Lição 3

Queue

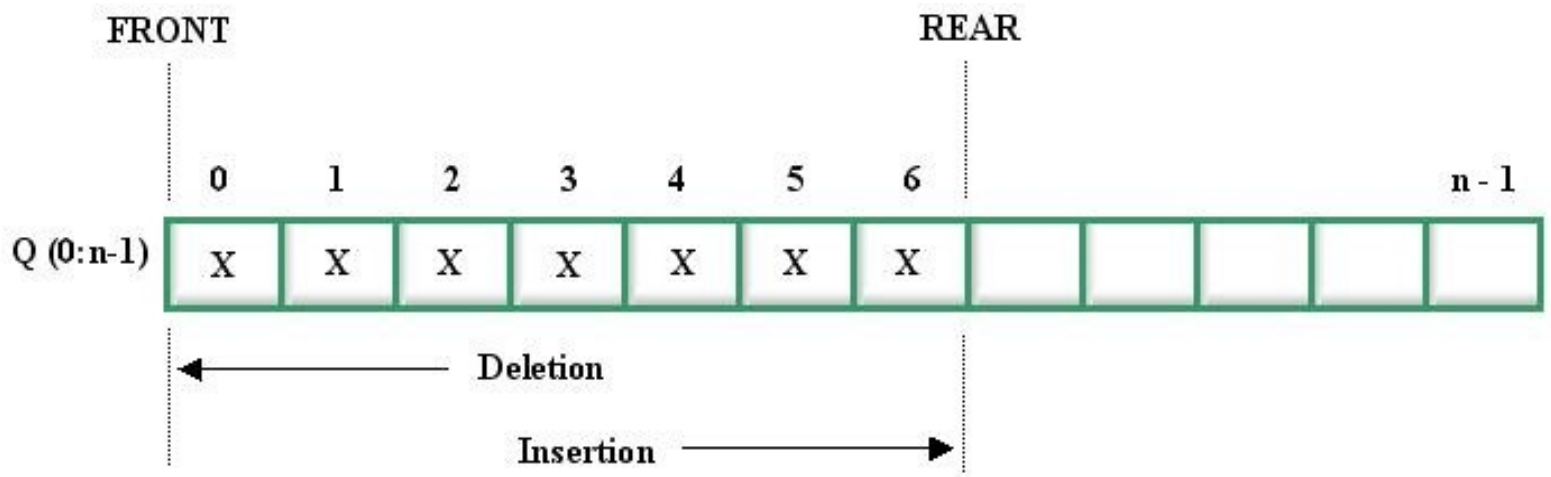
Objetivos

Ao final desta lição, o estudante será capaz de:

- Definir os conceitos básicos e operações com *queue* ADT
- Implementar uma *queue* ADT usando representação seqüencial e encadeada
- Realizar operações em *queue* circulares
- Usar ordenação topológica para produzir uma organização de elementos que satisfaça a um padrão estabelecido

Representação de *Queue*

- ***Queue*** – Conjunto de elementos linearmente ordenados obedecendo a lógica first-in, first-out (FIFO)
- Aplicações: relacionar elementos, ordenação topológica, gráficos transversais, etc.
- 2 operações básicas para manipulação de dados: inserção no final (*enqueue*) e remoção do primeiro elemento (*dequeue*)

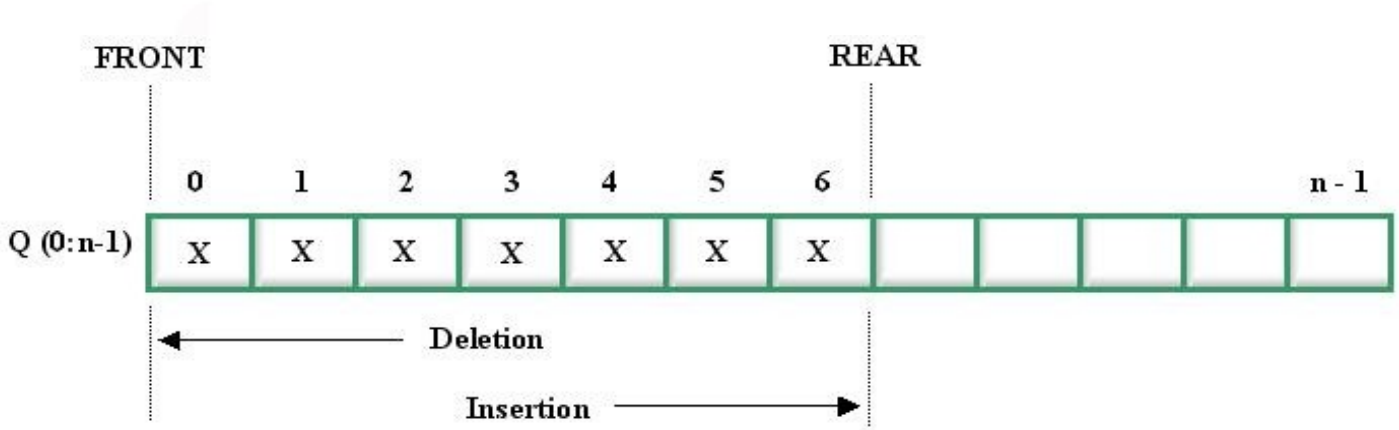


Representação Seqüencial

- Utiliza um array unidimensional
- Remover de uma *Queue* vazia causa um *Underflow*
- Inserir em uma *Queue* cheia causa um *Overflow*

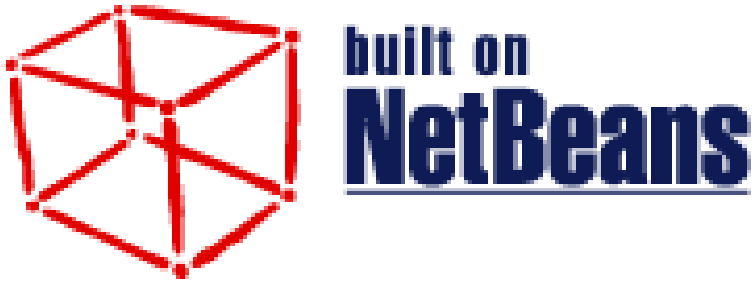
Representação Seqüencial

- *Front* aponta o primeiro elemento da *queue* enquanto que *rear* aponta para a célula seguinte à última ocupada
- *Queue* vazia se front=rear
- *Queue* cheia se front=0 e rear=n
- **Inicialização**: front = 0; rear = 0
- **Inserção** : Q[rear] = x; rear++;
- **Deleção** : x = Q[front]; front++;

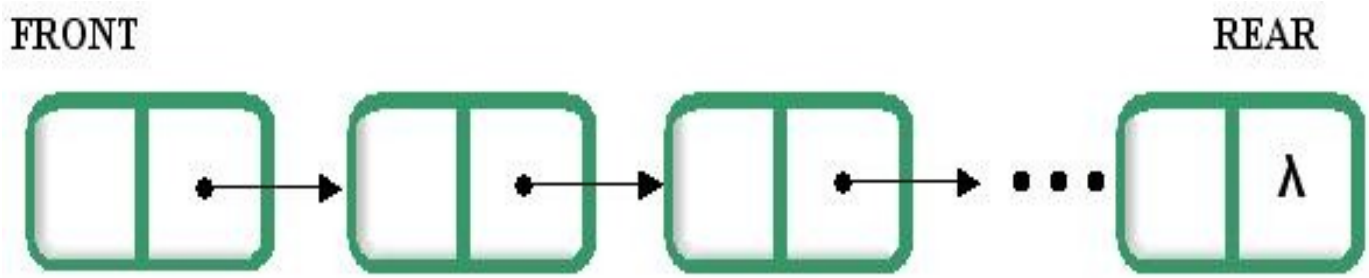


Representação Seqüencial

- Passaremos agora para o NetBeans



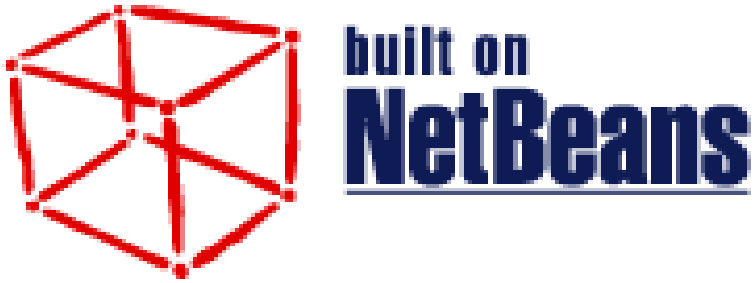
Representação Encadeada



- *Queue* vazia se front = null
- *Overflow* irá acontecer quando houver a tentativa de inserção e não existir espaço disponível

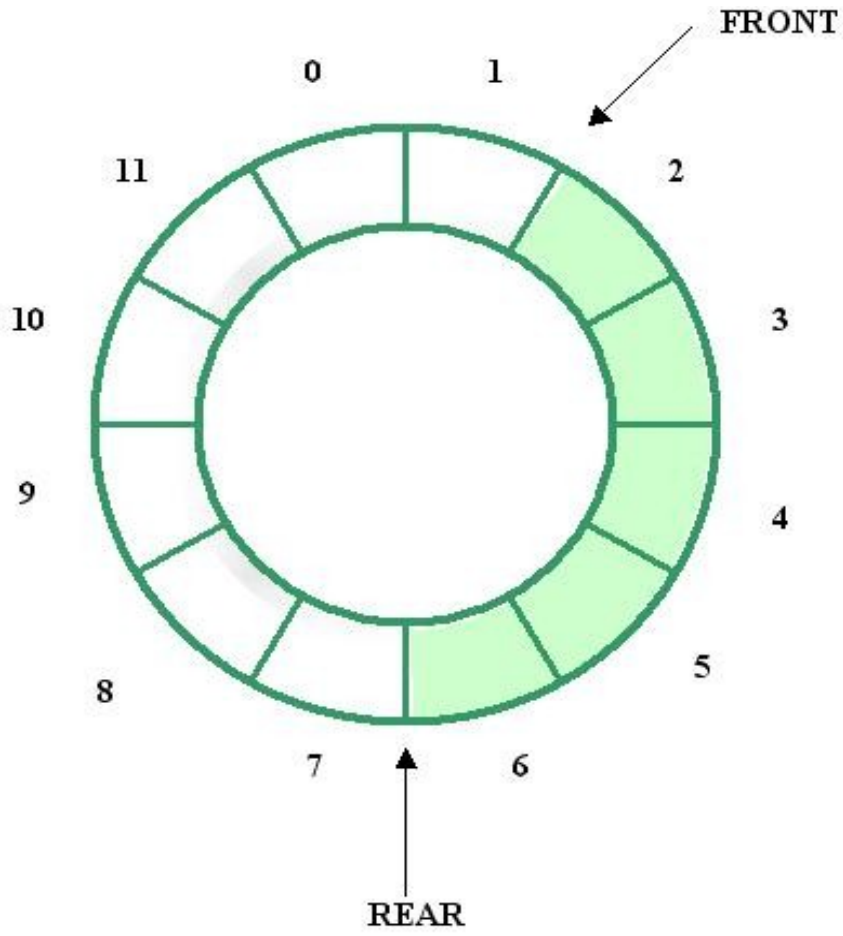
Representação Encadeada

- Passaremos agora para o NetBeans



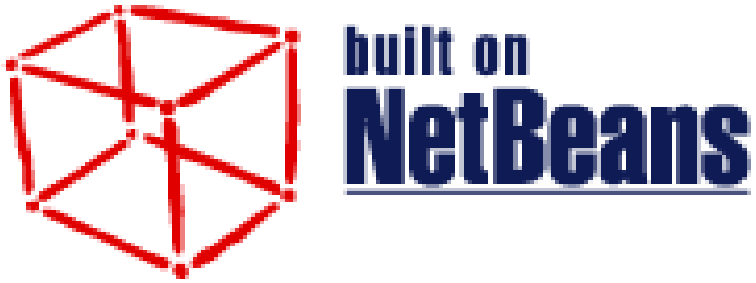
Queue Circular

- Células são consideradas como se estivem organizadas em um círculo
- ***Front*** aponta para o primeiro elemento da *queue*
- ***Rear*** aponta para a célula à direita do último elemento
- *Queue* cheia sempre possui uma célula não utilizada



Queue Circular

- Passaremos agora para o NetBeans



Aplicação: Classificação Topológica

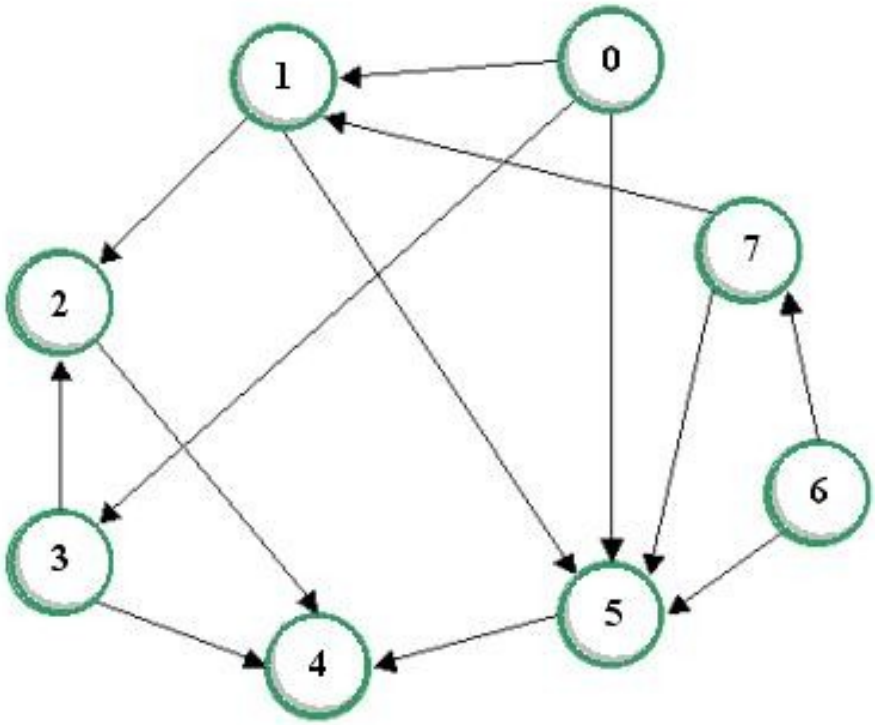
- É um problema característico de redes ativas
- Utiliza ambas as técnicas de representação, seqüencial e “linkada”, na qual a linked queue está inserida em um array seqüencial
- Utiliza técnicas simples para reduzir o tempo gasto
- Aplicada aos elementos de um conjunto no qual a ordenação parcial está definida

Aplicação: Classificação Topológica - Ordenação Parcial

- Ordenação Parcial
  - Um conjunto S, com seus elementos parcialmente ordenados, havendo uma relação entre seus elementos, caracterizada pelo símbolo $\preccurlyeq$, lido como “precede ou igual a”, satisfazendo as seguintes propriedades para quaisquer elementos x, y e z:
    - Ordenação parcial propriedades de $\preccurlyeq$:
      - Reflexividade : $x \preccurlyeq x$
      - Anti-simetria : se $x \preccurlyeq y$ e $y \preccurlyeq x$, então $x = y$
      - Transitividade : se $x \preccurlyeq y$ e $y \preccurlyeq z$, então $x \preccurlyeq z$
    - Resultado. Se $x \preccurlyeq y$ e $x \neq y$ então $x \prec y$. Equivalentemente,
      - Não-Reflexividade : $x \prec x$
      - Assimétrica : se $x \prec y$ então $y \prec x$
      - Transitividade : se $x \prec y$ e $y \prec z$, então $x \prec z$

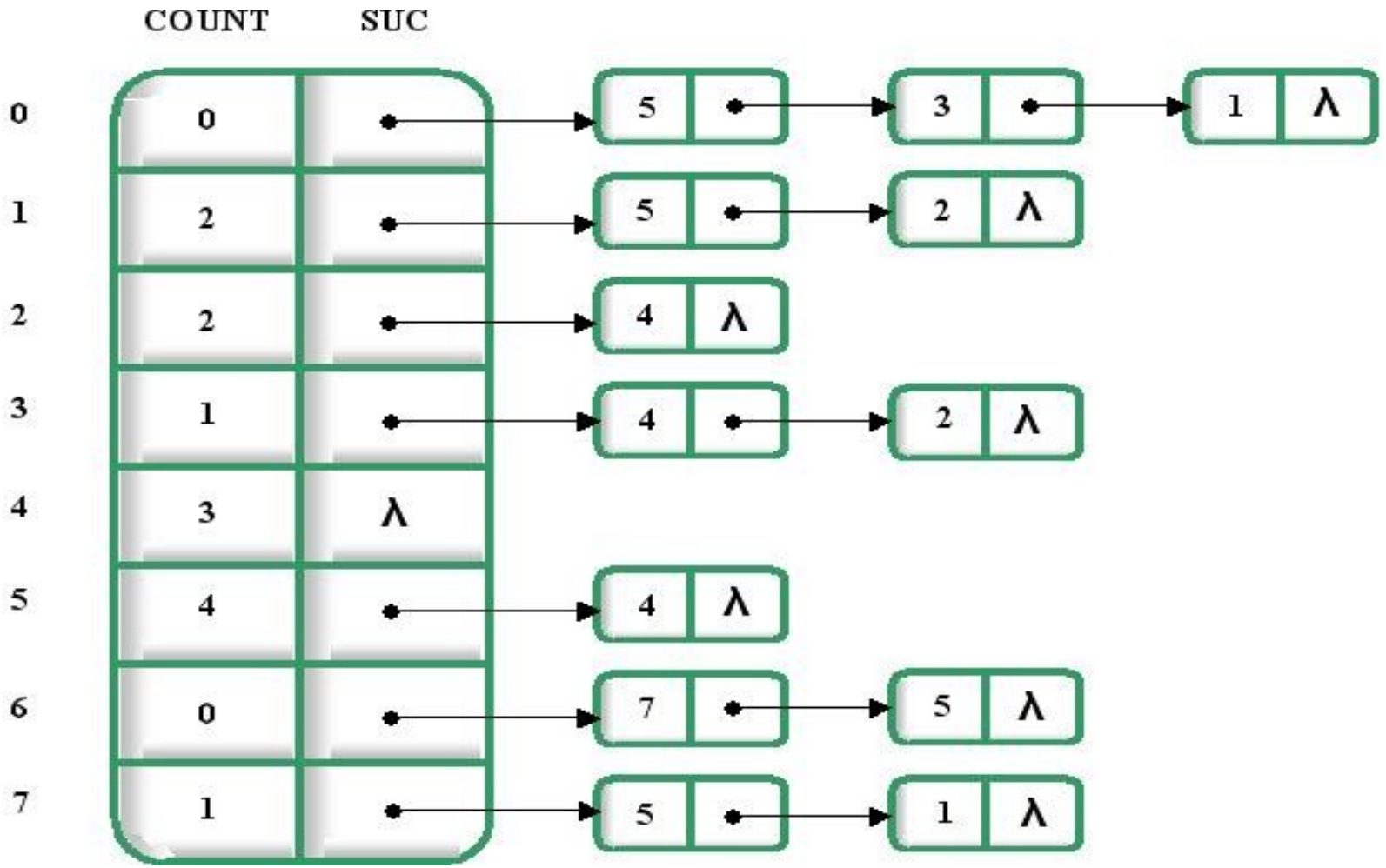
Aplicação: Classificação Topológica - Ordenação Parcial

0,1
0,3
0,5
1,2
1,5
2,4
3,2
3,4
5,4
6,5
6,7
7,1
7,5



Saída: 0 6 3 7 1 2 5 4

Aplicação: Classificação Topológica - Algoritmo



Aplicação: Classificação Topológica - Algoritmo

- Entrada
- Saída
- Algoritmo apropriado

Aplicação: Classificação Topológica - Algoritmo

- Uma *queue* encadeada pode ser usada para evitar percorrer todo o vetor COUNT repetidamente buscando por objetos com um contador igual a zero
  - QLINK[j] = k se k é o próximo item na fila

    = 0 se j for o último item na fila
  - O COUNT de cada item j na fila pode ser reutilizado como um campo “linkado”

Sumário

- Representação de *Queue*
  - Representação Seqüencial
  - Representação Encadeada
- *Queue* Circular
- Aplicação: Classificação Topológica
  - Ordenação Parcial
  - Algoritmo

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**