Mapas

Dicionários e tabelas

Prof. Marcelo de Souza

45EST – Algoritmos e Estruturas de Dados
Universidade do Estado de Santa Catarina

UDESC

Material de apoio

Leitura principal:

- Capítulo 10 de Goodrich et al. (2014)[1] – Mapas, tabelas hash e skip lists.

Leitura complementar:

- Capítulo 3 (3.1) de Sedgewick e Wayne (2011)[2] – Tabelas de símbolos.
- Capítulo 8 de Preiss (2001)[3] – Dispersão, tabelas de dispersão e tabelas de espalhamento.

[1] Michael T Goodrich et al. (2014). *Data structures and algorithms in Java*. 6ª ed. John Wiley & Sons.

[2] Robert Sedgewick e Kevin Wayne (2011). *Algorithms*. Addison-Wesley Professional.

[3] Bruno R Preiss (2001). *Estruturas de dados e algoritmos: padrões de projetos orientados a objetos com Java*. Campus.

Mapas

Conceitos básicos

Um **mapa** armazena entradas compostas por uma **chave** e um **valor**.

- A chave serve para buscar o registro.
- O valor armazena o registro associado à chave.

Mapas

Conceitos básicos

Um **mapa** armazena entradas compostas por uma **chave** e um **valor**.

- A chave serve para buscar o registro.
- O valor armazena o registro associado à chave.

Também chamado de **dicionário**, **tabela** ou **array associativo**, o mapa organiza e acessa as entradas pelas suas **chaves**, em vez das suas posições, i.e. a chave é o índice da estrutura.

Mapas

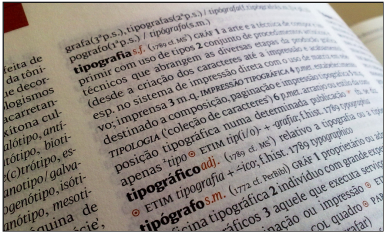
Conceitos básicos

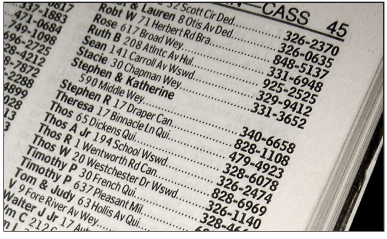
Um **mapa** armazena entradas compostas por uma **chave** e um **valor**.

- A chave serve para buscar o registro.
- O valor armazena o registro associado à chave.

Também chamado de **dicionário**, **tabela** ou **array associativo**, o mapa organiza e acessa as entradas pelas suas **chaves**, em vez das suas posições, i.e. a chave é o índice da estrutura.

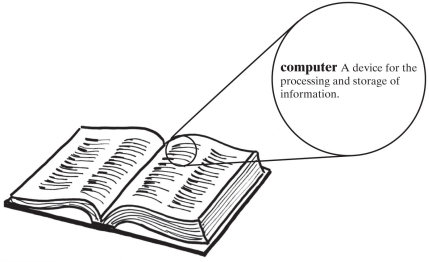
Exemplos no cotidiano: dicionários, listas telefônicas, cardápios, índices remissivos, ...





Características:

- Na maioria das implementações, **a chave é única**.
  - Ao inserir uma entrada com chave existente, o valor atual é substituído pelo novo valor.
- A chave pode ser um objeto de **qualquer classe**.
  - Necessário garantir a comparação de chaves.



Mapas

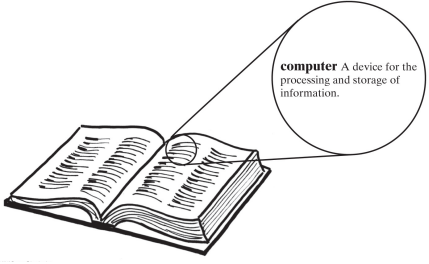
Conceitos básicos

Características:

- Na maioria das implementações, **a chave é única**.
  - Ao inserir uma entrada com chave existente, o valor atual é substituído pelo novo valor.
- A chave pode ser um objeto de **qualquer classe**.
  - Necessário garantir a comparação de chaves.

Operações:

- `get(k)`: retorna o valor associado à chave `k`.
- `put(k, v)`: insere uma entrada com chave `k` e valor v.
- `remove(k)`: remove a entrada com chave `k`.



Mapas

Análise de complexidade (diferentes implementações)

Podemos implementar um mapa **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

Mapas

Análise de complexidade (diferentes implementações)

Podemos implementar um mapa **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

- Qual a melhor escolha? ... vamos **analisar primeiro**, e escolher uma opção!

Mapas

Análise de complexidade (diferentes implementações)

Podemos implementar um mapa **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

- Qual a melhor escolha? ... vamos **analisar primeiro**, e escolher uma opção!

Operação	**Arranjo**		**Encadeamento**	
	Não ordenado	**Ordenado**	**Não ordenado**	**Ordenado**
`get`	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
`put`	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
`remove`	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

— a inserção (`put`) em um arranjo ordenado é feita em $\mathcal{O}(\log n)$ na substituição.

Mapas

Análise de complexidade (diferentes implementações)

Podemos implementar um mapa **arranjos** ou **encadeamento**. E ainda podemos manter a estrutura **não ordenada** ou **ordenada**.

- Qual a melhor escolha? ... vamos **analisar primeiro**, e escolher uma opção!

Operação	**Arranjo**		**Encadeamento**	
	Não ordenado	**Ordenado**	**Não ordenado**	**Ordenado**
`get`	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
`put`	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
`remove`	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

— a inserção (`put`) em um arranjo ordenado é feita em $\mathcal{O}(\log n)$ na substituição.

Conclusão: usaremos implementações baseadas em **arranjo** (ordenado e não ordenado).

45EST – Algoritmos e Estruturas de Dados
Prof. Marcelo de Souza