Estruturas de dados

João Paulo Dias de Almeida

Universidade Federal de Sergipe

O que vamos aprender hoje?

- Relembrar o que é uma estrutura de dado
- Entender as vantagens do uso de Arrays
- Compreender o uso da Lista encadeada
- Descrever o funcionamento da fila e da pilha
- Entender o comportamento do dicionário

Introdução

- Coleções de dados são fundamentais para programas de computador
  - Essas coleções são dinâmicas: aumentam, diminuem, ou são modificadas
- Em uma implementação típica de uma coleção, cada objeto pode ser manipulado como se houvesse um ponteiro para o objeto
  - Algumas coleções utilizam chaves únicas para identificar o objeto

Alocação dinâmica de memória

- Estruturas dinâmicas requerem alocação dinâmica de memória
  - Programa solicita mais memória durante a execução
- Em Java, a alocação de memória é feita no momento da instanciação do objeto
  - OutOfMemoryError caso não haja memória suficiente
- Não é necessário desalocar a memória alocada
  - Java realiza coleta de lixo automática de objetos que não são mais referenciados em um programa

Operações

- Lista de operações típicas executadas por coleções de dados:
  - Search (busca)
  - Insert (inserir)
  - Delete (deletar)
  - Minimun (menor objeto)
  - Maximum (maior objeto)
  - Sucessor (após determinado objeto)
  - Predecessor (anterior a determinado objeto)

Estruturas de dados

- Vamos conhecer algumas estruturas de dados básicas utilizadas por programas para armazenar coleções de dados
  - Arrays
  - Lista encadeada
  - Pilhas e filas
  - Árvore binária
  - Tabela hash

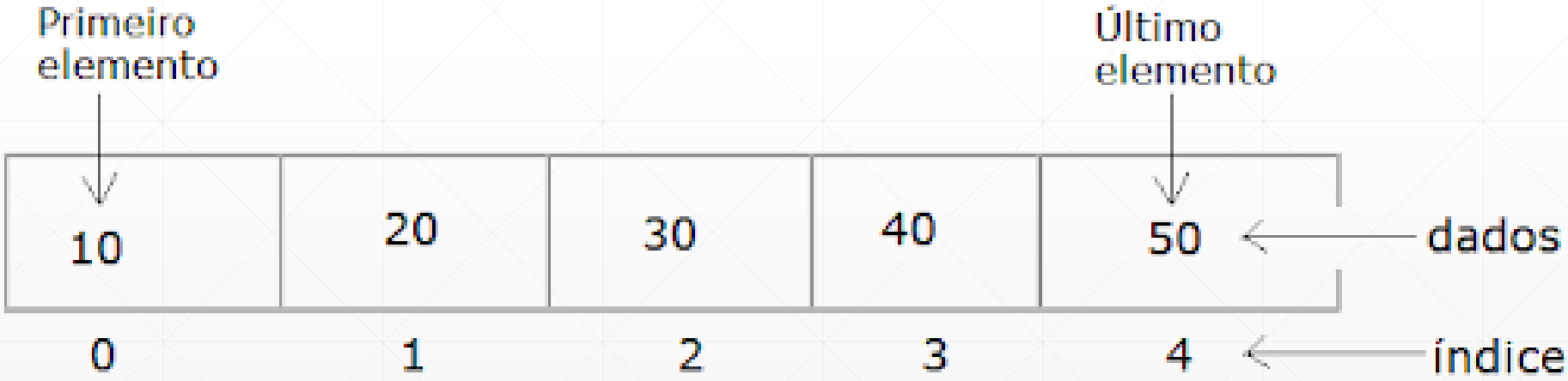
Dados contíguos vs Dados ligados

- Estruturas de dados podem ser classificadas como contíguas ou ligadas
  - Contíguas → baseadas em arrays
  - Ligadas → baseadas em ponteiros/referências
- Estruturas alocadas de forma contígua são compostas por blocos de memórias únicos.
  - Arrays e matrizes
  - Pilha
  - Tabela Hash

Arrays

Arrays: vetores e matrizes

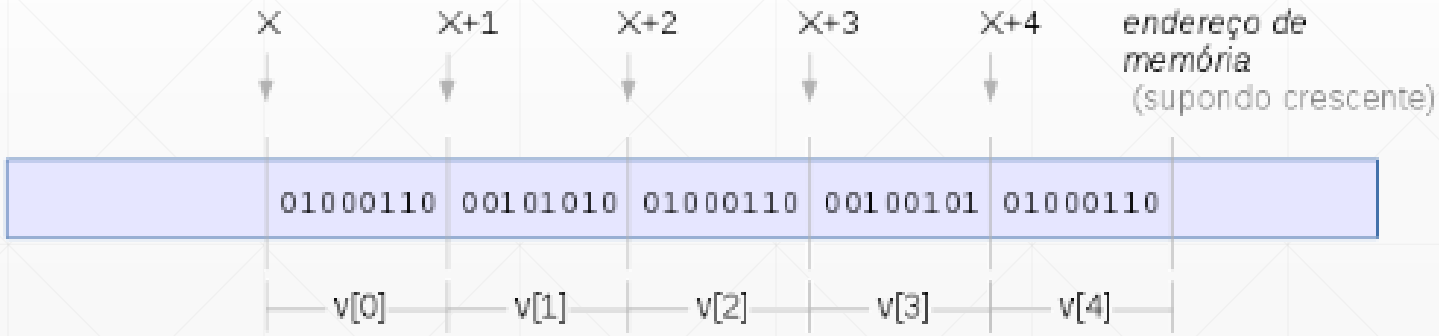
- Arrays são estruturas de tamanho **fixo**
  - Cada elemento pode ser acessado eficientemente através de um índice (endereço)



Arrays: vantagens

- Vantagens de arrays alocados de forma contígua:
  - **Acesso aos dados em O(1):**

    Como o índice de cada elemento direciona diretamente ao endereço de memória, podemos acessar qualquer informação instantaneamente.



Arrays: vantagens

- **Eficiência no armazenamento:**

  Nenhum espaço é dedicado para representar links ou outra informação de armazenamento. Nem mesmo um fim-de-arquivo é necessário

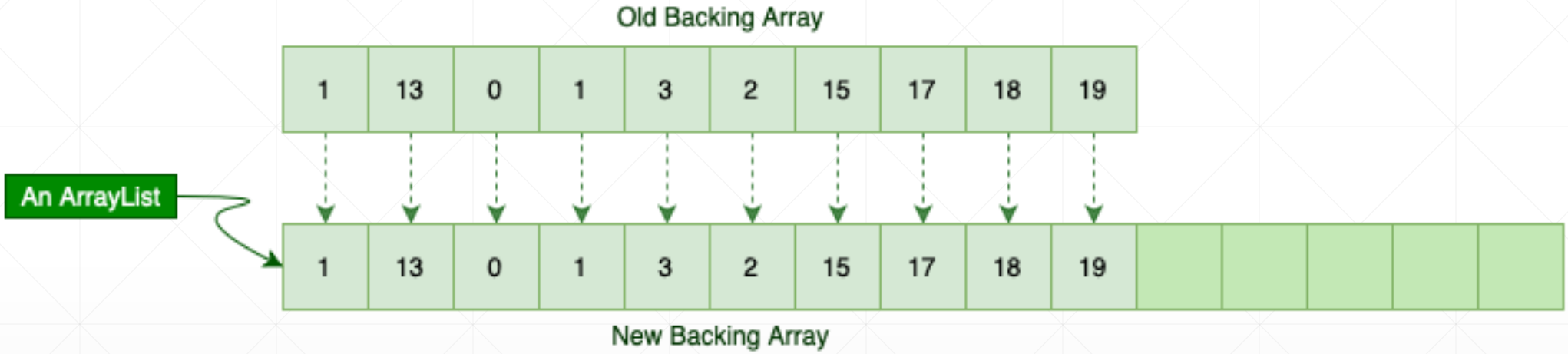
- **Localização na memória:**

  É comum um programa percorrer todos os elementos de um conjunto. Arrays permitem explorar a alta velocidade do cache pois os dados estão armazenados fisicamente um ao lado do outro

Arrays dinâmicos

- É possível utilizar Arrays dinâmicos (e.g. ArrayList) para resolver a limitação do tamanho fixo
  - Existe um custo para aumentar o tamanho do vetor: O(n)
  - Como existe um custo, um abordagem interessante seria dobrar o tamanho do vetor sempre que necessário
- Nos casos em que o vetor precisa aumentar de tamanho, a inserção de valores não será feita em O(1)

ArrayList: duplicação de tamanho




```
int possibleUpperBound = 10_000;
List<String> items = new ArrayList<>(possibleUpperBound);
```

ArrayList: Removendo item

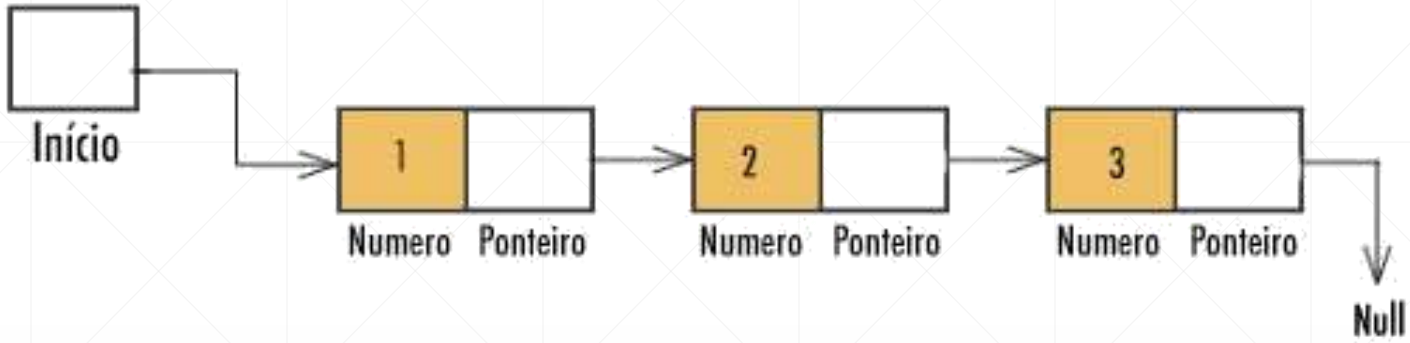
Lista encadeada

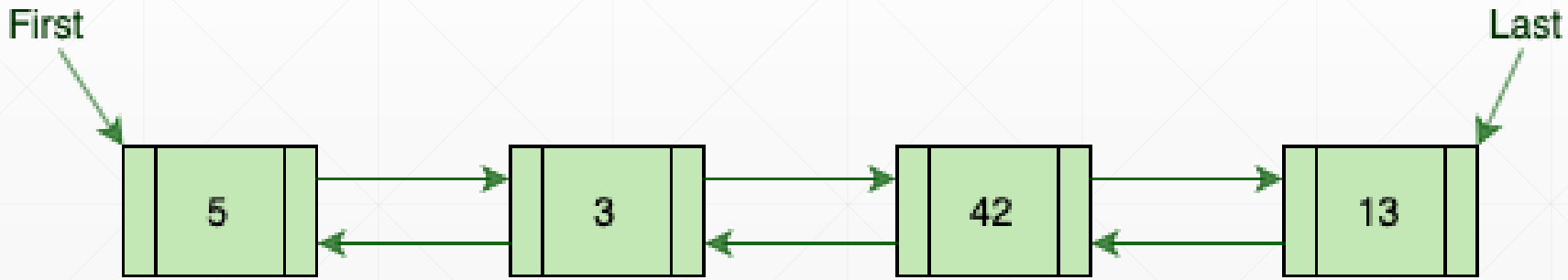
Lista encadeada

- Lista encadeada é uma coleção linear (em sequência) de objetos
  - Os objetos são conectados por links (ponteiros), por isto o nome
- Geralmente, um programa acessa uma lista encadeada por meio de uma referência ao primeiro nó
  - Cada nó subsequente é acessado por uma referência de link armazenado no nó anterior
  - O último nó da lista é definida como null para indicar o "fim da lista"

Lista encadeada

É necessário percorrer a lista para encontrar o item desejado





Listas

- **A lista é a estrutura ligada mais simples:**
  - Cada nó da lista contém um ou mais campos para conter os dados que desejamos armazenar
  - Cada nó contém um campo ponteiro para pelo menos um outro nó
    - Dessa forma, parte do espaço na memória é reservado para essa informação
  - É necessário ter um ponteiro para o inicio da lista

Exemplo – lista encadeada

Lista encadeada: operações

- **Busca:**

  Pode ser feita de forma iterativa ou recursiva

- **Inserção:**

  É feita atualizando os links de cada nó

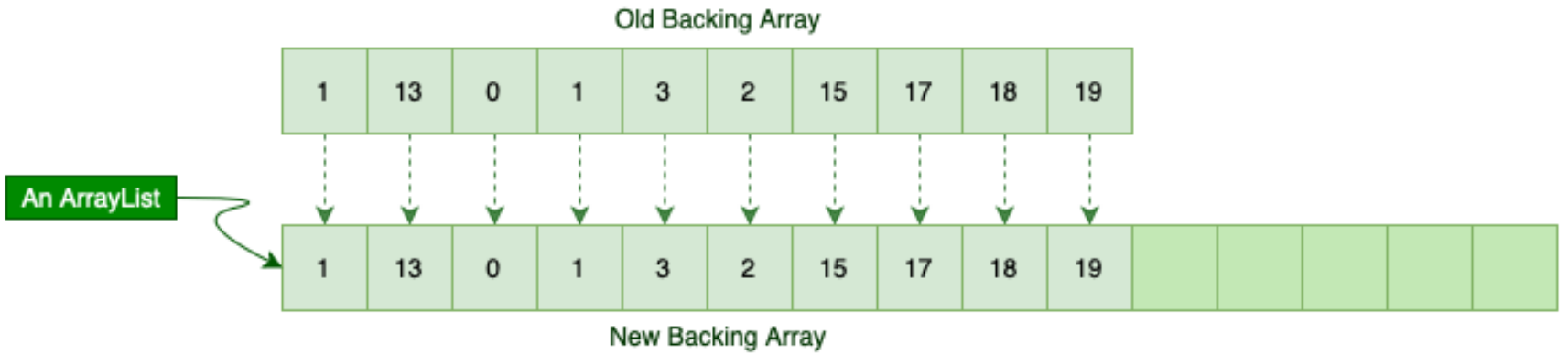
- **Remoção:**

  É necessário encontrar o nó anterior ao que se deseja remover e atualizar o link dele. No caso da remoção do nó inicial, é necessário atualizar a referência do inicio.

Lista encadeada vs ArrayList

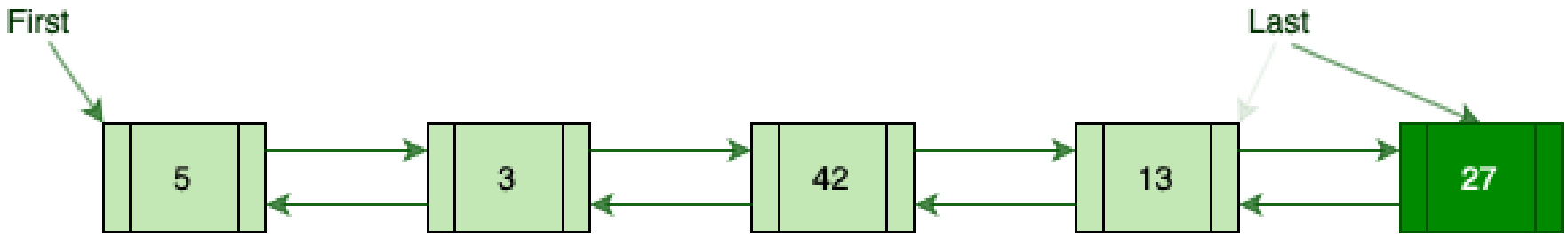
- ArrayList usa um Array para implementar uma lista
  - Arrays possuem tamanho fixo
  - Quando esse tamanho máximo é alcançado, um novo vetor precisa ser criado e todos os dados são copiados para o novo Array
    - Procedimento pode se tornar custoso a depender da quantidade de dados envolvidos na coleção

Lista encadeada vs ArrayList



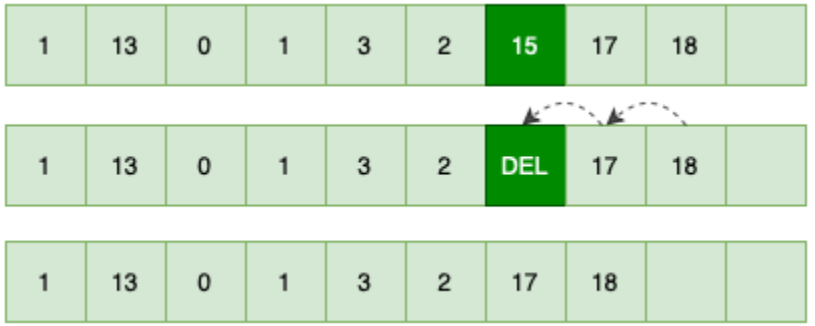

```
int possibleUpperBound = 10_000;
List<String> items = new ArrayList<>(possibleUpperBound);
```

Lista encadeada: inserção



ArrayList: Removendo item

Na lista encadeada, é necessário encontrar o item e depois atualizar os links



Lista: características

- Estouro de memória só ocorre se a memória inteira do computador estiver cheia
- Inserções e remoções são mais simples de fazer do que em arrays dinâmicos
  - Em grandes conjuntos de dados, é melhor atualizar uma referência ao nó do que movimentar todos os dados para direita ou para a esquerda

Lista: características

- Estruturas ligadas necessitam de espaço extra para armazenar os links (ponteiros)
- Listas encadeadas não permitem acesso aleatório aos seus itens
- Arrays fornecem melhor otimização e performance do cache no armazenamento dos dados na memória quando comparando aos saltos necessários nas estruturas ligadas

Quando usar?

- **Listas encadeadas** são mais adequadas quando a **inserção** de elementos na coleção é alta
  - Mais flexível em relação ao uso de memória
- **ArrayLists** são adequadas quando a **leitura** é a operação mais usada
  - Exceto quando interessado nos elementos da ponta da lista (cabeça e cauda)
- ArrayList também é adequado quando existe uma estimativa de tamanho máximo ou quando o tamanho da coleção não varia com frequência
  - Memória alocada constantemente

Fila

Fila

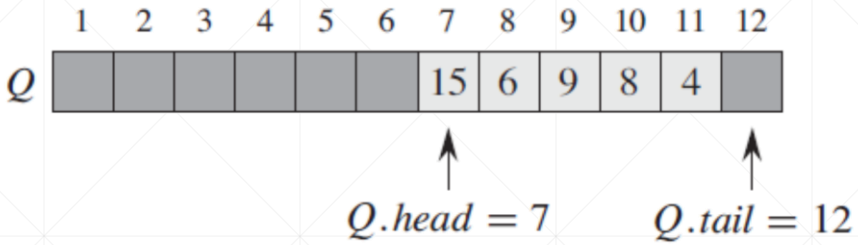
- A fila é uma lista encadeada que segue regras
- A fila segue a lógica do primeiro a chegar, é o primeiro a sair
  - FIFO (first-in, first-out)
  - Inserção e remoção também são conhecidas como enfileirar (**enqueue**) e desenfileirar (**dequeue**)
  - São geralmente utilizadas para gerenciar filas de impressão
    - Pacotes de rede também usam a lógica de fila

Fila - FIFO

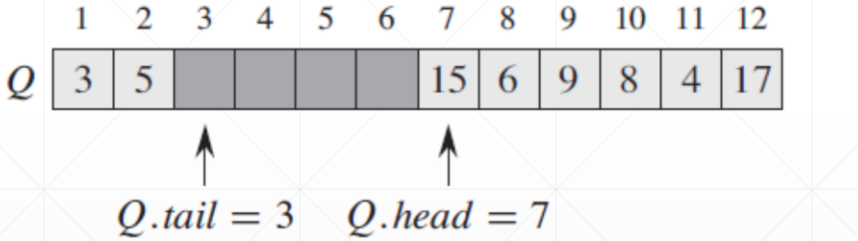
Fila

- Pode ser implementada utilizando lista ou array
- É necessário uma referência para a cabeça e para a cauda
  - Os elementos são inseridos na cauda (fim)
  - Os elementos são retirados da cabeça (inicio)

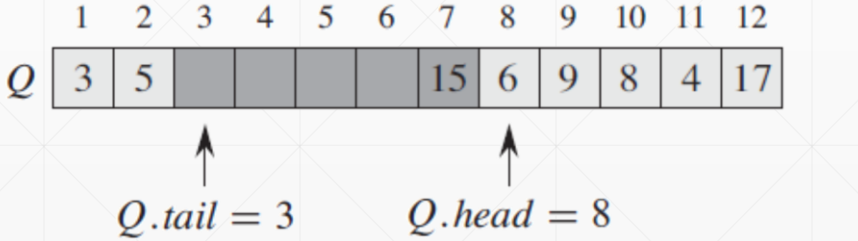
Fila



Queue(17)
Queue(3)
Queue(5)



Dequeue()



Fila baseada em Lista encadeada

Operations	Average case	Worst case
enqueue	O(1)	O(1)
dequeue	O(1)	O(1)

Exemplo - Fila

Fila de prioridade

Fila de prioridade

- Muitas aplicações necessitam que os itens sejam processados a partir de uma ordem específica
  - Exemplo: Definir ordem de execução de tarefas em um projeto
- Filas de prioridade permitem inserir itens na estrutura e organizá-los de acordo com sua prioridade
  - É mais eficiente inserir um item na fila de prioridade do que re-ordenar todos os itens de um conjunto

Fila de prioridade

- Uma fila de prioridade básica suporte três operações básicas:
  - Inserir(): Insere um item x com a chave k
  - EncontraMínimo() e EncontraMáximo(): retorna o item com menor, e maior, prioridade
  - RemoveMínimo() e RemoveMáximo(): remove o item com menor, e maior, prioridade

Exemplo – Fila de prioridade

Fila de prioridade: implementação

- Pode-se utilizar vetores (ordenados ou desordenados) para armazenar os itens
- Ao implementar uma fila de prioridade, é importante registrar a posição do elemento com maior prioridade
  - Assim, a remoção é feita em O(1)
- Após remover um item é necessário buscar pelo item com maior prioridade
  - Atualize sua referência

Fila de prioridade: implementação

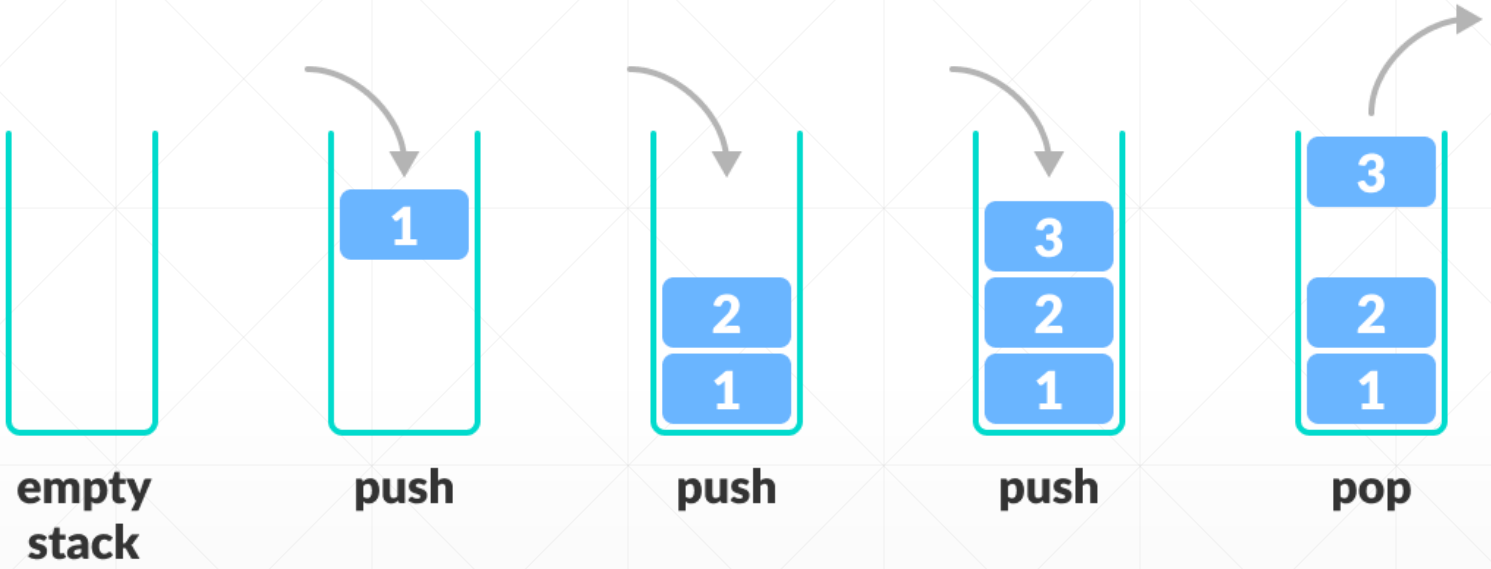
	Unsorted array	Sorted array
Insert(Q, x)	$O(1)$	$O(n)$
Find-Minimum(Q)	$O(1)$	$O(1)$
Delete-Minimum(Q)	$O(n)$	$O(1)$

Pilha

Pilha

- Assim como a fila, a pilha também é uma versão limitada da lista encadeada
- A pilha segue a lógica do último a chegar, é o primeiro a sair
  - LIFO (last-in, first-out)
- Operações principais da pilha se chamam push (inserir) e pop (remover)
- Usada no mecanismo de desfazer dos editores de texto ou voltar do navegador Web

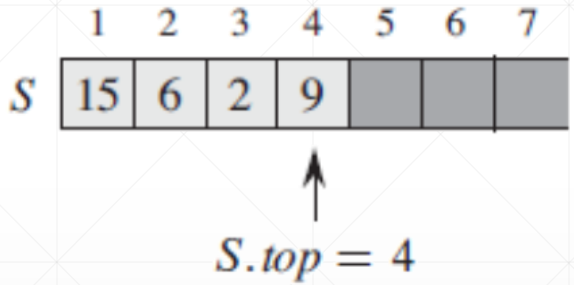
Pilha - Exemplo

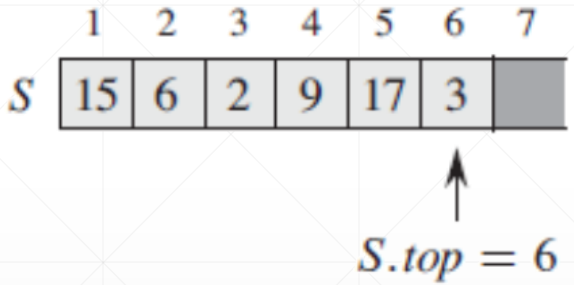


Pilha: implementação

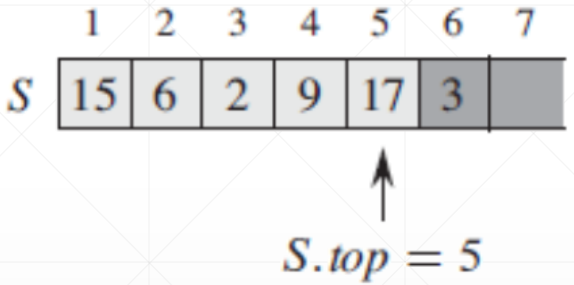
- A pilha utiliza um atributo para identificar o índice para o último elemento adicionado
  - Quando este atributo é igual a zero, a pilha está vazia
- Underflow → é o nome dado quando realizamos a operação pop() em uma pilha vazia
- Overflow → é quando realizamos a inserção em uma pilha que excede o seu tamanho (implementação usando array)

Pilha





Push(17)
Push(3)



Pop()

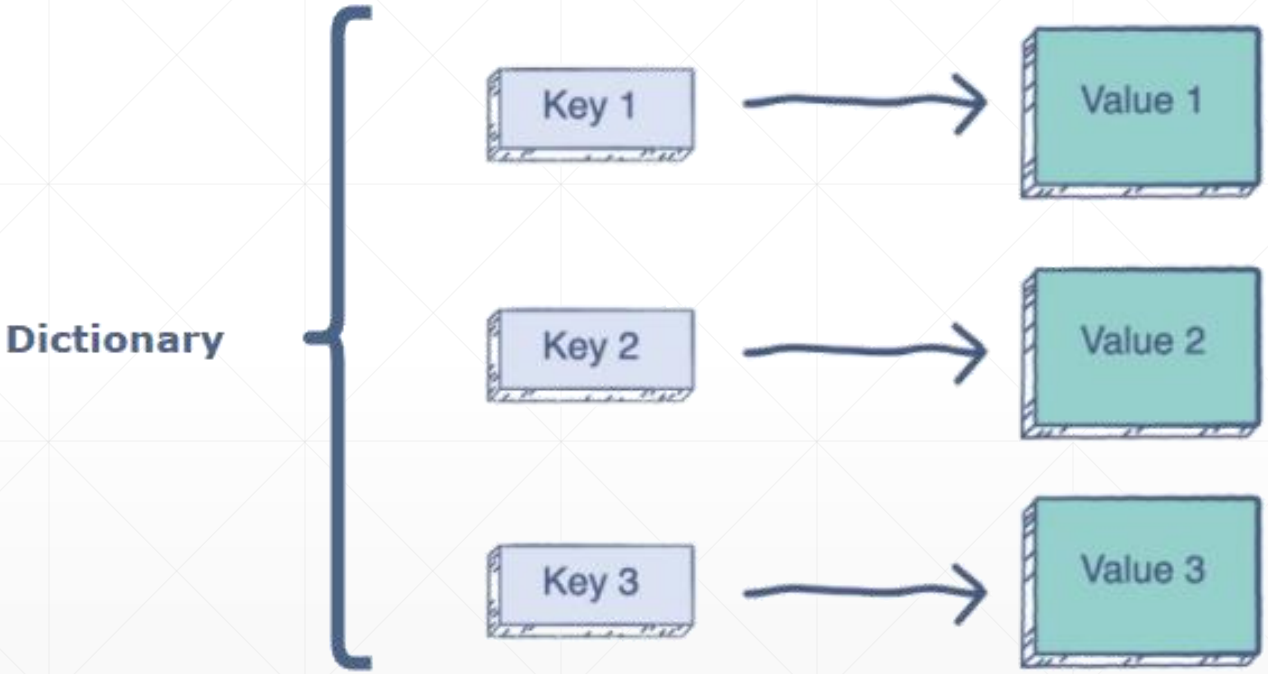
Exemplo – Pilha

Dicionários

Dicionário

- Permite acessar os dados a partir de seu conteúdo
  - É feito um mapeamento entre chave e valor
- Operações principais de um dicionário:
  - Busca()
  - Inserção()
  - Remoção()
  - Max() ou Min()
  - Predecessor ou Sucessor()

Dicionário



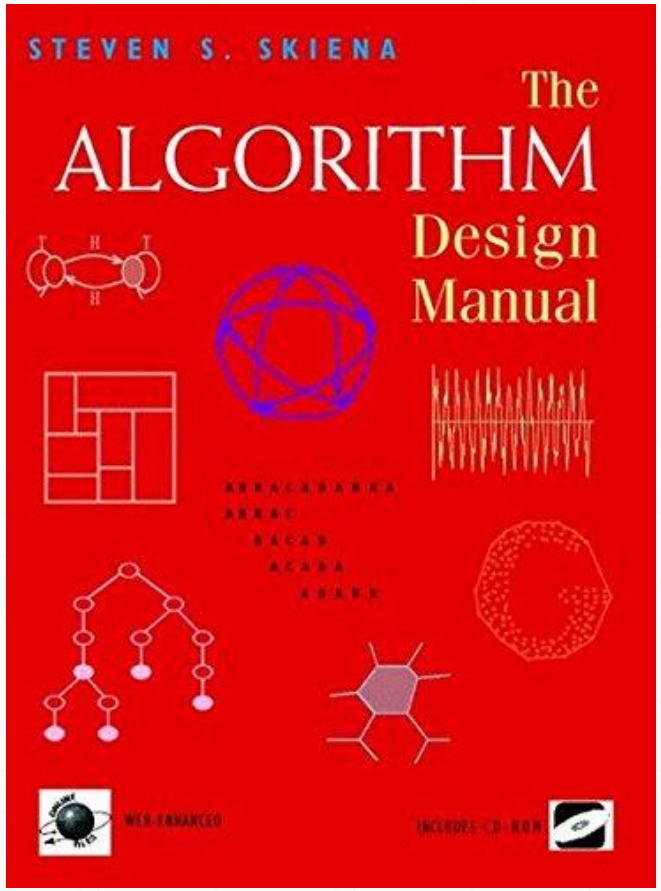
Dicionário: implementação

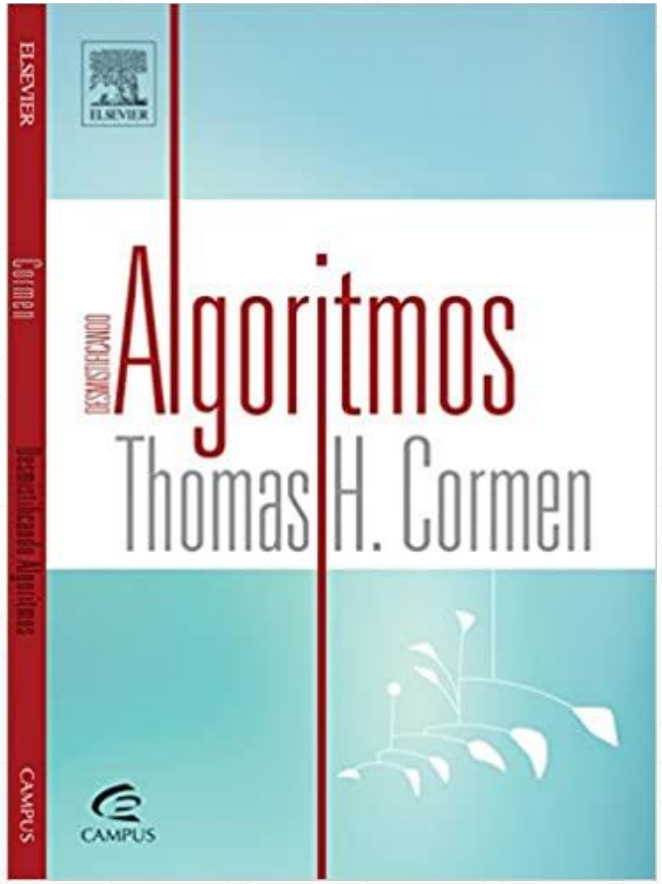
- Em Java a interface Map descreve as operações de um dicionário
- Podemos usar estruturas como HashTable para implementar o dicionário
  - Em Java, HashTable ou HashMap podem ser usados
  - Implementações utilizando árvores, arrays, e listas também são possíveis

Resumo

- Estruturas de dados dinâmicas podem crescer e encolher em tempo de execução
- O limite para alocação dinâmica de memória pode ser tão grande quanto a memória física disponível
- A lista encadeada é uma coleção de itens de dados vinculados
- Pilha e fila são versões limitadas da lista encadeada

Referências





Dúvidas?