общем случае отображение *f* представляет собой кусочно-непрерывную неубывающую функцию. Пример такого графа изображен на рис. 6.

Термин «гибридный» в данном определении используется как маркер того, что с алгоритмической точки зрения работа соответствующего рассматриваемому графу ассоциативного массива сочетает в себе поиск по бинарному дереву подынтервала и, одновременно, аналитического вида гладкой функции, отображающей данный подынтервал в адресное пространство на основе (4) – (6).

Отметим, что гладкая функция также является частным случаем кусочно-гладкой. Если в качестве *f* выбрана гладкая функция, то бинарная составляющая отсутствует и ГИПГ имеет вид, приведенный на рис. 5.

Возникает задача выбора ГИПГ для использования при хранении данных в информационно-поисковом массиве. Для простоты оценки в дальнейшем ограничимся рассмотрением в качестве гладких частей отображения *f* полиномиальных функций. Представленная ниже задача легко обобщается на функции произвольного вида.

Пусть G_X – множество всевозможных гибридных информационных графов, для хранения элементов *X*. Рассмотрим отображение $T : G_X \to R$, где *R* – множество действительных чисел, определяющее сложность решения задачи информационного поиска на произвольном графе $g \in G_X$ в худшем случае.

Зафиксируем некоторый граф $g \in G_X$. Пусть *r* – корневая вершина *g*, связанная с множеством вершин $r_0, r_1, ..., r_k$. Каждая из этих вершин является вершиной некоторого ГИПГ $g_0, g_1, ..., g_k$, имеющего свою сложность поиска в худшем случае, откуда

$$T(g) = \Theta(r) + \max\left\{T\left(g_0\right), T\left(g_1\right), ..., T\left(g_k\right)\right\}, \tag{7}$$

где $\Theta(r)$ – сложность вычислений в вершине *r*.

Важно отметить, что *r* может быть как бинарной вершиной и в этом случае $k = 2$, так и произвольной, выполнение поискового алгоритма на которой сводится к вычислению гладкой функции и нахождению адреса хранимого элемента на основе (6). Введем в рассмотрение множество B_g – множество всех бинарных вершин графа *g* и F_g – множество вершин, соответствующих полиному и вычисляющих адрес для хранения элемента на основе (6).

Для вычисления $\Theta(r)$ зададим время выполнения операций сравнения, сложения, умножения и вычисления (6) без учета *f* как $t_<, t_+, t_\times, t_6$ соответственно

$$\Theta(r) = \begin{cases} t_<, \ r \in B_g; \\ k\left(t_+ + t_\times\right) + t_6, \ r \in F_g. \end{cases} \tag{8}$$

где *k* – число элементов в многочлене, задающем функцию в вершине *r*.

Формулы (7) и (8) определяют вычисление сложности для обобщенного графа, в котором бинарные вершины и вершины, задаваемые многочленом, могут оказаться в качестве любого узла. Из определения ГИПГ следует, что вершины, соответствующие гладким функциям, связаны с листьями данного дерева. Для гибридного информационно-поискового графа (7) и (8) принимают вид

$$T(g) = \begin{cases} t_< + \max\left\{T\left(g_0\right), \ T\left(g_1\right)\right\}, \ r \in B_g; \\ k\left(t_+ + t_\times\right) + t_6, \ r \in F_g. \end{cases} \tag{9}$$

Заметим, что (9) задает рекуррентное соотношение вычислительной сложности ГИПГ.

Наилучшим, с точки зрения скорости информационного поиска, ГИПГ будет элемент G_X, обладающий наименьшим значением $T(g)$. Тогда рассмотрим задачу

$$T(g) \underset{g \in G_X}{\rightarrow} \min. \tag{10}$$

Заметим, что при отсутствии каких-либо ограничений решением (10) будет граф, формализуемый (1) – (3). Вместе с тем, если ограничить граф некоторым объемом памяти, то задача становится нетривиальной. Именно подобное ограничение является крайне важным в условиях работы с большими объемами данных в виртуальной памяти. Пусть $M : G_X \rightarrow N$ (N – множество натуральных чисел) – отображение, определяющее количество памяти необходимое для построения ассоциативного массива на основе некоторого графа. На практике значение $M(g)$, $g \in G_X$ легко определяется за счет учета объема памяти, необходимого для каждой вершины. Пусть при построении ассоциативного массива задано ограничение, на максимальный объем памяти $M_{\max}$. Тогда получаем оптимизационную задачу вида

$$\begin{cases} T(g) \underset{g \in G_X}{\rightarrow} \min; \\ M(g) \le M_{\max}. \end{cases} \tag{11}$$

Другой вариант постановки оптимизационной задачи с нетривиальным решением это достижение (10) при условии, что полученная на основе отображения соответствующего графу g хеш-таблица является совершенной [10, 11]. То есть в хеш-таблице отсутствуют пустые ячейки. Данная задача является приоритетной в случае отсутствия заранее заданного $M_{\max}$. Тем более что совершенные хеш-функции удобно использовать при работе с заранее известным набором хранимых элементов.

Формализуем рассматриваемую оптимизационную задачу. Рассмотрим отображение $D_g : F_g \rightarrow N$, определяющее число дополнительной неиспользуемой памяти.

Тогда исследуемая оптимизационная задача может быть сформулирована следующим образом

$$\begin{cases} T\left(g\right) \underset{g \in G_X}{\rightarrow} \min; \\ \sum\limits_{r \in F_g} D_g\left(r\right) = 0. \end{cases} \tag{12}$$

Системы (11) и (12) формализуют две оптимизационные задачи построения информационно-поискового графа с наименьшим для ГИПГ временем информационного поиска идентичного объекта в худшем случае.

Заключение

Решение оптимизационных задач (11) и (12), с одной стороны, позволит создавать ассоциативные массивы с константным в худшем случае временем поиска и ограничением по используемой памяти. С другой стороны, введенные в рассмотрение ГИПГ могут использоваться в теоретических выкладках для оценки качества хеш-таблиц, построение которых происходит без непосредственного учета хранимых элементов.

Решения задач (11) и (12) не являются тривиальными. Рассмотрим наивный алгоритм, заключающийся в переборе всех графов в целях поиска минимального, удовлетворяющего условию выбранной оптимизационной задачи. Заметим, что каждый граф разбивает X на полуинтервалы на основе части ГИПГ, соответствующей бинарному дереву. Такое разбиение происходит по элементам X. Пусть требуется разбить X на k полуинтервалов, $0 \le k \le n$, то есть выбрать k точек, по которым пройдет разбиение. Количество способов выбрать k элементов для разбиения вычисляется как число сочетаний из n по $k\left(C_n^k\right)$. Так как необходимо рассмотреть графы для всех $k = \overline{0,n}$, их общее число будет составлять $\sum_{k=0}^{n} C_n^k$ или 2^n. Кроме того, для каждого разбиения необходимо найти соответствующие многочлены. Так как их степень не превышает n, а в среднем будет пропорциональна $n/2$, исследуемый наивный алгоритм построения оптимального ГИПГ имеет асимптотическую сложность $O\left(n2^n\right)$.

Полученная асимптотическая сложность [5] является экспоненциальной и свидетельствует о невозможности использования данного алгоритма на практике при работе с большими объемами данных.

Вместе с тем любой подграф оптимального ГИПГ также должен быть оптимальным для полуинтервала, на котором он осуществляет поиск. Решение задач (11) и (12) обладает оптимальной подструктурой и удовлетворяет принципу Беллмана [5]. Последнее свидетельствует о возможности использования технологии динамического программирования для проектирования менее вычислительно сложного алгоритма построения оптимального ГИПГ.

В дальнейших исследованиях планируется осуществить проектирование алгоритма построения оптимального ГПИГ посредствам динамического программирования, провести оценку вычислительной сложности спроектированного алгоритма и исследовать возможности применения оптимальных ГИПГ на практике.

Список литературы

1. К 2020 году 50 млрд устройств будут связаны друг с другом через интернет [Электронный ресурс] // CyberSecurity : портал новостей высоких технологий и науки. – 2012. – 13 июля. – Режим доступа : http://www.cybersecurity.ru/news/155343.html (дата обращения: 17.10.2016).

2. Cisco: к 2020 году к сети будет подключено более 50 млрд устройств [Электронный ресурс] // CyberSecurity : портал новостей высоких технологий и науки. – 2013. – 2 авг. – Режим доступа : http://www.cybersecurity.ru/telecommunication/178937.html (дата обращения: 17.10.2016).

3. Duhigg, C. How Companies Learn Your Secrets [Электронный ресурс] / C. Duhigg // The New York Times. – 2012. – 16 Feb. – Режим доступа : http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=all&_r=0 (дата обращения: 17.10.2016).

4. Гасанов, Э. Э. Теория хранения и поиска информации / Э. Э. Гасанов, В. Б. Кудрявцев. – М. : ФИЗМАТЛИТ, 2002. – 288 с.

5. Алгоритмы: построение и анализ / Т. Кормен [и др.]. – М. : Вильямс, 2011. – 1290 с.

6. Кузнецов, С. Д. Методы сортировки и поиска [Электронный ресурс] / С. Д. Кузнецов // CITForum.ru : интернет-портал. – Режим доступа : http://citforum.ru/programming/theory/sorting/sorting2.shtml (дата обращения: 17.10.2016).

7. Theory and Practise of Monotone Minimal Perfect Hashing / D. Belazzougui [at al.] // Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX09) : New York, January 3, 2009 / Editors: Irene Finocchi and John Hershberger. – New York, USA, 2009. – P. 132 – 144. doi: 10.1137/1.9781611972894.13

8. Гасанов, Э.Э. Константный в худшем случае алгоритм поиска идентичных объектов / Э. Э. Гасанов, Ю. П. Луговская // Дискрет. математика. – 1999. – Т. 11, № 4. – С. 139 – 144. doi: 10.4213/dm396

9. Поляков, Д. В. Генератор монотонных хеш-функций для ассоциативного массива / Д. В. Поляков, А. И. Попов // Труды НГТУ им. Р.Е. Алексеева. – 2015. – № 2 (109). – С. 70 – 81.

10. Czech, Z. An Optimal Algorithm for Generating Minimal Perfect Hash Function / Z. Czech, G. Havas, B. Majewski // Information Processing Letters. – 1992. – Vol. 43 (5). – P. 257–264.

11. Monotone Minimal Perfect Hashing: Searching a Sorted Table with O(1) Accesses / D. Belazzougui [at al.] // Proceedings of the 20th Annual ACMSIAM Symposium On Discrete Mathematics (SODA). – New York, 2009. – P. 785 – 794.

On Construction of Optimal in Time Information Retrieval Count

D. V. Polyakov[1], A. I. Popov[2],
S. A. Duzkryatchenko[1], E. N. Lepyoshkin[1]

Department "Information Systems and Information Security" (1),
dimadress@yandex.ru; eLearning Department (2),
TSTU, Tambov, Russia

Keywords: associative array; interval search; information retrieval graph; optimization problem; hash table.

Abstract: This paper proposes an approach to the improvement of the constant in the worst-case algorithm of searching for identical objects. Definition of hybrid information retrieval count and the problem of optimization of its construction are given. Some approaches to solving this problem are offered.

References

1. http://www.cybersecurity.ru/news/155343.html (accessed 17 October 2016). (In Russ.)

2. http://www.cybersecurity.ru/telecommunication/178937.html (accessed 17 October 2016). (In Russ.)

3. Duhigg C. How Companies Learn Your Secrets, *The New York Times Magazine*, 2012, 16 Feb., available at: http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=all&_r=0 (accessed 17 October 2016).

4. Gasanov E.E., Kudryavtsev V.B. *Teoriya khraneniya i poiska informatsii* [The theory of information storage and retrieval], Moscow: FIZMATLIT, 2002, 288 p. (In Russ.)

5. Cormen Th.H., Leiserson Ch.E., Rivest R.L., Stein C. *Introduction to Algorithms*, MIT Press, 2009, 1312 p.

6. http://citforum.ru/programming/theory/sorting/sorting2.shtml (accessed 17 October 2016). (In Russ.)

7. Belazzougui, D., Boldi P., Pagh R., Vigna S. Theory and Practise of Monotone Minimal Perfect Hashing, *Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX09)*, New York, January 3, 2009, pp. 132-144, doi: 10.1137/1.9781611972894.13

8. Gasanov E.E. , Lugovskaya Yu.P. A constant, in the worst case, algorithm for the search for identical objects, *Discrete Mathematics and Applications*, 1999, vol. 9, no. 6, pp. 679-684.

9. Polyakov D.V., Popov A.I. [Generator of the monotone hash function for an associative array], *Trudy NGTU im. R.E. Alekseeva* [Transactions of Nizhni Novgorod State Technical University n.a. R.E. Alekseev], 2015, no. 2 (109), pp. 70-81. (In Russ., abstract in Eng.)

10. Czech Z., Havas G., Majewski B. An optimal algorithm for generating minimal perfect hash function, *Information Processing Letters*, 1992, vol. 43(5), pp. 257-264.

11. Belazzougui D., Boldi P., Pagh R., Vigna S. Monotone Minimal Perfect Hashing: Searching a Sorted Table with O(1) Accesses, *Proceedings of the 20th Annual ACMSIAM Symposium On Discrete Mathematics (SODA)*, New York, 2009, pp. 785-794.

Zur Frage der Konstruktion der nach der Zeit optimalen Informationssuchgrafen

Zusammenfassung: Es ist das Herangehen an die Verbesserung des schlimmstenfalls konstanten Suchalgorithmus der identischen Objekte angeboten. Es ist die Bestimmung des Hybridinformationssuchgrafen eingeführt und es ist die Optimisationsaufgabe seiner Konstruktion gestellt. Es sind die Herangehen an die Lösung der gestellten Aufgabe der Optimierung betrachtet.

Sur la question de la construction des graphes optimaux par le temps de recherche de l'information

Résumé: Est proposée une approche pour le perfectionnement de l'algorithme constant dans le pire cas de la recherche des objets identiques. Est introduite la définition du graphe de la recherche de l'information hybride; est mis le problème de l'optimisation. Sont examinées les approches pour la résolution du problème de l'optimisation.

Авторы: ***Поляков Дмитрий Вадимович*** – кандидат технических наук, старший преподаватель кафедры «Информационные системы и защита информации»; ***Попов Андрей Иванович*** – кандидат педагогических наук, доцент, начальник отдела электронного обучения; ***Дузькрятченко Светлана Александровна*** – студентка; ***Лепёшкин Евгений Николаевич*** – магистрант, ФГБОУ ВО «Тамбовский государственный технический университет», г. Тамбов, Россия.

Рецензент: ***Алексеев Владимир Витальевич*** – доктор технических наук. профессор кафедры «Информационные системы и защита информации», г. Тамбов, Россия.