

Computer science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra

Laboratório - Visão Geral TADs

Este laboratório é composto por exercícios selecionados das listas. Para algumas questões, o código do TAD apropriado é disponibilizado e é seu dever entendê-lo.

TAD Filas:

1. Um **deque** é um conjunto de itens a partir do qual podem ser eliminados e inseridos itens em ambas as extremidades. Chame as duas extremidades de um deque **esq** e **dir**. Como um deque pode ser representado como um vetor em C? Estenda a implementação da fila estática (disponível no AVA) e implemente as quatro funções seguintes:

RemDir, RemEsq, InsDir, InsEsq,

para remover e inserir elementos nas extremidades esquerda e direita de um deque, respectivamente.

Certifique-se de que as funções funcionem corretamente para o deque vazio e detectem o estouro e o underflow (tentativa de remoção quando a fila está vazia). Quais as desvantagens dessa implementação com relação a implementação por encadeamento/alocação dinâmica?

TAD Pilhas:

2. Desenvolva um método para manter duas pilhas dentro de um único vetor linear (um arranjo) de modo que nenhuma das pilhas incorra em estouro até que toda a memória seja usada, e toda uma pilha nunca seja deslocada para outro local dentro do vetor.

3. Utilizando uma das implementações de pilha disponíveis no AVA, escreva um programa para classificar uma pilha em ordem crescente.

Você não deve fazer nenhuma suposição sobre como a pilha é implementada. Isto é, seu programa deverá funcionar independentemente da forma que a pilha foi implementada.

As únicas funções que devem ser usadas para escrever este programa são as que estão no `pilha.h`: `Empilha` | `Desempilha` | `vazia`.

TAD Árvore:

4. Utilizando a implementação de árvore binária de busca (arvores.zip), escreva a função:

```
int ehBalanceada(ArvBin * raiz)
```

Retorna se a árvore é balanceada caso a diferença entre a subárvore direita e esquerda não seja maior que 1.

5. Escreva funções não recursivas para realizar os 3 tipos de percurso na árvore binária (dica: use uma pilha):

(a) pré-ordem

(b) em-ordem

(c) pós-ordem

Observação: Utilize a mesma implementação de árvore e pilha disponibilizada.

6. (FORTE CANDIDATA para PROVA) Você tem duas árvores binárias muito grandes: T1, com milhões de nós, e T2, com centenas de nós. Crie um algoritmo para decidir se T2 é uma subárvore de T1.

Você pode utilizar o mesmo código de árvore. Neste caso, deve-se criar uma função `int ehIgual(ArvBin * T1, ArvBin * T2)`.

Utilize os arquivos em input.zip para testar a sua solução. Diga se:

A) input-rand-1000-sub é uma subárvore de input-rand-1000.

B) input-rand-10000-sub é uma subárvore de input-rand-10000.

Dica: Você deve modificar o main para que carregue as duas árvores