МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

ОТЧЕТ К ЛАБОРАТОРНОЙ РАБОТЕ

«Реализация и экспериментальное сравнение базовых структур данных на примере телефонного справочника»

Выполнил: студент В. В. Пронин
Преподаватель: Н. С. Морозов

Нижний Новгород
2024

Содержание

1 Введение

Эффективность программных систем во многом определяется выбором способов организации данных в оперативной памяти. Задача разработки телефонного справочника является классическим примером, требующим баланса между скоростью вставки новых записей, поиском по ключу и эффективным удалением.

В рамках данной работы исследуются три фундаментальные структуры данных, реализованные «с нуля» в процедурной парадигме программирования на языке Python:

- **Связный список (Linked List)** — динамическая структура, позволяющая оценить базовые механизмы управления указателями и демонстрирующая линейную сложность операций $O(n)$.
- **Хеш-таблица (Hash Table)** — ассоциативный массив, использующий хеширование для обеспечения прямого доступа к данным. Реализация позволяет изучить методы разрешения коллизий и преимущества константной сложности $O(1)$.
- **Двоичное дерево поиска (BST)** — иерархическая структура, обеспечивающая логарифмическую скорость доступа $O(\log n)$ и поддерживающая упорядоченность данных «из коробки».

Цель работы: Изучить внутренние алгоритмы работы перечисленных структур, реализовать их без использования встроенных высокоуровневых контейнеров и экспериментально подтвердить теоретические оценки временной сложности на случайных и отсортированных наборах данных.

2 Реализация структур данных

2.1 Связный список

2.2 Хеш-таблица

2.3 Двоичное дерево поиска

3 Методика эксперимента

Замеры производились для наборов данных объемом $N = 500$ (и более) элементов. Использовались два сценария: перемешанные (*shuffled*) и отсортированные по алфавиту (*sorted*) записи. Каждая операция выполнялась 5 раз с последующим вычислением среднего арифметического значения с помощью функции `time.perf_counter()`.

4 Результаты и анализ

5 Заключение