Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ «Программной инженерии и компьютерной техники»

ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА «Проектирование и разработка систем больших данных»

НАПРАВЛЕНИЕ ПОДГОТОВКИ (СПЕЦИАЛЬНОСТЬ) «09.04.04 Программная инженерия»

ОТЧЕТ

по лабораторной работе №5

по курсу «Хранение и алгоритмы сжатия данных»

на тему: «Компактные структуры данных»

Обучающийся: Постнов С., P4135
(Ф.И.О, № группы)

Обучающийся: Нурджанян В., P4135
(Ф.И.О, № группы)

Обучающийся: Русинов Д., P4135
(Ф.И.О, № группы)

Преподаватель Бабаянц А. А., преподаватель
(Ф.И.О, должность)

2025 г.

СОДЕРЖАНИЕ

1 Теоретическая часть **3**

1.1 Классические n-арные деревья 3

1.2 LOUDS (Level-Order Unary Degree Sequence) 3

2 Практическая часть **5**

2.1 Результаты работы 5

2.1.1 Использование памяти 5

2.1.2 Производительность операций 6

2.1.3 Графики результатов сравнения 8

3 Выводы **9**

1 Теоретическая часть

1.1 Классические n-арные деревья

Классическое n-арное дерево — это фундаментальная структура данных, где каждый узел может иметь произвольное количество дочерних узлов. Реализуется посредством явных указателей: каждый узел содержит ссылку на родителя и список ссылок на детей.

Структура узла:

1) Значение узла (строка).

2) Указатель на родительский узел.

3) Список указателей на дочерние узлы.

Достоинства:

— Простая реализация.

— Быстрый доступ к соседним узлам (O(1) для дочерних узлов).

— Интуитивная структура для представления иерархических данных.

Недостатки:

— Высокие накладные расходы на память из-за указателей.

— O(n) сложность поиска узла по значению.

— Плохая локальность кэша при обходе дерева.

1.2 LOUDS (Level-Order Unary Degree Sequence)

LOUDS — это компактная структура данных для хранения деревьев, которая использует битовый массив вместо явных указателей. Представление основано на уровневом порядке обхода (BFS) и унарной кодировке степеней узлов.

Основная идея:

1) Узлы нумеруются в порядке обхода в ширину (BFS).

2) Для каждого узла с k детьми добавляется k единиц и один ноль в битовый массив.

3) Инвертированный индекс (map) хранит соответствие значению узла его индекс для быстрого поиска.

Операции над битовым массивом:

— **Rank(i, bit)** — подсчитать количество заданного бита до позиции i.

— **Select(n, bit)** — найти позицию n-го вхождения бита.

Достоинства LOUDS:

— Компактное хранение.

— Хорошая локальность кэша, последовательный доступ к битам.

— O(1) операции поиска дочерних узлов при использовании оптимизированного индекса.

Недостатки:

— Сложная реализация навигации (требуется работа с битами).

— O(n) для наивной реализации Select и Rank (может быть оптимизирована до O(log(log(N))) и O(1) соответственно).

2 Практическая часть

В лабораторной работе реализованы две структуры данных для представления n-арных деревьев:

1) Классическое дерево с явными указателями.

2) LOUDS дерево с компактным битовым представлением.

Произведено сравнение производительности обеих реализаций на основе четырех операций навигации:

— **FirstChild** — получить первого потомка узла.

— **LastChild** — получить последнего потомка узла.

— **ChildrenCount** — получить количество потомков узла.

— **Parent** — получить родителя узла.

2.1 Результаты работы

Замеры времени проводились на деревьях размером от 10 до 10000 узлов. Для каждой операции каждой размерности выполнялось 10000 операций, после чего вычислялось среднее значение.

2.1.1 Использование памяти

Таблица 2.1 – Сравнение использования памяти

Количество узлов	**Классическое дерево (байт)**	**LOUDS дерево (байт)**
10	239	129
50	1278	698
100	2611	1444
500	13038	7171
1000	26059	14317
5000	130105	71363
10000	260344	142852

2.1.2 Производительность операций

В таблицах 2.2- 2.3 представлены результаты сравнения производительности функций `FirstChild` и `Parent` соответственно.

Таблица 2.2 – Сравнение производительности: FirstChild (наносекунды)

Узлов	**Классическое дерево (нс)**	**LOUDS дерево (нс)**
10	34	53
50	203	81
100	294	118
500	1020	141
1000	1423	196
5000	9750	783
10000	28360	1543

Таблица 2.3 – Сравнение производительности: Parent (наносекунды)

Узлов	**Классическое дерево (нс)**	**LOUDS дерево (нс)**
10	38	48
50	162	81
100	220	130
500	746	272
1000	1303	428
5000	9487	1948
10000	27782	3795

В таблицах 2.4- 2.5 представлены результаты сравнения производительности функций `LastChild` и `ChildrenCount` соответственно.

Таблица 2.4 – Сравнение производительности: LastChild (наносекунды)

Узлов	**Классическое дерево (нс)**	**LOUDS дерево (нс)**
10	28	49
50	196	118
100	268	150
500	922	361
1000	1345	534
5000	9591	2458
10000	28152	4857

Таблица 2.5 – Сравнение производительности: ChildrenCount (наносекунды)

Узлов	**Классическое дерево (нс)**	**LOUDS дерево (нс)**
10	31	68
50	170	171
100	240	202
500	803	548
1000	1304	909
5000	9332	4215
10000	27737	8309

2.1.3 Графики результатов сравнения

Графики результатов сравнения представлены на рисунке 2.1:

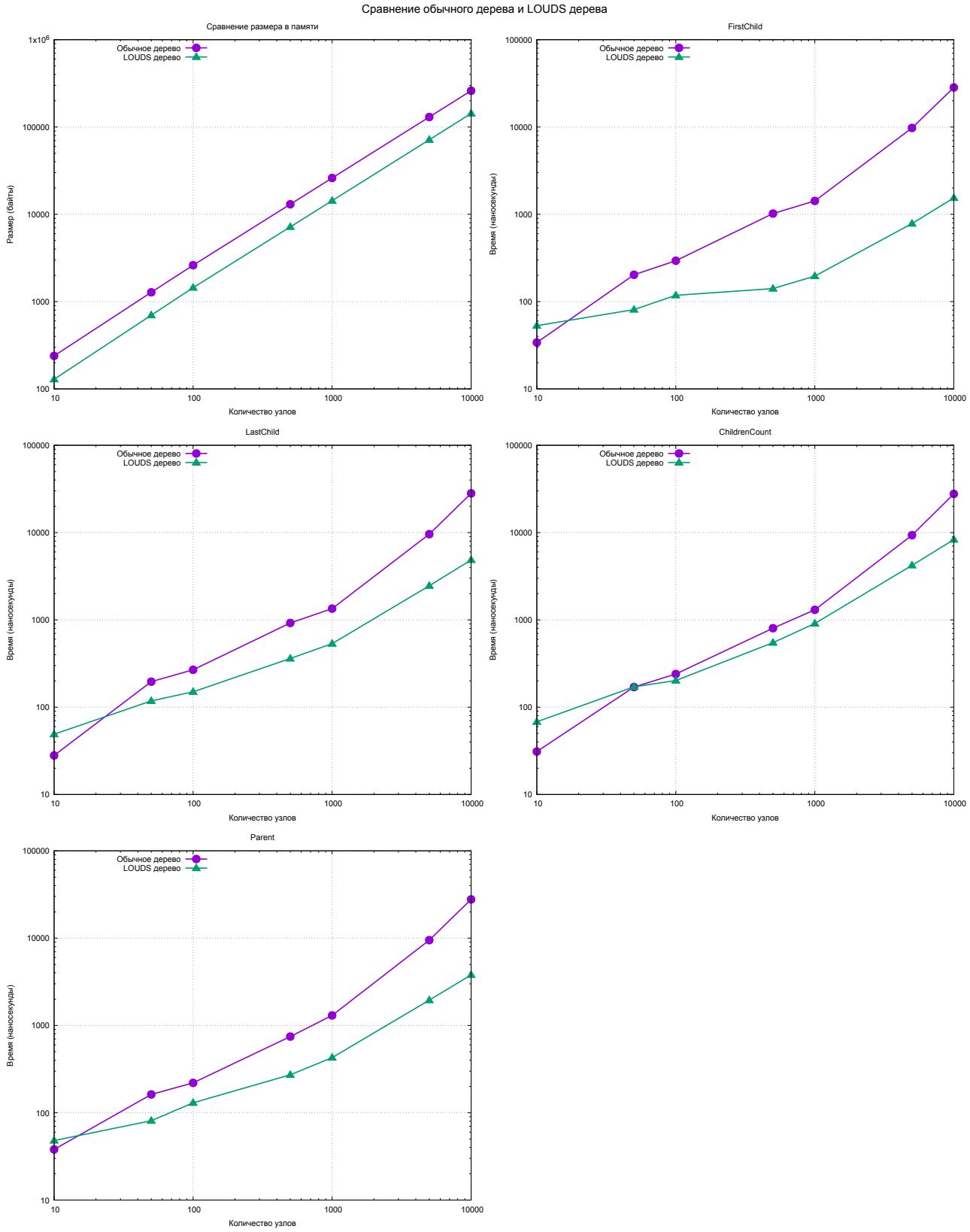

Рисунок 2.1 – Сравнение классического дерева и LOUDS дерева: память и производительность всех операций

3 Выводы

На основе проведенного анализа можно сделать следующие выводы:

1) LOUDS дерево является более экономным (на ∼45%) памяти по сравнению с классическим деревом, сохраняя полную функциональность.

2) LOUDS демонстрирует существенное преимущество по времени выполнения операций (по данным измерений на дереве из 10 000 узлов):

   — FirstChild: в среднем быстрее в ∼18,4 раза (28 360 нс против 1 543 нс).

   — Parent: в среднем быстрее в ∼7,3 раза (27 782 нс против 3 795 нс).

   — LastChild: в среднем быстрее в ∼5,8 раза (28 152 нс против 4 857 нс).

   — ChildrenCount: в среднем быстрее в ∼3,3 раза (27 737 нс против 8 309 нс).