**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ITMO University**

**Отчет по лабораторной работе № 2-2**

**По дисциплине** Алгоритмы и структуры данных

**Обучающийся** Овсянкин Даниил Витальевич

**Преподаватель**: Ромакина О.М

**Факультет** Инфокоммуникационных технологий

**Группа** К3244

**Направление подготовки** 45.03.04 Интеллектуальные системы в гуманитарной сфере

**Образовательная программа** Интеллектуальные системы в гуманитарной сфере

Санкт-Петербург

2025 г.

Изображение выглядит как текст, Шрифт, чек

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Текстовое объяснение решения:**

В задаче было реализовано три углублённых обхода бинарного дерева без рекурсии. Узлы читаются в массивы ключей и индексов левого/правого ребёнка, где отсутствие ребёнка кодируется значением -1, корень — индекс 0. Для центрированного обхода (in-order) используется классический «спуск по левому краю» со стеком; для прямого (pre-order) — стек с добавлением правого, затем левого ребёнка; для обратного (post-order) — стек состояний с флагом «посещён после детей». Такой подход имеет линейную сложность O(n) и не зависит от глубины дерева, поэтому надёжно работает при n≤105

**Результат на примере**:

Пример №1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Пример№2

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Вывод: Задача решалась итеративными DFS для трёх вариантов обхода; решение линейное по времени и памяти и устойчиво к глубоким, вырожденным деревьям.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Текстовое объяснение решения:**

В задаче было реализовано вычисление баланса каждой вершины двоичного дерева через итеративный пост-обход. Сначала находится корень по массиву parent. Затем стеком выполняется обход «дети -> вершина», для каждой вершины вычисляются высоты левого и правого поддеревьев (пустое - 0) и баланс как их разность hr - hl. Такой подход имеет линейную сложность O(N), не использует глубокую рекурсию.

**Результат на примере:**

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Вывод:

В задаче было реализовано итеративное вычисление баланса узлов через пост-обход: для каждого узла за один линейный проход считаются высоты левого и правого поддеревьев и печатается разность hr − hl. Решение работает за O(N) по времени и O(N) по памяти и устойчиво к глубоким деревьям.

Изображение выглядит как текст, снимок экрана, Шрифт, чек

Содержимое, созданное искусственным интеллектом, может быть неверным.

Код задачи получился очень объемным, так что его можно посмотреть на Githab <https://github.com/ovsyankaboi/Algorithms_2_sem/blob/main/lab2/task17/code/main.py>

**Текстовое объяснение решения:**

В задаче было реализовано динамическое множество на декартовом дереве по ключу, где каждая вершина хранит сумму своего поддерева. Операции добавления, удаления и поиска работают за O(logn) за счёт split/merge. Запрос суммы по диапазону сводится к разности двух префиксных сумм sum\_leq(R) − sum\_leq(L−1). Формат онлайн поддерживается сдвигом аргументов на x - результат прошлой суммы - по модулю M=109+1, в запросе s полученный диапазон упорядочивается по возрастанию.

**Результат на примере**:

Пример №1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Пример №2

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Пример №3

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Вывод: в задаче было реализовано декартово дерево с агрегированием суммы в поддеревьях; все операции выполняются за O(logn), а онлайн сдвиг параметров и модуль корректно учтены при обработке запросов.

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

Код задачи: <https://github.com/ovsyankaboi/Algorithms_2_sem/blob/main/lab2/task18/code/main.py>

**Текстовое объяснение решения:**

В задаче было реализовано Rope на декартовом дереве по неявному ключу — позиции. Узлы хранят **чанки** исходной строки и суммарные размеры поддеревьев. Операция запроса (i,j,k) выполняется двумя split для выделения куска S[i..j], затем дерево без этого куска собирается merge(left, right). После этого по условию индексы считаются в **оставшейся** строке, поэтому делается split по позиции k и вставка: merge(merge(L, mid), R). Получение ответа - симметричный обход, склеивающий чанки. Каждое действие занимает O(logN), а хранение чанков ускоряет реализацию.

**Результат на примере**:

Пример №1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Пример №2

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Вывод:** задача решалась с помощью Rope на имплицитном treap’е: вырезания и вставки подстрок сводятся к последовательностям split/merge, что обеспечивает эффективность и корректность при больших размерах строки и числе запросов.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

Код задачи: <https://github.com/ovsyankaboi/Algorithms_2_sem/blob/main/lab2/task16/code/main.py>

**Текстовое объяснение решения:**

В задаче было реализовано множество на декартовом дереве по ключу: в каждом узле хранится размер поддерева, что позволяет за O(logn) вставлять, удалять и находить элемент заданного ранга. Запрос 0 k трактуется как **k-й максимум при 0-индексации:** это (size−1−k)-й элемент по возрастанию, который извлекается спуском по дереву с учётом размеров левых поддеревьев. Вставка/удаление выполняются через стандартные split/merge

**Результат на примере**:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Вывод:** в задаче было реализовано дерево порядка (treap) с размером поддерева; все операции выполняются за O(logn), k-й максимум выдаётся как элемент нужного ранга, что соответствует формату входа и ограничениям.

**Вывод по лабораторной работе**

Практиковался и научился решать задачи на двоичные деревья поиска и сбалансированные деревья поиска