Estruturas de Dados I

Introdução

Prof. Bruno Azevedo

Instituto Federal de São Paulo

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Catanduva

O que são Estruturas de Dados?

- Conjuntos de dados são dinâmicos.
- Os manipulamos com algoritmos. Estes podem crescer, encolher e mudar com o tempo.
- Diversas operações podem ser feitas em conjuntos dinâmicos de dados, tais como: busca, inserção, deleção, mínimo e máximo.

Definição

Uma estrutura de dados é um formato para organização, gerenciamento, e armazenamento de dados.

- Estruturas de dados servem para organizar os dados de um problema de modo que eles possam ser processados de modo eficiente.
- Eficiência se relaciona com escalabilidade, ou seja, como o tempo de processamento cresce quando a quantidade de dados do problema aumenta.

Caixa de Ferramentas

- Se temos conjuntos de dados dinâmicos que queremos manipular, o primeiro passo é conhecer suficientes estruturas de dados, de modo que tenhamos uma **caixa de ferramentas** para utilizarmos.
- E assim, poderemos escolher as estruturas que melhor encaixem com o problema que temos em mão.
  - Se temos uma caixa de ferramentas pequena, ficaremos restritos as ferramentas que temos.
  - Pode ser que nenhuma delas seja realmente eficiente para o problema.
  - Talvez estejamos usando uma chave de fenda com um parafuso no qual uma chave philips seria mais adequada.
  - Mas ter uma caixa de ferramentas grande, com muitas ferramentas, não é suficiente.
  - Não é útil se não soubermos usar a ferramenta, ou se nos falta prática em seu uso.
- Portanto, quanto mais ferramentas tivermos, e quanto melhor conhecermos cada ferramenta, mais fácil será encontrar uma ferramenta adequada para nosso problema.

Apenas Estruturas de Dados?

- Mass não são apenas as estruturas de dados que devemos considerar. Também devemos considerar os **algoritmos utilizados**.
- Uma estrutura de dados simples, que vocês conhecem como o **array** (vetor) pode ser utilizado de um modo mais, ou menos, eficiente, dependendo do algoritmo utilizado.
- Se temos um problema de ordenação, utilizando a mesma estrutura (array), podemos chegar a resultados de diferentes eficiências dado o algoritmo utilizado.
- Portanto se conhecermos muitas estruturas de dados e muitos algoritmos que podem ser usados nessas estruturas de dados, maior a nossa chance de encontrarmos soluções eficentes
- Ou seja, a nossa caixa de ferramentas **não deve conter apenas estruturas, mas também algoritmos**.

Qual Estrutura Usar?

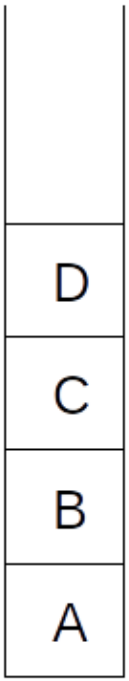
- Com o uso da estrutura de dados adequada, o problema pode:
  - Pode ser resolvido mais rápido.
  - Pode ser mais fácil de implementar.
  - Pode ser mais escalável.
- Vocês precisam ter uma caixa de ferramentas grande o suficiente e, com a prática, aprender a identificar quais são as mais eficientes para os problemas que irão encontrar.

Pilhas e Filas

- **Pilhas** e **Filas** são conjuntos dinâmicos no qual o elemento removido do conjunto é predefinido.
- Em uma pilha, o elemento removido é sempre o que foi inserido mais recentemente.
- Ou seja, a pilha implementa a estratégia **LIFO** – Last In First Out: o último que entra é o primeiro que sai.
- Em uma fila, o elemento removido é sempre o que está mais tempo no conjunto; o mais antigo.
- A fila implementa a estratégia **FIFO** – First In First Out: o primeiro que entra é o primeiro a sair.

Pilhas

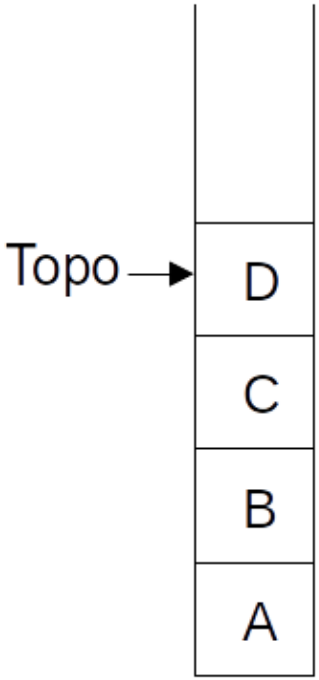
- Imaginem uma pilha de itens dentro de um recipiente de mesma largura.
- Apenas o item no topo é acessível.
- Para acessarmos qualquer outro item precisamos remover os itens que se encontram em cima dele.



- Podemos pensar em diversos exemplos reais, como uma pilha de pratos.

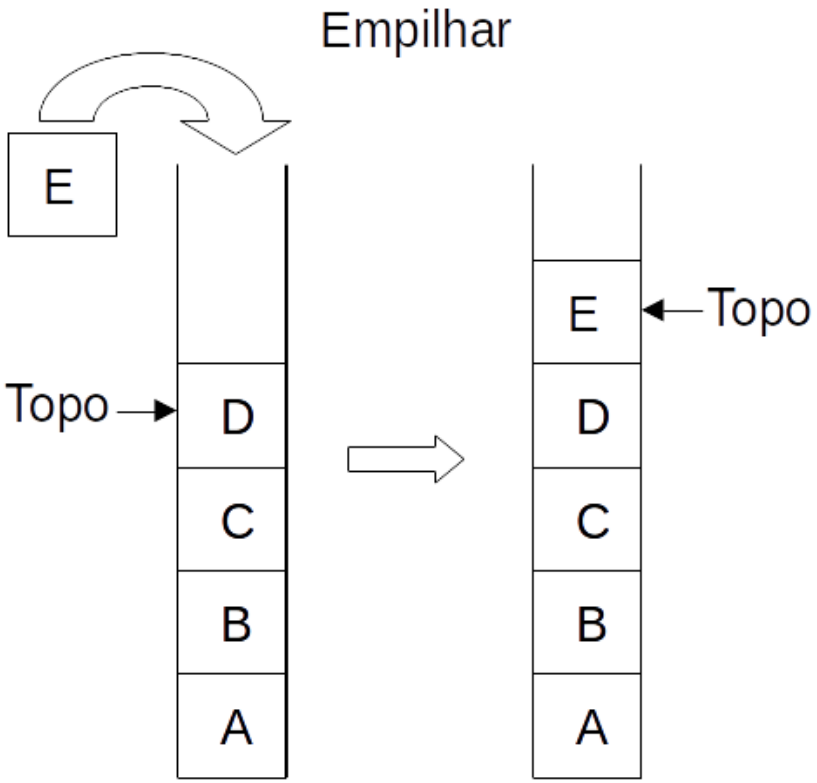
Pilhas

- Controlamos qual é o topo da pilha, para sempre sabermos qual item remover.



Pilhas

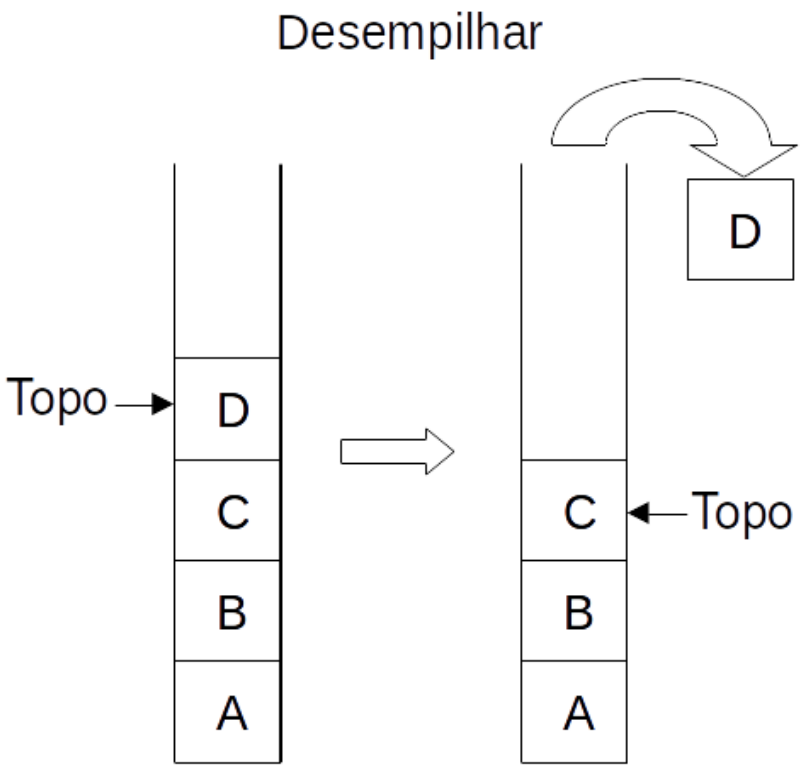
- A operação de inserção de um novo item em uma pilha é geralmente chamada de **empilhar** (ou push).



- Prestem atenção na mudança do **topo**. A operação de empilhar muda o topo para o elemento que foi inserido.

Pilhas

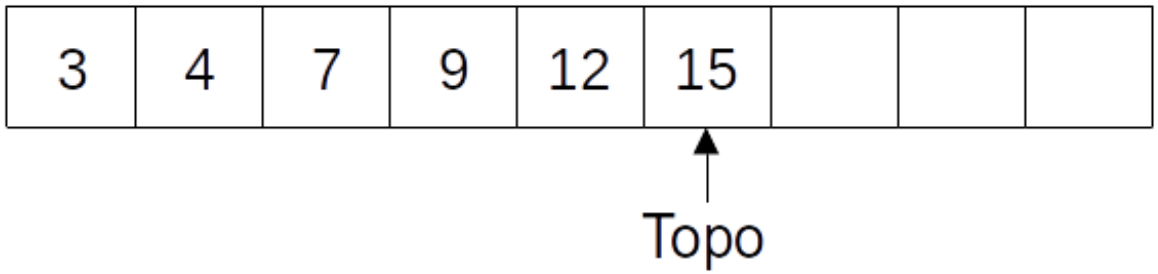
- A operação de remoção de um item em uma pilha é geralmente chamada de **desempilhar** (ou pop).



- Novamente, prestem atenção na mudança do **topo**. A operação de desempilhar muda o topo para o elemento que estava embaixo do elemento removido.

Pilhas

- Existem vários modos de se implementar pilhas, podemos até usar um simples vetor.
- Abaixo, temos uma pilha de capacidade máxima de 9 elementos implementada utilizando um vetor.



- Na posição 5 do vetor temos o **topo** da pilha.

Pilhas

- Vamos conhecer as operações da pilha implementadas em pseudo-código.
- Isso é interessante, porque estruturas de dados não pertencem a nenhuma linguagem.
- Elas são **abstrações** que podemos implementar na linguagem de nossa preferência.

Pilhas

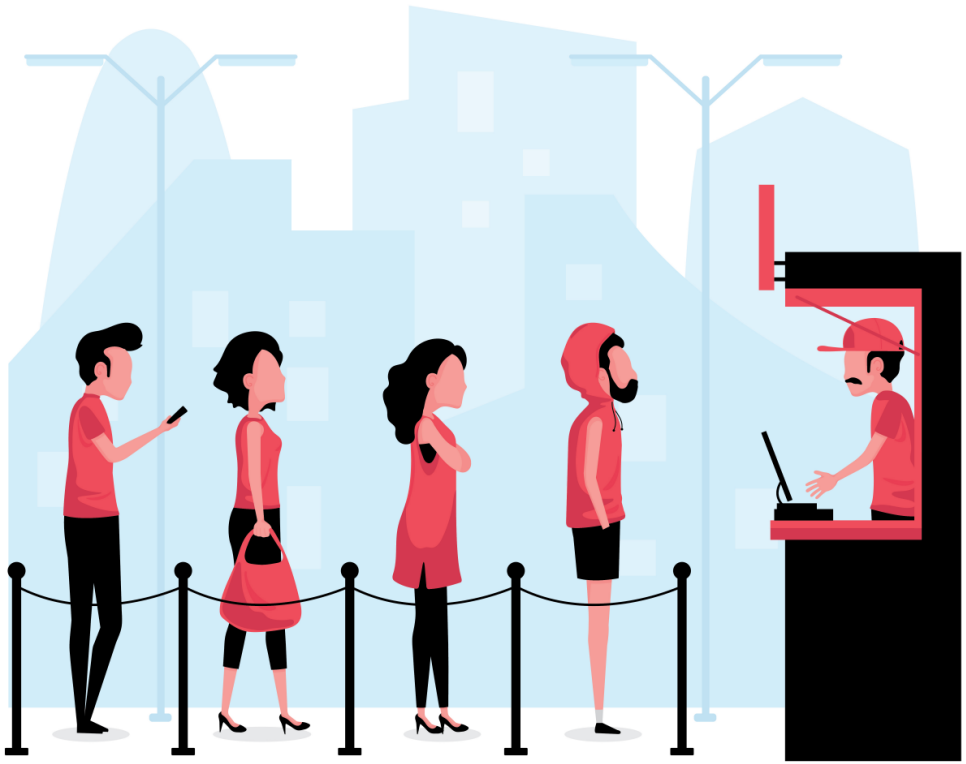
```
EMPILHAR(P, TopoPilha, topoMax, elemento)
    if TopoPilha = topoMax then
        "Erro de Overflow"
    else
        TopoPilha ← TopoPilha + 1
        P[TopoPilha] ← elemento
```

```
DESEMPILHAR(P, TopoPilha)
    if TopoPilha = -1 then
        "Erro de Underflow"
    else
        TopoPilha ← TopoPilha - 1
        return P[TopoPilha+1]
```

- topoMax indica o tamanho máximo da nossa implementação de pilha. Ou seja, o índice do último elemento **possível** da pilha.
- Por exemplo, se n = 9, e nosso índice inicia em zero, topoMax = 8.
- Se adicionamos em uma pilha cheia (TopoPilha = topoMax), teremos um erro de *overflow*.
- Quando TopoPilha = -1, a pilha está vazia.
- Se removemos de uma pilha vazia, temos um erro de *underflow*.

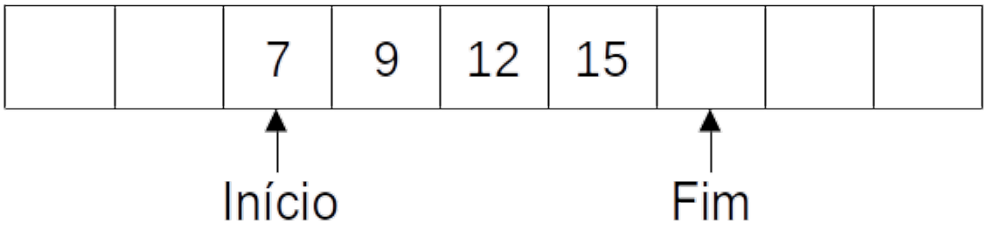
Filas

- Visualizem uma fila de pessoas.
- A primeira pessoa a sair da fila é a primeira que entrou na fila. A próxima pessoa a sair da fila é a segunda que entrou na fila.
- Portanto, a pessoa que está a mais tempo na fila será quem sairá **primeiro**.



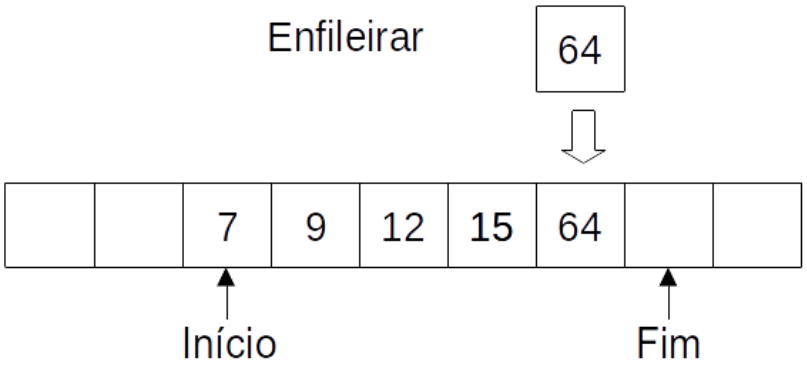
Filas

- Controlamos o início e o fim da fila, para sabermos qual elemento remover primeiro e onde inserir novos elementos, respectivamente.



Filas

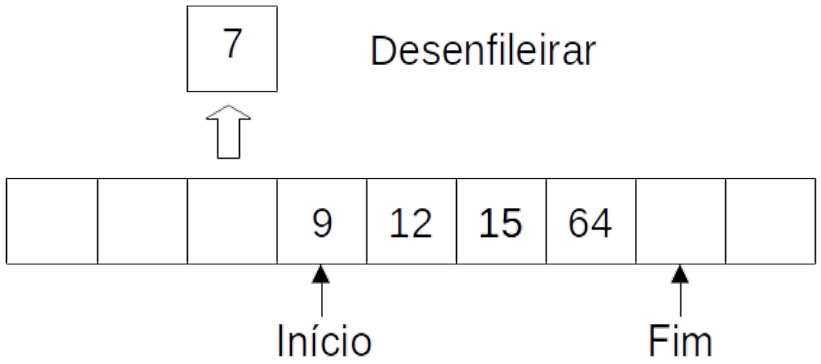
- A operação de inserção de um novo item em uma pilha é geralmente chamada de **enfileirar** (ou enqueue).



- Ocorre a mudança do **fim** da fila. A operação de enfileirar muda o fim para a próxima posição livre.

Filas

- A operação de remoção de um item em uma pilha é geralmente chamada de **desenfileirar** (ou dequeue).



- Ocorre a mudança do **fim** da fila. A operação de desenfileirar muda o início para o próximo item mais antigo da fila.

Filas

- Existem vários modos de se implementar filas, e como vimos anteriormente, podemos usar um vetor.
- Vamos conhecer as operações da fila implementadas em pseudo-código.

Filas

```
ENFILEIRAR(F, inicio, fim, comp, elemento)
    F[fim] ← elemento
    if fim = comp - 1 then
        fim = 0
    else
        fim = fim + 1
```

```
DESENFILEIRAR(F, inicio, comp)
    elemento = F[inicio]
    if inicio = comp - 1 then
        inicio = 0
    else
        inicio = inicio + 1
```

- inicio e fim indicam a posição inicial e final da fila, respectivamente.
- comp representa o comprimento da fila, ou seja, o tamanho do vetor.
- Em enfileirar, se fim = comp - 1 estamos inserindo na última posição do vetor, e portanto o fim mudará para a primeira posição do vetor.
- Em desinfileirar, se inicio = comp - 1, o primeiro elemento estava na última posição, e como essa é uma implementação circular, o próximo estará na primeira posição do vetor.
- Existe algum problema nesse pseudo-código?

Exercícios (16)

⇒ Pilhas

- Implemente duas pilhas em um vetor de tamanho n de modo que nenhuma das duas pilhas overflows a não ser que o número total de elementos em ambas as pilhas for n. Importante: as operações de empilhar e desempilhar executam em tempo constante (O(1)), mantenha isso em sua implementação.
- Implemente uma calculadora pós-fixa utilizando a estrutura de dados **pilha**. Em uma calculadora que utiliza a notação pós-fixa, os operadores vem depois dos operandos (ex: 64 128 +). Deve efetuar, no mínimo, as operações de adição, subtração, multiplicação e divisão.
- Implemente um programa utilizando a estrutura de dados **pilha** que verifique se uma expressão matemática (uma sequência de caracteres) possui parênteses balanceados. Ou seja, cada parêntese de abertura "(" deve ser correspondido por um parêntese de fechamento ")" na ordem correta.

Exercícios (17)

⇒ Filas

- Reescreva ENFILEIRAR e DESINFILEIRAR para detectar underflow e overflow.
- Um **Deque**, ou fila duplamente terminada, permite a inserção e deleção em ambos os lados. Implemente quatro procedimentos de tempo constante para inserir e deletar elementos de ambos os lados de um deque. O deque deve ser implementado com um vetor.
- Implemente um programa que simule uma fila de impressão considerando a prioridade de impressão de cada arquivo. Arquivos com prioridade alta são impressos antes dos arquivos com prioridade média e baixa. Dentro de cada prioridade, mantenha a ordem de chegada.

  Este programa deve possuir as seguintes funcionalidades:
  - Adicionar arquivos à fila de impressão com suas respectivas prioridades (alta, média, baixa).
  - Imprimir o próximo arquivo da fila, considerando as prioridades e a ordem de chegada.
  - Verificar o tamanho atual da fila de impressão.
- Implemente um programa utilizando a estrutura de dados **fila** que receba uma frase como entrada e a imprima de trás para frente. Dica: use a função strtok da biblioteca string.h.