Filas

Prof. Denio Duarte
duarte@uffs.edu.br
Prof. Claunir Pavan
claunir.pavan@uffs.edu.br

Fila (queue)

- Uma fila é uma estrutura de dados que é uma especialização da lista simplesmente encadeada
- Implementa o conceito de FIFO: first-in first-out, ou seja, o primeiro elemento que entrou é o primeiro a sair.
  - Não é possível inserir no meio ou no início. SEMPRE NO FIM
- Uma fila tem as seguintes propriedades:
  - Podemos realizar duas operações básicas:
    - **Inserir** um novo item na fila (enqueue)
    - **Remover** um item da fila (dequeue)
  - A operação de remoção sempre **remove o item que está há mais tempo** na fila (FIFO)

Fila (queue)

- O sistema operacional utiliza este conceito para:
  - Controlar a fila de impressão
  - Controla a execução do processos
- Dois conceitos baseados em fila:
  - Fila com prioridade em que a inserção obedece a prioridade do item a ser inserido
  - Fila circular: o tamanho da fila é fixo e é necessário controlar o início e fim da fila. Quando são iguais, a fila está vazia, quando o fim chegou na capacidade da fila, esta está cheia.

Exemplo de funcionamento

Entrada:

E S * T R D * A ...

Processamento:

- Se o caracter é uma letra: insere na fila
- Se é um *: remove da fila

Fila

Fila vazia

Saída:

Exemplo de funcionamento

Entrada:

S * T R D * A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila

Saída:

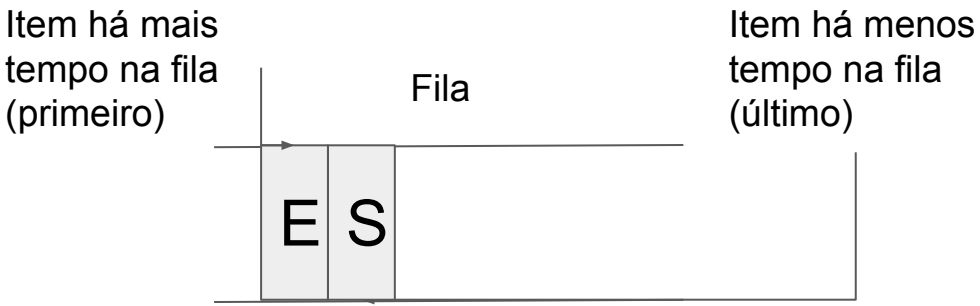
Exemplo de funcionamento

Entrada:

* T R D * A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila



Saída:

Exemplo de funcionamento

Entrada:

T R D * A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila

Saída:

E

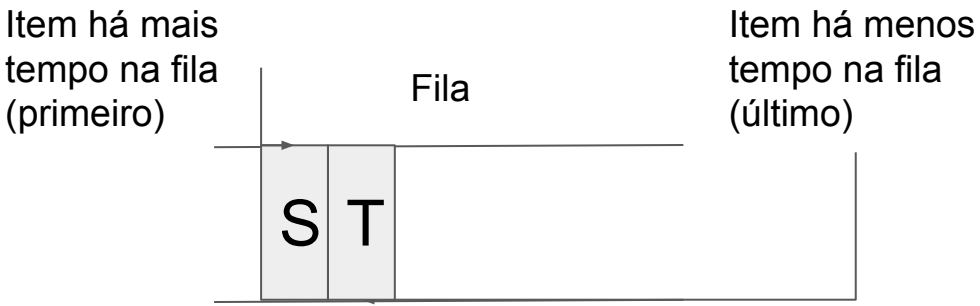
Exemplo de funcionamento

Entrada:

R D * A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila



Saída:

E

Exemplo de funcionamento

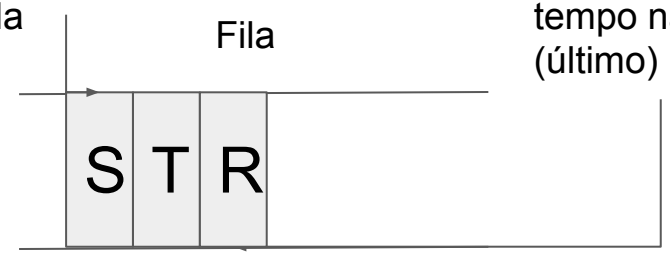
Entrada:

D * A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila

Item há mais tempo na fila (primeiro)



Item há menos tempo na fila (último)

Saída:

E

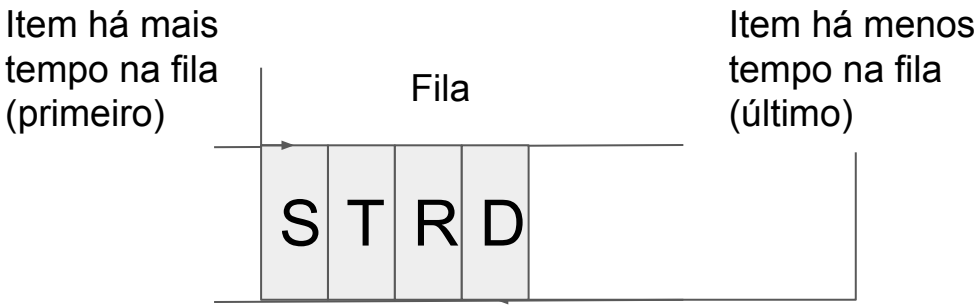
Exemplo de funcionamento

Entrada:

* A ...

Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila



Saída:

E

Exemplo de funcionamento

Entrada:

A ...

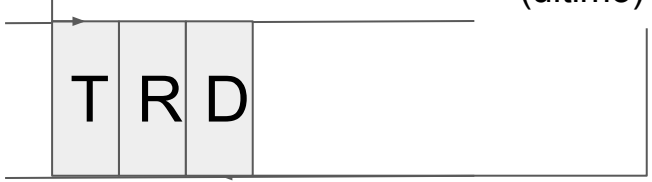
Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila

Item há mais tempo na fila (primeiro)

Fila

Item há menos tempo na fila (último)



Saída:

E S

Exemplo de funcionamento

Entrada:

...

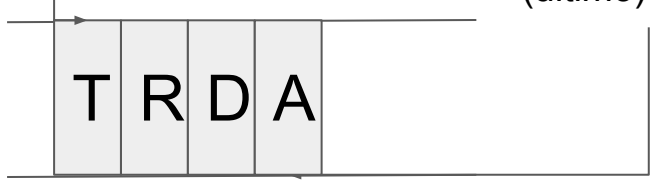
Processamento:

- Se é uma letra: insere na fila
- Se é um *: remove da fila

Item há mais tempo na fila (primeiro)

Fila

Item há menos tempo na fila (último)



Saída:

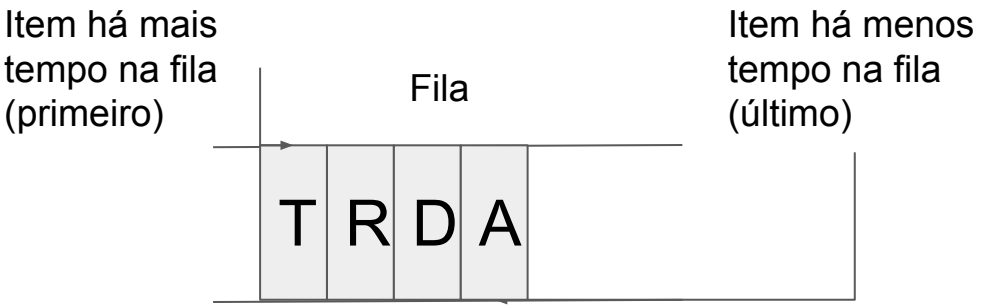
E S

Exemplo de funcionamento

Entrada:

O processamento continuaria...

...



Saída:

E S

Implementação

- Uma fila poder ser implementada de duas maneiras: usando um vetor ou usando uma lista encadeada simples
- Cada opção de implementação possui vantagens e desvantagens (as mesmas das opções de implementação de uma pilha)
- Usando um vetor
  - Desvantagem: É necessário definir um tamanho máximo da fila; uso ineficiente da memória total alocada
  - Vantagem: Inserção e remoção de itens não requerem alocação e liberação de memória
- Usando uma lista encadeada simples
  - Desvantagem: Inserção e remoção de itens requerem alocação e liberação de memória
  - Vantagem: Uso mais eficiente da memória total alocada

Implementação - lista encadeada simples

- Podemos declarar os seguintes tipos:

```
typedef int Item;
```

```
typedef struct TQueue {
    Item item;
    struct TQueue *next;
} EQueue;
```

```
typedef struct {
    EQueue *head;
    EQueue *tail;
} Queue;
```

Implementação - lista encadeada simples

- Podemos declarar os seguintes tipos:

```
typedef int Item;
```

```
typedef struct TQueue {
    Item item;
    struct TQueue *next;
} EQueue;
```

```
typedef struct {
    EQueue *head;
    EQueue *tail;
} Queue;
```

```
Queue *queue;
```

Implementação - lista encadeada simples

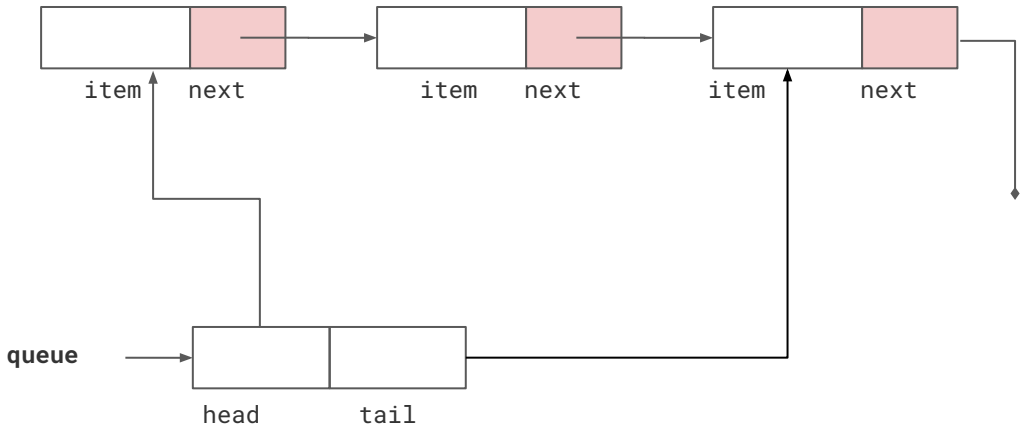
- Podemos declarar os seguintes tipos:

```c
typedef int Item;
```

```c
typedef struct TQueue {
    Item item;
    struct TQueue *next;
} EQueue;
```

```c
typedef struct {
    EQueue *head;
    EQueue *tail;
} Queue;
```

```c
Queue *queue;
```



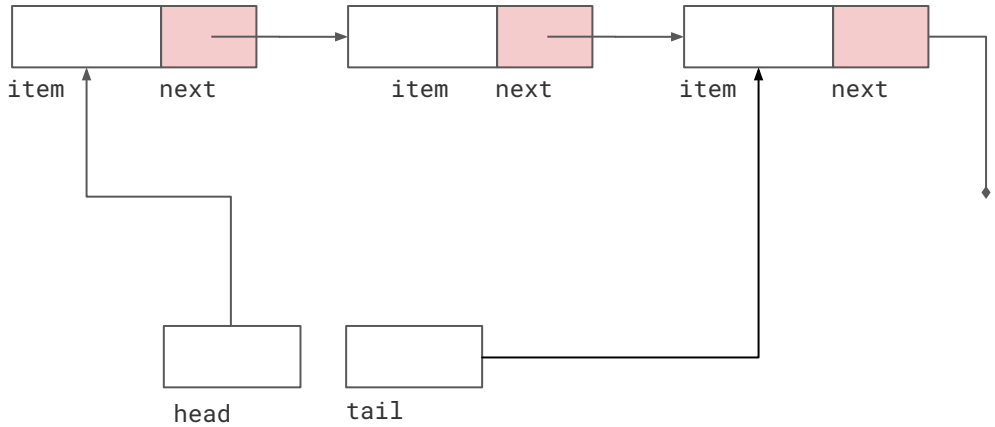
Implementação - lista encadeada simples

- Ou utilizando dois ponteiros para controlar início e fim.

```c
typedef int Item;
```

```c
typedef struct TQueue {
    Item item;
    struct TQueue *next;
} EQueue;
```

```c
EQueue *fila, *head, *tail;
```



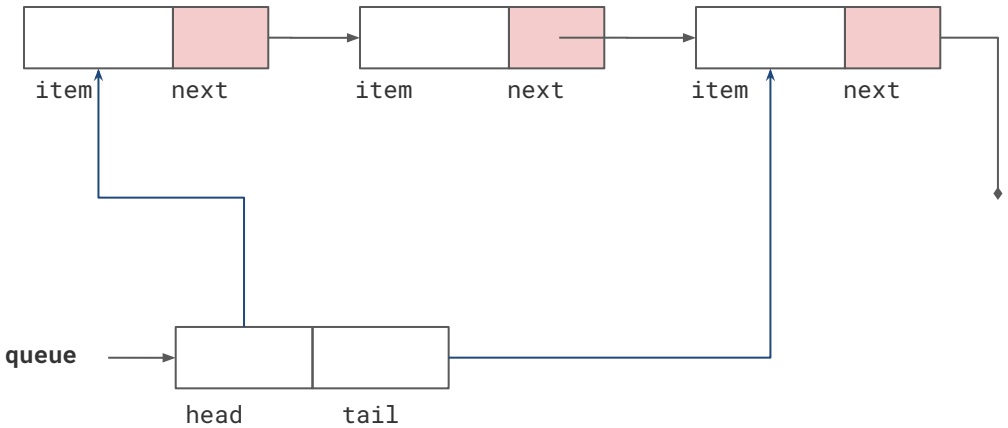
Implementação - lista encadeada simples

- Operações:

```
void enQueue(Queue *queue, Item item)

void deQueue(Queue *queue, Item *item)

void initQueue(Queue *queue)

int isEmpty(Queue *queue)

void freeAll(Queue *queue)
```

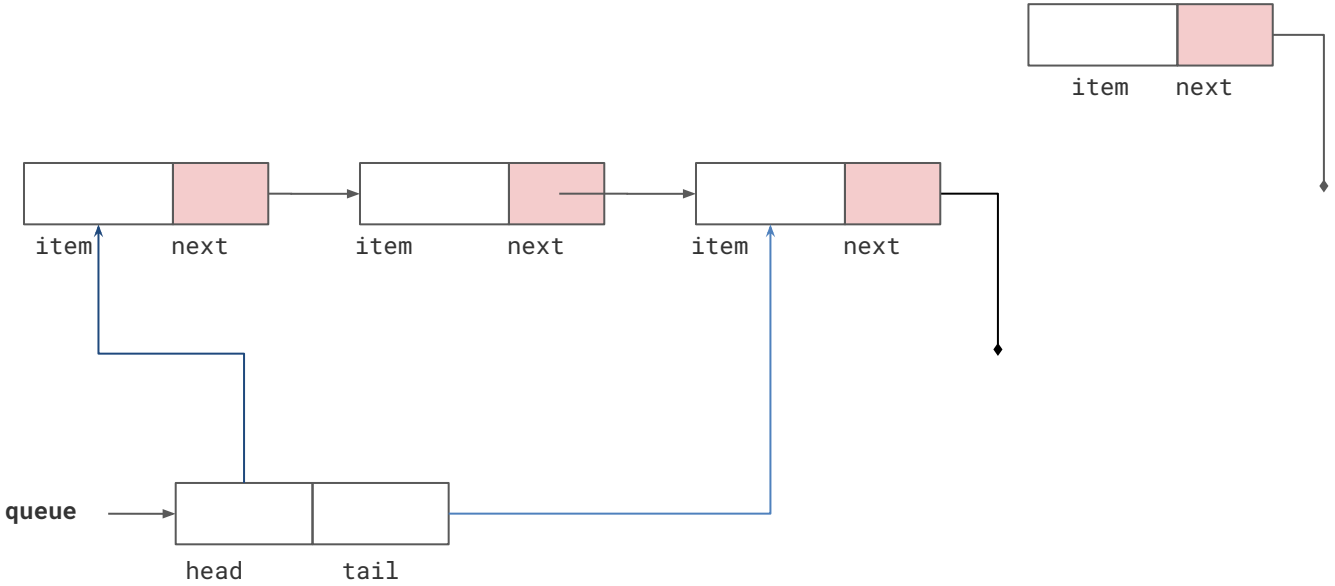
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



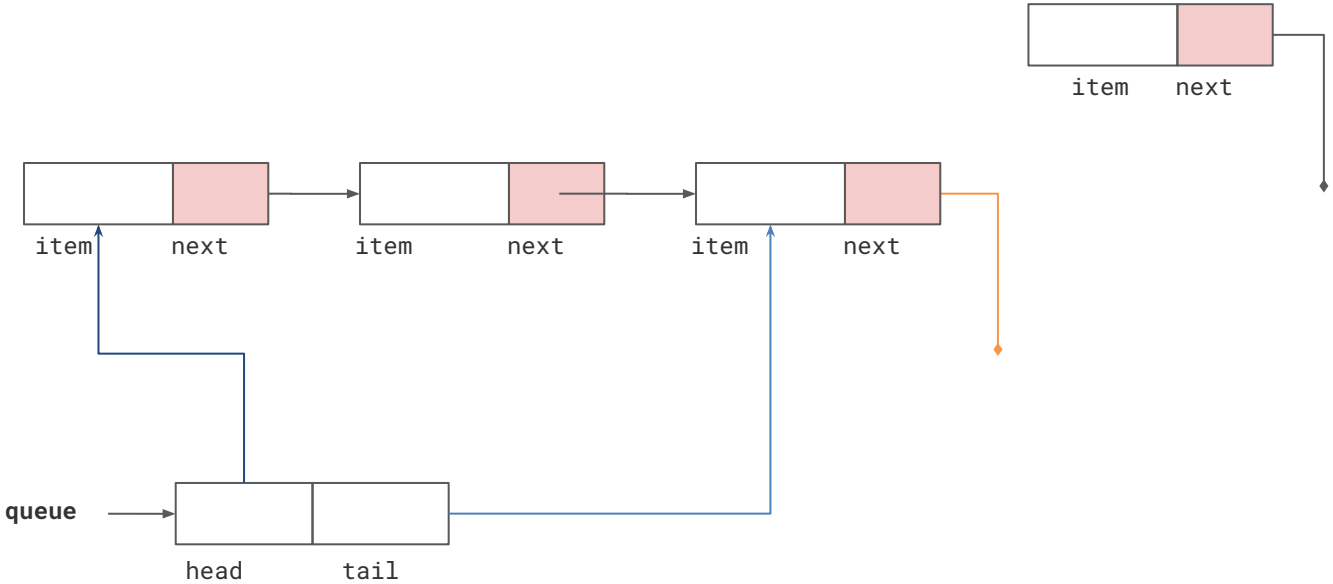
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



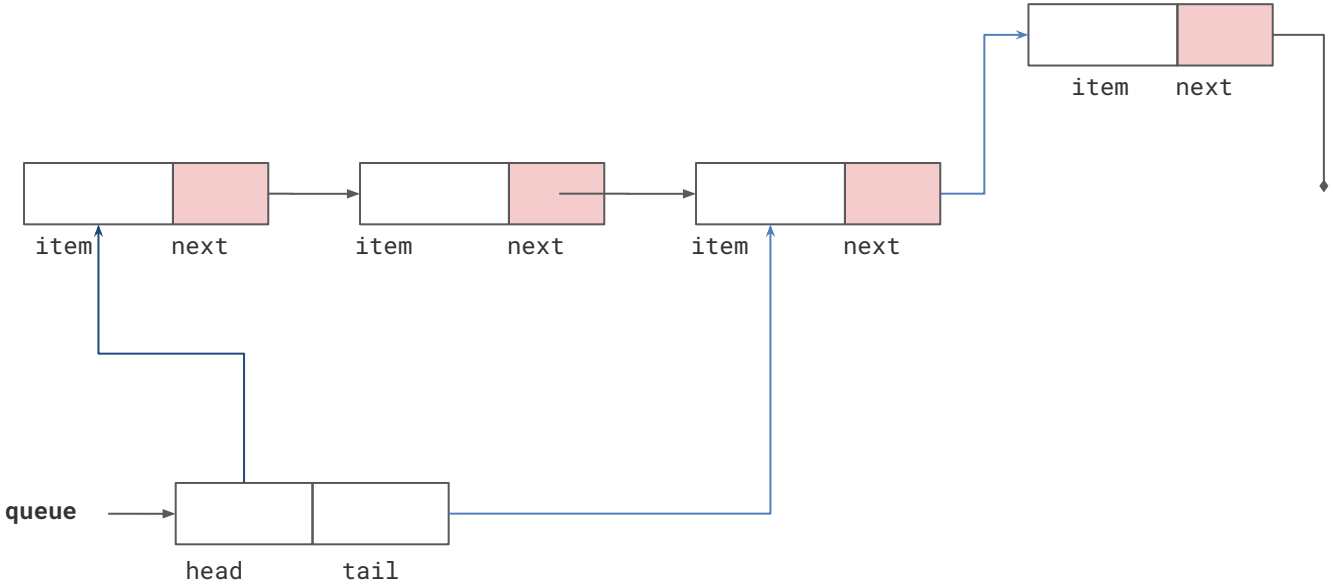
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



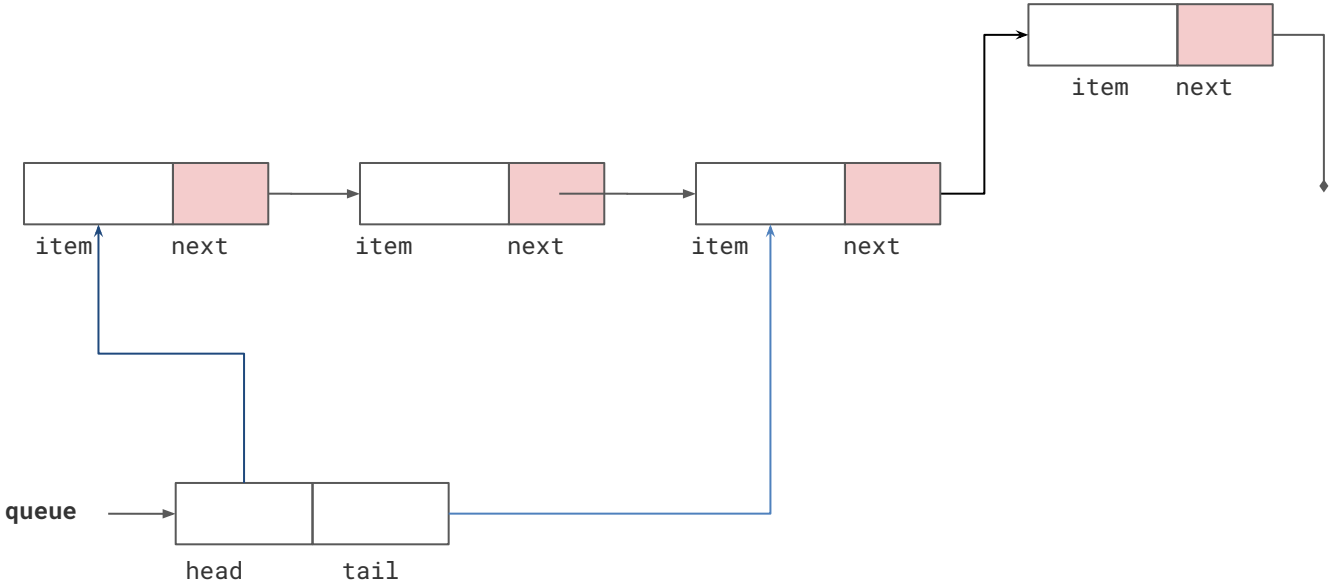
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



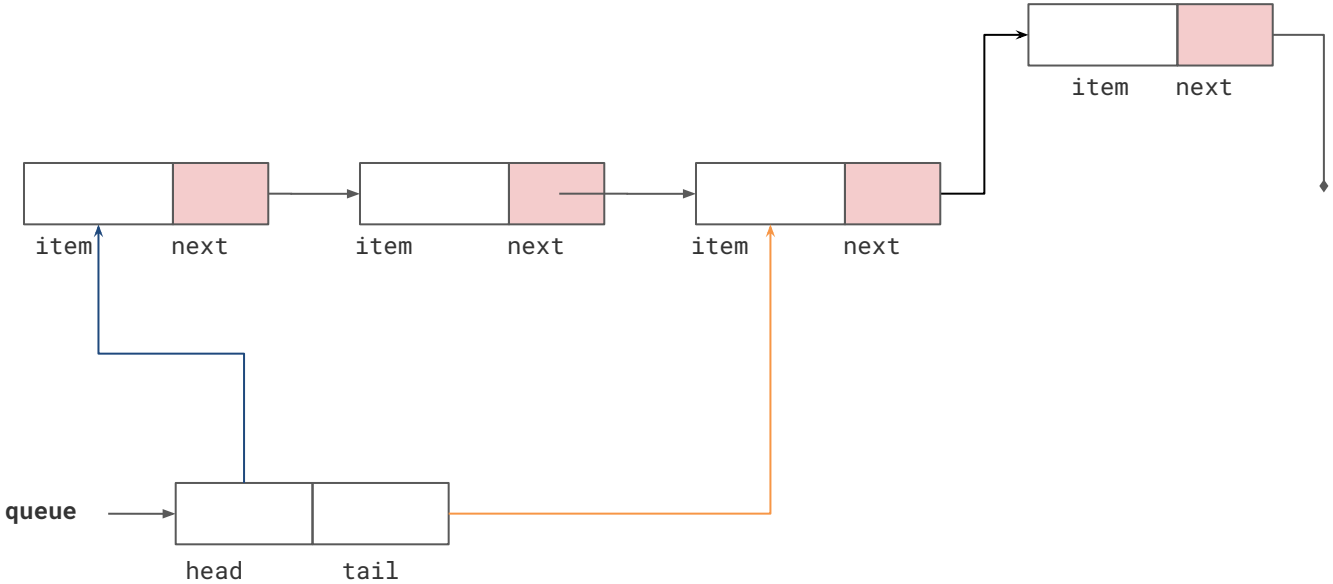
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



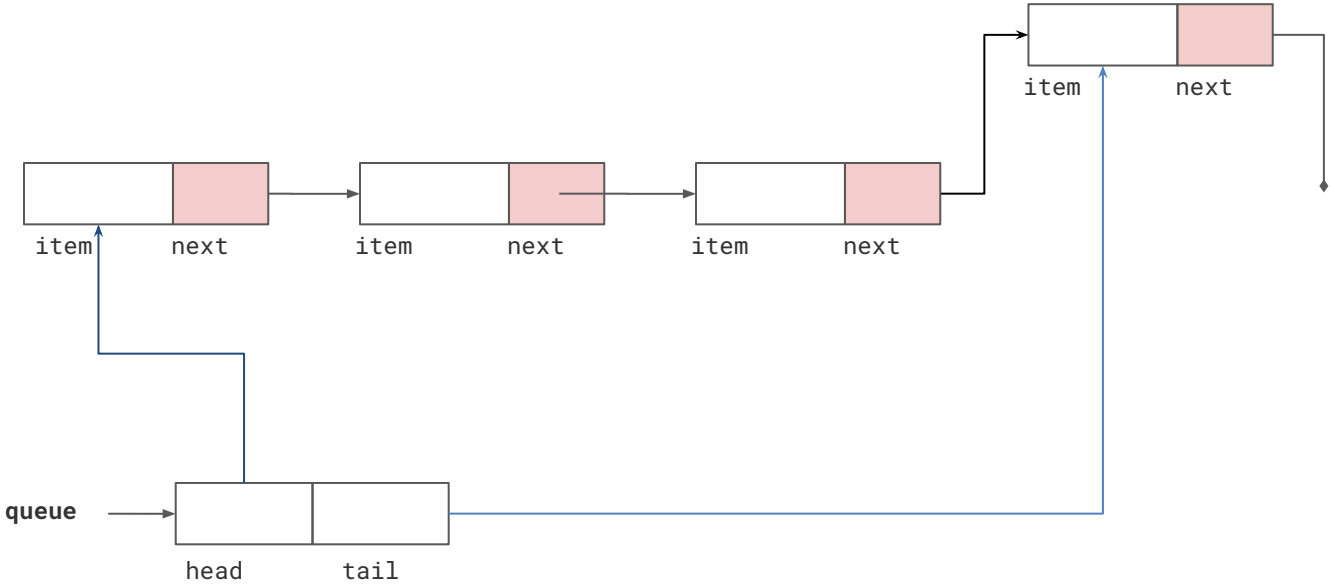
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:



Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:

```
void enQueue(Queue *queue, Item item) {
    EQueue *aux;

    // Cria um novo elemento da lista encadeada que representa a fila e
    // armazena neste novo elemento o item a ser inserido na fila
    aux = (Queue *) malloc(sizeof(Queue));
    aux->item = item;
    aux->next = NULL;

    // Continua no proximo slide
```

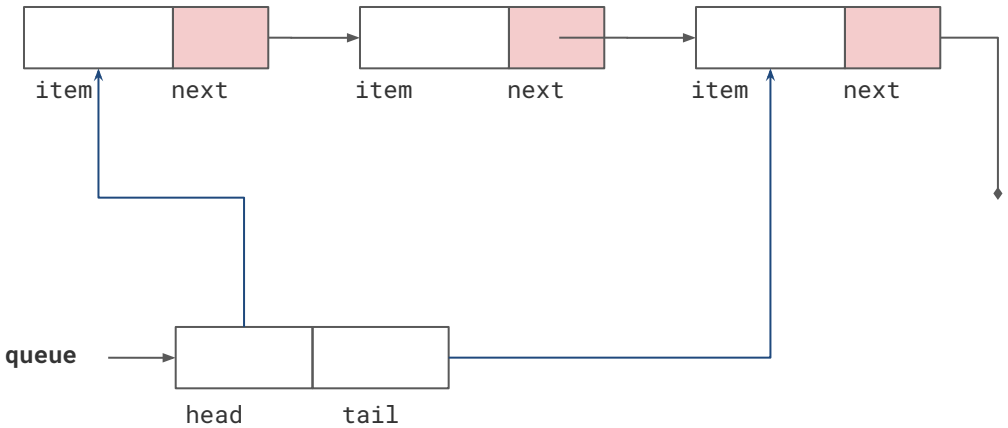
Implementação - lista encadeada simples

- Operação de inserir um novo elemento na fila:

```
    // Continuacao do slide anterior

    // Insere o novo elemento no fim da lista encadeada que representa a
    // fila
    if (queue->head == NULL) { // Se a fila esta vazia
        queue->head = aux;
        queue->tail = aux;
    }
    else { // Se a fila nao esta vazia
        queue->tail->next = aux;
        queue->tail = aux;
    }
}
```

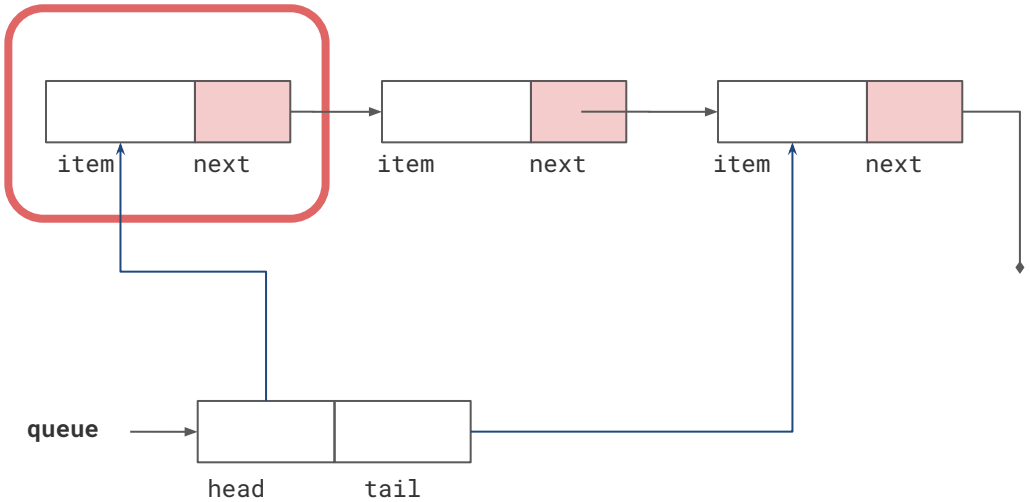
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



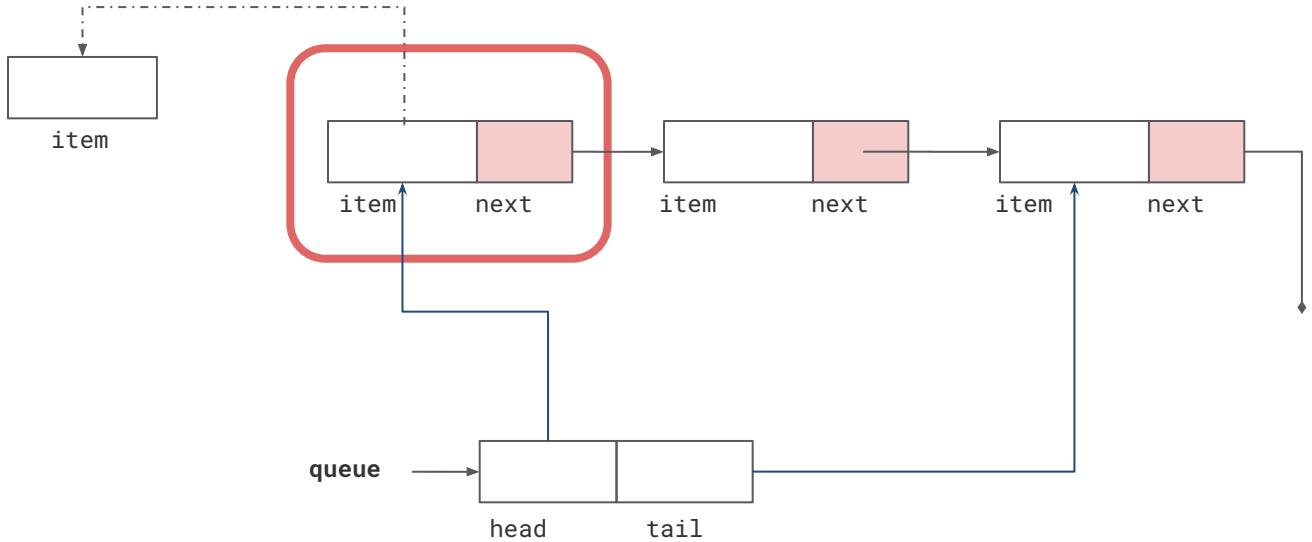
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



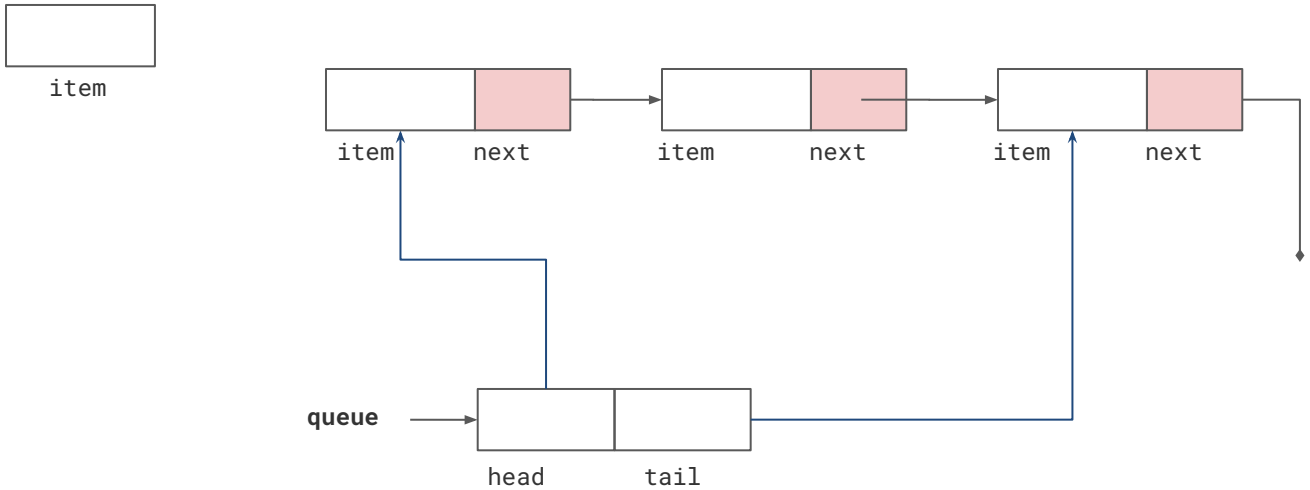
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



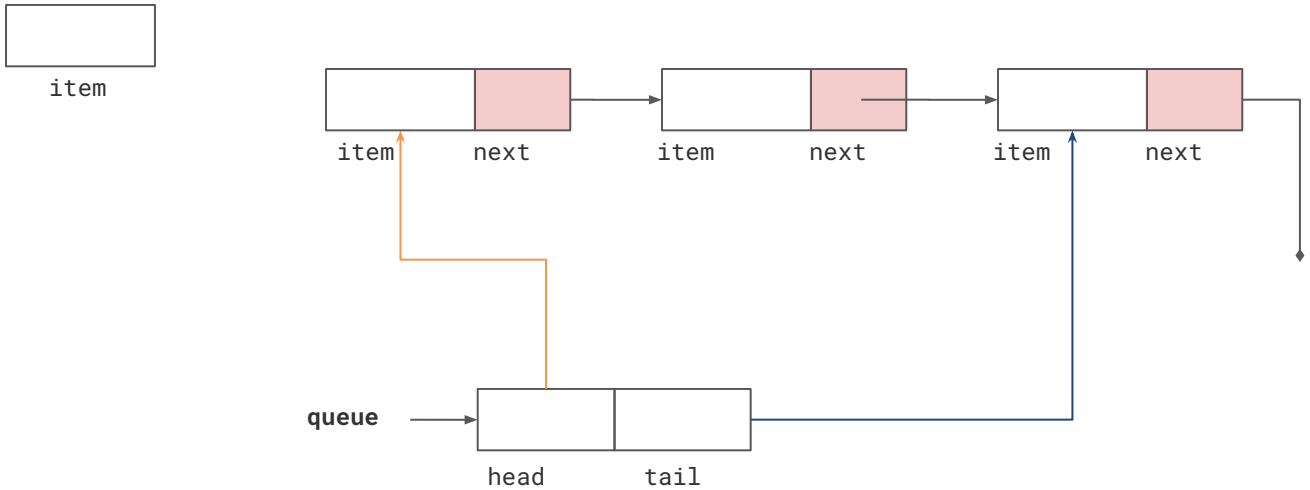
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



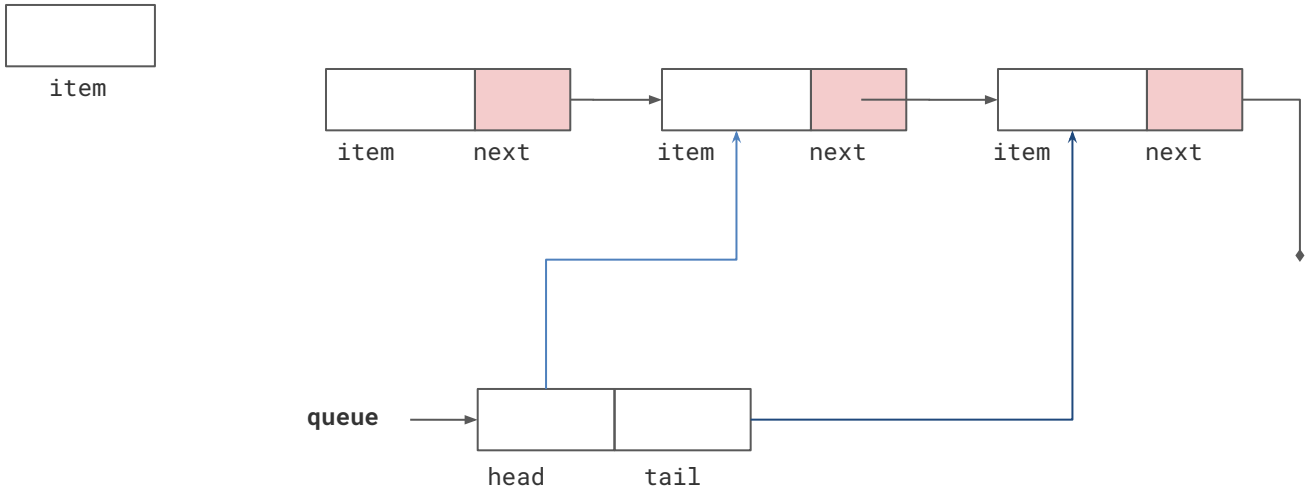
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



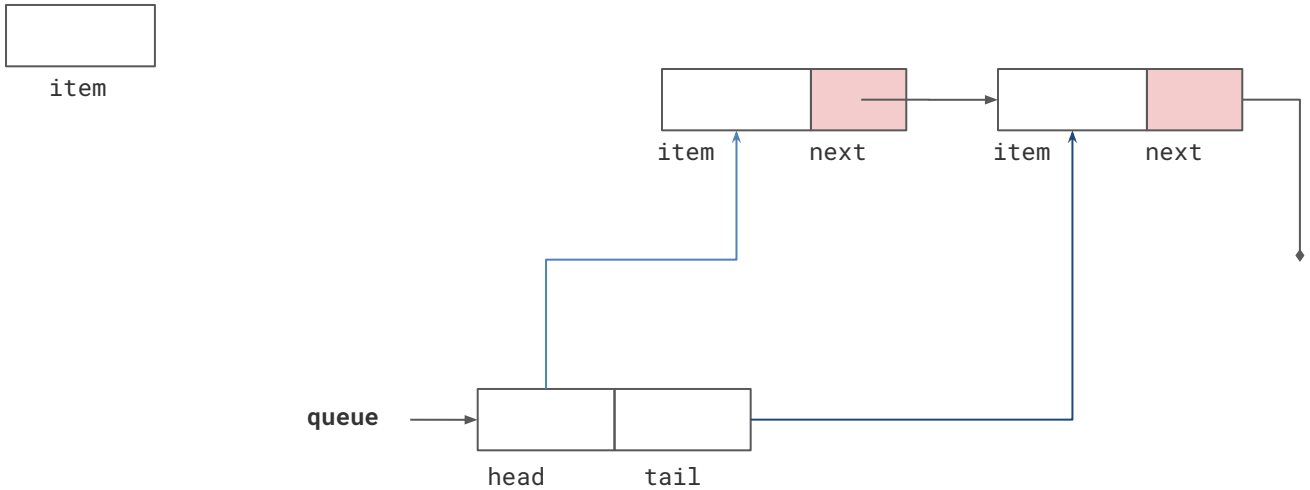
Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



Implementação - lista encadeada simples

- Operação de remover um elemento da fila:



Implementação - lista encadeada simples

- Operação de remover um elemento da fila:

```
void deQueue(Queue *queue, Item *item) {
    EQueue *aux;

    // Verificar se a fila esta vazia!

    // Armazena o item a ser removido da fila
    *item = queue->head->item; // ATENÇÃO: Depende da definição do tipo do
                               // item
    // Continua no proximo slide
```

Implementação - lista encadeada simples

- Operação de remover um elemento da fila:

```
    // Continuacao do slide anterior

    // Armazena o primeiro elemento da lista encadeada que representa a fila e
    // remove este primeiro elemento da lista
    aux = queue->head;
    if (queue->head == queue->tail) {
        queue->head = NULL;
        queue->tail = NULL;
    }
    else {
        queue->head = queue->head->next;
    }

    // Libera a memoria alocada para o elemento removido
    free(aux);
}
```

Implementação - lista encadeada simples

- Operação de remover um elemento da fila, retornando o **conteúdo** do nodo:

```c
typedef int Item;

typedef struct TEQueue {
    Item item;
    struct TEQueue *next;
} EQueue;

typedef struct TQueue {
    EQueue *head;
    EQueue *tail;
} Queue;

void initQueue(Queue *queue) {
    queue->head = NULL;
    queue->tail = NULL;
}

int isEmpty(Queue *queue){
    return (queue->head == NULL);
}
```

```c
int deQueue(Queue *queue) {
    if(isEmpty(queue)) return -1;

    int i = queue->head->item;
    EQueue *aux = queue->head;
    //desencadeia o nó
    if (queue->head == queue->tail) {
        queue->head = queue->tail = NULL;
    }
    else {
        queue->head = queue->head->next;
    }
    free(aux);
    return(i);
}
```

Implementação - lista encadeada simples

- Operação de remover um elemento da fila, retornando o **nodo**:

```
4   typedef int Item;
5
6   typedef struct TEQueue {
7       Item item;
8       struct TEQueue *next;
9   } EQueue;
10
11  typedef struct TQueue {
12      EQueue *head;
13      EQueue *tail;
14  } Queue;
15
16  void initQueue(Queue *queue) {
17      queue->head = NULL;
18      queue->tail = NULL;
19  }
20
21  int isEmpty(Queue *queue){
22      return (queue->head == NULL);
23  }
```

```
50  //retorna uma cópia de um nodo dequeueReturnNode
51  EQueue deQueueRN(Queue *queue) {
52      EQueue aux;
53
54      if(isEmpty(queue)){//retorna nodo com "lixo"
55          aux.next = NULL;
56          aux.item = -1;
57          return aux;
58      }
59
60      EQueue *nodeToFree = queue->head;
61      aux.item = queue->head->item;
62      // remove primeiro elemento da lista
63      if (queue->head == queue->tail) {
64          initQueue(queue);
65      }
66      else {
67          queue->head = queue->head->next;
68      }
69      free(nodeToFree);
70      return aux;
71  }
```

Implementação - lista encadeada simples

- Operação de inicializar a fila:

```
void initQueue(Queue *queue) {
    queue->head = NULL;
    queue->tail = NULL;
}
```

Implementação - lista encadeada simples

- Operação de testar se a fila está vazia:

```
int isEmpty(Queue *queue) {
    return (queue->head == NULL);
}
```

Implementação - lista encadeada simples

- Operação de destruir a fila (liberar a memória alocada para a fila):

```
void freeAll(Queue *queue) {
    EQueue *aux;

    while (queue->head != NULL) {
        // Armazena o primeiro elemento da lista encadeada que representa a
        // fila e remove este primeiro elemento da lista
        aux = queue->head;
        queue->head = queue->head->next;

        // Libera a memoria alocada para o elemento removido
        free(aux);
    }
    queue->tail = NULL;
}
```

Implementação - lista encadeada simples

- Usando a fila:

```c
int main() {
    Queue queue;
    Item item;

    initQueue(&queue);

    for (int i = 0; i < 10; i++) {
        item = i;

        printf("Inserindo na fila o item %d.\n", item);
        enQueue(&queue, item);
    }

    // Continua no proximo slide
```

Implementação - lista encadeada simples

- Usando a fila:

```
    // Continuacao do slide anterior

   while (isEmpty(&queue) == 0) { // Enquanto a fila não estiver vazia
        deQueue(&queue, &item);
        printf("Item %d removido da fila.\n", item);
    }

    freeAll(&queue); // Sem efeito se a fila já está vazia

    return 0;
}
```

Exercícios

1. Faça um programa que receba uma string, caractere por caractere
   - Cada caractere é colocado em uma fila
   - No fim da entrada, esvazie a fila imprimindo os caracteres armazenados

Referências

- Os exercícios desta apresentação são baseados no seguinte material: Pfenning, F., Platzer, A., Simmons, R., Lecture 9 - Stacks and Queues, Lecture Notes, Carnegie Mellon University, 2020. (https://www.cs.cmu.edu/~15122/handouts/09-stackqueue.pdf)