Programação 1: Pilhas e Filas

Prof. André Grégio

Introdução

Pilhas e filas são estruturas de dados elementares usadas para se representar conjuntos dinâmicos cujas operações de inserção e remoção possuem posições pré-definidas. Tais estruturas possuem operações simples e podem ser consideradas **listas especiais** nas quais as ações são feitas sobre o k-ésimo elemento, seja este o elemento que está no topo da estrutura (pilha) ou é apontado pelo primeiro/último elemento (fila).
Antes de discutir os tipos abstratos de dados e algoritmos relacionados a essas estruturas, é importante perceber que muitas aplicações computacionais reais do dia-a-dia as aplicam:

- **Navegador Web**: quando se visita mais de uma página, o botão "voltar" é habilitado, permitindo que se visite novamente as páginas anteriores (pela ordem da última a ser visitada, caminhando em direção à primeira).
- **Editor de texto**: após iniciar um texto, as N últimas alterações podem ser desfeitas com o botão *undo*, obedecendo também à ordem "última alteração feita, primeira desfeita".
- **Roteadores**: os pacotes que compõem o tráfego de rede devem ser organizados e entregues na ordem em que chegam.
- **Servidores de transações**: clientes de um *site* de *e-commerce* têm de ter suas compras processadas na ordem em que são pedidas, para garantir justiça e estoque.

Portanto, a compreensão das estruturas de dados "pilha" e "fila" são fundamentais para a resolução de problemas criados por necessidades de sistemas reais (e críticos), como manter a ordem de impressões dos usuários em uma impressora compartilhada, organizar pousos e decolagens, gerenciar transações de operadores de cartões, alocar variáveis criadas por funções em código e muitos outros, além dos exemplos já citados anteriormente.

Pilha

A pilha, ou *stack*, é uma estrutura onde o elemento a ser removido do conjunto é sempre o mesmo: aquele que foi inserido por último. Esse tipo de política de remoção é conhecida como LIFO, isto é, *last in, first out* (o último que entra é o primeiro que sai).
Em uma pilha, a operação de remoção é conhecida como **POP** e, como obedece à política LIFO, não é necessário passar nenhum argumento a esta função. A operação de inserção é conhecida como **PUSH**, que acrescenta um elemento ao topo da pilha. Pode-se comparar a estrutura pilha a um conjunto de bandejas empilhadas no restaurante universitário: tem-se acesso à bandeja que está no topo; uma nova bandeja é colocada sobre a que está no topo, transformando-se no novo topo; a bandeja removida é a que está no topo e, portanto, a última a ter sido empilhada.

TAD Pilha

O Tipo Abstrato de Dados "Pilha" possui um atributo para representar o elemento que está em seu **topo** e três operações básicas: `pilhaVazia`, que verifica se há ou não elementos em uma dada pilha; `PUSH`, para inserir um novo elemento na pilha; `POP`, para remover o elemento que se encontra no topo da pilha.

Pilha usando apontadores

A abordagem dinâmica consiste na implementação de pilhas usando **listas encadeadas**, isto é, a estrutura nó e apontadores para os próximos elementos. A característica da pilha com apontadores (quando implementada como uma lista) é que tanto a operação de **inserção** quanto a de **remoção** ocorrem na cabeça, aqui chamada

de topo. Recriar essas operações com apontadores torna-se simples, como mostrado nos exemplos a seguir, que consideram `P` uma estrutura pilha e `x`.

1. `PUSH(P, x)`
   - Cria-se um novo nó (`novo`) com valor "x".
   - `novo.prox = P.topo;`
   - `P.topo = novo;`
2. `POP(P)`
   - `y = P.topo->valor;`
   - `P.topo = P.topo->prox;`
   - Libera-se a memória do nó que estava no topo;
   - Retorna y.

Aplicação de pilhas

1. Transformação da notação infixa para pós-fixa:
   - Leia os dados de entrada, um por vez.
   - Se for um operando, escreva na saída.
   - Se for um operador e pilha vazia, PUSH.
   - Se pilha não vazia, POP dados com prioridade maior ou igual ao encontrado antes de PUSH o operador novo.
   - Se for um '(', PUSH.
   - Se for um ')', POP até '('.
   - Após ler todos os dados, POP até o último elemento da pilha.
   - Precedência crescente: '+','-','*','/'.
   - Ex.: $3 + (5 * 2)$ é convertido para $352 * +$.
2. Avaliação de expressões aritméticas.
   - Algoritmo de duas pilhas de Dijkstra (anos 1960):
     (a) PUSH *operandos* na pilha de operandos.
     (b) PUSH *operadores* na pilha de operadores.
     (c) Ignore parênteses esquerdos.
     (d) Ao encontrar ')':
       – POP um operador;
       – POP o número correto de operandos;
       – PUSH o resultado na pilha de operandos.

Fila

A fila, ou *queue*, também é uma estrutura onde o elemento a ser removido do conjunto é sempre o mesmo: aquele que foi inserido primeiro. Esse tipo de política de remoção é conhecida como FIFO, isto é, *first in, first out*.
Em uma fila, a operação de remoção é conhecida como **DEQUEUE** e, como obedece à política FIFO, não é necessário passar nenhum argumento a esta função. A operação de inserção é conhecida como **ENQUEUE**, que acrescenta um elemento ao fim da fila. Pode-se comparar a estrutura fila a um conjunto de pessoas esperando para pegar comida no restaurante universitário: a pessoa que chegou primeiro e, portanto, está na frente da fila, será atendida primeiro; se uma nova pessoa entra na fila, ela passa a ser a última a ser atendida, isto é, está no fim da fila.

TAD Fila

O Tipo Abstrato de Dados "Fila" pode possuir três atributos principais para representar o elemento que está no início da fila (**cabeça** ou *head*), o elemento que está no fim da fila (**cauda** ou *tail*) e o comprimento de uma fila com n elementos (**tamanho**). As operações básicas sobre filas também são três: `filaVazia`; `ENQUEUE` para inserir um novo elemento na cauda; `DEQUEUE` para remover o elemento da cabeça.

Fila usando apontadores

Similar às pilhas, a abordagem dinâmica para a construção de filas consiste na implementação usando listas encadeadas contendo nós e apontadores para os próximos elementos. As peculiaridades da fila com apontadores são que a operação de **inserção** ocorre na cauda e a **remoção** na cabeça. A seguir, as direções para se implementar as operações citadas usando nós e apontadores, considerando uma estrutura de fila `F`.

1. `ENQUEUE(F, x)`
   - Cria-se um novo nó com valor "x" (ex.: `novo->valor = x`).
   - `F.cauda->prox = novo;`
   - `F.cauda = novo;`
2. `DEQUEUE(F)`
   - `y = F.cabeca->valor;`
   - `F.cabeca = F.cabeca->prox;`
   - Libera-se a memória do nó que estava na cabeça;
   - Retorna y.

Aplicação de filas

1. Tamanho da fila do processador (*Processor Queue Length*):
   - Coleção de *threads* prontas para executar, mas que não o fazem devido a uma outra *thread* ativa em execução no processador.
2. Processamento de *buffers* de mensagens:
   - WhatsApp
     - ≈50 bilhões de mensagens/dia.
     - Entrega das mensagens mais antigas primeiro no cliente.
     - Gerenciamento do tamanho da fila de mensagens no servidor.

Revisão

1. **Pilha**: também chamada de *stack*.
   - Inserção feita no topo (`PUSH`);
   - Remoção do elemento mais recente (`POP`, não precisa de argumento);
   - Política LIFO (*last in, first out*).
2. **Fila**: também chamada de *queue*.
   - Inserção feita na cauda (`ENQUEUE`);
   - Remoção do elemento mais antigo (`DEQUEUE`, não precisa de argumento);
   - Política FIFO (*first in, first out*).