TRABALHO T1

1. Definição de conceitos

Tipos de Dados Abstratos (TDA) são modelos matemáticos que fazem o encapsulamento de um conjunto de elementos. O conjunto SET trata-se de uma estrutura de dados, mas também é um tipo de dado abstrato que armazena apenas coleções de elementos sem repetição, ou seja, os elementos devem ser únicos e não possuem qualquer ordem específica. Nesse caso, podem ser realizadas algumas operações básicas, como de inserção, remoção e verificação da existência de um elemento no conjunto.

Quanto as estruturas de dados, elas são uma coleção de técnicas e algoritmos que permitem a organização e manipulação de dados. Para isso, há diversos exemplos de estruturas que podem ser implementadas, dependendo da função e objetivo do conjunto de dados. Podem ser citados:

- **Array**: Armazena uma coleção de elementos do mesmo tipo;
- **Pilha**: Permite inserção e remoção de dados em ordem LIFO;
- **Fila**: Permite a inserção e remoção de elementos em ordem FIFO;
- **Lista ligada**: Cada elemento tem um ponteiro para o próximo da lista;
- **Tabela Hash**: Armazena pares de chave-valor, sendo cada chave única para mapear um valor correspondente;
- **Árvore**: Estrutura de dados hierárquica que consiste em raiz, nós e nós-filhos;
- **Grafo**: Conjunto de vértices conectados por arestas, com cada aresta possuindo um peso associado.

Seguindo esta lógica, é importante levar em consideração alguns fatores para escolher a estrutura de dados mais adequada para o tipo de dados que está sendo trabalhado. Assim, a definição de suas características e limitações são pontos de análise necessários. Entre os fatores que podem influenciar, estão:

- Tipos de Dados;
- Operações como inserção e remoção de elementos;
- Eficiência;
- Facilidade de Implementação e Manutenção.

2. Implementação do tipo de dado abstrato SET

Para a implementação, a estrutura de dados da Tabela Hash mostrou-se como a mais eficiente para a execução do programa. Isso ocorre porque a Tabela Hash permite acesso, inserção e remoção de elementos em tempo médio constante O(1) em cenários ideais, desde que a função Hash seja eficaz e a própria tabela não esteja muito cheia. Além disso, trata-se de uma boa escolha para coleções de dados devido a sua eficiência para operações básicas.

Tratando-se da complexidade de tempo, pode ser dividida entre as operações que se espera realizar com o conjunto de dados:

- Inserção: O (1) amortizado;
- Exclusão: O (1) amortizado;
- Procura: O (1) amortizado;

Para fins de detalhamento, o significado de O (1) amortizado indica que o tempo de execução é constante, ou seja, não depende do tamanho da entrada. E amortizado significa que a sequência de operações está sendo analisada como um todo, ao invés de cada operação individualmente.

Por fim, a complexidade de espaço armazenado é definida por O(n) no pior caso, sendo 'n' o número de dados do conjunto. Isso significa que a quantidade de memória necessária para executar o algoritmo aumenta conforme o tamanho da entrada também aumenta, o que pode ser uma desvantagem se 'n' for grande. No entanto, se 'n' é pequeno ou moderado, então a complexidade de espaço O(n) pode não ser um problema. Esse fator depende muito do contexto específico ao qual o algoritmo está sendo utilizado.

3. Algoritmo

implement-set.java

```java
import java.util.HashSet;

public class SetImplementation {
    private HashSet<Integer> set;

    public SetImplementation() {
        set = new HashSet<>();
    }

    public void add(int element) {
        set.add(element);
    }

    public void remove(int element) {
        set.remove(element);
    }

    public boolean contains(int element) {
        return set.contains(element);
    }

    public static void main(String[] args) {
        SetImplementation setImpl = new SetImplementation();
        setImpl.add(5);
        setImpl.add(10);
        setImpl.add(5);

        System.out.println("Contains 5: " + setImpl.contains(5));  // true
        System.out.println("Contains 15: " + setImpl.contains(15));  // false

        setImpl.remove(10);
        System.out.println("Contains 10 after removal: " + setImpl.contains(10));  // false
    }
}
```

Figura 1. Exemplo da implementação de código em JAVA, do tipo de dado abstrato SET com base na Tabela Hash.

Explicação detalhada do algoritmo:

- **Importação de bibliotecas**: Importa a classe HashSet da biblioteca padrão do Java, que é uma implementação de conjunto baseada em uma tabela de hash.

  'import java.util.Hashset'

- **Classe SetImplemention**: Declaração da classe SetImplementation que contém um campo set do tipo HashSet que armazenará os elementos do conjunto.

  'public class SetImplemention {
  private HashSet<Integer> set }'

- **Construtor**: Construtor da classe que inicializa o conjunto set como um novo HashSet.

  'public SetImplementation() {
  set = new HashSet<>();}'

- **Métodos**:
  add(int element): adiciona um elemento ao conjunto;
  remove(int element): remove um elemento do conjunto;
  contains(int element): verifica se o conjunto contém um determinado elemento.

- **Método Main (Ponto de Entrada)**:
  Instancia um objeto da classe SetImplementation.
  Adiciona elementos (5 e 10) ao conjunto.
  Verifica se o conjunto contém os elementos 5 e 15.
  Remove o elemento 10 do conjunto.
  Verifica novamente se o conjunto contém o elemento 10 após a remoção e imprime os resultados na saída padrão.

```
public static void main(String[] args) {
SetImplementation setImpl = new SetImplementation();
setImpl.add(5);
setImpl.add(10);
setImpl.add(5);

 System.out.println("Contains 5: " + setImpl.contains(5));
 System.out.println("Contains 15: " + setImpl.contains(15));

 setImpl.remove(10);
 System.out.println("Contains 10 after removal: " + setImpl.contains(10));}
```

Nomes: Geovanna Cristine Corrêa Ribeiro (202010137), Luana Roza de Oliveira (202010258)

Curso: Ciência da Computação
Data: 02 de outubro de 2023