На правах рукописи

Чабанюк Денис Андреевич

ПРЕОБРАЗОВАНИЕ ИНФОРМАЦИОННЫХ ДАННЫХ И ДВОИЧНЫХ СТРУКТУР С МИНИМИЗАЦИЕЙ ВРЕМЕННОЙ СЛОЖНОСТИ НА ОСНОВЕ АЛГОРИТМОВ СОРТИРОВКИ

Специальность:
05.13.17 – Теоретические основы информатики

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Таганрог – 2018

Работа выполнена на кафедре информатики Таганрогского института имени А.П. Чехова (филиал) ФГБОУ ВО "Ростовский государственный экономический университет (РИНХ)"

Научный руководитель: **Ромм Яков Евсеевич**,
доктор технических наук, профессор, заведующий кафедрой информатики Таганрогского института имени А.П. Чехова (филиал) ФГБОУ ВО "РГЭУ (РИНХ)" (г. Таганрог)

Официальные оппоненты: **Зинкин Сергей Александрович**,
доктор технических наук, профессор кафедры «Вычислительная техника» ФГБОУ ВО ПГУ (г. Пенза)

Турулин Игорь Ильич,
доктор технических наук, профессор кафедры информационных измерительных технологий и систем ИТА ЮФУ (г. Таганрог)

Ведущая организация: ФГБОУ ВО Ростовский государственный университет путей сообщения (РГУПС) (г. Ростов-на-Дону)

Защита состоится « 27 » июня 2018 г. в ____ на заседании диссертационного совета Д 999.065.02, созданного на базе ФГАОУ ВО "ЮФУ" и ФГБОУ ВО "ЮРГПУ (НПИ) им. М.И. Платова" по адресу: 347928, г. Таганрог, ГСП-17А, пер. Некрасовский, 44, ауд. Д-406.

С диссертацией можно ознакомиться в Зональной научной библиотеке Южного федерального университета по адресу: 344000, г. Ростов-на-Дону, ул. Пушкинская, 148.

Автореферат разослан « » 2018 г.

Ученый секретарь
диссертационного совета Д 999.065.02
доктор технических наук, профессор Целых А.Н.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Развитие компьютерных технологий и создание высокопроизводительных процессоров обуславливает разработку структур данных, адаптированных к массовому параллелизму. При существующем многообразии архитектур процессоров и способов представления информации одной из важнейших задач является ускорение обработки данных на основе эффективно распараллеливаемых алгоритмов. Большинство существующих методов параллельных вычислений направлено на решение сложных задач адаптации вычислительного алгоритма к архитектурам многопроцессорных вычислительных систем. В качестве одной из таких задач целесообразно рассматривать параллельное построение и обработку древовидных структур данных. Распараллеливание организуется как средство повышения эффективности представления и обработки динамических данных с целью быстрого поиска информации. Известны различные способы построения структур данных, включая параллельные, реализуемые под конкретные задачи. Классическим структурам данных посвящены работы Д.Э. Кнута, А. Ахо, Дж. Ульмана, Н. Вирта, R. Seidel, C. Aragon. Предложенные в этих работах структуры данных и методы выполнения в них базовых операций создали основу современных технологий информационного поиска. Исследование классических структур и возможности их адаптации к параллелизму представлено в работах А.А. Могилко, А.А. Плетнева, Л.И. Тимченко, С.А. Харченко, A. Lagana, V. Kumar, C. Tan, P. Chalermsook, A. Amir, M. Farach и др. Наряду с тем созданы комплексы и системы структур данных, которые используют параллельные вычисления: Oracle Database, IBM DB2, Microsoft SQL Server, MySQL и др. При этом остается актуальной проблема адаптации структур и средств обработки данных к быстро развивающимся архитектурам параллельных и последовательных вычислительных систем, к технологиям создания современных информационных систем. В качестве стимула сохраняется необходимость быстрой и эффективной обработки огромных объемов сложно структурированной информации, характеризующихся неуклонным ростом. Наличие множества исследований наряду с технологическими факторами подтверждает актуальность создания эффективных, обладающих достаточной простотой программной реализации, алгоритмов построения и обработки параллельных структур данных.

Целью диссертационной работы является разработка и исследование способов ускорения организации и преобразований структур данных для информационной обработки на основе алгоритмов устойчивой адресной сортировки в применении к базовым операциям информационного поиска. В число исследуемых входят структуры декартовых и двоичных деревьев, в число способов – алгоритмические и разрядные преобразования информационных данных во взаимно независимой форме. На основе разрабатываемых алгоритмов требуется представить конструктивный способ извлечения структурных закономерностей информационных данных, скрытых в массивах входной информации.

Для достижения поставленной цели в диссертационной работе решаются следующие **задачи**:

1. Разработать способ сравнения данных числового и строкового типа на основе вертикальной обработки в качестве компонента информационного поиска, при

котором не используется операция вычисления переноса и выполняется взаимно независимый анализ разрядов данных.

2. Разработать алгоритм ускоренного построения декартова дерева информационных данных на основе сортировки, который выполняется с логарифмической временной сложностью.

3. Разработать алгоритм ускоренного построения двоичного дерева информационных данных на основе устойчивой адресной сортировки, который выполнялся бы с временной сложностью $O(\log_2 N)$.

4. Разработать алгоритм ускоренного построения двоичного дерева информационных данных на основе сортировки и априорного вычисления хранимых индексов корней поддеревьев, который выполнялся бы с оценкой временной сложности $T(1)=O(1)$.

5. Показать, что из произвольного расположения данных в массиве можно извлечь априори скрытую полную информацию об экстремальных закономерностях, а также информацию о структурных закономерностях двоичного дерева.

6. Получить дополнительное ускорение предложенных алгоритмов за счет отсутствия вычисления переноса и взаимной независимости обработки разрядов данных при выполнении операций сравнения.

Методы исследования опираются на теоретические основы информатики, теорию сложности, методы организации структур данных, методы синтеза и анализа параллельных алгоритмов применительно к поиску и сортировке.

Достоверность результатов вытекает из их математического обоснования, подтверждается результатами численных и программных экспериментов.

Научная новизна

1. Предложен способ сравнения данных числового и строкового типа на основе вертикальной обработки в качестве компонента информационного поиска. Способ отличается от известных отсутствием вычисления переноса и взаимной независимостью анализа разрядов данных, что позволяет выполнять сравнение с временной сложностью $O(1)$ независимо от числа символов строк (С. 55 – 72).

2. Разработан алгоритм построения декартова дерева информационных данных на основе устойчивой адресной сортировки, отличающийся от аналогов структурой и логарифмическим количеством последовательных шагов, что позволяет достигать ускорения относительно аналогов $O(N)$, где N – число данных (С. 87 – 93, 103 – 113).

3. Разработан алгоритм построения двоичного дерева информационных данных на основе устойчивой адресной сортировки, отличающийся от аналогов структурой и логарифмической оценкой временной сложности, что позволяет достигать ускорения относительно аналогов $O\left(N^{\alpha}\right)$, $\alpha \geq 1$ (С. 115 – 122, 130 – 142).

4. Разработан алгоритм построения двоичного дерева информационных данных на основе сортировки и априорного вычисления хранимых индексов корней поддеревьев, который отличается от известных структурой и оценкой временной сложности $T(1)=O(1)$, что позволяет достигать ускорения относительно аналогов $O\left(N^{\alpha}\right)$, $\alpha > 1$ (С. 115 – 123, 130 – 136, 143 – 145).

5. Показано, что на основе предложенного алгоритма из произвольного расположения данных в массиве можно извлечь полную информацию об экстремальных закономерностях и, аналогично, информацию о структурных закономерностях двоичного дерева, априори скрытую во входных данных. Способ отличается от аналогов точным решением с помощью конструктивного детерминированного алгоритма (С. 123 – 126).

6. Показано, что предложенные алгоритмы построения декартовых и двоичных деревьев информационных данных получают дополнительное ускорение за счет отсутствия вычисления переноса и взаимной независимости обработки разрядов данных при выполнении операций сравнения, что позволяет увеличить ускорение относительно известных аналогов до $O\left(nN^{\alpha}\right)$, $\alpha \geq 1$, где n – количество разрядов данных (С. 128 – 130, 148 – 152).

Основные положения, выносимые на защиту

1. Способ сравнения данных числового и строкового типа на основе вертикальной обработки в качестве компонента информационного поиска, отличающийся отсутствием вычисления переноса и взаимной независимостью анализа разрядов данных, позволяющий выполнять сравнения с временной сложностью $O(1)$ независимо от числа символов строк.

2. Алгоритм построения структуры декартова дерева информационных данных на основе устойчивой адресной сортировки с логарифмическим количеством последовательных шагов.

3. Алгоритм построения структуры двоичного дерева информационных данных из N элементов на основе устойчивой адресной сортировки с временной сложностью $T\left(N^2\right)=O\left(\log_2 N\right)$.

4. Алгоритм построения двоичного дерева информационных данных на основе сортировки и априорного вычисления хранимых индексов корней поддеревьев.

5. Способ извлечения из произвольного расположения данных в массиве полной информации об экстремальных закономерностях, а также информации о структурных закономерностях двоичного дерева, априори скрытой во входном массиве данных.

6. Способ дополнительного ускорения алгоритмов построения декартовых и двоичных деревьев информационных данных за счет отсутствия операций вычисления переноса и взаимной независимости обработки разрядов данных при выполнении сравнений.

Теоретическая значимость научных результатов состоит в разработке конструктивных способов ускорения построения и обработки данных древовидных структур информационных данных на основе алгоритмов устойчивой адресной сортировки, в улучшении существующих оценок временной сложности построения декартова и двоичного дерева, в использовании вертикальной разрядной обработки слов для ускорения операций в древовидных структурах данных. В целом предложенные способы и алгоритмы могут составить алгоритмическую основу для ускоренного детерминированного поиска в реляционных базах данных и информационных системах.

Практическая ценность диссертационного исследования заключается в

прикладном характере предложенных методов и алгоритмов. Разработанная схема параллельного поиска с применением разрядного распараллеливания на основе вертикальной обработки может быть реализована для ускорения, повышения точности и расширения функциональных возможностей поиска. Параллельное на уровне алгоритмических и разрядных операций построение декартова и двоичного дерева может рассматриваться в качестве основы для ускорения обработки и управления в реляционных базах данных. Предложенные методы и алгоритмы ориентированы на создание эффективных методов динамической обработки баз данных, дают возможность ускорить процесс обработки и управления базами данных, повысить скорость и точность систем информационного поиска.

Внедрение и использование результатов работы. Полученные в работе результаты использованы:

1. В АО НКБ ВС (г. Таганрог, Ростовская обл.) приняты к использованию способ сравнения слов числового и строкового типа на основе вертикальной обработки данных; алгоритм построения декартова дерева на основе устойчивой адресной сортировки; алгоритм построения двоичного дерева на основе устойчивой адресной сортировки с временной сложностью $T\left(N^2\right)=O\left(\log_2 N\right)$; алгоритм построения двоичного дерева на основе сортировки и априорного вычисления хранимых индексов корней поддеревьев; способ дополнительного ускорения алгоритмов построения декартовых и двоичных деревьев за счет отсутствия вычисления переноса и взаимной независимости обработки разрядов данных. Перечисленные результаты приняты с целью использования при разработке быстродействующих систем преобразования и поиска в реляционных базах данных, а также при разработке систем управления робототехническими комплексами в реальном масштабе времени.

2. В учебном процессе кафедры информатики Таганрогского института имени А.П. Чехова (филиал) ФГБОУ ВО "РГЭУ (РИНХ)" в курсах «Базы данных», «Программирование», «Основы алгоритмизации и программирования», «Информационные системы», «Теория алгоритмов».

Апробация работы. Основные результаты работы докладывались на следующих семинарах и конференциях:

– V Международной научно-практической конференции «Информационные ресурсы и системы в экономике, науке и образовании» (г. Пенза, апрель 2015);

– Пятьдесят девятой научно-теоретической конференции профессорско-преподавательского состава Таганрогского института имени А.П. Чехова (филиала) РГЭУ (РИНХ) (г. Таганрог, апрель 2015);

– XVI Всероссийском симпозиуме по прикладной и промышленной математике (летняя сессия) (г. Челябинск, 21-27 июня 2015 г.);

– Международной научно-практической конференции «Вопросы образования и науки: теоретический и методический аспекты» (г. Тамбов, июнь 2015);

– II Международных научных чтениях (памяти С.Ф. Ковалевской) (г. Москва, 19.09.2016);

– Всероссийской научно-практической конференции с международным участием «Аспекты развития науки, образования и модернизации промышленности» (г. Таганрог, 20-21 апреля 2017);

– XXXII Международной научно-практической конференции (г. Москва, 10.07.2017).

Публикации. По материалам диссертационной работы опубликовано 13 печатных работ, в том числе 3 статьи в рецензируемых журналах, входящих в Перечень ведущих научных журналов и изданий, утвержденных ВАК.

Структура и объем работы. Диссертационная работа состоит из введения, 3 глав основного раздела, заключения, списка литературы и приложения, включающего код программы и акты о внедрении. Основное содержание работы изложено на 172 страницах, включая 10 таблиц, 14 рисунков и библиографии из 179 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность темы диссертационного исследования, охарактеризовано современное состояние методов ускорения базовых операций в структурах данных, в частности, способов ускорения параллельного информационного поиска. Представлен аналитический обзор наиболее важных резервов повышения эффективности систем обработки информации для актуальных разновидностей структур данных, обзор включает системы с алгоритмами параллельного поиска. На основе анализа существующих методов с учетом нерешенных задач определены цель и задачи исследования, сформулированы основные положения, выносимые на защиту. Представлены компоненты научной новизны, используемые методы, а также оценка достоверности результатов, теоретическая и практическая значимость работы, характеристика публикаций и апробации, структура и объем диссертационного исследования.

В **первой главе** разрабатывается и исследуется способ сравнения слов числового и строкового типа на основе взаимной независимости разрядных преобразований алгебраического сложения двоичных чисел, который применяется к алгоритмам сортировки строк в качестве компонента информационного поиска. Способ отличается отсутствием вычисления переноса и максимальным разрядным параллелизмом, что позволяет выполнять сравнение с единичной оценкой временной сложности $O(1)$ независимо от числа символов строк. Способ рассматривается в качестве вспомогательного средства ускорения информационного поиска. В качестве исходного представлен алгоритм поразрядно-параллельного сравнения целых двоичных $(n+1)$-разрядных чисел:

β_n	β_{n-1}	···	β_1	β_0	$\mathrm{PC}_{\text{вх}}^{(0)}$
γ_n	γ_{n-1}	···	γ_1	γ_0	$\mathrm{PC}_{\text{вх}}^{(1)}$

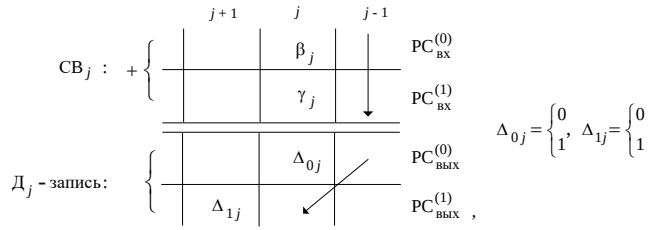
,

где $\mathrm{PC}_{\text{вх}}^{(0)}$, $\mathrm{PC}_{\text{вх}}^{(1)}$ – входные разрядные сетки. Аналогично, $\mathrm{PC}_{\text{вых}}^{(0)}$, $\mathrm{PC}_{\text{вых}}^{(1)}$ обозначают выходные разрядные сетки Сравнение двоичных чисел A и B выполняется по следующему алгоритму:

Алгоритм 1.

а) Число B записывается в обратном коде (единицы заменяются нулями, а нули – единицами). Это выполняется с целью тождественного преобразования:

$A-B=A+\left(\left(2^{n+1}-1\right)-B\right)-\left(2^{n+1}-1\right)$, с учетом $2^{n+1}-1=2^n+2^{n-1}+2^{n-2}+\ldots+2^1+2^0$, или, $2^{n+1}-1=\underbrace{1\,1\,1\ldots1\,1}_{n+1}$. Синхронно и взаимно независимо по всем номерам разрядов выполняется операция суммирования двоичных чисел A и инверсного числа B по вертикали (CB_j):



.

б) Производится перезапись каждого разряда с номером j из $\mathrm{РС}^{(0)}_{\text{вых}}$ в $(j+1)$-й разряд $\mathrm{РС}^{(1)}_{\text{вых}}$ и одновременно инвертируется знак в j-м разряде $\mathrm{РС}^{(0)}_{\text{вых}}$, образуя диагональную от j-го разряда запись.

в) СВН-параллельно по всем номерам разрядов j выполняется операция CB_j суммирования полученных знакоразрядных двоичных чисел. Очевидно, что результат будет иметь вид однорядного двоичного числа в знакоразрядном коде.

г) Для восстановления правильного результата вычитания из полученного результата следует вычесть $2^{n+1}-1$. Иными словами, к младшему разряду добавляется $+1$, а к разряду веса 2^{n+1} добавляется -1.

д) Знак сравнения двух чисел A и B определяется знаком старшего ненулевого разряда однорядного знакоразрядного двоичного числа, полученного на выходе предыдущего шага. Если знак этого разряда отрицательный, то $A<B$, если положительный, то $A>B$, если все разряды нулевые, то $A=B$. Это обусловлено тем, что $2^j>2^{j-1}+2^{j-2}+\ldots+2^1+2^0>\pm2^{j-1}\pm2^{j-2}\pm\ldots\pm2^1\pm2^0$.

Изложенный способ означает возможность поразрядно-параллельного сравнения целых двоичных чисел с единичной оценкой временной сложности при любом количестве разрядов сравниваемых чисел: $T(R)=O(1)$, где число параллельных компонентов устройства сравнения пропорционально числу разрядов, $R=O(n)$. При сравнении слов строкового типа используется двоичный код символов слов, из которых состоит текст в ASCII-коде. Код каждого символа представляется в двоичном виде, а набор двоичных кодов всех символов в порядке расположения интерпретируется как единое числовое значение. Сравнение полученного двоичного кода выполняется по алгоритму 1 с той разницей, что в соответствии с лексикографическим порядком слов выравнивание весов разрядов выполняется не по младшим, а по старшим разрядам. Согласно изложенному способу сравнение слов в двоичном представлении на основе взаимной независимости разрядных преобразований выполнимо с единичной оценкой временной сложности при любом количестве символов сравниваемых слов: $T(R)=O(1)$, где число компонентов устройства сравнения пропорционально числу разрядов двоичного представления меньшего по длине из

сравниваемых слов, $R = O(n)$. На этой основе выполняется параллельный поиск подстроки в строке. Пусть даны подстрока W, состоящая из M символов, и строка P из N символов, и для определенности $M < N$. Выполняется выравнивание начального символа подстроки W с каждым последовательным символом строки P от ее начала. От каждого отдельно фиксированного символа строки P без учета предшествующих символов реализуется сравнение с подстрокой W по алгоритму 1. Сравнение параллельно по всем символам и СВН-параллельно по всем сдвинутым для выравнивания символам. С учетом сдвигов единичная оценка времени корректируется по числу компонентов: $T(R) = O(1)$, $R = O\left(n^2\right)$.

Возникает проблема минимизации временной сложности выделения старшего ненулевого разряда знакоразрядного двоичного слова. В главе предлагается три способа ее решения. Первым способом является выделение старшего ненулевого разряда на основе устойчивой максимально параллельной сортировки подсчетом по матрице сравнений (она приводится ниже при изложении содержания второй главы), применяемой к двоичным битам результата сравнения, эта сортировка сохраняет единичную оценку поразрядно-параллельного сравнения слов в двоичном представлении. Для выделения старшего ненулевого разряда на основе сортировки подсчетом по матрице сравнений необходимо на вход сортировки подать абсолютные величины всех сортируемых элементов входного массива, для которых составляется матрица сравнений. Если слева от первого ненулевого разряда в упорядоченном по возрастанию массиве есть 0, то адрес данного ненулевого элемента во входном массиве знакоразрядных битовых элементов будет указывать знак результата сравнения слов (данная сортировка на выходе хранит индексы входных элементов). При отсутствии ненулевых разрядов – слова равны.

Второй способ является схемотехническим, выполняется за время $O(1)$ путем синхронного распространения сигнала запрета из каждого разряда одновременно во все разряды младшего веса (рис. 1):

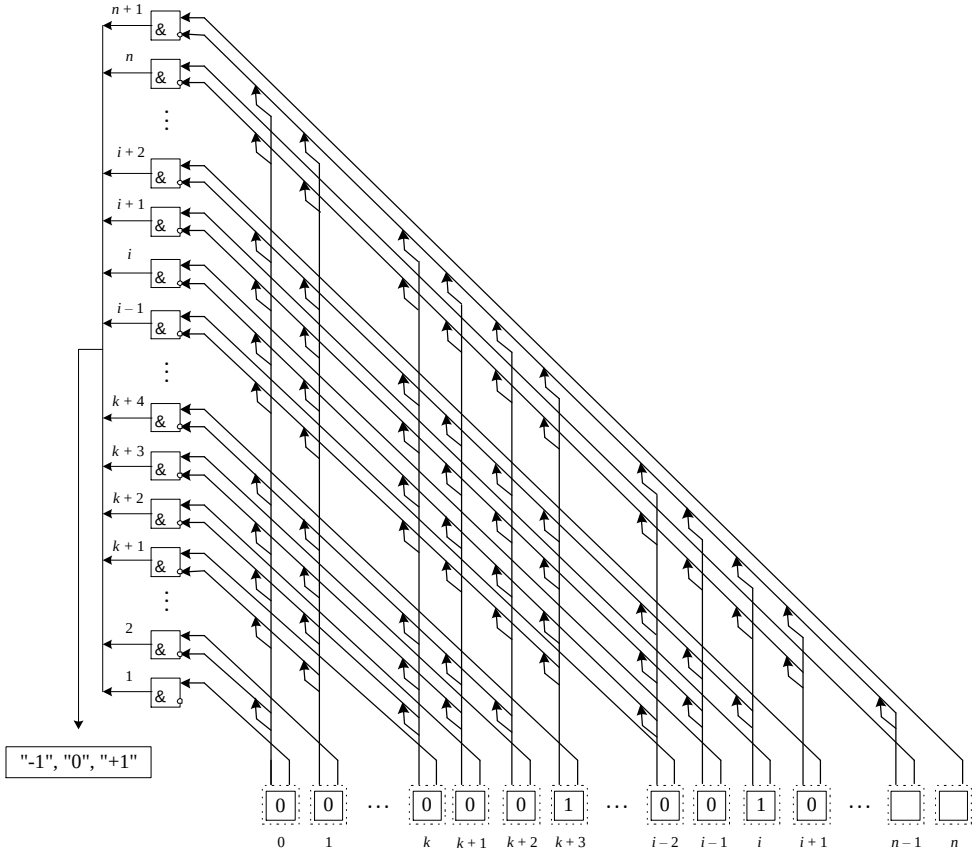


Рис. 1. Схема выделения старшего ненулевого разряда

Схема работает в два такта: в первом – параллельно коммутируются соединения, во втором – параллельно подаются все значения бит со знаками. На открытый выход поступает значение старшего ненулевого разряда со своим знаком.

Данные логические операции поддерживаются в системе команд языка программирования, поэтому схема может быть реализована программно на параллельных процессорных элементах. Схема взаимно независима по разрядным преобразованиям и выделяет знак старшего ненулевого разряда за время $T(R)=O(1)$, где число процессорных элементов пропорционально числу разрядов: $R=O(n)$. Аналогичное обозначение временной сложности используется всюду в дальнейшем. Оценка инвариантна относительно числа символов слова.

Третьим способом является параллельное выделение старшего ненулевого разряда по аналогии с нормализацией двоичных мантисс в формате числа с плавающей точкой. Пусть на выходе алгоритма 1 необходимо выделить знак старшего ненулевого разряда. Двоичное число, являющееся результатом сравнения, располагается по вертикали, как для суммирования одноразрядных чисел посредством опе-

рации $СВ_j$, причем исходными старшими разрядами вниз. В первом такте коммутируются соединения, во втором – передаются значения всех бит со знаком. В первом такте на вход схемы подаются абсолютные значения всех n бит. Пусть для простоты n – целая степень по основанию 2.

Шаг 1. Все n входных слагаемых разбиваются на $\frac{n}{2}$ непересекающихся пар. В каждой паре производится сдвиг верхнего слагаемого на одну позицию вниз, если нижнее слагаемое пары равно нулю. Иначе сдвиг не производится. Преобразование выполняется СВН-параллельно по всем парам. Его результат в виде упорядоченных групп по два слагаемых передается на вход следующего шага.

Шаг $k \geq 2$. Все $\frac{n}{2^{k-1}}$ упорядоченных групп по 2^{k-1} слагаемых, поступивших на вход шага, разбиваются на $\frac{n}{2^k}$ пар. В каждой из таких пар производится сдвиг одновременно всех слагаемых верхней группы данной пары вниз на число позиций, которое равно числу нулей нижней ее группы. Если в последней нулей нет, сдвиг не производится (рис. 2).

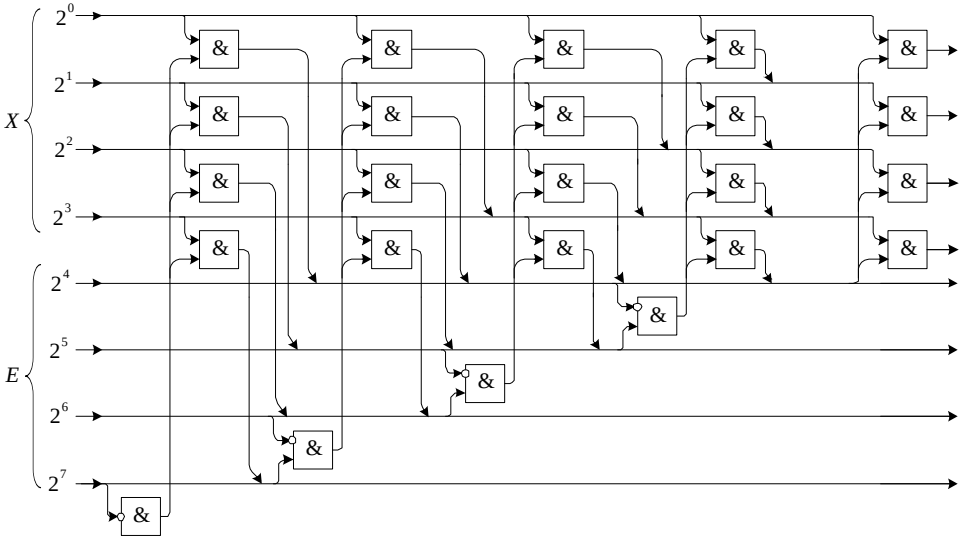


Рис. 2. Схема выполнения k -го шага ($k = 3$)

Преобразование выполняется СВН-параллельно по всем парам групп. Его результат в виде $\frac{n}{2^k}$ упорядоченных групп по 2^k слагаемых в каждой передается на вход следующего шага, если $k < \log_2 n$; иначе результат подается на выход. Выходом является нижний по расположению сверху вниз элемент схемы. На выходе появля-

ется нижняя (в результате сдвига) единица, если она имелась, иначе на выходе будет ноль. Схема срабатывает с минимальной временной сложностью, выполняя k -й шаг за время $T_k = O(1)$, а всю совокупность шагов – за время $T = O(\log_2 n)$. Во втором такте на выход по коммутированным по шагам соединениям элементов схемы поступит знак старшего ненулевого разряда.

Отличием предложенного способа от известных, является то, что он распространяется как на сравнение чисел, в том числе с плавающей точкой, так и без принципиальных изменений – на сравнение слов. Отсюда метод поиска, основанный на таком способе, может включать поиск чисел и слов строкового типа, при этом он обладает максимальным параллелизмом на уровне битовых операций.

Во второй главе и в дальнейшем метод поразрядно-параллельной обработки применяется для ускорения базовых операций в древовидных структурах данных. Основное содержание главы составляет разработка алгоритмов построения декартова дерева с минимизацией временной сложности, для дополнительного ускорения применяются методы и алгоритмы гл. 1. Построение декартова дерева выполняется с применением максимально параллельной сортировки подсчетом на основе матриц сравнений (другая в работе не используется, поэтому ниже для краткости – просто сортировка). Краткое пояснение особенностей сортировки состоит в следующем. Пусть $A = (a_1, a_2, ..., a_n)$ – входной массив, $C = (c_1, c_2, ..., c_n)$ – выходной массив отсортированных элементов. Матрица сравнений носит вспомогательный характер для объяснения сортировки, при программировании она не используется. Для сортировки трех элементов, $n = 3$, матрица имеет вид:

a_i \ a_j	3	-11	5
3	0	–	+
-11	+	0	+
5	–	–	0

Элемент a_{ij} этой матрицы определяется как результат сравнения: если $a_j > a_i$, то $a_{ij} = 1$, если $a_j = a_i$, то $a_{ij} = 0$, если $a_j < a_i$, то $a_{ij} = -1$, $i, j = 1, 2, ..., n$. Значения a_{ij} заменяются соответственно символами "+", "0", "–". Номер входного элемента a_j в выходной отсортированной последовательности определяется количеством нулей и плюсов в j -м столбце до главной диагонали включительно, сложенному с количеством только плюсов в этом же столбце ниже главной диагонали. Выходные индексы располагаются в порядке отсортированных элементов: $c[k] := a[j]$, $e[k] := j$. В данном примере $a_1 = 3 \to c_{(1+1)} = c_2$; $a_2 = -11 \to c_{(1+0)} = c_1$; $a_3 = 5 \to c_{(3+0)} = c_3$, $e = (2, 1, 3)$. Сортировка является устойчивой – сохраняет порядок равных элементов, в явном виде задает взаимно однозначное соответствие входных и выходных индексов. Все сравнения взаимно независимы, поэтому сортировка максимально параллельна с временной сложностью $T\left(\frac{n^2}{2}\right) = O(1)$. Для минимизации временной сложности построения декартова дерева сортировка используется следующим образом. Пусть задано некоторое множество пар (X_i, Y_i). Ко всем компонентам Y_i – элементов множества (X_i, Y_i) применяется сортировка по неубыванию. Пары уже отсортированных по Y_i элементов (X_i, Y_i) располагаются с сохранением входного индекса без изменения единич-

ной оценки времени. Наибольший среди Y_i элемент окажется в правом конце отсортированного массива вместе с входным индексом пары (X_i,Y_i). Пусть эта пара обозначена $(X_i,Y_i)_N$, а ее входной индекс в отсортированном массиве – E_N, $E_N = i$, таким образом, определен корень декартова дерева $(X_i,Y_i)_N$. Далее, без изменения его местоположения во входном массиве, которое идентифицируется по индексу E_N, этот элемент объявляется серединным в этом же входном массиве. Относительно середины выполняется упорядочение по X_i: все элементы в левой части массива, которые меньше X_i из $(X_i,Y_i)_N$, остаются в левом подмассиве с сохранением исходного взаимного порядка; все элементы в левой части массива, которые больше X_i из $(X_i,Y_i)_N$, переносятся из левой части в правый подмассив, также с сохранением исходного взаимного порядка. Все элементы правой части массива, которые больше X_i из $(X_i,Y_i)_N$, остаются в правом подмассиве с сохранением исходного взаимного порядка; все элементы правой части массива, которые меньше X_i из $(X_i,Y_i)_N$, переносятся из правой части в левый подмассив, также с сохранением исходного взаимного порядка. Выполнение упорядочения относительно середины производится аналогично сортировке подсчетом и представляет собой шаг сортировки Хоара в параллельной форме. Например, при $N = 5$ в случае множества пар (1; 1), (2; 9), (2; 0), (0; 5), (3; 2) корнем декартова дерева является (2; 9), относительно корня выполняется упорядочение по X_i. В результате определятся элементы левого поддерева (1; 1), (2; 0), (0; 5) и правого поддерева (3; 2) искомого декартова дерева с корнем (2; 9). В общем случае описанные действия параллельно повторяются отдельно в каждом подмассиве, при этом наибольший среди Y_i элемент левого подмассива окажется левым потомком корня декартова дерева $(X_i,Y_i)_N$, наибольший среди Y_i элемент правого подмассива окажется правым его потомком. Действия рекуррентно воспроизводятся в каждом новом подмассиве, образуемом потомком как корнем поддерева, причем параллельно по всем подмассивам, соответственным корням текущего уровня. На рис. 3 изображен результат построения декартова дерева в стандартной нотации для массива пар (1; 1), (11; 4), (2; 9), (8; 9), (2; 0), (10; 8), (0; 5), (5; 8), (4; 10), (7; 7), (3; 2):

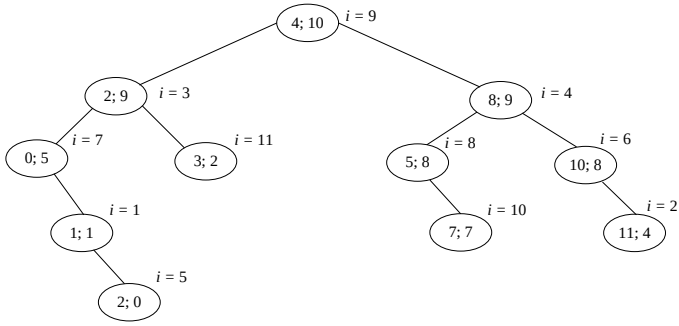


Рис. 3. Пример декартова дерева

Последовательные шаги алгоритма продолжаются до полного построения декартова дерева, их количество не превосходит $\log_2 N$. В общем случае, соответственно уровню корней поддеревьев с номером $\log_2 k$, $k \le N$, количество сортируемых по всем подмассивам элементов удовлетворяет равенству $\sum_{i=1}^{\log_2 k} N_i = N$. В этом случае количество процессоров при параллельном по всем i выполнении описанных операций сравнения удовлетворяет соотношению $\sum_{i=1}^{\log_2 k} R_i \le \frac{1}{2} N \left(\sum_{i=1}^{\log_2 k} N_i - 1 \right) = \frac{N^2 - N}{2}$.

Таким образом, с учетом единичного времени каждого последовательного шага и числа шагов $\log_2 N$ имеет место

Теорема 1. Построение декартова дерева может быть выполнено с помощью детерминированного параллельного алгоритма за логарифмическое число шагов с временной сложностью $T\left(\frac{N^2 - N}{2}\right) = O\left(\log_2 N\right)$.

При перемещении входных индексов пар (X_i, Y_i) по уровням и подмассивам в соответствии шагам изложенного построения декартова дерева возможна адресация от элементов окончательно построенного декартова дерева непосредственно к элементам входного массива и обратно. Сопоставление предложенного алгоритма с известными аналогами представлено в табл. 1.

Таблица 1

Сравнительные оценки временной сложности последовательных и параллельных алгоритмов построения декартова дерева в сопоставлении с предложенным алгоритмом

Метод выполнения построения	Временная сложность алгоритма	Ускорение при использовании предложенного алгоритма
Примитивный алгоритм (1996)	$O\left(N \log_2 N\right)$	$O\left(N\right)$
Линейный алгоритм (1994)	$O\left(N\right)$	$O\left(\frac{N}{\log_2 N}\right)$
Параллельный алгоритм построения (2014)	$O\left(N \log_2^2 N\right)$	$O\left(N \log_2 N\right)$
Параллельный алгоритм ANSV (2012)	$O\left(N \log_2 N\right)$	$O\left(N\right)$
Параллельный алгоритм CREW PRAM (2013)	$O\left(N \log_2 N\right)$	$O\left(N\right)$
Алгоритм минимальных промежутков (2011)	$O\left(\frac{N}{\log_2 N} \log_2 \frac{N}{\log_2 N}\right)$	$O\left(N \ln 2\right)$
Предложенный алгоритм (2015)	$O\left(\log_2 N\right)$	–

Согласно таблице предложенный алгоритм улучшает оценки временной сложности известных аналогов. При использовании поразрядно-параллельных сравнений по методу первой главы ускорение предложенного алгоритма по отношению к известным, представленным в табл. 1, возрастет и составит соответственно: $O(N)n$; $O\left(\frac{N}{\log_2 N}\right)n$; $O(N\log_2 N)n$; $O(N)n$; $O(N)n$; $O(N\ln 2)n$, где n – число символов сравниваемых элементов строкового типа.

Помимо отмеченных особенностей, улучшение известных оценок дополнительно обусловлено тем, что в предложенном построении декартова дерева отсутствует этап предварительной обработки входных данных.

В третьей главе излагается построение двоичного дерева с минимизацией временной сложности при помощи сортировки. Метод включает детерминированные алгоритмы, временная сложность которых дает оценку ускорения существующих последовательных и параллельных алгоритмов, используемых для этой цели. Первый алгоритм строится следующим образом. Пусть задано некоторое множество из N элементов X_i с отношением порядка $\le$, расположенных в виде одномерного массива. Массив сортируется по данному отношению с единичной оценкой временной сложности. В качестве корня двоичного дерева выбирается серединный элемент отсортированного массива C с округлением индекса середины до ближайшего целого, не меньшего самого числа: $j_{\text{ср}} = \left\lceil \frac{N}{2} \right\rceil$. С учетом устойчивости сортировки все элементы отсортированного массива слева от $C_{j_{\text{ср}}}$ меньше $C_{j_{\text{ср}}}$ (в смысле данного отношения порядка), они автоматически определяются принадлежащими левому поддереву (левому подмассиву). Аналогично, все элементы справа от $C_{j_{\text{ср}}}$ больше $C_{j_{\text{ср}}}$ и определяются как принадлежащие правому поддереву (правому подмассиву). Далее, левый подмассив рассматривается как новый массив для аналогичного определения его корня по номеру: $j_{\text{ср. лев. }1/2} = \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1\right) \right\rceil = \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil$. Такой корень, $C_{j_{\text{ср. лев. }1/2}}$, станет ближайшим слева потомком ранее найденного корня всего дерева $C_{j_{\text{ср}}}$, а также корнем левого поддерева. В силу сортировки левое поддерево с корнем $C_{j_{\text{ср. лев. }1/2}}$ сохраняет требуемые свойства: все элементы подмассива слева от $C_{j_{\text{ср. лев. }1/2}}$ не превосходят $C_{j_{\text{ср. лев. }1/2}}$, все элементы подмассива справа, аналогично, не меньше $C_{j_{\text{ср. лев. }1/2}}$. Одновременно с левым правый подмассив рассматривается как новый массив для аналогичного определения его корня по номеру: $j_{\text{ср. прав. }1/2} = \left\lceil \frac{N}{2} \right\rceil + \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1\right) \right\rceil = j_{\text{ср}} + \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil$. Такой корень, $C_{j_{\text{ср. прав. }1/2}}$, станет ближайшим справа потомком ранее найденного корня всего дерева $C_{j_{\text{ср}}}$, а также корнем правого поддерева. Правое поддерево с корнем $C_{j_{\text{ср. прав. }1/2}}$ сохраняет требуемые свойства: все элементы справа от $C_{j_{\text{ср. прав. }1/2}}$ больше $C_{j_{\text{ср. прав. }1/2}}$, все элементы слева меньше $C_{j_{\text{ср. прав. }1/2}}$. Далее, процесс вычисления индексов узлов итера-

тивно воспроизводится по отношению к каждой паре смежных подмассивов слева и, одновременно, справа:

$$j_{\text{ср. лев. }1/4,1}=\left\lceil\frac{j_{\text{ср. лев. }1/2}-1}{2}\right\rceil,\quad j_{\text{ср. лев. }1/4,2}=j_{\text{ср. лев. }1/2}+\left\lceil\frac{j_{\text{ср. лев. }1/2}-1}{2}\right\rceil,\quad j_{\text{ср. прав. }1/4,1}=j_{\text{ср. прав. }1/2}-\left\lceil\frac{j_{\text{ср. прав. }1/2}-j_{\text{ср}}-1}{2}\right\rceil,$$

$$j_{\text{ср. прав. }1/4,2}=j_{\text{ср. прав. }1/2}+\left\lceil\frac{j_{\text{ср. прав. }1/2}-j_{\text{ср}}-1}{2}\right\rceil,\ \ldots,\ j_{\text{ср. лев. }1/2^i,1}=\left\lceil\frac{j_{\text{ср. лев. }1/2^{i-1}}-1}{2}\right\rceil,\ j_{\text{ср. лев. }1/2^i,2}=j_{\text{ср. лев. }1/2^{i-1}}+\left\lceil\frac{j_{\text{ср. лев. }1/2^{i-1}}-1}{2}\right\rceil,$$

$$j_{\text{ср. прав. }1/2^i,1}=j_{\text{ср. прав. }1/2^{i-1}}-\left\lceil\frac{j_{\text{ср. прав. }1/2^{i-1}}-j_{\text{ср. прав. }1/2^{i-2}}-1}{2}\right\rceil,\ j_{\text{ср. прав. }1/2^i,2}=j_{\text{ср. прав. }1/2^{i-1}}+\left\lceil\frac{j_{\text{ср. прав. }1/2^{i-1}}-j_{\text{ср. прав. }1/2^{i-2}}-1}{2}\right\rceil,$$

где $i=1, 2, \ldots, \lceil\log_2 N\rceil$. В результате за время $O(1)$ сформируются все элементы нижестоящего уровня двоичного дерева. Процесс можно продолжать до исчерпания $\lceil\log_2 N\rceil$ уровней двоичного дерева. Число шагов параллельного алгоритма построения двоичного дерева складывается из шага сортировки и последовательности шагов вычисления индексов корней поддеревьев. Число индексов равно целой части логарифма числа элементов входного множества. В этом случае временную сложность оценивает

Теорема 2. Двоичное дерево для массива из N элементов может быть построено параллельно на основе сортировки с логарифмической оценкой временной сложности $T\left(\frac{N^2}{2}\right)=O\left(\log_2 N\right)$.

Очевидно, что индексы всех серединных элементов всех уровней могут быть вычислены за один шаг. При этом адресоваться к соответственным корням и элементам поддеревьев можно по ссылкам на основе явного задания взаимно однозначного соответствия входных и выходных индексов сортируемых элементов. Поэтому как следствие теоремы интерпретируется следующее утверждение.

Следствие 1. В условиях теоремы 2 утверждение теоремы может быть выполнено с временной сложностью $T\left(\frac{N^2}{2}\right)=O(1)$ на основе априорного вычисления индексов всех корней и потомков.

Пусть, например, требуется построить двоичное дерево для массива из $N=15$ элементов $X=(14,\ 9,\ 24,\ 7,\ 11,\ 20,\ 28,\ 3,\ 8,\ 10,\ 13,\ 17,\ 21,\ 25,\ 30)$. Выполняется сортировка массива X, результатом которой является массив $C=(3,\ 7,\ 8,\ 9,\ 10,\ 11,\ 13,\ 14,\ 17,\ 20,\ 21,\ 24,\ 25,\ 28,\ 30)$. Корнем двоичного дерева является серединный элемент массива C: $j_{\text{ср}}=\left\lceil\frac{15}{2}\right\rceil=8,\ \ C_8=14$. Левый подмассив рассматривается как новый массив для аналогичного определения корня, $j_{\text{ср. лев. }1/2}=\left\lceil\frac{8-1}{2}\right\rceil=4$, элемент $C_4=9$ является ближайшим слева потомком корня $C_{j_{\text{ср}}}$ и корнем левого поддерева. Одновременно правый подмассив рассматривается для аналогичного определения его корня, $j_{\text{ср. прав. }1/2}=8+\left\lceil\frac{8-1}{2}\right\rceil=12$, элемент $C_{12}=24$ – ближайший справа потомок корня $C_{j_{\text{ср}}}$ и корень правого поддерева. Далее, $j_{\text{ср. лев. }1/4,1}=\left\lceil\frac{4-1}{2}\right\rceil=2,\ \ C_2=7$; $j_{\text{ср. лев. }1/4,2}=4+\left\lceil\frac{4-1}{2}\right\rceil=6,\ \ C_6=11$; $j_{\text{ср. прав. }1/4,1}=12-\left\lceil\frac{12-8-1}{2}\right\rceil=10,\ \ C_{10}=20$; $j_{\text{ср. прав. }1/4,2}=12+\left\lceil\frac{12-8-1}{2}\right\rceil=14$,

$C_{14} = 28$. Слева и справа от каждого из четырех найденных корней текущего уровня осталось по одному потомку, которые составят нижний уровень дерева (рис. 4):

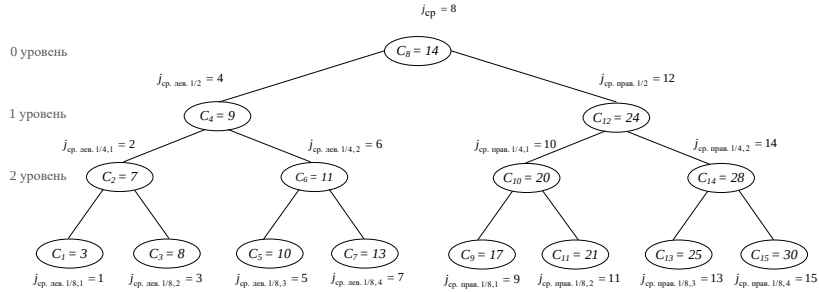


Рис. 4. Пример построения двоичного дерева на основе сортировки

В силу устойчивости сортировки двоичное дерево изложенным способом строится со свойством единственности. Формулы для вычисления индексов узлов зависят только от количества входных элементов N и не зависят от их взаимного расположения. Поэтому такие вычисления можно выполнить для каждого значения N в отдельности априори. При хранении в памяти компьютера общим адресом массива индексов являет значение числа N. Адресом же каждого индекса является указанные в примере нижние индексы с отметкой адресации вправо или влево. Например, $j_{ср.\ прав.\ 1/4,2}$ означает, что требуется правый от корня дерева индекс узла на втором от корня уровне. Для упрощения обращения к памяти индексы упорядочиваются слева направо на каждом уровне и располагаются по возрастанию уровней. Тогда при обращении к памяти по ключу N считывается вся совокупность упорядоченных индексов узлов. После этого по считанным адресам располагаются отсортированные элементы дерева. Таким образом, после сортировки, для получения дерева не требуется дополнительных вычислений, требуется лишь однократное обращение к памяти. Значения индексов могут быть априори рассчитаны для всех значений N в некоторых реальных границах и храниться в памяти компьютера.

Теорема 3. Двоичное дерево для массива из N элементов может быть построено на основе сортировки и априорного вычисления индексов, хранимых в памяти компьютера, с единичной оценкой временной сложности $T\left(\frac{N^2}{2}\right) = O(1)$.

Рассматриваемая сортировка позволяет из произвольного расположения элементов в массиве извлекать информацию обо всех экстремальных закономерностях. Пусть по отношению «$\le$» требуется упорядочить элементы массива $\{c_l\}_{l=1}^{n}$, $n < \infty$, в результате образуется массив $C1 = (c1[1], c1[2], ..., c1[n])$, где выполнено $c1[k] := c[j]$ и $e[k] := j$. При каждом k условие локализации минимального элемента проверяется в виде системы неравенств $|e[k] - e[k-L]| > \varepsilon$, $L = 1,2,...,k-1$. Выполнение одновременно всех этих неравенств означает, что внутри ε-окрестности входного индекса элемента массива c_l с индексом $e[k]$ нет индекса элемента массива $C1$, меньшего по значению, чем элемент $c1[k]$. Таким образом, идентифицируются локальные минимумы

при всех значениях радиуса локализации ε. Аналогично, по условию $|e[k]-e[k+L]|>\varepsilon$, $L=1,2,...,N-k$, идентифицируются все локальные максимумы. Для идентификации глобального минимума, достаточно воспользоваться условием $|e[k]-e[k-L]|\geq 1$, $L=1,2,...,k-1$, а глобального максимума – $|e[k]-e[k+L]|\geq 1$, $L=1,2,...,N-k$. Эти условия относятся исключительно к взаимному входному расположению индексов $e[k]$, $e[k-L]$, $e[k+L]$. Сортировка, располагая эти индексы в порядке отсортированных элементов $c[k]$, $c[k-L]$, $c[k+L]$, позволяет проверить соотношения и без преобразований и вычислений извлечь всю информацию об экстремальных закономерностях. Согласно изложенному этот факт можно дополнить следующим утверждением: вся информация о структуре дерева для входного массива содержится в индексах элементов отсортированного массива. Но она содержится и непосредственно в самом входном массиве, к элементам которого можно адресоваться по ссылкам $e[k]$, $e[f(k)]$, где $f(k)$ – вычисленные адреса узлов двоичного дерева. Поэтому рассматриваемая сортировка входного массива является универсальным алгоритмом извлечения закономерностей структуры двоичного дерева из хаоса расположения входных элементов.

В главе приведен алгоритм последовательного построения двоичного дерева, основанный на алгоритме устойчивой последовательной сортировки слиянием с явным заданием взаимно однозначного соответствия входных и выходных индексов, ее временная сложность $\sim N\log_2 N\tau$, где τ – время бинарного сравнения. Отсюда справедливо следующее утверждение.

Теорема 4. Последовательное построение двоичного дерева для массива из N элементов может быть выполнено с временной сложностью $T(1)=O(N\log_2 N)$.

Изложенные алгоритмы имеют конструктивную форму, являются детерминированными и с единственностью определяют каждый свой шаг. Поэтому двоичное дерево с единственностью строится на основе сортировки. Между двоичным деревом и отсортированной последовательностью его элементов существует взаимно однозначное соответствие, которое в обе стороны (обход сверху-вниз, либо снизу-вверх) реализуется конструктивным алгоритмом. В табл. 2 дано сравнение временной сложности предложенных алгоритмов с известными аналогами.

Таблица 2

Сравнительные оценки временной сложности последовательных и параллельных алгоритмов построения двоичного дерева

Алгоритм построения двоичного дерева	Ускорение при использовании предложенного последовательного алгоритма (теорема 4)	Ускорение при использовании предложенного параллельного алгоритма с логарифмическим числом шагов (теорема 2)	Ускорение при использовании предложенного алгоритма с априорной записью адресов в память (теорема 3)
Алгоритм (Lagana A., Kumar V., Tan C.) $O(N)$	$O\left(\frac{1}{\log_2 N}\right)$	$O(N\ln 2)$	$O(N)$
Алгоритм (Chalermsook P.) $O(N^2)$	$O(N)$	$O(N^2\ln 2)$	$O(N^2)$

Полиномиальный алгоритм $O(N^3)$	$O(N^2)$	$O(N^3 \ln 2)$	$O(N^3)$
LCRS-алгоритм $O(N^2)$	$O(N)$	$O(N^2 \ln 2)$	$O(N^2)$
Алгоритм на основе шаблонов $O(\|D\|\log_2 D)$	$O\left(\frac{\|D\|\log_2 D}{N \log_2 N}\right)$	$O\left(\frac{\|D\|\log_2 D}{\log_2 N}\right)$	$O(\|D\|\log_2 D)$

В табл. 2 D – мощность словаря шаблонов, N – число элементов двоичного дерева.

При использовании поразрядно-параллельных сравнений по методу первой главы ускорение известных аналогов, представленных в табл. 2, дополнительно увеличится в $O(n)$, где n – число разрядов представления данных. В частности, суммарное ускорение по алгоритму теоремы 3 составит: для алгоритма (Lagana A., Kumar V., Tan C.) – $O(Nn)$, для алгоритма (Chalermsook P.) – $O(N^2 n)$, для полиномиального алгоритма – $O(N^3 n)$; для LCRS-алгоритма – $O(N^2 n)$; для алгоритма на основе шаблонов – $O(n|D|\log_2 D)$.

В целом улучшение известных оценок обусловлено тем, что в предложенном построении двоичного дерева отсутствует этап предварительной обработки входных данных, это, а также структуры предложенных алгоритмов позволяют существенно ускорить построение двоичного дерева относительно известных аналогов.

В заключении обобщаются основные результаты диссертационной работы, характеризуется их научная новизна, отмечается практическое значение проведенных исследований.

Основной результат диссертации заключается в разработке и исследовании способов ускорения организации и преобразований структур данных для информационной обработки на основе алгоритмов устойчивой адресной сортировки в применении к базовым операциям информационного поиска. Как средство ускорения процессов организации и обработки данных предложены алгоритмические и разрядные преобразования информационных данных во взаимно независимой форме, на основе разработанных алгоритмов представлен конструктивный способ извлечения структурных закономерностей информационных данных, скрытых в массивах входной информации.

Наряду с основным следующие результаты отличаются **научной новизной**:

1. Разработан способ сравнения данных числового и строкового типа на основе вертикальной обработки в качестве компонента информационного поиска без вычисления переноса и с взаимной независимостью анализа разрядов данных.

2. На основе устойчивой адресной сортировки разработан алгоритм построения декартова дерева информационных данных с логарифмической временной сложностью.

3. Разработан алгоритм построения двоичного дерева информационных данных на основе устойчивой адресной сортировки с временной сложностью $O(\log_2 N)$.

4. На основе сортировки и априорного вычисления хранимых индексов корней поддеревьев разработан алгоритм построения двоичного дерева информационных данных с минимизацией временной сложности до значения $O(1)$.

5. Разработаны алгоритмы построения декартовых и двоичных деревьев информационных данных с дополнительным ускорением за счет отсутствия вычисления переноса и взаимной независимости обработки разрядов данных при выполнении операций сравнения.

6. Показано, на основе предложенного алгоритма построения двоичного дерева информационных данных, из произвольного расположения данных в массиве можно извлечь полную информацию об экстремальных закономерностях и, аналогично, информацию о структурных закономерностях двоичного дерева, априори скрытую во входных данных.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИОННОЙ РАБОТЫ

Публикации в ведущих рецензируемых изданиях, рекомендованных ВАК

1. Ромм Я.Е., Чабанюк Д.А. Сравнение слов с единичной временной сложностью // Известия ЮФУ. Технические науки. – № 7(156). – 2014. – С. 230-238.

2. Ромм Я.Е., Чабанюк Д.А. Параллельное построение декартова дерева с логарифмической оценкой временной сложности // Современные проблемы науки и образования. – 2015. – № 1; URL: http://www.science-education.ru/121-18604 (дата обращения: 25.02.2017).

3. Ромм Я.Е., Чабанюк Д.А. Построение двоичного дерева на основе параллельной сортировки // Фундаментальные исследования. – 2015. – № 8 (часть 3). – С. 509-513.

Публикации в других изданиях

4. Ромм Я.Е., Чабанюк Д.А. Применение метода вертикальной обработки информации к операциям сравнения и поиска / ТГПИ – Таганрог, 2013. – 32 с. Деп. в ВИНИТИ 20.05.13, № 141 – В 2013.

5. Ромм Я.Е., Чабанюк Д.А. Сравнение слов с единичной временной сложностью / ТГПИ – Таганрог, 2013. – 30 с. Деп. в ВИНИТИ 30.10.13, № 306 – В 2013.

6. Ромм Я.Е., Чабанюк Д.А. Поразрядно-параллельное сравнение ключей в некоторых древовидных структурах данных / Таганрог. ин-т им. А.П. Чехова (филиал) ФГБОУ ВПО «РГЭУ (РИНХ)». – Таганрог, 2014. – 41 с. Деп. в ВИНИТИ 04.09.14, № 244 – В2014.

7. Ромм Я.Е., Чабанюк Д.А. Параллельные алгоритмы обработки структур данных и последовательное моделирование построения декартова дерева / Таганрог. ин-т им. А.П. Чехова (филиал) ФГБОУ ВПО «РГЭУ (РИНХ)». – Таганрог, 2015. – 53 с. Деп. в ВИНИТИ 16.01.15, № 9 – В2015.

8. Чабанюк Д.А. Сравнение слов за единичное время с применением схем вертикальной обработки данных // Информационные ресурсы и системы в экономике, науке и образовании. Сборник статей V Международной научно-практической конференции. – Пенза, 2015. – вып. 5 – С. 97-102.

9. Ромм Я.Е., Чабанюк Д.А. Параллельное построение декартова дерева // Обозрение прикладной и промышленной математики. – Т.22, № 1, Челябинск,

2015. - С. 87-88.

10. Ромм Я.Е., Чабанюк Д.А. Выделение старшего ненулевого разряда при поразрядно-параллельном способе сравнения слов // Вопросы образования и науки: теоретический и методический аспекты. – Т. 5., Тамбов, 2015. – С. 124-125.

11. Ромм Я.Е., Чабанюк Д.А. Сопоставление оценок временной сложности параллельного построения декартова дерева // II Международные научные чтения (памяти С.Ф. Ковалевской). Сборник статей Международной научно-практической конференции (Москва, 19.09.2016 г.). – Москва: РИЦ ЕФИР, 2016. – С. 18-21.

12. Ромм Я.Е., Чабанюк Д.А. Параллельное построение двоичного дерева на основе сортировки // Материалы Всероссийской научно-практической конференции с международным участием "Аспекты развития науки, образования и модернизации промышленности" – Ростов-на-Дону: ДГТУ. – С. 77-84.

13. Ромм Я.Е., Чабанюк Д.А. Параллельное построение двоичного дерева на основе сортировки с логарифмической и единичной временной сложностью // Актуальные вопросы науки: Материалы XXXII Международной научно-практической конференции (10.07.2017) М.: Спутник +, 2017. – С. 118-123.

Личный вклад автора в работах, опубликованных в соавторстве:

– [1, 4, 5, 8] в качестве компонента информационного поиска предложены способы выделения старшего ненулевого разряда при сравнении слов числового и строкового типа;

– [2, 7, 9, 11] предложен алгоритм параллельного построения декартова дерева на основе устойчивой сортировки;

– [3, 6, 12, 13] разработан алгоритм параллельного построения двоичного дерева на основе устойчивой сортировки с логарифмической временной сложностью;

– [10, 12] предложен способ априорного вычисления хранимых индексов корней поддеревьев с минимизацией на этой основе временной сложности построения двоичного дерева до значения $O(1)$ в параллельном и последовательном вариантах.

Соискатель  Чабанюк Денис Андреевич