На правах рукописи

Андрианов Игорь Александрович

АНАЛИЗ И РАЗРАБОТКА СПОСОБОВ ИНДЕКСИРОВАНИЯ ТЕКСТОВ НА ОСНОВЕ ОБОБЩЕННЫХ И НЕПЛОТНЫХ СУФФИКСНЫХ ДЕРЕВЬЕВ

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург – 2005

Работа выполнена в Вологодском государственном техническом университете

Научный руководитель – кандидат технических наук, доцент Ржеуцкая С. Ю.

Официальные оппоненты – доктор технических наук, профессор Геппенер В.В.
кандидат технических наук Тупицин А.В.

Ведущая организация – Вологодский научно-координационный центр ЦЭМИ РАН

Защита состоится "13" декабря 2005 года в 10^{00} часов на заседании диссертационного совета Д 212.238.01 Санкт-Петербургского государственного электротехнического университета "ЛЭТИ" имени В.И. Ульянова (Ленина) по адресу: 197376, Санкт-Петербург, ул. Проф. Попова, 5.

С диссертацией можно ознакомиться в библиотеке университета

Автореферат разослан "10" ноября 2005г.

Ученый секретарь
диссертационного совета Пантелеев М.Г.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность проблемы. В связи с развитием компьютерных технологий в различных областях деятельности в настоящее время накоплены огромные массивы информации, в том числе условно называемой "текстовой". Это может быть HTML-наполнение web-сайтов, тематические электронные библиотеки, архивы сообщений электронной почты, исходный программный код ведущихся проектов и др.

Для того, чтобы эффективно выполнять полнотекстовый информационный поиск в данных массивах, необходимо иметь соответствующее математическое и программное обеспечение. При этом важную роль играет как скорость поиска, так и возможность выполнять различные его виды при гибкой формулировке запросов.

На сегодняшний момент хорошо теоретически проработан и реализован в большом количестве поисковых систем словарный поиск – т.е. поиск слов и их грамматических форм. Несколько хуже обстоит дело с другими видами поиска, которые также важны и востребованы на практике – поиск по регулярным выражениям или их подмножествам, поиск по сходству и некоторые другие. Специфика таких задач состоит в том, что не предполагается естественного разбиения текста на некоторые составные единицы (слова, предложения и т.п.), т. е. сам термин «текстовый документ» рассматривается в широком смысле как простая последовательность символов.

Вопросам проектирования структур данных и алгоритмов информационного поиска посвящены работы Р. Баеза-Ятеса, Г. Гоннета, Д. Гасфилда, Э. Укконена, С. Куртца и др. Из отечественных публикаций можно выделить работы Л.М. Бойцова, Е.В. Андриенко, А.В.Сокирко, О.С. Бартунова, Т.Г. Сигаева и др.

Существуют различные способы хранения текстовой информации, например, в виде файловых коллекций, в документо-ориентированных и фактографических базах данных. В любом случае, для ускорения или даже обеспечения возможности выполнения поиска необходима предварительная обработка текста с целью создания дополнительных структур данных для поддержки поиска – индексов.

Перспективным подходом к построению индексов для рассматриваемых видов поиска является использование суффиксных деревьев (СД) и их расширений. СД представляет собой одну из наиболее универсальных структур данных для поддержки поиска в символьных последовательностях – известно по крайней мере несколько десятков задач, эффективно решаемых с её помощью. Однако, в силу своей универсальности СД имеют сравнительно высокие требования к памяти. Поэтому представляется целесообразным использовать данную структуру для индексирования коллекций документов среднего размера (до нескольких гигабайт).

Применение СД для задач полнотекстового поиска исследовано в ряде работ. Однако, в этой области имеется ещё много нерешенных вопросов как теоретического, так и прикладного характера. В частности, известные варианты представления данной структуры в памяти позволяют эффективно выполнять лишь определённые подмножества операций над СД – конкретный способ представления дерева выбирается исходя из решаемой задачи. Но для того, чтобы применять СД как основу индекса сразу для нескольких видов поиска, крайне желательно эффективное выполнение всего набора операций. Важным является и тот факт, что

при построении СД во внешней памяти время работы существующих вероятностных алгоритмов для реальных входных данных может во много раз отличаться от теоретического среднего. Исследование перечисленных и других проблем, связанных с использованием суффиксных деревьев для индексирования текстов, является достаточно актуальной задачей.

Цель работы. Целью работы является разработка способов организации индексов на основе СД и алгоритмов их эффективного построения и использования для выполнения отдельных видов несловарного поиска – по шаблонам в виде регулярных выражений, а также по сходству фрагментов текста. Возможны и другие применения разработанных структур и алгоритмов.

Для достижения поставленной цели в диссертации решается ряд задач: анализ работ, посвященных построению и использованию СД, разработка способов представления, структур и алгоритмов для реализации индексов для случаев оперативной и внешней памяти, применение полученных теоретических результатов для решаемых задач поиска, разработка программных реализаций, получение экспериментальных данных, сравнение с известными аналогами, внедрение результатов в деятельность конкретных организаций.

Методы исследования. В работе использованы методы теории множеств, теории графов, анализа алгоритмов, теории автоматов, математического анализа, а также методы объектно-ориентированного программирования.

Научную новизну составляют следующие результаты:

1. Разработан метод эффективного построения обобщенных суффиксных деревьев, основанный на кодировании символов исходного текста с помощью алфавита значительно меньшего размера, чем исходный. В результате зависимость времени построения дерева от размера исходного алфавита снижается с линейной до логарифмической, аналогично уменьшается и время поиска. Требования к памяти не изменяются.

2. Предложен подход к построению обобщенных суффиксных деревьев над текстом, сжатым по методу Хаффмана. Это позволяет повысить скорость построения дерева и увеличить возможный размер входных данных в соответствии с достигнутой степенью сжатия. Модифицирован алгоритм Каркайнена и Укконена для выполнения такого построения.

3. Предложено обобщение алгоритма Укконена для построения неравномерно неплотных суффиксных деревьев с ограничениями на позиции суффиксов, доказана его корректность, получены асимптотические оценки сложности.

4. Разработан способ построения обобщенных суффиксных деревьев во внешней памяти, значительно менее чувствительный к отличиям реальных входных данных от среднего случая и сравнимый с аналогами по другим параметрам.

Основные научные положения, выносимые на защиту.

1. Метод построения обобщенных суффиксных деревьев, использующий кодирование исходного текста в алфавит меньшего размера.

2. Подход к построению обобщенных суффиксных деревьев над текстом, сжатым по методу Хаффмана. Соответствующая модификация алгоритма Каркайнена и Укконена.

3. Обобщение алгоритма Укконена для построения неравномерно неплотных суффиксных деревьев с ограничениями на позиции суффиксов.

4. Способ построения обобщенных суффиксных деревьев во внешней памяти.

Практическая ценность работы заключается в следующем: реализовано программное обеспечение для ускорения поиска по регулярным выражениям и по сходству в программном коде, в том числе разработан и внедрен новый вид индекса в СУБД PostgreSQL с открытым исходным кодом.

При реализации поиска по регулярным выражениям расширен класс т.н. префиксных регулярных выражений, поиск на соответствие которым в суффиксных деревьях выполняется с гарантированной эффективностью. Для выполнения поиска по сходству в программном коде предложена и реализована методика, основанная на предобработке откомпилированного кода с помощью суффиксных деревьев.

Реализованное программное обеспечение может быть легко адаптировано и для других видов поиска.

Реализация результатов работы. Прикладные результаты данной работы внедрены в НОУ "УЦ "Мезон" и ООО "Вологодский трубопрокатный завод". Для НОУ «УЦ «Мезон» улучшена фильтрация электронных сообщений по наборам регулярных выражений и реализовано дополнение к автоматической проверяющей системе для поиска подозрительно похожего программного кода. В ООО "Вологодский трубопрокатный завод" данные результаты применены для поиска дублирующегося программного кода в разрабатываемом ПО, а также для ускорения поиска по LIKE-шаблонам в текстовых полях СУБД.

Кроме того, результаты работы используются в учебном процессе кафедры автоматики и вычислительной техники Вологодского государственного технического университета при преподавании курсов "Структуры и алгоритмы обработки данных", "Программирование на языке высокого уровня".

Апробация работы. Основные результаты работы докладывались и получили положительную оценку на следующих конференциях, симпозиумах и семинарах: X юбилейная международная студенческая школа-семинар (Судак, МГИЭМ, 2002 г.); Всероссийская научная конференция студентов и аспирантов "Молодые *исследователи - регионам" (Вологда, ВоГТУ, 2003, 2004, 2005 гг.);* IV Международная научно-техническая конференция "Повышение эффективности теплообменных процессов и систем" (Вологда, ВоГТУ, 2004 г.); VI Международный симпозиум "Интеллектуальные системы" (Саратов, СГТУ, 2004 г.); вторая Всероссийская научно-техническая конференция "Модернизация образования. Региональный аспект" (Вологда, ВоГТУ, 2004 г.); XI Международная конференция "Современные технологии обучения: международный опыт и российские традиции "СТО-2005" (Санкт-Петербург, СПбГЭТУ (ЛЭТИ), 2005 г.); Всероссийская научно-практическая конференция "Образование, наука, бизнес: особенности регионального развития и интеграции" (Череповец, филиал ИМИТ СПбГПУ, 2005 г.); XVI Международная конференция "Применение новых технологий в образовании" (Троицк, Байтик, 2005 г.).

Публикации. По теме диссертации опубликовано 13 научных работ, из них – 6 статей и тезисы к 7-ми докладам на международных и всероссийских научно-тенических конференциях.

Структура и объём работы. Диссертация состоит из введения, четырёх глав, заключения, списка литературы, включающего 107 наименований, и трёх

приложений. Основная часть работы изложена на 138 страницах машинописного текста. Работа содержит 34 рисунка и 13 таблиц.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность проблемы, показывается место исследуемой в диссертации задачи среди совокупности задач обработки текстов, определяются цели и задачи исследования. Сформулированы научные результаты, выносимые на защиту, определены их научная новизна и практическая ценность, приведены сведения об апробации и внедрении работы.

В первой главе определяются и уточняются цели диссертационной работы, выполняется анализ существующих подходов, определяются направления и конкретные задачи диссертационного исследования.

В начале главы выполнен обзор структуры данных «суффиксное дерево» (далее СД) и её расширений – обобщенных и неплотных суффиксных деревьев (ОСД и НСД). СД представляет собой дерево, каждая дуга которого нагружена меткой - подстрокой исходной строки (для экономии памяти на дугах хранятся не сами подстроки, а их позиции в исходной строке). Каждому суффиксу исходной строки однозначно соответствует конкатенация меток на пути от корня дерева до одного из его листьев. Аналогично, каждый узел дерева соответствует некоторой подстроке – конкатенации меток на пути от корня до этого узла.

ОСД отличается тем, что оно строится не над одной исходной строкой, а над их множеством. В связи с этим в структуру добавляются операции вставки и удаления строк. НСД строится не над всеми суффиксами исходного текста (или текстов), а лишь над определённой их частью.

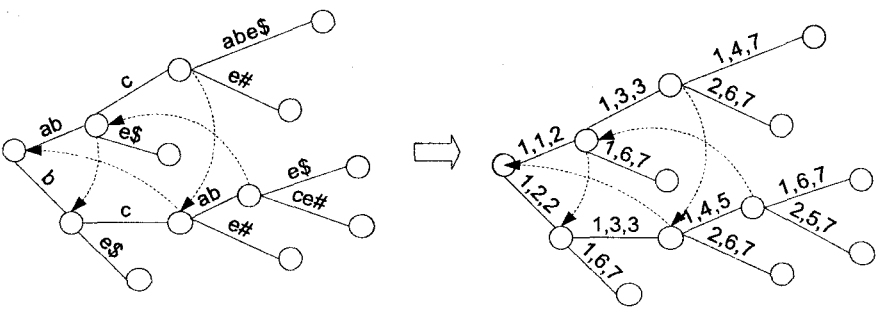


Рис.1. Пример обобщенного неплотного суффиксного дерева для строк {'abcabe$','bcabce#'}. В дерево включены только суффиксы, начинающиеся с символов 'a' и 'b'. Дуги нагружены подстроками исходной строки (дерево слева), реально на дугах хранят позиции подстрок в исходных строках (дерево справа). Пунктиром показаны суффиксные связи.

Пример обобщенного и одновременно неплотного СД показан на рис. 1. Использование сокращения "СД" в дальнейшем будет означать, что результаты применимы также к ОСД и НСД, если явно не сказано обратное.

В работе отмечается, что такие структуры могут применяться для решения большого количества задач, связанных с обработкой текстовой информации – в настоящее время известны десятки различных применений. Поскольку основным недостатком СД являются сравнительно высокие требования к памяти (10-15 байт на входной символ), то делается вывод: СД могут быть использованы как в высокой степени универсальный и расширяемый индекс для текстовой информации в информационных системах со средними объёмами данных.

Далее сформулированы практические задачи, решаемые автором в данной работе. Первая задача заключается в ускорении поиска по регулярным выражениям (далее РВ). РВ – гибкий и удобный способ для извлечения информации. Например, для поиска документов, содержащих ссылки на mp3-файлы и адреса электронной почты в домене ru, можно написать следующие (для примера несколько упрощенные) выражения:

'<a\s+href[^>]\.mp3', '\w+@\w+(.\w+)*\.ru'.

Вторая задача практического плана, для решения которой применяются результаты диссертационного исследования, заключается в поиске по сходству программного кода. Данная задача в первую очередь направлена на поиск *подозрительно похожих решений в дистанционном лабораторном практикуме по* программированию. Однако, результаты могут быть использованы и для поиска дублирующегося кода в разрабатываемых программных проектах, а также других задачах.

Для эффективного решения обеих решаемых задач целесообразно использовать структуры на основе ОСД (поскольку обоснование этого требует достаточно детального рассмотрения прикладных задач, оно вынесено в главу 4).

Далее в первой главе анализируются конкретные алгоритмы построения и способы компактного представления в памяти суффиксных деревьев. Рассматриваются два основных способа представления дуг СД — хеширование и связные списки, а также эффективные по памяти способы хранения меток дуг. При этом подробно разбирается алгоритм Укконена, так как дальнейшие результаты во многом опираются на используемые в нём идеи. На основе анализа делается вывод, что существующие методы имеют ряд недостатков и ограничений:

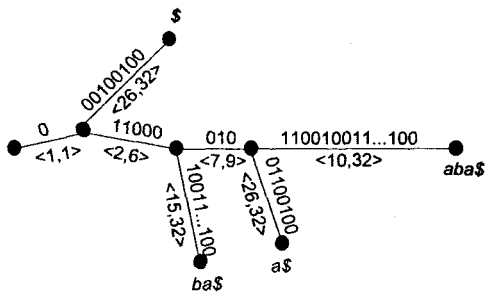
- основные схемы компактного представления в памяти обычных СД напрямую неприменимы для случая ОСД в связи с возможностью динамического изменения данных в таких деревьях;
- существующие способы представления деревьев в оперативной памяти для реальных данных с алфавитами большого размера позволяют эффективно выполнять лишь отдельные подмножества операций над СД – см. табл. 1. Следует также учитывать, что представление на базе хеширования имеет примерно на 50% большие требования к памяти;
- алгоритмы построения обобщенных суффиксных деревьев в оперативной памяти непригодны для данных больших объёмов в случае использования внешней памяти. Используемые же для этого вероятностные алгоритмы имеют в худшем случае квадратичную сложность, и для реальных данных время их работы зачастую во много раз отличается от теоретического среднего случая.

Таблица 1. Асимптотические оценки реализации отдельных операций над СД с использованием двух основных способов представления их в памяти

Операции / способ представления в памяти	Построение	Поиск нужной дуги, выходящей из заданного узла	Перечисление дуг, исходящих из узла (для реализации обхода)
Хеширование	O(n)	O(1)	O(|Σ|)
Связные списки	O(n·|Σ|), где |Σ| - размер алфавита	O(|Σ|)	O(*l*), где *l* – количество дуг

Во второй главе рассматриваются отличия ОСД от обычных СД, выполняется анализ существующих способов компактного представлению данных структур в памяти. Отмечаются ограничения способа, предложенного Р. Гигерихом и др., не позволяющие реализовать динамическое удаление данных за линейное время. Показана применимость альтернативного способа, предложенного в диссертации С. Куртца, и особенности его использования в случае ОСД.

Далее большое внимание уделяется проблеме эффективного построения и использования СД для поиска в текстах, для которых размер алфавита достаточно велик, чтобы существенно влиять на время выполнения алгоритмов. Рассмотрен подход А. Андерсона и др., предлагающих преобразовывать метки дуг СД к бинарному алфавиту после его обычного построения для повышения скорости поиска. На этой основе в диссертации предложена модификация алгоритма Каркайнена и Укконена, позволяющая реализовать непосредственное построение СД в новом алфавите. В результате достигается ускорение не только поиска, но и построения дерева. Пример дерева, построенного с использованием нового алфавита, приведен на рис. 2.



Исходный текст	a	b	a	$
Кодированный текст	01100001	01100010	01100001	00100100
Начала суффиксов для НСД	▲	▲	▲	▲

Рис. 2. Пример НСД над текстом в двоичном представлении

Временная сложность построения снижается при этом с O(n·|Σ|) до O(n·log(|Σ|)).

Дальше во второй главе рассматривается вопрос выбора размера алфавита, в котором будет выполняться построение дерева. Переход к бинарному алфавиту представляется нецелесообразным, так как при этом число внутренних узлов в любом СД с n листьями будет равняться в точности n-1. Для алфавитов же бóльших размеров число внутренних узлов составляет порядка 0,6n-0,8n. Теоретическая оценка среднего числа внутренних узлов дерева в зависимости от размера исходного алфавита получена в работе В. Шпанковски, она имеет вид:

$$EL_n = \frac{n}{h_1} \cdot (1 + P_4(\log n)) + O(1),$$

где n – размер текста, h_1 – энтропия: $h_1 = -\sum_{i=1}^{k} p_i \cdot \log(p_i)$. Данная формула выведена для текстов, где вероятности появления символов жестко заданы, но схожий результат получается опытным путём и для реальных текстов. Пример графика такой зависимости приведён на рис. 3 слева. Как видно из графика, вначале при увеличении размера алфавита число внутренних узлов в дереве резко снижается, но данный процесс быстро стабилизируется.

На рис. 3 справа показана типичная зависимость времени построения дерева от размера алфавита. Сопоставив данные зависимости, делается вывод, что наиболее подходящие значения k, обеспечивающие разумный компромисс между размером дерева и временем его построения, лежат в интервале от 3 до 5. Учитывая соображения программной реализации, в качестве размера алфавита целесообразно взять значение 4, т.к. при этом обработка каждого входного символа эффективно реализуется несколькими инструкциями процессора.

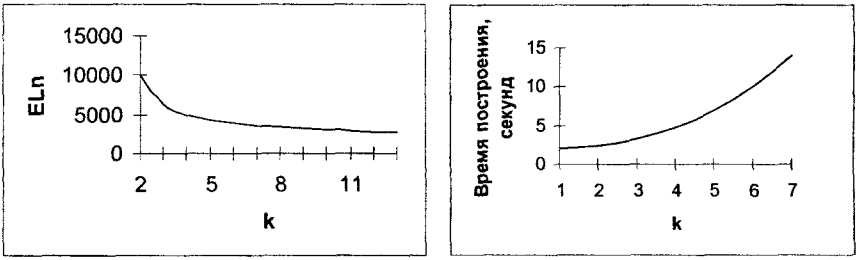


Рис. 3. График типичной зависимости числа внутренних узлов в СД от размера k исходного алфавита (слева) и времени построения СД от k (справа)

Как дальнейшее развитие данной идеи, в работе показывается, что алгоритм Каркайнена и Укконена применим для построения суффиксного дерева над текстом, сжатым по алгоритму Хаффмана (предполагается вариант алгоритма Хаффмана, формирующего кодовые последовательности в произвольном, не обязательно бинарном алфавите). При этом размер дерева остаётся таким же, но:

- повышается скорость построения дерева в соответствии со степенью сжатия;
- соответственно увеличивается размер входных данных, которые могут

разместиться в оперативной памяти для построения СД без обращения к низкоскоростной внешней памяти.

Предложенный способ перехода к новому алфавиту приводит к необходимости соответствующей модификации существующих алгоритмов, использующих СД. Но внесение таких изменений в алгоритмы и программный код требует затрат, и желательно этого избежать. Для этого показывается, что СД, построенное над кодированным текстом, может быть рассмотрено как реализация абстрактного типа данных СД, построенного над оригинальным исходным текстом.

Пусть T – оригинальное ОСД, T' – ОСД, построенное над данными в новом алфавите. В работе выделено множество из 11 базовых элементарных операций АТД "Обобщенное суффиксное дерево" (примеры операций приведены в таблице 2) и показано, что каждая такая операция над T физически может реализовываться над T', причём с менее высокой стоимостью. Для каждой из операций описывается алгоритм её реализации над T' для случая как кодов фиксированной длины, так и префиксных кодов.

В основе данных алгоритмов лежит следующее утверждение: Для каждого узла дерева T существует однозначно соответствующая ему позиция в T' (возможно, лежащая на дуге), т.ч. переход как в ту, так и в другую сторону может быть выполнен за константное время. Пример показан на рисунке 4.

Таблица 2. Примеры базовых элементарных операций, определенных для ОСД в исходном алфавите и реализуемых для ОСД в новом алфавите. Обозначения: GST – обобщенное суффиксное дерево, Pos – позиция в нём.

Операция	Пример применения	Описание
find_child : GST ⊗ Pos ⊗ Σ → Pos	q := tree.find_child (p,a)	Входные данные: p – позиция в T' некоторого узла u в T, символ a ∈ Σ (где T-исходное дерево, Σ - исходный алфавит, T'-дерево в новом алфавите). Результаты: Если у узла u существует такой сын v, что дуга u→v начинается с символа a, то q – *позиция узла v в T', иначе q = NULL (пустая позиция)*
suffixlink: GST ⊗ Pos → Pos	q:= tree.suffixlink(p)	Входные данные: позиция p соответствует узлу u в T. Результаты: позиция q в T', соответствующая узлу u.suffixlink в T (переход по суффиксной связи)

Установив такое соответствие и определив реализацию базовых функций, мы тем самым показываем, что полученная структура данных является полноценным компактным представлением исходного ОСД и сохраняет все его свойства.

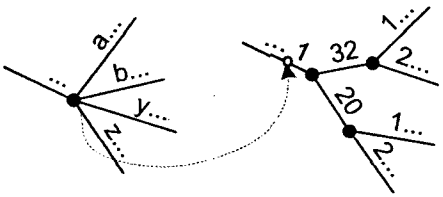


Рис. 4. Иллюстрация к соответствию между исходным деревом T (часть дерева слева) и T' (часть дерева справа). Для примера исходным символам соответствуют коды: 'a'→'1201', 'b'→'1202', 'y→'1321', 'z'→'1322'. Узлы дерева T однозначно отображаются в позиции на дугах дерева T'.

В третьей главе доказывается следующее утверждение, являющееся основой предлагаемого далее способа построения ОСД во внешней памяти. Пусть дана строка s длины n и функция f(s,j)→{true, false}, возвращающая true, если суффикс s[j..n] должен быть включен в неплотное суффиксное дерево. Функция f должна удовлетворять следующим ограничениям: а). Её результат зависит только от подстроки s[j-1..j+k-1], где k – некоторая константа. б). Функция f вычисляется за константное время. Тогда возможно построить соответствующее НСД с временной сложностью O(n·log(|Σ|)) и пространственной сложностью O(|T|), где |T| - размер конечного результата.

В процессе доказательства приводятся изменения, которые должны быть внесены в алгоритм Укконена, при этом дополняется определение суффиксных связей дерева. Основным изменением является введение в алгоритм операции обратного сдвига фазы для корректного продолжения последнего суффикса в каждой фазе алгоритма. Показывается, что такая операция не увеличивает сложность алгоритма, и он остаётся корректным.

Утверждение применимо, по крайней мере, для двух случаев.

- Во-первых, оно имеет самостоятельное значение, так как позволяет гибко задавать подмножества суффиксов, требуемых для решения конкретных задач.
- Во-вторых, оно лежит в основе алгоритма построения обобщенных суффиксных деревьев во внешней памяти, рассматриваемого далее.

По аналогии с работой Е. Хант, построение дерева разбивается на независимые построения отдельных его поддеревьев. Однако, предложенный подход для построения поддеревьев использует уже не вероятностный алгоритм с квадратичным в худшем случае временем работы, а модифицированный алгоритм Укконена. Время построения одного поддерева составляет O(n) независимо от входных данных (опуская зависимость от алфавита, т.к. для случая внешней памяти она не столь критична). Приведенное выше утверждение даёт возможность задавать подмножества суффиксов, входящие в поддеревья так, чтобы строящиеся поддеревья имели одинаковый заранее определённый размер. Если M – размер доступной оперативной памяти, m – среднее число байт на один суффикс поддерева, n – длина исходного текста, то при n<M алгоритм требует выполнения $\frac{n \cdot m}{M - n}$ проходов по исходным данным, на каждом из которых строится одно

поддерево. Общая стоимость построения будет равна $O(\frac{n^2 \cdot m}{M - n})$.

Далее в диссертации рассматривается возможность использования данного алгоритма для случая, когда n близко к M или превосходит его. Как и для других алгоритмов построения СД во внешней памяти, последовательность обращений к исходным данным носит псевдослучайный характер. Это ограничивает их возможный размер: он должен быть сравним с размером оперативной памяти (на практике превышать его не более чем в 1.5 – 2 раза).

Для увеличения размера входных данных возможно применение предложенного в предыдущей главе способа построения СД над текстом, сжатым по методу Хаффмана. Однако, если размер текста значительно превышает объём доступной памяти, требуется найти соотношение между размером кэш-буфера под исходный текст и памятью под поддеревья, минимизирующее число дисковых операций. Пусть B – размер одной страницы ввода-вывода, N – среднее число суффиксов в одном поддереве – настраиваемая величина. В работе получена следующая оценка для числа чтений с диска для предложенного алгоритма:

$$Q(N) = (\frac{n}{NB} + 1) \times (n - M + mN)$$

Пример графика данной зависимости изображен на рис. 5.

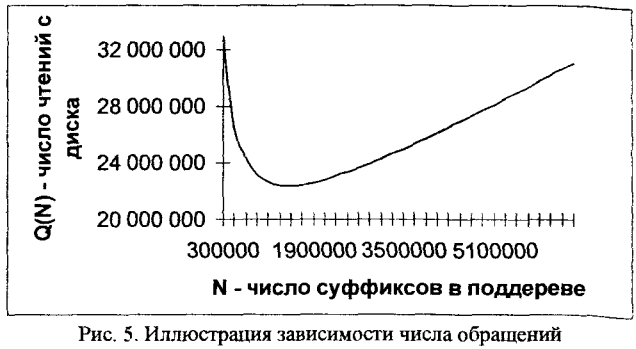


Рис. 5. Иллюстрация зависимости числа обращений к диску от размера строящихся поддеревьев

Размер поддерева, для которого число обращений к диску минимально, тогда можно найти как:

$$N = \frac{[-m \cdot B \cdot n \cdot (-n + M)]^{0.5}}{m \cdot B}$$

В ходе дальнейшего улучшении алгоритма предлагается приём, позволяющий дополнительно сократить число обращений к исходным данным как в процессе построения дерева, так и при его использовании при поиске. Он заключается в хранении дополнительно на каждой дуге дерева первого символа подстроки, связанной с данной дугой. Тем самым получается структура данных, сочетающая в

себе свойства как суффиксных, так и деревьев цифрового поиска. В результате:

- операция поиска нужной дуги, выходящей из вершины дерева, не требует обращений к исходной строке. Поскольку данная операция выполняется при работе алгоритме наибольшее число раз, то время построения уменьшается более чем на 100% даже для случая оперативной памяти;
- при выполнении каждой фазы алгоритма Укконена удаётся избежать обращений к исходному тексту для всех её продолжений, кроме первого;
- при выполнении поиска подстрок в дереве можно работать с ним как с цифровым деревом поиска – т.е. обращаться к исходному тексту только по завершении поиска для окончательной проверки найденных вхождений.

Полученная в итоге программная реализация превосходит известные аналоги для реальных текстовых данных и оказывается сравнимой с ними для случайно сгенерированных и близких к ним последовательностей. Для сравнения был взят алгоритм PWOTD, широко применяемый для построения СД во внешней памяти, и его реализация в виде утилиты TDD, используемой для исследований в области вычислительной биологии в университете г. Мичиган. Пример экспериментальных результатов сравнения приведен на рис. 6. Время выполнения TDD оказалось примерно в 3 раза меньше на случайно сгенерированных символьных последовательностях. Однако, выполненная автором реализация показала примерно на 50% более высокую производительность на случайно сформированной коллекции HTML-текстов и на несколько порядков более высокую – на подборке выпусков одного из электронных журналов. Последнее можно объяснить тем, что журнальные статьи в HTML-формате содержали более высокое число повторяющихся подстрок – похожие участки HTML-кода, повторяющиеся названия, URL-адреса и др. Данный факт не оказывает влияния на наш алгоритм, но оказывается очень существенным для квадратичного в худшем случае алгоритма PWOTD.

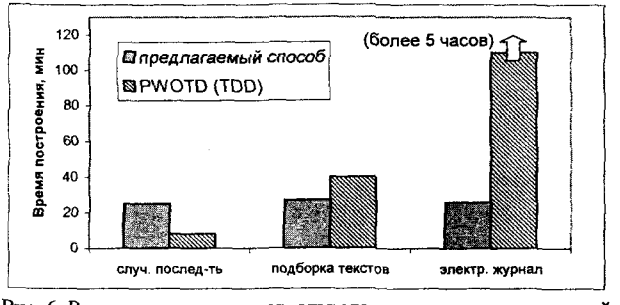


Рис. 6. Результаты экспериментального сравнения предложенной реализации и реализации алгоритма PWOTD

Таким образом, можно сделать вывод, что предложенный в работе алгоритм построения ОСД во внешней памяти значительно менее чувствителен к возможным отличиям реальных входных данных от теоретического среднего случая, и может служить основой для построения индексов для реальных текстов.

В четвертой главе рассматривается применение полученных теоретических результатов к решению практических задач.

Первая из них заключается в разработке индекса для ускорения поиска по регулярным выражениям (РВ). В работе выполнен обзор существующих подходов к построению таких индексов. Один из подходов, предложенный Баеза-Ятесом и др., заключается в применении построенного по РВ конечного автомата не к исходным текстам, а к индексу на основе СД. В результате среднее время поиска оказывается сублинейным. Альтернативный подход, предложенный Чо и др., использует тот факт, что из многих РВ можно извлечь отдельные подстроки (граммы) с низкой селективностью (т.е. встречающиеся в небольшой доле исходных документов). На их основе строится сокращенное синтаксическое дерево (см. рис. 7), которое используется для выполнения поиска. Найденные документы требуют дополнительной проверки на соответствие исходному РВ.

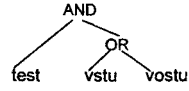


Рис.7. Сокращенное синтаксическое дерево для регулярного выражения 'avt.*@.*(vstu|vostu)'.

Чтобы получить для граммы её подмножество документов, в оригинальной статье предложена специализированная индексная структура. В диссертации показано, что для этой цели вполне применимо ОСД, при этом достигаются:

- возможность совместного использования двух описанных подходов, эффективных для разных классов РВ, используя одну и ту же структуру данных и базовые алгоритмы работы с ней;
- возможность динамического обновления индекса (в оригинальной статье Чо и др. индекс предполагается статическим, и способы эффективной его модификации при изменении данных не предусмотрены);
- при поиске возвращается меньшее количество неподходящих документов, отбрасываемых в ходе дальнейшей проверки.

В методе, предложенном Баеза-Ятесом и др., введен класс т.н. префиксных РВ, поиск по которым в суффиксном дереве выполняется за время, пропорциональное их длине. Основное свойство таких выражений заключается в том, что для каждого из них существует уникальный конечный набор ключевых слов, таких что:

- для любого другого слова языка одно из них является его собственным префиксом;
- никакое ключевое слово не является собственным префиксом другого.

Однако, класс префиксных РВ не охватывает многие практически важные выражения, поиск по которым может быть выполнен с высокой эффективностью. Например, выражение 'a(b|c)d+' является префиксным, тогда как похожее выражение '(a|b)cd+' – не является. В связи с этим предложено расширить данный класс выражений, включив в него РВ, для которых выполняется приведённое свойство и при этом поиск требует не более O(mK) операций, где m – длина РВ, K – максимальное количество ключевых слов (параметр настройки). Соответственно дополнено определение расширенных РВ и модифицирован алгоритм их

идентификации – см. табл. 3.

Таблица 3. Алгоритм проверки, является ли регулярное выражение расширенным префиксным (EPRE), и определение его ключевых слов (prewords)

выражение	EPRE(выражение)	prewords(выражение)
$\varnothing$	True	$\varnothing$
ε	True	$\{\varepsilon\}$
$x \in \Sigma$	True	$\{x\}$
$r \mid s$	EPRE(r) и EPRE(s)	prewords(r) $\cup$ prewords(s)
Rs	а). (r – выражение, задающее конечный язык и EPRE(s)) или б). (EPRE(r) и $\varepsilon \in$ L(s))	prewords(r)prewords(s)
r^*	True	$\{\varepsilon\}$

Вторая практическая задача состоит в поиске по сходству программного кода. Известны различные подходы, используемые для этого: сравнение (в т.ч. параметризованное) участков исходного текста, сопоставление синтаксических деревьев и графов выполнения программы, анализ стиля программирования и др. Их недостатки - либо сильная привязка к конкретным языкам программирования и их диалектам, либо отсутствие возможностей обнаружения даже простых структурных изменений в коде.

В диссертации предложена простая легко реализуемая методика, дающая на практике приемлемые результаты. Идея заключается в реализации поиска по сходству не по исходному тексту, а по обработанному промежуточному коду, получаемому на выходе компиляции (т.е. ассемблерному тексту, байт-коду либо объектному коду.) Такая проверка нечувствительна к переформатированию текста, замене имен идентификаторов, вставке неиспользуемых переменных или кода и др. Пример показан в таблице 4.

Таблица 4. Простые изменения в исходном коде не влияют на конечный результат

	Первый текст	Второй текст

Исходный код	`for(int i=0; i<20; i++)`
	`  { c+=a; a*=2; }`
		`int k=0, i=0;`
		`while(i<20)`
		`  { c+=a; k++; a*=2; i++; }`

Откомпилированный код	`  xor edx,edx`
	`@3: add ecx,eax`
	`  mov ebx,eax`
	`  add ebx,ebx`
	`  mov eax,ebx`
	`@5: inc edx`
	`  cmp edx,20`
	`  jl short @3`
		`  xor edx,edx`
		`@2: add ecx,eax`
		`  mov ebx,eax`
		`  add ebx,ebx`
		`  mov eax,ebx`
		`  inc edx`
		`  cmp edx,20`
		`  jl short @2`

Таким образом, задача сводится к поиску совпадающих фрагментов в текстах, которая эффективно решается с помощью ОСД (работы Моностори и др.)

Предложенная методика была реализована для поиска подозрительно похожих

решений в автоматизированной проверяющей системе. При этом были выделены два способа контроля поступающего решения – сравнение с решениями сдаваемой задачи либо со всеми решениями в базе данных (предполагая заимствование участков кода из разных мест).

В первом случае ОСД, используемое для поиска по сходству, строится для относительно небольшого количества решений и кэшируются в оперативной памяти. Это позволяет выполнять проверку за время O(m+k), где m – длина проверяемого решения, k – результирующее количество совпадений.

Во втором случае ОСД более выгодно строить, наоборот, над множеством присланных на проверку решений. При этом временная сложность алгоритма оказывается линейной от суммарной длины решений в базе данных (без учета размера алфавита).

Проверка в реальных условиях проводилось на проверяющей системе, используемой в ВоГТУ для проведения лабораторных работ по программированию. Результаты показали, что сравнение поступающего решения только с решениями данной задачи выполняется значительно меньше секунды при количестве сравниваемых решений около 100, со всеми решениями (несколько тысяч) – порядка одной секунды (без учета времени компиляции). Данные результаты являются вполне приемлемыми и показывают наличие достаточных резервов для дальнейшего наращивания объёмов данных.

Далее в работе исследуются особенности реализации индексного метода доступа на основе ОСД для СУБД PostgreSQL. Проанализированы основные способы внедрения новых индексных методов доступа в данную СУБД – разработка индексов "с нуля", используя соответствующие интерфейсы, и применение обобщенных деревьев поиска информации (GIST). Сделан вывод, что текущая реализация GIST не предоставляет требуемых возможностей для реализации ОСД, и предложен вариант реализации новых видов индексов, позволяющий упростить их разработку, особенно для программистов - не специалистов по особенностям внутренней работы СУБД. Суть подхода состоит в применении для хранения и доступа к индексным данным высокоуровневых средств самой СУБД, что позволяет использовать готовые средства обеспечения корректного параллельного доступа к индексу, размещения данных в дисковых страницах, компрессии и др. Важным моментом является то, что по своему использованию новые виды индексов не отличаются от встроенных в СУБД.

На основе данного варианта реализации разработан индекс для PostgreSQL, позволяющий ускорять поиск по шаблонам в виде PB для текстовых полей. При хранении ОСД разбивается на отдельные поддеревья, занимающие 1-3 дисковых страниц, над которыми, в свою очередь, строится индекс стандартными средствами СУБД. В итоге выполнение поиска для низкоселективных запросов требует чтения всего лишь нескольких дисковых станиц.

Важным вопросом для любого индекса является возможность его модификации для приведения в соответствии с изменившимися данными и стоимость такой операции. Модификация ОСД при вставке/удалении записи длины m выполняется в среднем за время O(m·log(m)). Непосредственное выполнение такой операции при каждом изменении данных неприемлемо для большинства приложений ввиду ощутимых задержек. Поэтому использован другой вариант – в структуру индекса

дополнительно вводится список изменённых документов, которые включаются в основную структуру отдельным процессом в фоновом режиме.

Результаты тестирования показали более чем десятикратное среднее ускорение поиска по сравнению с использованием стандартных средств СУБД. При этом поиск отдельных РВ выполняется быстрее на 2 порядка и более. Платой за такое ускорение служит объём созданного индекса, который с учётом возможностей сжатия данных, предоставляемых СУБД, превышает объём входных данных в 10-12 раз.

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ РАБОТЫ

1. Кодирование символов исходного текста с использованием алфавита меньшего размера позволяет снизить зависимость от размера алфавита времени построения суффиксного дерева и поиска в нём с линейной до логарифмической. Требования к оперативной памяти при этом не изменяются.

2. Использование оптимальных префиксных кодов позволяет снизить сложность построения дерева и поиска в нем в соответствии со степенью сжатия исходного текста. Для построения такого дерева предложена модификация алгоритма Каркайнена-Укконена.

3. Обобщен алгоритм Укконена для построения неравномерно неплотных суффиксных деревьев. По сравнению с существующим подходами множество суффиксов, входящее в дерево, может быть задано более гибко.

4. Суффиксное дерево, построенное над закодированным текстом, эквивалентно оригинальному дереву (построенному над исходным текстом). Множество элементарных операций оригинального дерева может быть физически реализовано над деревом в новом алфавите с меньшей либо такой же временной сложностью.

5. Предложен способ построения суффиксных деревьев во внешней памяти, при котором время построения дерева значительно менее зависит от отличий реальных входных данных от среднего случая.

6. Разработана методика выполнения поиска по сходству в исходном программном коде, основанная на предобработке кода на промежуточном языке с помощью суффиксных деревьев после выполнения трансляции.

7. Реализован новый вид индекса для СУБД с открытым исходным кодом для ускорения поиска по регулярным выражениям.

8. Реализовано и внедрено прикладное программное обеспечение для выполнения рассмотренных видов поиска.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИОННОЙ РАБОТЫ

1. Андрианов, И.А. Средства контроля знаний в обучающих средах / И.А.Андрианов, В.Р. Габитова // X юбилейная Междунар. студенч. школа-семинар: тез. докл, г. Судак, 14-21 мая 2002 г. - М., 2002. – Т. 1. - С. 305 - 306.

2. Андрианов, И.А. Подходы к разработке легкоуправляемых сайтов / И.А. Андрианов, С.Ю. Ржеуцкая // Молодые исследователи – региону: материалы Всерос. науч. конф. студентов и аспирантов, г. Вологда, 19-20 апр. 2003 г. – Вологда, 2003. – С. 148 - 149.

3. Андрианов, И.А. Структуры данных для повышения эффективности выполнения специальных видов поиска /И.А. Андрианов, В.Н. Минин // Молодые исследователи – региону: материалы Всерос. науч. конф. студентов и аспирантов, г. Вологда, 22-23 апр. 2004 г. – Вологда, 2004. – С. 154 - 155.
4. Андрианов, И.А. Один пример использования неплотных суффиксных деревьев для обработки текстов / И.А. Андрианов // Повышение эффективности теплообменных процессов и систем: материалы IV Международной науч.-техн. конф., г. Вологда, 25-27 окт. 2004 г. – Вологда, 2004. – С. 304 - 307.
5. Андрианов, И.А. Построение индексов для расширенного поиска по текстовым полям / И.А. Андрианов // Интеллектуальные системы: материалы шестого Междунар. симп., г. Саратов, 29 июн. - 2 июл. 2004 г. – М., 2004. – С. 279 - 282.
6. Андрианов, И.А. Подходы к организации подсистемы поиска по сайту в условиях хостинга с ограниченными возможностями / И.А. Андрианов // Моделирование, оптимизация и интенсификация производственных процессов и систем: материалы Междунар. науч.-техн. конф., г. Вологда, 19-21 мая 2004 г. – Вологда, 2004. – С. 270 - 273.
7. Андрианов, И.А. Опыт проведения занятий по программированию в форме on-line-соревнований / И.А. Андрианов, С.Ю. Ржеуцкая // Модернизация образования. Региональный аспект: материалы второй Всерос. науч.-метод. конф., г. Вологда, 11-12 мар. 2004 г. – Вологда, 2004. С. 199 - 200.
8. Андрианов, И.А. Применение неплотных суффиксных деревьев для поиска наибольшей общей подстроки / И.А. Андрианов // Методы и системы обработки информации / Муромский ин-т (филиал) Владимирского гос. ун-та. – М., 2004. – С. 77 - 82.
9. Андрианов, И.А. Использование высокоуровневых интерфейсов для разработки индексных методов доступа в СУБД PostgreSQL / И.А. Адрианов // Образование, наука, бизнес: особенности регионального развития и интеграции: тр. Всерос. науч.-практ. конф., г. Санкт-Петербург, 26-27 мая 2005 г. – СПб., 2005. – С. 230 - 235.
10. Андрианов, И.А., Проблема поиска по регулярным выражениям в текстовых базах данных / И.А Андрианов, В.Н. Минин // Образование, наука, бизнес: особенности регионального развития и интеграции: тр. Всерос. науч.-практ. конф., г. Санкт-Петербург, 26-27 мая 2005 г. – СПб., 2005. – С. 284 - 287.
11. Андрианов, И.А. Применение обобщенных суффиксных деревьев для анализа программного кода при обучении программированию / И.А. Андрианов // Применение новых технологий в образовании: материалы XVI Междунар. конф., г. Троицк, 28-29 июня 2005 г. – Троицк, 2005. – С 11 - 12.
12. Андрианов, И.А. Одна задача обработки исходных текстов в автоматизированных системах обучения программированию / И.А.Андрианов, В.Н Минин // Современные технологии обучения: международный опыт и российские традиции «СТО-2005»: материалы XI Междунар. конф., г. Санкт-Петербург, 20 апр. 2005 г. – СПб., 2005. - С. 102 - 104
13. Андрианов, И.А. Применение GIST индексов для ускорения поиска по шаблонам / И.А.Андрианов // Молодые исследователи – регионам: материалы Всерос. науч. конф. студентов и аспирантов, г. Вологда, 21-22 апреля 2005 г. – Вологда, 2005. – С. 271 – 272.