Algoritmos e Estrutura de Dados II

Pilha, Fila e Lista
(Alocação Estática)

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia

Roteiro

Objetivos

Esta aula tem como objetivos:

1. Apresentar os conceitos básicos sobre filas, pilhas e listas;
2. Explicitar as diferenças, vantagens e desvantagens de cada um;
3. Exemplificar os algoritmos por meio de pseudo-códigos.

Referências bibliográficas

Introdução

- As estruturas de dados são formas de distribuir e relacionar os dados disponíveis, de modo a tornar mais eficientes os algoritmos que manipulam esses dados;
- Quando o programador cria um algoritmo para solucionar um problema, ele também cria uma estrutura de dados que é manipulada pelo algoritmo.

Introdução

- A escolha de uma determinada estrutura pode afetar substancialmente a quantidade de área de armazenamento requerida para o processamento bem como o tempo deste processamento;
- É, portanto, de grande importância o estudo de diferentes estruturas que possam ser utilizadas eventualmente na resolução de um problema, de forma a simplificar a sua implementação prática.

Implementação Estática
Pilhas

Pilhas

Pilha

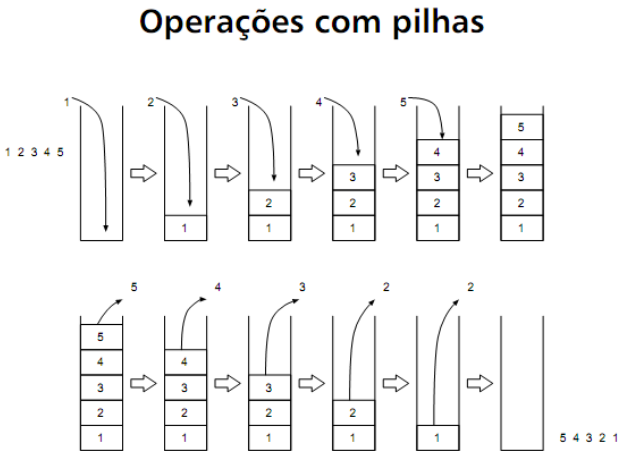
Segundo **??**), uma pilha é uma lista linear em que todas as inserções, retiradas e todos os acessos são feitos em apenas um extremo da lista.

- Adotam a política **LIFO** (Last In, First Out) para manipulação de elementos;
- Inserções e retiradas são realizadas no topo.

Pilhas - Aplicações

- Ideal para estruturas aninhadas (verificação de parênteses, controle de sequências de subprogramas e etc.);
- Armazenar histórico de ações realizadas (páginas visitadas em um navegador, controle de destazer/refazer em um editor de testos);
- Utilizadas para implementar a recursividade (pilha de recursão).

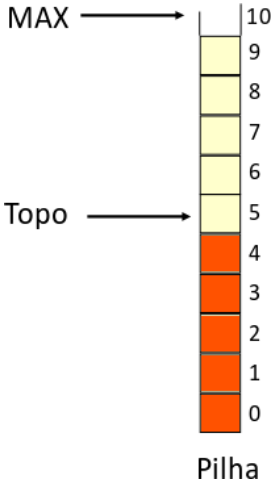
Pilhas - Operações básicas



Pilhas - Operações básicas

1. CriaPilhaVazia(S)
2. PilhaVazia(S)
3. Empilha(S,x)
4. Desempilha(S)

Pilhas - Implementação com arranjo



Pilhas - Implementação com arranjo

Algoritmo 1: CriaPilhaVazia

Entrada: Pilha S

```
1 início
2     S.topo ← 0
```

Algoritmo 2: PilhaVazia

Entrada: Pilha S
Saída: Booleano informando se S está vazia

```
1 início
2     se (S.topo = 0) então
3         retorna Verdadeiro
4     senão
5         retorna Falso
```

Adaptado de ??) e ??).

Pilhas - Implementação com arranjo

Algoritmo 3: Empilhar

Entrada: Pilha S, item x a ser empilhado

```
1 início
2     se (S.topo = S.tamanhoMáximo) então
3         Erro overflow: pilha cheia.
4     senão
5         S.itens[S.topo] ← x
6         S.topo ← S.topo + 1
```

Adaptado de **??**) e **??**).

Pilhas - Implementação com arranjo

Algoritmo 4: Desempilhar

Entrada: Pilha S
Saída: Elemento desempilhado

```
1 início
2     se (PilhaVazia(S)) então
3         retorna underflow: pilha vazia.
4     senão
5         x ← S.itens[ S.topo - 1]
6         S.topo ← S.topo - 1
7         retorna x
```

Adaptado de ??) e ??).

Implementação Estática
Filas

Filas

Fila

De acordo com **??**), uma fila é uma lista linear em que todas as inserções são realizadas em um extremo da lista, e todas as retiradas e acessos são realizados no outro extremo da lista.

- Adotam a política **FIFO** (First In, First Out) para manipulação de elementos;
- Inserções são realizadas no final da fila;
- Remoções são realizadas no início da fila.

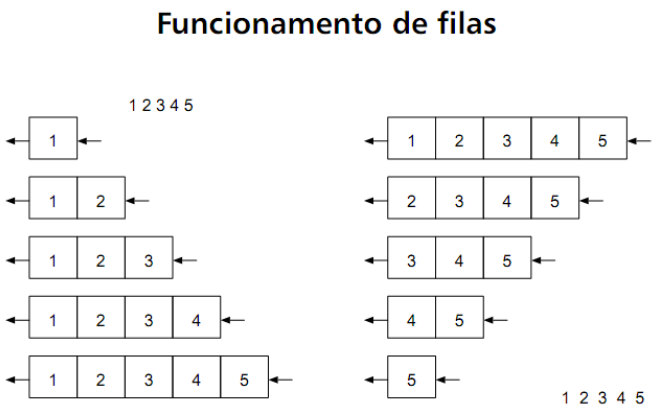
Filas - Aplicações

- Alocação de recursos para impressão de documentos em uma impressora;
- Atendimento de processos requisitados ao sistema operacional;
- Ordenação do encaminhamento de pacotes em um roteador;
- Buffer para gravação de dados em mídia.

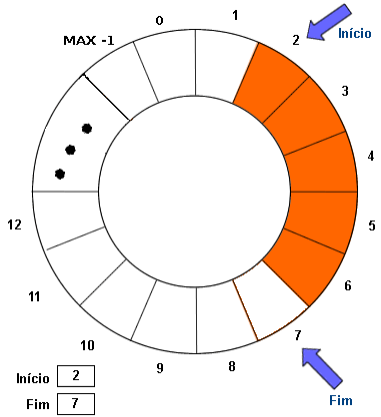
Filas - Operações básicas

1. CriaFilaVazia(Q)
2. FilaVazia(Q)
3. Enfileira(Q,x)
4. Desenfileira(Q)

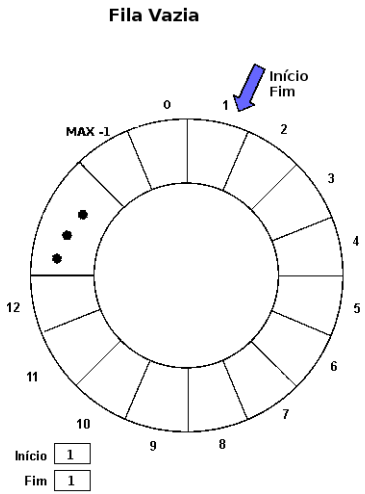
Filas - Operações básicas

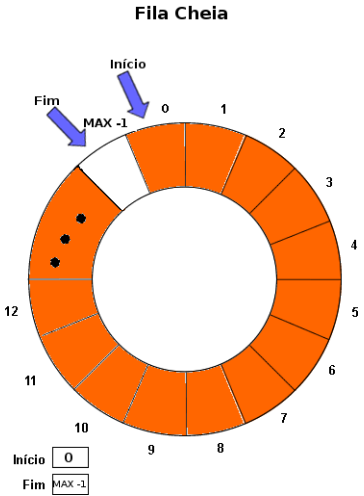


Filas - Implementação com arranjo



Filas - Implementação com arranjo





Fila - Implementação com arranjo

Algoritmo 5: CriaFilaVazia

Entrada: Fila Q

```
1 início
2     Q.início ← 0
3     Q.fim ← Q.início
```

Algoritmo 6: FilaVazia

Entrada: Fila Q.

```
1 início
2     se (Q.início = Q.fim) então
3         retorna Verdadeiro
4     senão
5         retorna Falso
```

Adaptado de **??**) e **??**).

Fila - Implementação com arranjo

Algoritmo 7: Enfileira

Entrada: Fila Q, item x

```
1 início
2     se ( (Q.fim) MOD (Q.tamanhoMáximo) + 1 = Q.início) então
3         Erro overflow: fila cheia.
4     senão
5         Q.itens[Q.fim] ← x
6         Q.fim ← (Q.fim) MOD (Q.tamanhoMáximo) + 1
```

Adaptado de ??) e ??).

Fila - Implementação com arranjo

Algoritmo 8: Desenfileira

Entrada: Fila Q
Saída: Elemento desenfileirado

```
1 início
2     se FilaVazia(Q) então
3         retorna Erro underflow: fila vazia.
4     senão
5         x ← Q.itens[Q.início]
6         Q.início ← (Q.início) MOD (Q.tamanhoMáximo) + 1
7         retorna x
```

Adaptado de ??) e ??).

Implementação Estática
Listas

Listas

Lista

Na lista, ao inserir ou remover elementos não existe nenhuma restrição, podendo realizar uma determinada operação em qualquer posição.

- Ao inserir um elemento em uma posição, é necessário deslocar os elementos para que nenhum elemento seja sobreposto.
- Ao remover um elemento é necessário deslocar os elementos de modo a ocupar a posição vazia.

Listas - Operações básicas

1. CriaListaVazia(L)
2. ListaVazia(L)
3. Inserir(L, x, p)
4. Remover(L, p)

Lista - Implementação com arranjo

Algoritmo 9: CriaListaVazia

Entrada: Lista L

```
1 início
2     L.primeiro ← 0
3     L.ultimo ← L.primeiro
```

Algoritmo 10: ListaVazia

Entrada: Lista L.

```
1 início
2     se (L.primeiro = L.ultimo) então
3         retorna Verdadeiro
4     senão
5         retorna Falso
```

Adaptado de **??**) e **??**).

Lista - Implementação com arranjo

Algoritmo 11: Inserir

Entrada: Lista L, elemento x, posição p

```
1 início
2     se ( (L.ultimo + 1) MOD L.tamanhoMáximo = L.primeiro ) então
3         Erro overflow: lista cheia.
4     senão
5         para ( i ← L.ultimo até p) faça
6             L.itens[ i+1 ] ← L.itens[ i ]
7         L.itens[ p ] ← x
8         L.ultimo = L.ultimo + 1
```

Adaptado de **??**) e **??**).

Lista - Implementação com arranjo

Algoritmo 12: Remove

Entrada: Lista L, posição p
Saída: Elemento removido

```
1 início
2     se ListaVazia(L) então
3         retorna Erro underflow: lista vazia.
4     senão
5         x ← L.itens[ p ]
6         para ( i ← p até L.ultimo ) faça
7             L.itens[ i ] ← L.itens[ i+1 ]
8         L.ultimo = L.ultimo - 1
9         retorna x
```

Conclusão

Conclusão

Material de apoio

Animações das operações disponíveis em http://www.ime.usp.br/~nelio/ensino/2002-1/ed/

Próxima aula

Pilha, Fila e Lista com alocação dinâmica.

Algoritmos e Estrutura de Dados II

Pilha, Fila e Lista
(Alocação Estática)

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia