Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Языковые процессы интеллектуальных систем»

# Тема: «Хеш-таблицы»

Выполнил:

Студент 4 курса

Группы ИИ-21

Кирилович А. А.

Проверил:

Монтик Н.С.

Брест 2024

**Цель работы**: изучить основные методы организации хеш-таблиц, получить представление о преимуществах и недостатках, присущих различным методам организации таблиц

Вариант 17. Хеш-таблица с цепочками и учетом длины цепочек: таблица строится с использованием хеш-функции, возвращающей код первой буквы идентификатора. При коллизиях используется метод цепочек. Подсчитывается средняя длина цепочек и максимальная длина цепочки.

import 'dart:io';

class Node {

  String key;

  Node? next;

  Node(this.key);

}

class HashTable {

  final int size;

  List<Node?> table;

  int totalChains = 0;

  HashTable(this.size) : table = List<Node?>.filled(size, null);

  int \_hash(String key) {

    return key.isNotEmpty ? (key.codeUnitAt(0) % size) : 0;

  }

  void insert(String key) {

    int index = \_hash(key);

    Node newNode = Node(key);

    if (table[index] == null) {

      table[index] = newNode;

    } else {

      Node? current = table[index];

      while (current?.next != null) {

        current = current?.next;

      }

      current?.next = newNode;

    }

    totalChains++;

  }

  bool search(String key) {

    int index = \_hash(key);

    Node? current = table[index];

    while (current != null) {

      if (current.key == key) {

        return true;

      }

      current = current.next;

    }

    return false;

  }

  bool delete(String key) {

    int index = \_hash(key);

    Node? current = table[index];

    Node? prev;

    if (current == null) return false;

    if (current.key == key) {

      table[index] = current.next;

      totalChains--;

      return true;

    }

    while (current != null && current.key != key) {

      prev = current;

      current = current.next;

    }

    if (current == null) return false;

    prev?.next = current.next;

    totalChains--;

    return true;

  }

  double averageChainLength() {

    int totalLength = 0;

    int nonEmptyChains = 0;

    for (var node in table) {

      if (node != null) {

        nonEmptyChains++;

        int chainLength = 0;

        Node? current = node;

        while (current != null) {

          chainLength++;

          current = current.next;

        }

        totalLength += chainLength;

      }

    }

    return nonEmptyChains > 0 ? totalLength / nonEmptyChains : 0;

  }

  int maxChainLength() {

    int maxLength = 0;

    for (var node in table) {

      if (node != null) {

        int chainLength = 0;

        Node? current = node;

        while (current != null) {

          chainLength++;

          current = current.next;

        }

        maxLength = maxLength > chainLength ? maxLength : chainLength;

      }

    }

    return maxLength;

  }

}

void main() async {

  HashTable hashTable = HashTable(10);

  final file = File('identifiers.txt');

  if (await file.exists()) {

    List<String> lines = await file.readAsLines();

    for (String line in lines) {

      hashTable.insert(line.trim());

    }

    print("Идентификаторы успешно добавлены в хеш-таблицу.");

  } else {

    print("Файл identifiers.txt не найден.");

  }

  print("Поиск ключа 'banana': ${hashTable.search("banana")}");

  print("Средняя длина цепочек: ${hashTable.averageChainLength()}");

  print("Максимальная длина цепочки: ${hashTable.maxChainLength()}");

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Вывод:** в ходе выполнения лабораторной работы изучил основные методы организации хеш-таблиц, получил представление о преимуществах и недостатках, присущих различным методам организации таблиц.