Curso Estruturas de Dados e Algoritmos Expert

Prof. Nelio Alves

Conjuntos, dicionários

DEVSUPERIOR

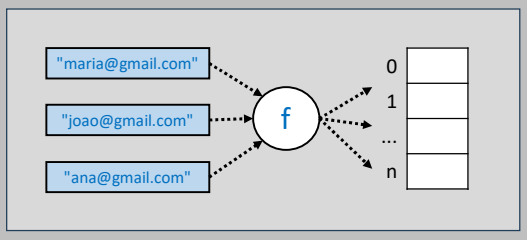
https://devsuperior.com.br

1

Visão geral

Neste módulo vamos conhecer os conjuntos e dicionários, que são estruturas de dados onde o armazenamento e recuperação dos dados é tipicamente não ordenado e baseado em tabelas hash.

O acesso aos elementos é feito por meio de uma chave, e é extremamente eficiente, com complexidade constante O(1).



2

Conjuntos

Curso Estruturas de Dados e Algoritmos Expert

Conjunto

Um conjunto é uma coleção não ordenada de elementos distintos.

Não ordenado:

- Não há uma sequência entre os elementos (primeiro, segundo, etc.).
- Não tem como acessar um elemento por uma posição, diferentemente de arrays e listas.

Elementos distintos:

- Conjunto não admite repetição de elementos, diferentemente de arrays e listas.
- Cada elemento de um conjunto precisa ter um critério de identificação, para comparar se um objeto é igual a outro.
- Se tentar adicionar um elemento repetido a um conjunto, nada acontece.

Aplicações comuns de conjuntos

Problemas que envolvem eliminação de objetos duplicados.

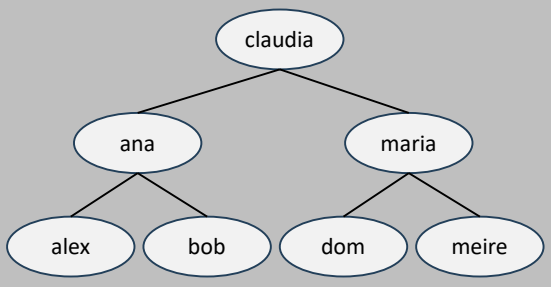
Problemas que envolvem operações de conjuntos (união, interseção, diferença).

Problemas matemáticos/algébricos.

Principais implementações

TreeSet: mais lento, porém ordenado e com velocidade mais previsível. Implementação interna com árvore binária de busca balanceada (AVL) ou árvore rubro-negra.

Complexidade de acesso, inserção e remoção:
O(log n)

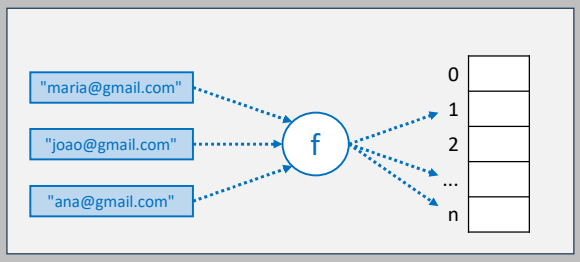


Principais implementações

HashSet: mais rápido, não ordenado.
Implementação interna com **tabela hash**.

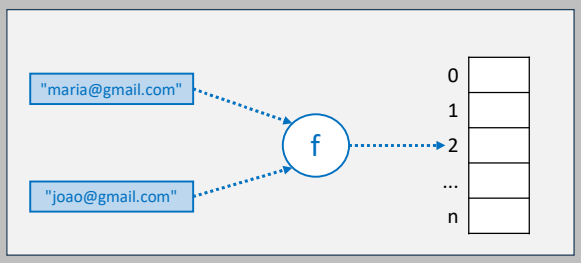
Complexidade de acesso, inserção e remoção:

Idealmente O(1), ou seja, constante. No entanto, dependendo da qualidade da função hash e do tratamento de colisões, essas operações podem se degradar para O(n) em casos extremos.



7

Colisão: ocorre quando a função hash gera o mesmo código hash para duas chaves diferentes.



Tratamento de colisões: método adotado para tratar objetos colididos. Dentre os mais utilizados está o "chaining", que utiliza uma estrutura de dados auxiliar (lista ou árvore) para armazenar os objetos colididos.

8

Operações de um conjunto

Operação	Efeito
add(T item)	Adicionar um elemento ao conjunto.
boolean remove(T item)	Remove o elemento do conjunto. Retorna true se o elemento existia.
boolean contains(T item)	Testa se um elemento está contido no conjunto.
boolean isEmpty()	Testa se o conjunto está vazio.
int size()	Retorna o número de elementos do conjunto.
union(Set<T> other)	Operação união (adiciona todos elementos do outro conjunto).
intersection(Set<T> other)	Operação intersecção (remove todos elementos que não existem no outro conjunto).
difference(Set<T> other)	Operação diferença (remove todos elementos que existem no outro conjunto).

Dicionários

Curso Estruturas de Dados e Algoritmos Expert

Dicionário

Também chamado de **mapa (map)** .

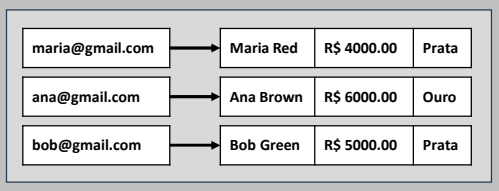
É uma coleção não ordenada de pares chave-valor onde cada chave é única.

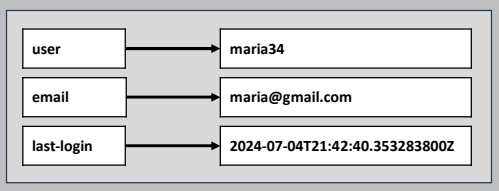
Não ordenado:

- Não há uma sequência entre os elementos (primeiro, segundo, etc.).
- Não tem como acessar um elemento por uma posição, diferentemente de arrays e listas. O acesso é feito pela chave.

Chave única:

- Não é permitido que mais de um elemento possua a mesma chave no dicionário.





Aplicações comuns de dicionários

Contagem de frequência: número de ocorrências de itens em uma coleção, de palavras em um texto, etc.

Caching e memoização: resultados de operações caras são armazenados para evitar repetições desnecessárias.

Indexação: Em sistemas de banco de dados ou em qualquer aplicação onde dados precisam ser recuperados rapidamente.

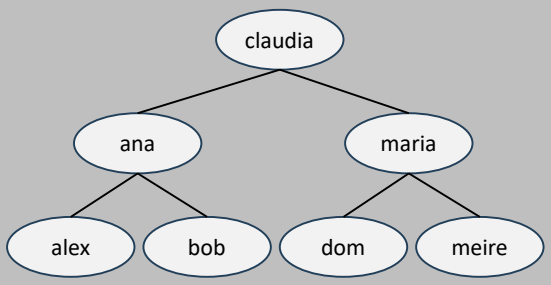
Qualquer mecanismo de armazenamento que faça mapeamento entre uma chave e seu valor correspondente: cookies, local storage web, carrinhos de compras, compiladores, etc.

Principais implementações

Árvores: mais lento, porém ordenado e com velocidade mais previsível. Implementação interna com árvore binária de busca balanceada, tais como árvores AVL e árvore rubro-negra.

Complexidade de acesso, inserção e remoção:

O(log n)

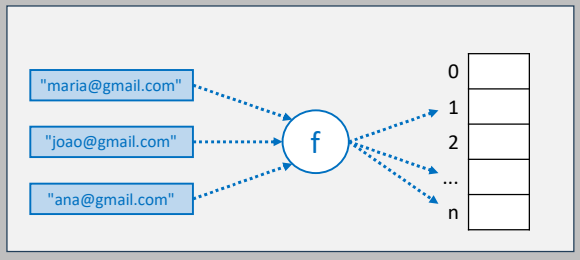


Principais implementações

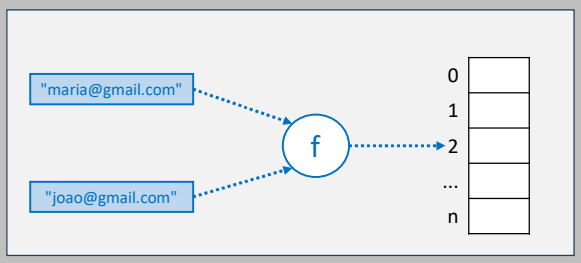
HashMap: mais rápido, não ordenado. Implementação interna com **tabela hash**.

Complexidade de acesso, inserção e remoção:

Idealmente O(1), ou seja, constante. No entanto, dependendo da qualidade da função hash e do tratamento de colisões, essas operações podem se degradar para O(n) em casos extremos.



Colisão: ocorre quando a função hash gera o mesmo código hash para duas chaves diferentes.



Tratamento de colisões: método adotado para tratar objetos colididos. Dentre os mais utilizados está o "chaining", que utiliza uma estrutura de dados auxiliar (lista ou árvore) para armazenar os objetos colididos.

15

Operações de um dicionário

Operação	Efeito
put(K key, V value)	Adicionar o par chave-valor ao dicionário (substitui o valor se a chave já existir).
V remove(K key)	Remove o elemento do dicionário, retornando-o.
V get(K key)	Obtém o valor associado à chave.
boolean contains(K key)	Testa se o dicionário contém um elemento.
boolean isEmpty()	Testa se o dicionário está vazio.
int size()	Retorna o número de elementos do dicionário.
Collection<K> keys()	Retorna uma coleção contendo as chaves do dicionário.
Collection<V> values()	Retorna uma coleção contendo os valores do dicionário.

16