Aula 10

Estruturas de Dados

Listas Ligadas

Programação II, 2014-2015

v0.8, 13-06-2015

DETI, Universidade de Aveiro

10.1

Objectivos:
- Estrutura de dados lista ligada;
- Implementar pilhas e filas com listas ligadas como representação interna.

Conteúdo

10.2

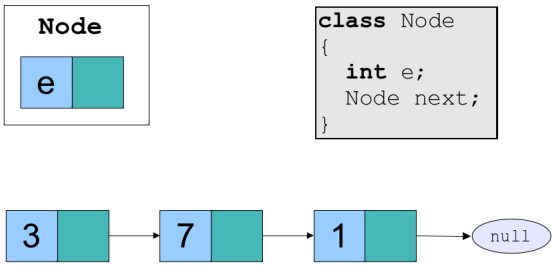
1 Lista Ligada

Lista Ligada

- Estrutura de dados sequencial em que cada elemento da lista contém uma referência para o próximo elemento.
  - Essa referência terá o valor null caso esse elemento não exista.
- É uma estrutura de dados **recursiva** (dado que contém uma referência para si própria).
- Ao contrário do array, é completamente dinâmica.
  - No entanto, obriga a um acesso sequencial.
- Requer a criação de um tipo Nó para cada elemento.

10.3

Lista Ligada: Exemplo



10.4

2 Pilha: Representação Interna com Lista Ligada

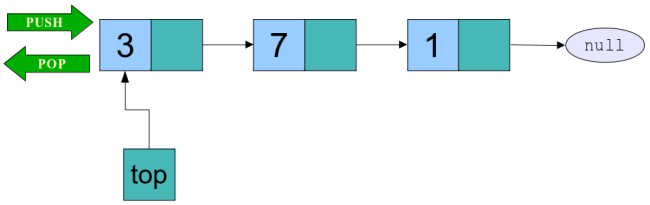
Pilha: módulo

- Nome do módulo:
  - `Stack`
- Serviços:
  - `push`: insere um elemento no topo da pilha
  - `pop`: retira o elemento no topo da pilha
  - `top`: retorna o elemento no topo da pilha
  - `isEmpty`: verifica se a pilha está vazia
  - `isFull`: verifica se a pilha está cheia
  - `size`: retorna a dimensão actual da pilha
  - `clear`: limpa a pilha (retira todos os elementos)

10.5

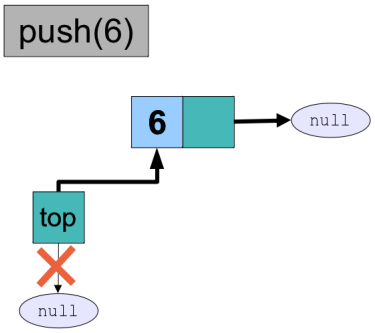
Pilha: Implementação com Lista Ligada

- Os novos elementos (i.e. dados) são inseridos e retirados pelo topo (`top`)
  - LIFO vazio: `top=null`



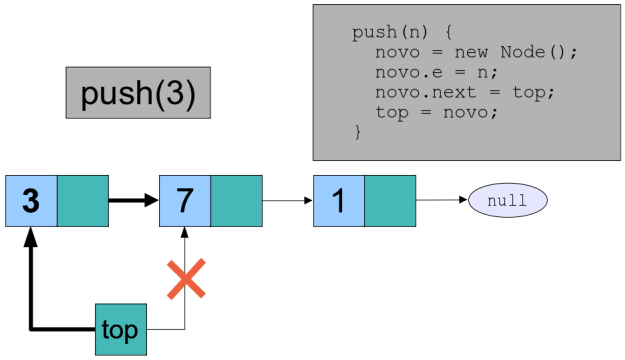
10.6

- Inserção do primeiro elemento



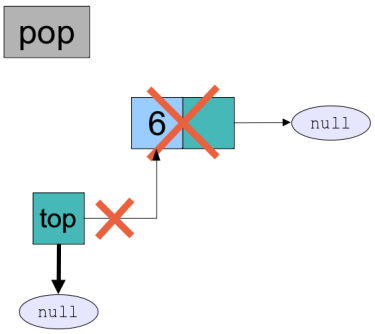
10.7

- Inserção de outros elementos



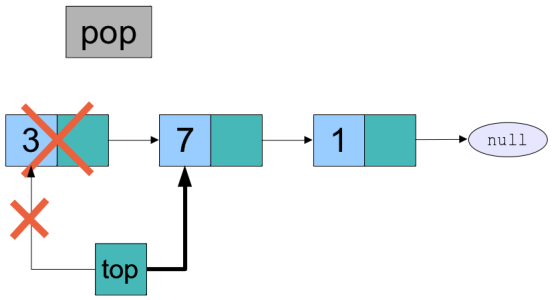
10.8

- Remoção do último elemento



10.9

- Remoção de outro elemento



10.10

```
public class Stack<T> {

   public void push(T e) {
      assert !isFull();

      Node<T> n = new Node<T>();
      n.e = e;
      n.next = top;
      top = n;
      size++;
   }

   public void pop() {
      assert !isEmpty();

      top = top.next;
      size--;
   }

   public T top() {
      assert !isEmpty();

      return top.e;
   }
```

```
   public boolean isEmpty() {
      return top == null;
   }

   public int size() {
      return size;
   }

   private Node<T> top = null;
   private int size = 0;

   private class Node<T> {
      T e;
      Node<T> next;
   }
}
```

10.11

3 Fila: Representação Interna com Lista Ligada

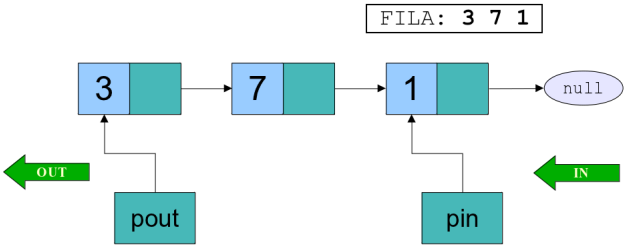
Fila: módulo

- Nome do módulo:
  - `Queue`
- Serviços:
  - `in`: insere um elemento no fim da fila
  - `out`: retira elemento do início da fila
  - `peek`: retorna o elemento do inicio da fila
  - `isEmpty`: verifica se a fila está vazia
  - `isFull`: verifica se a fila está cheia
  - `size`: retorna a dimensão actual da fila
  - `clear`: limpa a fila (retira todos os elementos)

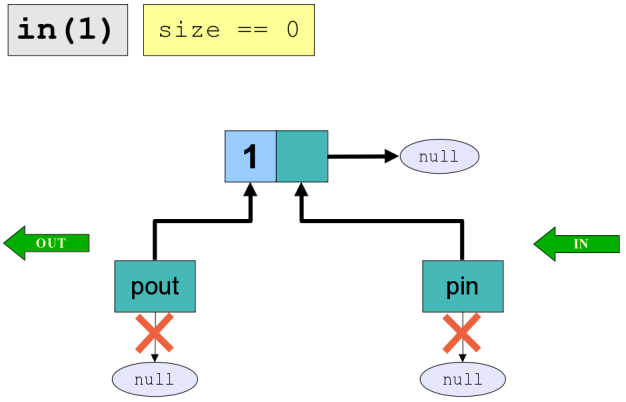
10.12

Fila: Implementação com Lista Ligada

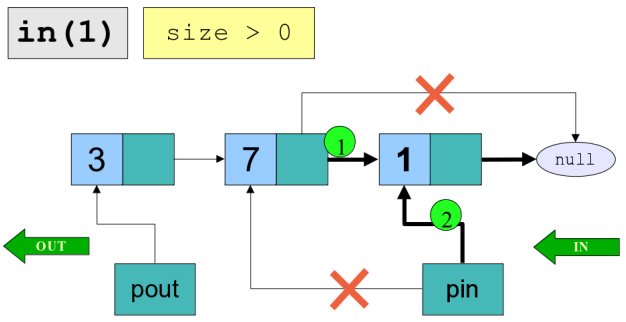
- Os novos elementos (i.e. dados) são inseridos pelo nó pin
- E os elementos são retirados pelo nó da outra extremidade (`pout`)
  - FIFO vazio: `pout=pin=null`



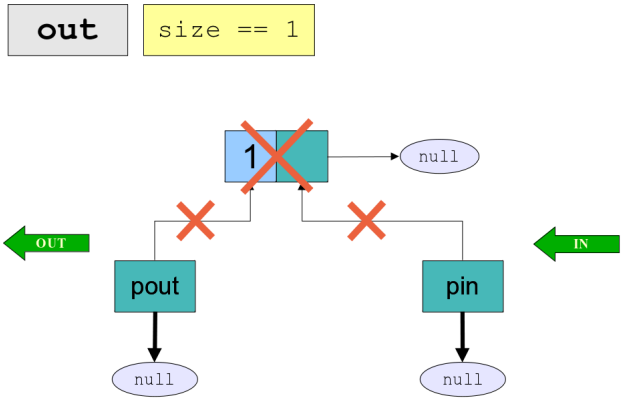
10.13



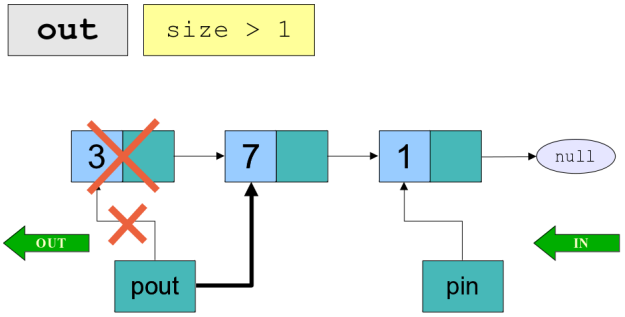
10.14



10.15



10.16



10.17

```
public class Queue<T> {

   public void in(T e) {
      //assert !isFull();

      Node<T> n = new Node<T>();
      n.e = e;
      if (pout == null)
         pout = n;
      else
         pin.next = n;
      pin = n;
      size++;
   }

   public void out() {
      assert !isEmpty();

      size--;
      pout = pout.next;
      if (pout == null)
         pin = null;
   }
```

```
   public T peek() {
      assert !isEmpty();

      return pout.e;
   }

   public int size() {
      return size;
   }

   public boolean isEmpty() {
      return size() == 0;
   }

   private Node<T> pout = null;
   private Node<T> pin = null;
   private int size = 0;

   private class Node<T> {
      T e;
      Node<T> next = null;
   }
}
```

10.18

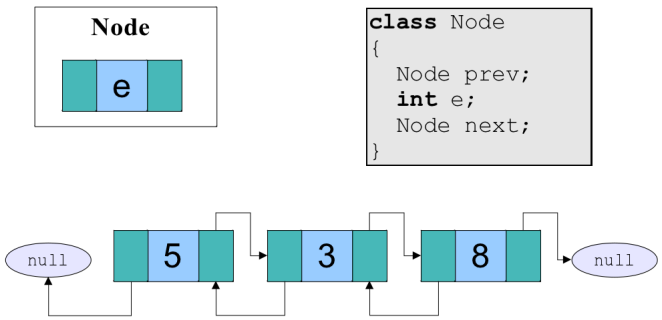
4 Lista Biligada

Lista Biligada

- Estrutura de dados sequencial em que cada elemento da lista contém uma referência para o próximo elemento e outra para o anterior.
  - Essas referências terão o valor `null` caso o elemento a que se refere não exista.
- Ao contrário da lista ligada, permite um acesso sequencial directo do fim para o início.
  - Facilita as operações de inserção e remoção de elementos (`insereTopo`, `insereCauda`, ...).

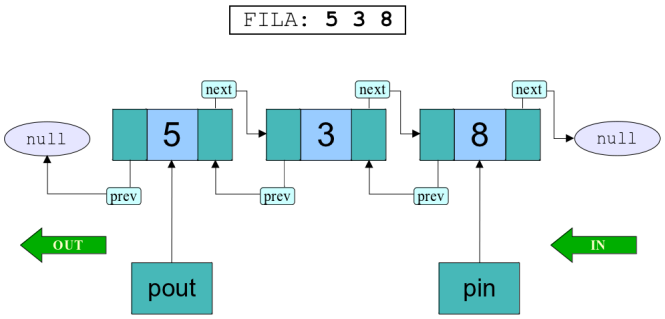
10.19

Lista Biligada

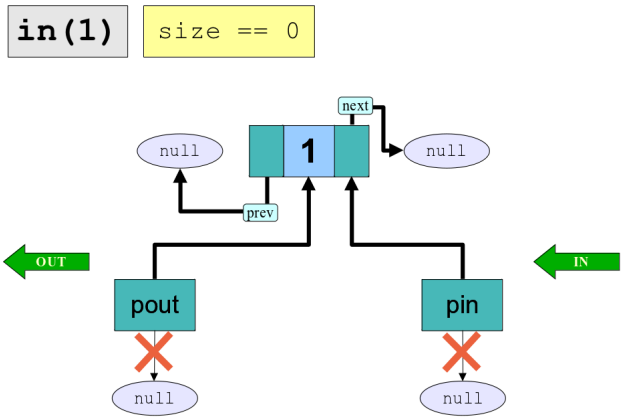


10.20

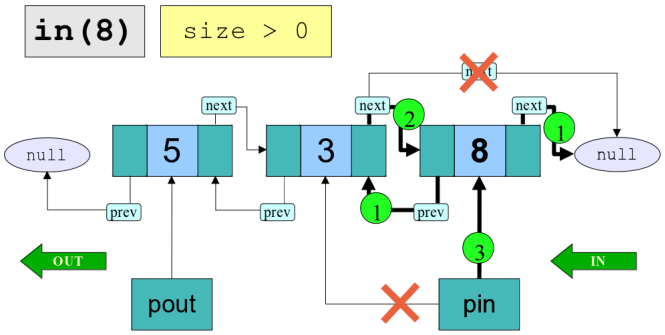
Fila: Implementação com Lista Biligada



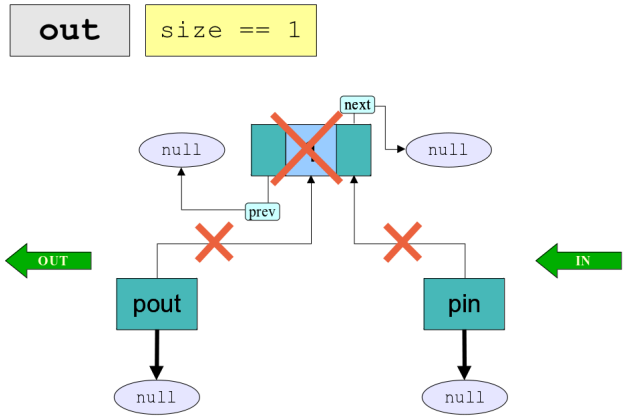
10.21



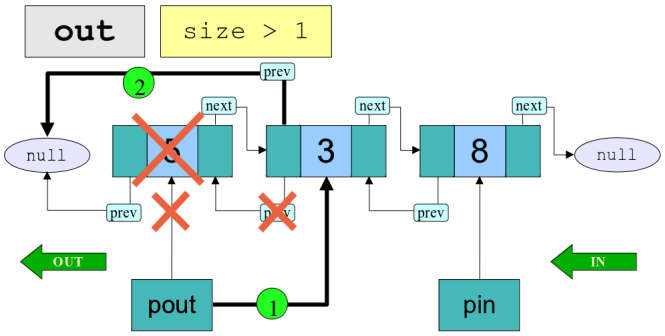
10.22



10.23



10.24



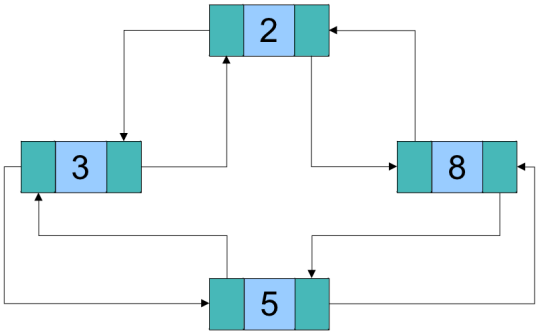
10.25

Fila: Implementação com Lista Biligada

- Se comparada com a implementação com lista ligada *não há nenhuma vantagem* nesta nova implementação!
- Antes pelo contrário, algumas operações tornam-se desnecessariamente mais complexas.
- No entanto, podemos ainda dar outra forma às listas (bi)ligadas: em vez de listas lineares (terminadas com uma referência nula), podemos implementar listas fechadas (as duas extremidades passam a estar unidas).

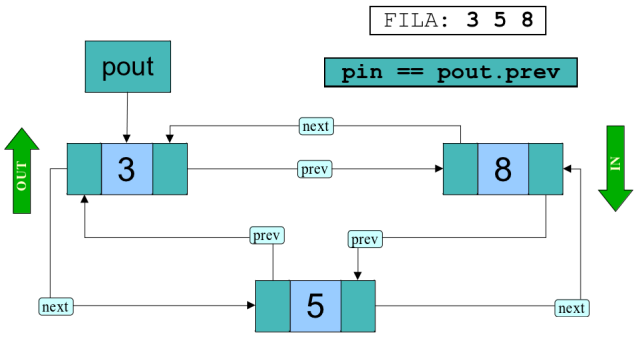
10.26

4.1 Lista Biligada Fechada

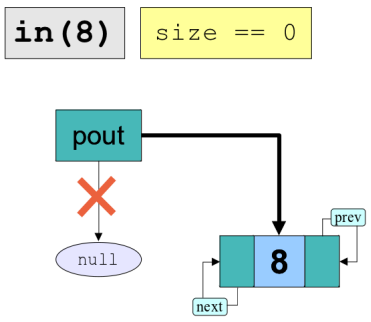


10.27

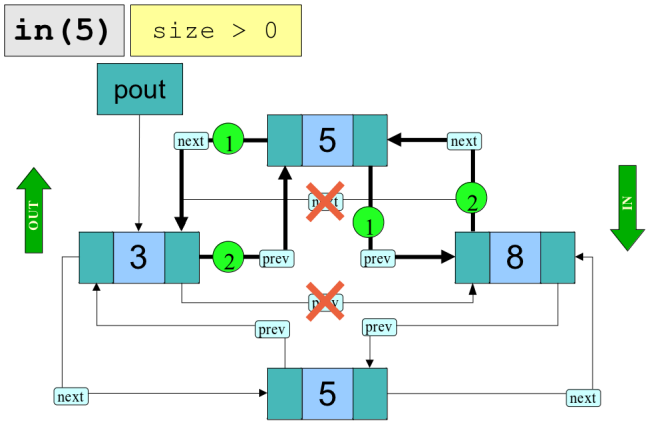
Fila: Implementação com Lista Biligada Fechada



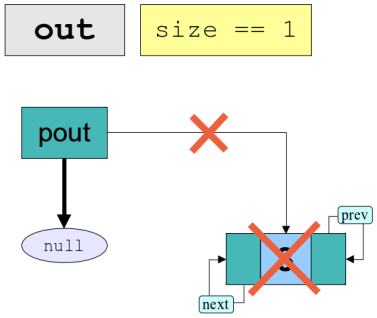
10.28



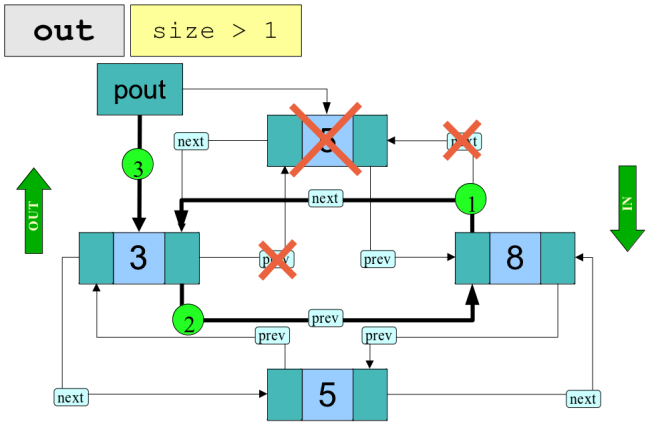
10.29



10.30



10.31



10.32

5 Comparação com Tipos de Listas Ligadas

Tipo de Lista	Simples	Simples	Circular Simples	Biligada	Circular Biligada
Atributos / **Operações**	first	first last	last	first last	first (last)
insert first	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
remove first	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
insert last	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
remove last	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
scan forward	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
scan backward	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n)$
insert middle	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
remove middle	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

10.33