Ф

УДК 004.6

В. И. Финаев[1], Г. В. Дорохина[2]
[1]Южный федеральный университет, г. Ростов-на-Дону, Россия
344006, г. Ростов-на-Дону, ул. Большая Садовая, 105/42, Россия
[2]Государственное учреждение «Институт проблем искусственного интеллекта», г. Донецк
83048, г. Донецк, ул. Артема, 118-б

ПРИМЕНЕНИЯ УСОВЕРШЕНСТВОВАННЫХ ДЕРЕВЬЕВ ЦИФРОВОГО ПОИСКА

V. I. Finayev[1], G. V. Dorokhina[2]
[1]Southern Federal University, Rostov-on-Don, Russia
344006, Rostov-on-Don, 105 / 42 Bolshaya Sadovaya Str., Russia
[2]Public institution «Institute of Problems of Artificial intelligence», Donetsk
83048, Donetsk, Artema str., 118-b

APPLICATIONS OF IMPROVED DIGITAL SEARCH TREES

В. І. Фінаєв[1], Г. В. Дорохіна[2]
[1]Південний федеральний університет, м. Ростов-на-Дону, Росія
344006, м. Ростов-на-Дону, вул. Велика Садова, 105/42, Росія
[2]Державна установа «Інститут проблем штучного інтелекту», м. Донецьк
83048, м. Донецьк, вул. Артема, буд. 118-б

ЗАСТОСУВАННЯ УДОСКОНАЛЕНИХ ДЕРЕВ ЦИФРОВОГО ПОШУКУ

В статье на основе усовершенствованного дерева цифрового поиска разработано усовершенствованное дерево Patricia для хранения и поиска строковых величин, которое обеспечило для рассмотренного словаря строк снижение затрат памяти более, чем на 31% в сравнении с использованием дерева Patricia, как ссылочной структуры, одновременно с массивом строк, и снижение более, чем на 36% в сравнении с усовершенствованным деревом цифрового поиска. Затраты времени на добавление и удаление элементов в древовидные структуры на несколько порядков меньше аналогичных величин для упорядоченного массива строк и массива строк с таблицей индексов.
Ключевые слова: усовершенствованное дерево цифрового поиска, усовершенствованное trie-дерево, усовершенствованное дерево Patricia, представление множеств сложных объектов, применение при решении оптимизационных задач.

In the article, on the basis of an improved digital search tree, an improved Patricia tree for storing and searching string values was developed, which provided for the string dictionary under consideration a reduction in memory costs by more than 31% compared to using the Patricia tree as a reference structure simultaneously with an array of strings., and a decrease of more than 36% compared to the advanced digital search tree. The time required on adding and removing elements to tree-like structures is several orders of magnitude less than the corresponding values for an ordered array of rows and an array of rows with an index table.
Keywords: improved digital search tree, improved trie-tree, improved Patricia tree, representation of sets of complex objects, application in solving optimization problems.

У статті на основі удосконаленого дерева цифрового пошуку розроблено удосконалене дерево Patricia для зберігання і пошуку строкових величин, яке забезпечило для розглянутого словника рядків зниження витрат пам'яті більш, ніж на 31% в порівнянні з використанням дерева Patricia, як посилальної структури, одночасно з масивом рядків, і зниження більш, ніж на 36% в порівнянні з удосконаленим деревом цифрового пошуку. Витрати часу на додавання і видалення елементів в деревовидні структури на кілька порядків менше аналогічних величин для упорядкованого масиву рядків і масиву рядків до таблиці індексів.
Ключові слова: вдосконалене дерево цифрового пошуку, вдосконалене trie-дерево, вдосконалене дерево Patricia, уявлення множин складних об'єктів, застосування при вирішенні оптимізаційних задач.

Введение

Деревья цифрового поиска – это структуры данных, которые представляют множества строк в соответствии со структурой их общих префиксов. Самые фундаментальные из таких деревьев – trie-деревья (лучевой поиск Р. Брианде). В них каждая строка множества представлена ветвью – последовательностью ребер или вершин (в зависимости от выбранного способа реализации), представляющих одну букву строки, начиная от корня дерева и заканчивая листом, родительское ребро (вершина) которого соответствует последней букве самого длинного префикса, отличающего строку от любой другой строки множества [1].

Отмечают два недостатка trie-деревьев: однонаправленные ветвления приводят к созданию дополнительных узлов; trie-деревья содержат два различных типа узлов (внутренние и листовые вершины), что усложняет алгоритмы.

Кроме того, trie-деревья являются ссылочной структурой – содержат отсылки к искомым данным, но не сами данные. Предложенное в работе [2] усовершенствование позволяет не только искать, но и хранить данные в древовидной структуре. Таким образом, исчезает необходимость дополнительно хранить множество строк в отдельной области памяти. В данной работе также предложено решение проблемы однонаправленного ветвления.

Представление данных для описания потенциальных решений оптимизационных задач в виде древовидных структур и множеств последовательностей

Среди методов поиска решений оптимизационных задач выделяют точные методы (прямого перебора, динамического программирования ветвей и границ), приближенные методы (метод множителей Лагранжа, градиентный метод) [3] и эвристические методы. Авторы работы [4] приводят следующую классификацию методов поиска решений: техники, основанные на исчислении (прямой метод, косвенный метод), техники управляемого случайного поиска (имитация отжига, эволюционные алгоритмы), перечислительные техники (динамическое программирование).

Существенное влияние на вычислительные ресурсы (затраты времени и памяти), требуемые для решения задачи, оказывает способ представления данных для описания потенциальных решений. Многие реальные приложения, такие как веб-майнинг, анализ текста, биоинформатика, системная диагностика и распознавание действий, имеют дело с последовательными данными. В этих приложениях решается задача поиска содержательных шаблонов или создания эффективных прогностических моделей [5]. В таких приложениях применяют методы машинного обучения, например, K-means или Support Vector Machine (SVM), к последовательным данным, чтобы находить глубинные шаблоны или создавать эффективные прогностические модели [5]. Однако методы машинного обучения обычно требуют входных данных в виде векторов фиксированной длины, которые неприменимы к последовательностям, что усложняет задачу.

Известное решение в области интеллектуального анализа данных заключается в использовании последовательных шаблонов [6]. Этот подход сначала добывает последовательные шаблоны из набора данных, а затем представляет каждую последовательность в наборе данных как вектор признаков с двоичными компонентами, указывающими, содержит ли эта последовательность конкретный последовательный шаблон. Число последовательных шаблонов часто очень велико. Это приводит к проблемам с высокой размерностью и разреженностью данных [6].

Известна задача кластеризации сложных (составных) объектов для выделения часто повторяющихся последовательностей в потоке данных [7]. В ней рассматриваются объекты, которые описаны как последовательности других, более простых данных. В этом описании важен не только характер компонентов, но и порядок их появления. Ketterlin [7] утверждает, что невозможно кластеризовать сложные объекты без предварительной кластеризации их компонентов; чтобы построить классы сложных объектов, нужен некоторый «словарь» для формулирования этих классов. Этот словарь может состоять из имен классов. Такой подход сделает понятными специалисту-человеку результаты кластеризации, особенно при подходе «без учителя», где очень мало (если таковые имеются) объективных мер для оценки результатов эксперимента.

Для кластеризации сложных объектов Ketterlin [7] выполняет иерархическую кластеризацию компонентов объектов. Листовые вершины соответствуют классам, вершины более высоких уровней иерархии – обобщениям (классам-обобщениям). Каждому классу, в том числе классам-обобщениям, поставлена в соответствие количественная оценка «точности»: очень узкие (или специфические) классы имеют более высокий балл, чем более общие.

Предложенный Ketterlin [7] механизм обобщения для двух последовательностей подобъектов заключается в следующем: классифицируют компоненты объектов, заменяя объект именем наиболее конкретного класса, охватывающего его; преобразуют последовательности компонентов в последовательности меток классов; находят наиболее конкретное обобщение, охватывающее обе последовательности. При обобщении последовательности порядок элементов сохраняется, несколько элементов последовательности могут быть отображены на один и тот же элемент обобщения. Ketterlin [7] указывает, что получить такое обобщение проблематично, так как исчерпывающий поиск в пространстве обобщений имеет экспоненциальную стоимость длины последовательностей.

Бесконечно глубокие бесконечно ветвящиеся деревья используются в байесовских непараметрических методах для вывода распределений на гибких структурах данных [8]. Blei, Griffiths, Jordan [8] используют вложенный китайский ресторанный процесс (nested Chinese restaurant process, nCRP)* в качестве предварительного распределения в байесовской непараметрической модели коллекций документов. Они представляют приложение для поиска информации, в котором документы моделируются как пути вниз по случайному дереву, а динамика преимущественного вложения в nCRP приводит к кластеризации документов в соответствии с разделением тем на нескольких уровнях абстракции.

В генетических алгоритмах [9], относящихся к техникам управляемого случайного поиска, потенциальное решение кодируют строкой – последовательностью символов (двоичных в классическом генетическом алгоритме). Само потенциальное решение называют особью, а множество решений на некоторой итерации – популяцией. Строку, которой кодируют потенциальное решение, называют хромосомой, а отдельный символ этой строки – геном. Потенциальные решения (особи) с лучшими значениями целевой (фитнес) функции принимают участие в «скрещивании» и формировании новой популяции. Таким образом, в генетических алгоритмах множества закодированных потенциальных решений представлены как множества строк. С этими строками на

* Вложенный китайский ресторанный процесс (nested Chinese restaurant process, nCRP) – стохастический процесс, который присваивает распределения вероятности бесконечно глубоким, бесконечно ветвящимся деревьям [8].

каждой итерации выполняют следующие операции: генерируют множество строк-хромосом текущей популяции; вычисляют и запоминают фитнес-функцию для каждой строки; выбирают подмножество строк с наилучшим значением фитнес-функции; преобразуют строки выбранного подмножества для получения решений следующей итерации (новой популяции).

Особенности и свойства усовершенствованных деревьев цифрового поиска

Усовершенствованное дерево цифрового поиска представляет собой дерево цифрового поиска (trie-дерево – метод лучевого поиска Р. Брианде), в вершину которого дополнительно введена ссылка на родительскую вершину, а также сохраняются ссылки на вершины окончания строк [2].

Это позволяет использовать древовидную структуру не только для скоростного поиска, но и для хранения множеств строк. В получаемом таким образом «хранилище строк» каждой строке соответствует уникальный идентификатор. Основные операции: получение строки по идентификатору; поиск строки – получение идентификатора строки по написанию (значение 0 указывает на отсутствие строки в хранилище). Дополнительные операции: добавление и удаление строк.

Особенность авторской реализации усовершенствованного дерева цифрового поиска, апробированной в работах [10-15], состоит в использовании вершин одного типа (нет разделения вершин на внутренние и листовые). Во все вершины введён «признак конца строки» – положительное число для вершин окончания строк (идентификатор строки) и ноль для остальных.

С точки зрения авторов, одним из недостатков практических реализаций деревьев цифрового поиска и trie-деревьев является хранение ссылок на потомков внутри вершины [16-20]. Это приводит либо к различному размеру вершин в одном дереве, либо к выделению избыточной памяти (под хранение максимально возможного количества потомков, равного размеру алфавита). Если информацию о потомках вершины хранить в виде списков [20], [21], это приводит к невозможности осуществлять прямой доступ к потомкам, а значит, замедляет поиск.

Ещё одна особенность предложенных авторами реализаций дерева цифрового поиска [10-15] и усовершенствованного дерева цифрового поиска – любая вершина (независимо от числа потомков) имеет фиксированный размер; её можно хранить в массивах (и других структурах с прямым доступом) и адресовать по номеру. Это достигается за счет того, что ссылки на вершины-потомки вынесены из вершины в упорядоченное множество «массивов потомков». В самой же вершине хранится номер соответствующего «массива потомков».

Структура вершин дерева цифрового поиска (trie-tree) и усовершенствованного дерева цифрового поиска (Adv. Trie-tree) представлена в табл. 1.

Таблица 1 – Структура вершин trie-tree и Adv. Trie-tree.

Поле	Trie-tree	Adv. Trie-tree
SymbNum	Символ (или номер символа в алфавите)	
ParentNum	–	Номер родительской вершины в массиве вершин
StrNum	Идентификатор строки, оканчивающейся в данной вершине	
ChArrNum	Номер «массива потомков» – массива номеров вершин-потомков – в упорядоченном множестве «массивов потомков» *Children* (табл. 2).	

Размер числа, которое используется для хранения ссылки на вершину (*ParentNum*, элемент «массива потомков»), зависит от ожидаемого количества вершин древовидной структуры, а оно больше максимального количества строк, которые предполагается хранить.

Размер числа, которое используется для хранения поля *SymbNum*, зависит от размера алфавита, из символов которого состоят строки. В общем случае, описанное усовершенствованное дерево не является бинарным.

Использование усовершенствованных деревьев цифрового поиска

Авторами работы [10] предложено для описания сложных объектов, представимых в виде последовательностей компонент, использовать их обозначение с помощью последовательностей идентификаторов компонент (или строковых величин). Множество таких сложных объектов будет обозначено с помощью словаря строк. Далее, множества строк предлагается хранить и обрабатывать в древовидной структуре – усовершенствованном дереве цифрового поиска (УДЦП) [2], [10], [11]. Между строками множества и их идентификаторами УДЦП устанавливает взаимно однозначное соответствие. Эта иерархическая структура (УДЦП) обеспечивает обработку множеств последовательностей, при которой общие начальные части последовательностей анализируют единожды. Под обработкой можно понимать, например, поиск последовательности наиболее близкой ко входной или совпадающей с ней.

Работы [12-14] используют УДЦП как средство представления словаря строк, которое обеспечивает скоростной поиск строк. В работе [15] словарь распознавания представлен множеством транскрипций слов словаря (множеством строк, состоящим из транскрипционных символов) и эталонами аллофонов, которые обозначаются транскрипционными символами. Множество транскрипций хранится в УДЦП, распознавание выполняется в процессе обхода этой древовидной структуры.

Интересным представляется применение УДЦП в задачах, описанных в работах [5-8]. Этому вопросу будут посвящены последующие исследования.

Рассмотрим перспективу применения УДЦП в генетических алгоритмах [9]. Хранение множеств хромосом в этой древовидной структуре позволяет сохранять все рассмотренные решения (хромосомы всех особей) и их фитнес-функции. Возможность скоростного поиска хромосомы в УДЦП позволяет использовать ранее вычисленные значения фитнес-функции для особей, встречавшихся в предыдущих популяциях. Благодаря организации УДЦП, рост множества хромосом не приводит к существенному замедлению поиска.

Для процедур вычисления фитнес-функции, представимых в виде простой интеграции (суммы или произведения) неких локальных значений от порядкового номера и значения гена, позволит использовать часть значения фитнес-функции, которое предварительно вычислено для хромосомы, имеющей общую начальную часть с данной.

Такое вычисление аналогично приведенному в работе [15], где мера расхождения между распознаваемой речевой командой вычисляется как сумма мер расхождения между эталонами аллофонов и соответствующими им фрагментами речевого сигнала. Множество речевых команд словаря задано множеством транскрипций – строковых величин, в которых каждый символ обозначает аллофон. Это множество транскрипций хранится в УДЦП. Общие начальные символы транскрипций нескольких речевых команд представлены ветвью древовидной структуры. Значение меры расхождения вдоль ветви вычисляется последовательно вершина за вершиной, начиная от корня.

Для каждой вершины получают частичную меру расхождения с речевым сигналом и границу аллофона в распознаваемом сигнале; их сохраняют в области памяти, ассоциированной с этой вершиной.

Разработка усовершенствованного дерева Патриция на основе усовершенствованного дерева цифрового поиска

Древовидная структура Patricia – это trie-дерево, в котором неразветвленная ветвь сжимается в одно ребро (вершину) [1]. Оно также как и это trie-дерево является ссылочной структурой, то есть позволяет найти ссылку на искомую строку, хранящуюся вне дерева.

Из усовершенствованного trie-дерева (УДЦП) получим усовершенствованное дерево Patricia. Сжатие неразветвлённых ветвей усовершенствованного trie-дерева выполним, храня в поле *SymbNum* вершины дерева не символ, а «ссылку» на «строку вершины». В этом случае каждая строка будет храниться отдельно. Представим ссылку на «строку вершины» в виде её номера во множестве (массиве строк), получим усовершенствованное дерево Patricia, вершина которого имеет ту же структуру, что и вершина усовершенствованного trie-дерева (табл. 1). Усовершенствованное дерево Patricia, как и усовершенствованное trie-дерево, пригодно и для скоростного поиска, и для хранения множеств строк.

Создадим и проанализируем с точки зрения затрат памяти и времени несколько вариантов реализации усовершенствованного дерева Patricia (Adv. Patricia-tree), сравнив их с деревом Patricia (Patricia-tree) и усовершенствованным деревом цифрового поиска (Adv. trie-tree). Реализация этих деревьев описана в табл. 2.

Авторами предложена реализация дерева Patricia с вершинами одного вида – без разделения вершин на внутренние и листовые. Вершины так же, как и описанные выше для trie-tree и Adv. trie-tree, имеют фиксированный размер. Структура вершин дерева Patricia и усовершенствованного дерева Patricia аналогична структуре дерева цифрового поиска и усовершенствованного дерева цифрового поиска (табл. 1). Описание того, как используются поля данной структуры в дереве Patricia и усовершенствованном дереве Patricia, приведено в табл. 3.

Номера вершин-потомков *Children*[*i*] упорядочены по возрастанию поля первого символа «строки вершины», номер (идентификатор) которой хранится в поле *SymbNum* вершин, причём в *Children*[i] входят номера вершин с разными первыми символами «строки вершины».

В дереве Patricia число элементов в массиве «строк вершины» *StrsOfKnots* совпадает с количеством вершин в массиве *KnotHeap*. Каждой вершине ставится в соответствие «строка вершины».

В усовершенствованном дереве Patricia «строки вершины» собраны в массиве *StringData*, причём этот массив не содержит повторений строк. *Alphabet* – дерево цифрового поиска, которое позволяет быстро искать строки в массиве *StringData*. Такой поиск необходим при добавлении строк в усовершенствованное дерево Patricia. *StrKnotConnect*[*i*] – массив номеров вершин, для которых «строкой вершины» является *StringData*[*i*]. Массив номеров вершин *StrKnotConnect* упорядочен по возрастанию.

И в дереве Patricia, и в усовершенствованном дереве Patricia вершина не может иметь разных потомков, у которых первые символы «строки вершины» совпадают. Для обеспечения этого при добавлении новых строк выполняется расщепление вершины (пример см. на рис. 1). Расщепление вершины ведёт к её удалению из дерева, взамен добавляются две новые вершины.

Таблица 2 – Структура Patricia-tree, Adv. trie-tree и разработанного Adv. Patricia-tree

<table>
<tr><th>Дерево / Данные</th><th>Adv. trie-tree</th><th>Adv. Patricia-tree</th><th>Patricia-tree*</th></tr>
<tr><td>KnotHeap</td><td>Массив вершин, структура которых описана в табл.1.</td><td colspan="2">Массив вершин (структура описана в табл. 3).</td></tr>
<tr><td>DeletedKnots</td><td colspan="3">Массив номеров удалённых вершин.</td></tr>
<tr><td rowspan="2">Children</td><td colspan="3">Упорядоченное множество массивов номеров вершин-потомков.</td></tr>
<tr><td>Элементы Children[i] (номера вершин-потомков) упорядочены по возрастанию поля SymbNum этих вершин.</td><td colspan="2">Элементы Children[i] (номера вершин) упорядочены по возрастанию поля первого символа «строки вершины», номер (идентификатор) которой хранится в поле SymbNum этих вершин. В Children[i] не может быть двух различных номеров вершин, у которых бы совпадал первый символ «строки вершины».</td></tr>
<tr><td>DeletedChArr</td><td colspan="3">Массивов номеров удалённых «массивов номеров вершин-потомков».</td></tr>
<tr><td>Strings</td><td colspan="2">Массив номеров вершин, в которых оканчиваются строки, причём если Strings[i] = k, k≠0, то KnotHeap[k].StrNum = i</td><td>–</td></tr>
<tr><td>DeletedStrings</td><td colspan="3">Массив идентификаторов (номеров) удалённых строк. Упорядочен по возрастанию.</td></tr>
<tr><td rowspan="3">Множество «строк вершины»</td><td rowspan="3">–</td><td>Alphabet – trie-tree для быстрого поиска «строк вершины», хранящихся в массиве StringData.</td><td rowspan="3">StrsOfKnots – массив «строк вершин». Вершине с номером i соответствует строка с номером i.

DltdStrsKn – массив номеров удалённых «строк вершины»</td></tr>
<tr><td>StringData – массив строк, хранящий уникальные написания «строк вершины»</td></tr>
<tr><td>StrKnotConnect – множество массивов переменной длины, хранящих номера вершин в массиве KnotHeap.
StrKnotConnect[i] – массив номеров вершин, для которых «строкой вершины» является StringData[i]. Каждый массив StrKnotConnect[i] упорядочен по возрастанию номера вершины.</td></tr>
<tr><td>Массив строк StrArray</td><td>–</td><td>–</td><td>Массив строк, совместно с которым используется Patricia-tree</td></tr>
</table>

Таблица 3 – Структура и описание полей вершин Patricia-tree и Adv. Patricia-tree.

Поле	Patricia-tree	Adv. Patricia-tree
SymbNum	Номер строки в массиве «строк вершины» *StrsOfKnots*	Идентификатор строки вершины в дереве цифрового поиска *Alphabet*.
ParentNum	–	Номер родительской вершины в массиве вершин.
StrNum	Для вершины, в которой оканчивается строка, номер этой строки в массиве строк *StrArray*.	Идентификатор строки, оканчивающейся в данной вершине
ChArrNum	Номер «массива потомков» – массива номеров вершин-потомков – в упорядоченном множестве «массивов потомков» *Children* (табл. 2).	

* Для простоты сравнения Patricia-tree реализовано авторами с помощь вершин одного вида (без введения дополнительных листовых вершин)

На рис. 1 число в верхней части окружности обозначает номер вершины, а последовательность символов в нижней части окружности – «строку вершины», символом ⮟ обозначены вершины окончания строк. На рис. 1а) показано дерево Patricia для строк: «*он*», «*она*», «*они*», «*след*», «слон». Добавление слова «*опт*» приводит к расщеплению вершины номер 8. Усовершенствованные деревья Patricia, соответствующие приведенным на рис. 1, показаны на рис. 2.

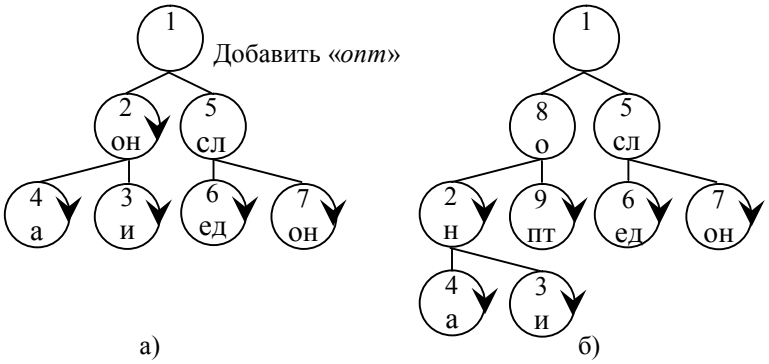
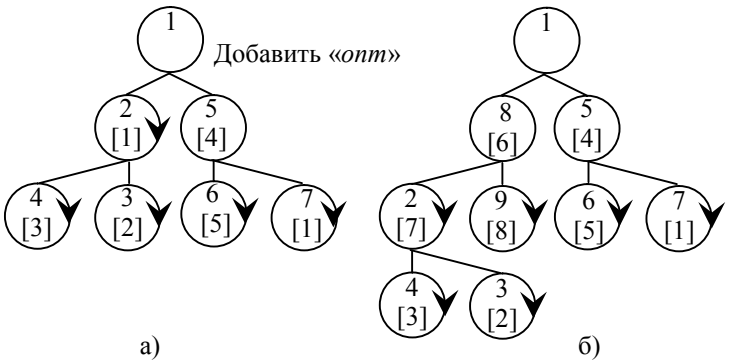


Рисунок 1 – Расщепление вершины при добавлении слова «*опт*» в дерево Patricia

В дереве Patricia удаление вершины с номером *i* ведёт к удалению «строки вершины» из множества «строк вершины» путём очистки строки *StrsOfKnots* [*KnotHeap*[*i*].*SymbNum*] и внесения *KnotHeap*[*i*].*SymbNum* в массив номеров удалённых «строк вершины» *DltdStrsKn*.



Множество «строк вершины»

Идентификатор в trie-tree *Alphabet*		Написание в *StringData*	
а)	б)	а)	б)
1		он	
2		и	
3		а	
4		сл	
5		ед	
	6		о
	7		н
	8		пт

Рисунок 2 – Расщепление вершины при добавлении слова «*опт*» в усовершенствованное дерево Patricia

В усовершенствованном дереве Patricia каждую «строку вершины» хранят единожды. Если у нескольких вершин совпадают «строки вершины», то в этих вершинах хранят один и тот же идентификатор (рис. 2). Информация о том, с какими вершинами связана определённая «строка вершины» хранится в упорядоченном множестве массивов переменной длины *StrKnotConnect*. Эта информация используется для удаления из *StringData* и *Alphabet* «строк вершины», которые больше не связаны ни с одной вершиной.

Если помещать в *StrKnotConnect* ссылки на все вершины дерева, то суммарное количество ссылок будет на 2 меньше числа вершин (так как вершина с номером 0 не используется, а вершина с номером 1 – корневая вершина – не имеет строки вершины).

Число «строк вершины», состоящих из 1 или 2 символов, невелико. Их удаление из *StringData* и *Alphabet* при удалении соответствующих вершин усовершенствованного дерева Patricia не приведёт к существенному сокращению затрат памяти. В то же время, как показал описанный ниже эксперимент, если не хранить в *StrKnotConnect* ссылки на вершины, для «строк вершины», чья длина не превышает 2, то можно значительно сократить затраты памяти. В таблице 4 модификация усовершенствованного дерева Patricia, в которой *StrKnotConnect* хранит ссылки на все вершины дерева помечена «(полное)». Модификация, в которой в *StrKnotConnect* помещают ссылки только для «строк вершины», чья длина превышает 2, помечена «(сокр.)».

Численное исследование усовершенствованного дерева Patricia

Численное исследование усовершенствованного дерева Patricia проведено на словаре словоформ русского языка, содержащем 1 987 907 уникальных строк (табл. 4).

Таблица 4 – Оценка затрат памяти древовидных структур

Элемент данных	Характеристика	Обозначение	Trie-tree	Adv. trie-tree	Adv. Patricia-tree		Patricia-tree
					полное	сокр.	
KnotHeap	Кол-во вершин	*pKc*	3026633		2252594		
	Размер вершины*, байт	*pKs*	12	16	16		12
Children	Кол-во массивов потомков	*pCh_c*	1713867		939828		
	Суммарное число ссылок на вершины-потомки	*pChKLc*	3026631		2252592		
	Размер ссылки на вершину-потомка, байт	*pKLs*	4				
Strings	Кол-во строк	*pSc*		1987907			
	Размер ссылки на вершину, байт	*pKLn*		4			
Alphabet. KnotHeap	Кол-во вершин	*aKc*			25198	25230	
	Размер вершины, байт	*aKs*			8		
Alphabet. Children	Кол-во массивов потомков	*aCh_c*			17839		
	Суммарное число ссылок на вершины-потомки	*aChKLc*			25196	25228	
	Размер ссылки на вершину-потомка, байт	*aKLs*			2		
StringData	Кол-во строк	*aSc*			10309	10386	
	Суммарный размер, байт	*aStr_s*			106264	106803	
StrKnotConnect	Кол-во массивов ссылок	*aSc*			10309	10386	
	Суммарное число ссылок на вершины Patricia-tree	*aSLPc*			2252593	56469	
	Размер ссылки на номера вершины Patricia-tree, байт	*aSLPs*			4		
Массив «строк вершины» *StrsOfKnots*	Кол-во «строк вершины»	*pKc*					2252594
	Размер множества «строк вершины», байт	*KnStr_s*					14289599
Множество строк *StrArr*	Кол-во строк	*pSc*	1987907				1987907
	Размер множества строк, байт	*StrArr_s*	32115425				32115425

* Размер структуры данных, описывающей вершину, выровнен по границе 4 байта.

В табл. 4 показаны затраты памяти на хранение одного и того же множества строк в усовершенствованном trie-дереве (Adv. trie-tree) и усовершенствованном дереве Patricia (Adv. Patricia-tree), а также затраты памяти для дерева Patricia (Patricia-tree), как ссылочной (обеспечивающей только поиск) структуры, совместно с массивом строк (StrArr).

Затраты памяти на усовершенствованное дерево цифрового поиска (Adv. Patricia-tree) будем рассматривать в двух вариантах: с сохранением в *StrKnotConnect* всех ссылок на вершины усовершенствованного дерева Patricia (полное) и с сохранением в *StrKnotConnect* ссылок на только на те вершины, у которых длина «строки вершины» больше 2 (сокр.).

Рассмотрим затраты памяти на хранение вершины усовершенствованного дерева Patricia – *pKs* и вершины дерева цифрового поиска *Alphabet – aKs*.

Вершина усовершенствованного дерева Patricia включает поля: *SymbNum, ParentNum, StrNum, ChArrNum*. Исходя из количества вершин, строк и количества массивов потомков, для полей *ParentNum, StrNum* и *ChArrNum* нужно по 4 байта.

Так как для набора из 1987907 уникальных строк в проведенном эксперименте было создано усовершенствованное дерево Patricia, имеющее около 10 тысяч уникальных «строк вершины», то для хранения их идентификаторов в вершине дерева Patricia (поле *SymbNum*) необходимо 2 байта.

Из-за выравнивания по границе 4 байта, *pKs* = 16.

В trie-tree *Alphabet* количество вершин, строк и массивов потомков позволяет для полей, *StrNum* и *ChArrNum* выделять по 2 байта. Полю *SymbNum* в вершине алфавита достаточно 1 байта. Таким образом, учитывая выравнивание по границе 4 байта, *aKs* = 8.

Вершины дерева Patricia, в отличие от вершины усовершенствованного дерева Patricia, не содержат поле *ParentNum* (размер 4 байта). Таким образом, для него размер вершины *pKs* = 12.

Подсчёт затрат памяти на хранение массива строк. Для простоты будем считать, что 1 символ занимает *aSybm_s* = 1 байт, строки оканчиваются специальным символом – признаком конца строки. Размер данных P(*s*) для хранения строки *s* составляет:

$$\text{P}(s) = aSybm\_s \cdot (\text{длина}(s) + 1) + Ls,$$

где *Ls* – размер ссылки на область памяти начала строки.

Тогда затраты памяти на хранение массивов и множеств строк будут следующими.

Количество байт для хранения исходного множества уникальных строк *StrArr,* совместно с которым используется дерево Patricia, в таблице 4 обозначено *StrArr_s* и вычисляется как:

$$StrArr\_s = aSymb\_s \cdot \sum_{m_i=1}^{lp} (i+1) \cdot m_i + Ls \cdot pSc,$$

где i – длина строки; lp – максимальная длина строки во множестве строк *StrArr*; m_i – количество строк длины i во множестве строк *StrArr*, *pSc* – количество строк во множестве строк *StrArr*, $pSc = \sum_{n_i=1}^{lp} m_i$.

Количество байт для хранения массива «строк вершины» *StrsOfKnots* (используется дереве Patricia), в таблице 4 обозначено *KnStr_s* и вычисляется как:

$$KnStr\_s = aSymb\_s \cdot \sum_{n_i=1}^{la} (i+1) \cdot n_i + Ls \cdot pKc,$$

где *la* – максимальная длина строки в массиве «строк вершины»; n_i – количество строк длины *i* в массиве «строк вершины»; *pKc* – количество строк во множестве «строк вершины» *StrsOfKnots* (совпадает с количеством вершин дерева Patricia), $pKc = \sum_{n_i=1}^{la} n_i$.

Количество байт для хранения множества «строк вершины» *StringData* (используется в усовершенствованном дереве Patricia, не содержит одинаковых строк), в таблице 4 обозначено *aStr_s* и вычисляется как:

$$aStr\_s = aSymb\_s \cdot \sum_{n_i=1}^{la} (i+1) \cdot x_i + Ls \cdot aSc ,$$

где *la* – максимальная длина строки в массиве «строк вершины»; x_i – количество строк длины *i* в множестве «строк вершины» *StringData*; a*Kc* – количество строк во множестве «строк вершины» *StringData*, $aKc = \sum_{n_i=1}^{la} x_i$.

Величина *KnStr_s* зависит от распределения длин «строк вершины» в массиве «строк вершины» *StrsOfKnots*, а величина *StrArr_s* – от распределения длин строк в исходном массиве строк. Эти распределения для рассматриваемого примера приведены в табл. 5.

Таблица 5 – Распределение длин «строк вершины» и строк массива строк

i	n_i	m_i	x_i		***i***	n_i	m_i	x_i	***i***	n_i	m_i	x_i
			полн.	*сокр.*								
1	1579346	10	33	33	**10**	301	259434	243	**19**		11775	
2	616778	119	492	569	**11**	184	259505	168	**20**		5643	
3	33962	1332	1760		**12**	92	241665	84	**21**		2857	
4	13871	6910	2250		**13**	44	200377	42	**22**		1297	
5	3342	24714	1796		**14**	30	155711	30	**23**		547	
6	2022	57910	1354		**15**	11	109789	11	**24**		304	
7	1206	113576	930		**16**	0	69510	0	**25**		96	
8	873	176170	679		**17**	1	41275	1	**26**		41	
9	527	225165	434		**18**	2	22165	2	**27**		10	

Кроме введённых в табл. 3 обозначений, используем *Ls* – размер ссылки на области памяти начала и окончания массива переменной длины. Используем их при оценке затрат памяти на хранение упорядоченных множеств массивов переменной длины *Children*, *StrKnotConnect*, *Alphabet.Children*:

$$\text{P}(Children) = pCh\_c \cdot 2 \cdot Ls + pChKLc \cdot pKLs,$$
$$\text{P}(Alphabet.Children) = aCh\_c \cdot 2 \cdot Ls + aChKLc \cdot aKLs,$$
$$\text{P}(StrKnotConnect) = aSc \cdot 2 \cdot Ls + aSLPc \cdot aSLPs.$$

Расчёт затрат памяти на хранение древовидных структур из таблицы выполним по формулам, приведенным ниже.

$$\text{P}(Adv.\ trie\text{-}tree) = pKc \cdot pKs + \text{P}(Children) + pSc \cdot pKLn$$
$$\text{P}(Adv.\ Patricia\text{-}tree) = pKc \cdot pKs + \text{P}(Children) + pSc \cdot pKLn + \text{P}(Alphabet),$$

где

$$P(Alphabet) = aKc \cdot aKs + P(Alphabet.Children) + P(StrKnotConnect) + aStr\_s$$

Затраты памяти на хранение дерева Patricia совместно с массивом строк:

$$P(Patricia\text{-}tree) = pKc \cdot pKs + P(Children) + KnStr\_s + StrArr\_s,$$

В табл. 6 собраны результаты расчёта затрат памяти усовершенствованного trie-дерева (Adv. trie-tree) и усовершенствованного дерева Patricia (Adv. Patricia-tree), хранящих одно и то же множество строк, а также дерева Patricia (Patricia-tree), как ссылочной (обеспечивающей только поиск) структуры, совместно с массивом строк (*StrArr*).

Таблица 6 – Численная оценка затрат памяти структур данных. Результаты расчета

Затраты памяти	Trie-tree	Adv. trie-tree	Adv.Patricia-tree		Patricia-tree	Упорядоченный массив	Массив с таблицей индексов
			полное	сокращ.			
Абсолютное значение, байт	94252481	82195216	70115920	61332899	89965144	32115425	40067053
Относительное значение, %	104,77	91,36	77,94	68,17	100	35,70	44,54

Результаты эксперимента по оценке времени выполнения основных операций приведены в табл. 7. В ней приведено отношение времени выполнения операции указанной в структуре данных ко времени выполнения той же операции в упорядоченном массиве.

Таблица 7 – Относительные средние затраты времени на выполнение основных операций

Представление данных / Операция	Adv. trie-tree	Adv. Patricia-tree	Упорядоченный массив	Массив с таблицей индексов
Получение написания	3,09	8,73	1	1,02
Поиск	0,63	0,89	1	эксперимент не проводился
Удаление	$1{,}25 \cdot 10^{-4}$	$4{,}63 \cdot 10^{-5}$	1	$4{,}77 \cdot 10^{-2}$
Добавление	$2{,}02 \cdot 10^{-3}$	$1{,}22 \cdot 10^{-3}$	1	$4{,}88 \cdot 10^{-2}$

Из табл. 7 видно, что на получение написания строки из усовершенствованного дерева цифрового поиска тратится в 3 раза больше времени, чем на аналогичную операцию в упорядоченном массиве строк, а в усовершенствованном дереве Patricia – в 9 раз больше. При этом, усовершенствованное дерево цифрового поиска выполняет поиск строк значительно быстрее бинарного поиска в упорядоченном массиве. В проведенном эксперименте затраты на поиск строки в усовершенствованном дереве Patricia составили 0,89 от времени бинарного поиска в упорядоченном массиве.

Отметим, что операция удаления и добавления элементов в упорядоченный массив большого размера требует существенных затрат памяти. Аналогичные операции для массива с таблицей индексов требуют на два порядка меньше времени. Операции добавления и удаления строки для усовершенствованного дерева цифрового поиска и усовершенствованного дерева Patricia требуют на три и более порядка меньше времени, чем аналогичные операции для упорядоченного массива.

Выводы

В ряде методов решения оптимизационных задач потенциальные решения представимы в виде множеств последовательностей и бесконечно глубоких бесконечно ветвящихся деревьев. Деревья цифрового поиска, организованные по методу Р. Бранде (лучевая память, trie-tree), являются бесконечно глубокими бесконечно ветвящимися деревьями. Они используются как ссылочная структура для скоростного поиска строк в словарях.

Усовершенствованные деревья цифрового поиска позволяют хранить, быстро искать и обрабатывать множества строк. Их можно использовать, в том числе при распознавании сложных объектов, представимых как последовательности более простых объектов (пофонемное распознавание речи).

Высказано предположение, что усовершенствованные деревья цифрового поиска могут найти применение в генетических алгоритмах при некоторых видах фитнес-функций.

На основе усовершенствованного дерева цифрового поиска разработано усовершенствованное дерево Patricia для хранения и поиска строковых величин, которое обеспечило для рассмотренного словаря строк снижение затрат памяти более, чем на 32% в сравнении с использованием дерева Patricia, как ссылочной структуры, одновременно с массивом строк.

В этом же случае усовершенствованное дерево цифрового поиска требует 8,24% меньше затрат памяти, чем дерево Patricia.

Рассмотренные древовидные структуры медленнее осуществляют получение написания строки: усовершенствованное дерево цифрового поиска – в 3 раза, а усовершенствованное дерево Patricia – почти в 9 раз, зато позволяют быстрее искать по сравнению с бинарным поиском. Отношение времени поиска в указанных древовидных структурах ко времени бинарного поиска составило 0,63 для усовершенствованного дерева цифрового поиска и 0,89 для усовершенствованного дерева Patricia.

В пользу применения древовидных структур при работе с большими массивами строк говорит также тот факт, что затраты времени на добавление и удаление элементов в древовидные структуры на несколько порядков меньше аналогичных величин для упорядоченного массива строк и массива строк с таблицей индексов.

Список литературы

1. Magner, Abram, Profles of PATRICIA Tries [Электронный ресурс] // Open Access Dissertations. – 2015. – 1311. – URL: https://docs.lib.purdue.edu/open_access_dissertations/1311
2. Патент України № 78806 «Пристрій для збереження і пошуку рядкових величин та спосіб збереження і пошуку рядкових величин» [Текст] / Дорохіна Г. В. ; заявник та власник: Інститут проблем штучного інтелекту, винихідник // Промислова власність. – Бюл. №5. – 25.04.2007.
3. Хорольский В. Я. Прикладные методы для решения задач электроэнергетики и агроинженерии: учебное пособие [Текст] / В. Я. Хорольский, М. А. Таранов и др. – М. : Форум : НИЦ ИНФРА-М, 2015. – 176 с.
4. Sanjay Jangid. Evolutionary Algorithms: A Critical Review and its Future Prospects [Электронный ресурс] / Sanjay Jangid, Ravinder Puri and Thilak kumar // International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Conference Proceeding | NCRBDIC-19, March 2019. – URL: http://www.ijtrd.com/papers/IJTRD20409.pdf
5. Sqn2Vec: Learning Sequence Representation via Sequential Patterns with a Gap Constraint Nguyen, Dang & Luo, Wei & Nguyen, Tu & Venkatesh, Svetha & Phung, Dinh [Электронный ресурс] // Machine Learning and Knowledge Discovery in Databases. – 2018. – European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II. – P. 569–584.
URL: http://www.ecmlpkdd2018.org/wp-content/uploads/2018/09/362.pdf

6. Fradkin D. Mining sequential patterns for classification [Текст] / Dmitriy Fradkin and Fabian Mörchen // Knowledge and Information Systems. – 2015. – № 45 (3). – P. 731–749.
7. Ketterlin A. Clustering sequences of complex objects [Электронный ресурс] // KDD'97 Proceedings of the Third International Conference on Knowledge Discovery and Data Mining [Newport Beach, CA – August 14 - 17, 1997]. – P. 215–218. – URL: https://www.aaai.org/Papers/KDD/1997/KDD97-043.pdf [дата обращения: 10.05.2019]
8. David M. Blei, Thomas L. Griffiths, Michael I. Jordan The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies [Электронный ресурс] // Submitted on 3 Oct 2007 (v1), last revised 27 Aug 2009 (version, v3). – URL: https://arxiv.org/abs/0710.0845
9. Скобцов А. Ю. Основы эволюционных вычислений [Текст] : учебное пособие / А. Ю. Скобцов. – Донецк : ДонНТУ, 2008. – 326 с.
10. Дорохина Г. В. Способ представления множеств последовательностей [Электронный ресурс] / Г. В. Дорохина, В. Н. Павлыш [Электронный ресурс] // Информатика и кибернетика. – № 1(3). – Донецк : ДонНТУ, 2016. – С. 55-64. – URL : http://infcyb.donntu.org/IC_3.pdf
11. Дорохина Г. В. Сравнение затрат памяти для метода деревьев цифрового поиска и его усовершенствования [Текст] / Г. В. Дорохина // Искусственный интеллект . – 2009. – № 4. – С. 338–343.
12. Свідотство про реєстрацію авторського права на твір Комп'ютернапрограма «Модуль декларативного морфологічного аналізу слів російської мови» («РДМА_ІПШІ») [Текст] / Дорохіна Г. В.; заявник та власник : Інститут проблем штучного інтелекту. – № 43188 від 09.04.2012.
13. Дорохина Г. В. Модуль морфологического анализа без словаря слов русского языка [Текст] / Г. В. Дорохина, В. Ю. Трунов, Е. В. Шилова // Искусственный интеллект. – 2010. – № 2. – С. 32–36.
14. Дорохина Г.В. Ограничение количества гипотез фразы при распознавании слитной речи [Текст] / Г. В. Дорохина // Известия ТРТУ – 2005. – № 10. – С. 54–60.
15. Дорохина Г.В. Модификация алгоритма DTW для пофонемного распознавания слов [Текст] // Проблемы искусственного интеллекта. – № 0, 2015 (1). – С. 38 – 49.
16. Таранов И. С. Использование префиксного дерева для хранения и поиска строк во внешней памяти [Электронный ресурс] / И. С. Таранов // Труды Института системного программирования РАН. – Vol. 20, 2011. – P. 283–296. – URL : https://cyberleninka.ru/article/n/ispolzovanie-prefiksnogo-dereva-dlya-hraneniya-i-poiska-strok-vo-vneshney-pamyati (дата обращения: 29.11.2019).
17. James Aspnes Notes on Data Structures and Programming Techniques [Электронный ресурс] (CPSC 223, Spring 2018). 2019-05-17. – URL: http://www.cs.yale.edu/homes/aspnes/classes/223/notes.pdf (дата обращения: 29.11.2019).
18. Thenmozhi M. An Analysis on the Performance of Tree and Trie based Dictionary Implementations with Different Data Usage Models [Электронный ресурс] / M. Thenmozhi and H. Srimathi // Indian Journal of Science and Technology. – February 2015. – Vol. 8(4). – P. 364–375. – URL: http://www.indjst.org/index.php/indjst/article/view/59865.
19. Метод быстрого поиска узлов семантической сети по точному совпадению словоформы [Электронный ресурс] / Покид А.В., Клименков С.В., Цопа Е.А., Жмылёв С.А., Ткешелашвили Н.М. // Известия высших учебных заведений. Приборостроение. – Vol. 60, no. 10. – 2017. – С. 932–939. URL: https://cyberleninka.ru/article/n/metod-bystrogo-poiska-uzlov-semanticheskoy-seti-po-tochnomu-sovpadeniyu-slovoformy (дата обращения: 29.11.2019).
20. Структуры данных и алгоритмы [Текст] / Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д.; пер. с англ. – М. : Издательский дом «Вильямс», 2003. – 384 с.
21. Деревья [Электронный ресурс] // Структуры данных и алгоритмы / Ахо Альфред В., Хопкрофт Джон Э., Ульман Джеффри Д. – : М. : Издательский дом «Вильямс», 2018. – с. 79-104. URL: http://www.williamspublishing.com/PDF/978-5-8459-1610-5/part.pdf (дата обращения: 29.11.2019).

References

1. Magner, Abram, "Profles of PATRICIA Tries" (2015). // Open Access Dissertations. 1311. URL: https://docs.lib.purdue.edu/open_access_dissertations/1311
2. Patent of Ukraine No. 78806 “Device for saving and searching for lowercase values and method for saving and searching for lowercase values” Owner: Institute of problems of artificial intelligence, inventor Dorokhina G.V. // Industrial property. Bull. No. 5, 25.04.2007.
3. Khorolsky V.Ya. Applied methods for solving problems of electric power and agricultural engineering: textbook / V.Ya. Khorolsky, M.A. Taranov, etc. – M.: Forum: SIC INFRA-M, 2015. – 176 p.

4. Sanjay Jangid, Ravinder Puri and Thilak kumar Evolutionary Algorithms: A Critical Review and its Future Prospects // International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Conference Proceeding | NCRBDIC-19, March 2019,
URL: http://www.ijtrd.com/papers/IJTRD20409.pdf
5. Nguyen, Dang & Luo, Wei & Nguyen, Tu & Venkatesh, Svetha & Phung, Dinh. (2018). Sqn2Vec: Learning Sequence Representation via Sequential Patterns with a Gap Constraint // Machine Learning and Knowledge Discovery in Databases. – European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II. – pp. 569-584. URL: http://www.ecmlpkdd2018.org/wp-content/uploads/2018/09/362.pdf
6. Dmitriy Fradkin and Fabian Mörchen. Mining sequential patterns for classification. // Knowledge and Information Systems, 45(3):731-749, 2015.
7. Ketterlin A. Clustering sequences of complex objects // KDD'97 Proceedings of the Third International Conference on Knowledge Discovery and Data Mining [Newport Beach, CA – August 14 - 17, 1997]. – P. 215-218 . URL: https://www.aaai.org/Papers/KDD/1997/KDD97-043.pdf [access date: 10.05.2019]
8. David M. Blei, Thomas L. Griffiths, Michael I. Jordan The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies (Submitted on 3 Oct 2007 (v1), last revised 27 Aug 2009 (version, v3)). URL: https://arxiv.org/abs/0710.0845
9. Skobtsov A.Yu. Bases of evolutionary calculations. – Textbook. – Donetsk: DonNTU, 2008. – 326 p.
10. Dorokhina G.V., Pavlysh V.N. The method of representation of sets of sequences // Informatics and Cybernetics, No. 1(3), – Donetsk: DonNTU, 2016. – pp. 55-64.
URL: http://infcyb.donntu.org/IC_3.pdf
11. Dorokhina G.V. Comparison of memory costs for the digital search tree method and its improvement // Artificial intelligence. – 2009. – No. 4. – pp. 338–343.
12. Certificate of registration of copyright for the work Computer program “Module of declarative morphological analysis of words of the Russian language» (“RDMA_IPSHI”) No. 43188 from 09.04.2012. Author: G.V. Dorokhina, Owner: Institute of problems of artificial intelligence.
13. Dorokhina G.V., Trunov V.Yu., Shilova E.V. Module of morphological analysis of Russian words without dictionary // Artificial intelligence. – 2010. – No. 2. – pp. 32–36.
14. Dorokhina G.V. Limiting the number of hypotheses of a phrase in recognition of a merged speech // Izvestiya TRTU – 2005. – No. 10. – pp. 54–60.
15. Dorokhina G.V. Modification of the DTW algorithm for phonemic word recognition // Problems of artificial intelligence. – No. 0, 2015 (1). – pp. 38 – 49.
16. Taranov I.S. Using a prefix tree for storing and searching strings in external memory // Proceedings of the Institute of system programming (RAS), vol. 20, 2011, pp. 283-296. – URL: https://cyberleninka.ru/article/n/ispolzovanie-prefiksnogo-dereva-dlya-hraneniya-i-poiska-strok-vo-vneshney-pamyati (access date: 29.11.2019).
17. James Aspnes Notes on Data Structures and Programming Techniques (CPSC 223, Spring 2018). 2019-05-17. – URL: http://www.cs.yale.edu/homes/aspnes/classes/223/notes.pdf (access date: 29.11.2019).
18. M. Thenmozhi* and H. Srimathi An Analysis on the Performance of Tree and Trie based Dictionary Implementations with Different Data Usage Models // Indian Journal of Science and Technology, Vol. 8(4), 364–375, February 2015. – URL: http://www.indjst.org/index.php/indjst/article/view/59865.
19. Pokid A.V., Klimenkov S.V., Tsopa E.A., Zhmylev S.A., Tkeshelashvili N.M. Method of fast search of semantic network nodes by exact word-form coincidence // News of higher educational institutions. Instrument engineering. – Vol. 60, No. 10. – 2017, pp. 932-939.
URL: https://cyberleninka.ru/article/n/metod-bystrogo-poiska-uzlov-semanticheskoy-seti-po-tochnomu-sovpadeniyu-slovoformy (access date: 29.11.2019).
20. Aho, Alfred, V. Hopcroft, John, Jeffrey, D. Structures and algorithms.: Transl. for eng. – M.: Publishing house "Williams", 2003. — 384 p.
21. Trees //Data structures and algorithms/ Aho Alfred, V., Hopcroft John E., Ullman Jeffrey D. – M.: Publishing house “Williams”, 2018. – pp. 79-104.
URL: http://www.williamspublishing.com/PDF/978-5-8459-1610-5/part.pdf (access date: 29.11.2019).

RESUME

V. I. Finayev, G. V. Dorokhina

Applications of Improved Digital Search Trees

Computing resources for solving problems depend on how potential solutions are presented. Potential solutions for a number of methods for solving optimization problems are represented as sets of sequences and tree structures. The features of improved digital search trees, data on the experience of their application in tasks related to storing and searching strings and recognition of complex objects that are represented as sequences of simpler objects (phonemic speech recognition) are presented. It is suggested that improved digital search trees can be used in genetic algorithms for some types of fitness functions.

On the basis of the improved digital search tree, an improved Patricia tree for storing and searching string values was developed, which provided for the considered string dictionary a memory cost reduction of more than 31% in comparison with the use of the Patricia tree as a reference structure, simultaneously with an array of the strings and a reduction of more than 36% compared to the improved digital search tree.

The time to get the string written in the improved digital search tree and the improved Patricia tree is 3 and 9 times longer than the time to get a string written from the array, respectively. But the ratio of search time in these tree-like structures to the time of binary search in an orderly array of rows 0.63 and 0.89, respectively. And the time of deleting and inserting elements in the considered tree-like structures is several orders of magnitude less than similar values for an ordered array of rows.

РЕЗЮМЕ

В. И. Финаев, Г. В. Дорохина

Применения усовершенствованных деревьев цифрового поиска

Вычислительные ресурсы при решении задач зависят от способа представления потенциальных решений. Потенциальные решения для ряда методов решения оптимизационных задач представимы в виде множеств последовательностей и древовидных структур. Приведены особенности усовершенствованных деревьев цифрового поиска, данные об опыте их применения в задачах, связанных с хранением и поиском строк и с распознаванием сложных объектов, представимых как последовательности более простых объектов (пофонемное распознавание речи). Высказано предположение, что усовершенствованные деревья цифрового поиска могут найти применение в генетических алгоритмах при некоторых видах фитнес-функций.

На основе усовершенствованного дерева цифрового поиска разработано усовершенствованное дерево Patricia для хранения и поиска строковых величин, которое обеспечило для рассмотренного словаря строк снижение затрат памяти более, чем на 31% в сравнении с использованием дерева Patricia, как ссылочной структуры, одновременно с массивом строк, и снижение более, чем на 36% в сравнении с усовершенствованным деревом цифрового поиска.

Время получения написания строки в усовершенствованном дереве цифрового поиска и усовершенствованном дереве Patricia больше времени получения написания строки из массива в 3 и 9 раз, соответственно. Зато отношение времени поиска в этих древовидных структурах ко времени бинарного поиска в упорядоченном массиве строк 0,63 и 0,89, соответственно. А время удаления и вставки элементов в рассматриваемые древовидные структуры на несколько порядков меньше аналогичных величин для упорядоченного массива строк.

Статья поступила в редакцию 12.12.2019.