# TADs lineares[1][2]

## Parte I: Pilhas e filas

Pilhas e filas são tipos abstratos de dados (TADs) lineares que se distinguem pelas políticas usadas para controlar a inserção e a remoção de elementos. A política LIFO (do inglês *last in, first out*) usada por pilhas estabelece que os elementos mais recentemente adicionados têm prioridade sobre os mais antigos. Assim, em qualquer instante da existência de uma pilha, o único elemento diretamente acessível é o último elemento adicionado. De fato, para ter acesso aos elementos mais antigos da pilha é necessário *desempilhar* os que se encontram acima deste. Por sua vez, filas implementam a política FIFO (do inglês *last in, first out*), isto é, os elementos mais antigos têm prioridade sobre os mais recentes. Assim, a inserção e a remoção de elementos em uma fila são realizadas em extremidades opostas. No entanto, listas comumente oferecem acesso somente ao elemento mais antigo (o próximo a sair da lista).

## 1 Operações principais

A implementação tradicional de pilhas fornece apenas três operações: empilhar/desempilhar elementos e visualizar o elemento atualmente no topo da pilha. Ainda que o conceito de uma pilha remeta a elementos sendo armazenados verticalmente, na prática pilhas são tipicamente implementadas através de estruturas lineares como vetores e listas encadeadas, com a restrição adicional de só haver inserção e remoção em seu final. Assim, visualizar o elemento no topo da pilha é simplesmente visualizar o último elemento da estrutura. Analogamente, empilhar ou desempilhar um elemento significa, respectivamente, adicioná-lo ao fim da estrutura ou removê-la do fim da mesma.

Assim como as pilhas, as filas também oferecem apenas operações de inserção, remoção e a possibilidade de visualizar o elemento mais próximo de ser removido (neste caso, o mais antigo). Em termos de implementação, filas podem ser implementadas usando (i) listas encadeadas ou (ii) armazenamento circular em um vetor. A principal limitação da segunda alternativa se dá quando a quantidade de elementos da fila atinge a capacidade máxima originalmente escolhida, sendo necessário redimesioná-la a um custo computacional linear.

## 2 Análise da complexidade

Uma vez que pilhas são tradicionalmente implementadas usando vetores, que por sua vez oferecem o acesso à qualquer posição em tempo constante, todas as operações envolvendo pilhas também apresentam complexidade de tempo constante. Em termos de espaço, a única informação adicional que necessita ser armazenada é o tamanho atual da pilha, o que significa complexidade constante. Analogamente às pilhas, a implementação de filas baseadas em vetores garante operações em tempo constante. No entanto, o redimensionamento de uma fila é uma operação de custo computacional linear. No caso de uma fila implementada através de uma lista encadeada, todas as operações podem ser realizadas em tempo constante desde que se armazene uma sentinela apontando

[1] Roteiro de estudo fornecido na disciplina de Estruturas de Dados Básicas 1 (EDB1) da Universidade Federal do Rio Grande do Norte (UFRN).

[2] Autor: Leonardo Bezerra.

para o último elemento da fila. Em ambas as implementações, a complexidade em termos de espaço das filas é constante.

# 3 Exemplos de aplicação

Ainda que conceitualmente simples, a aplicabilidade de pilhas é enorme na prática devido ao elevado número de situações que requerem o respeito à política LIFO, dentre as quais destacamos duas:

**Navegação de históricos:** sistemas com botões *voltar* (*browsers*) ou *desfazer* (editores de texto);

**Implementações recursivas:** compiladores usam pilhas para armazenar o contexto de cada nível de recursão e posteriormente executar o *backtracking* de forma correta;

Da mesma forma, a aplicabilidade prática de filas está diretamente relacionada à política FIFO:

**Requisições,** quer em um servidor web, sistema operacional ou qualquer outro contexto onde haja uma fila de solicitações;

**Buffers**, particularmente em contextos de transmissões de dados (*streaming*);

# 4 Exercícios sugeridos

[3]

1. Implemente uma pilha que, além de permitir empilhar e desempilhar elementos, consegue retornar o elemento mínimo. Todas as operações deverão apresentar complexidade de pior caso constante.
2. Imagine uma pilha (literal) de pratos. Se a pilha ficar muito alta, ela poderá tombar. Assim, no mundo real, nós gostaríamos de começar uma nova pilha quando a pilha anterior excede um determinado limite de elementos. Implemente um TAD chamado ConjuntoDePilhas que possua este comportamento. Este TAD deverá ser composto de várias pilhas, criando uma nova pilha a cada vez que a pilha atual exceda um limite. No entanto, as funções de empilhar e desempilhar devem ter comportamento idêntico à de pilhas (identificar a pilha atual e realizar as operações sobre ela).
3. Implemente um algoritmo para solucionar o problema da Torre de Hanói sem recursão (usando uma pilha).
4. Implemente um programa que leia uma sequência de caracteres e determine se os parênteses, colchetes e chaves presentes na sequência estão balanceados.
5. Implemente:
   (a) uma fila com duas pilhas. Qual a complexidade das operações de adicionar e remover elementos?
   (b) uma pilha com duas filas. Qual a complexidade das operações de empilhar e desempilhar?

[3]Fonte: MCDOWELL, Gayle Laakmann. **Cracking the coding interview**. CarrerCup, 2011.
BEGON, Jean-Michel. **Advanced computer programming. Exercise session 3: Stack, queue, list, vector and sequence**. 2014.