CI1057 - Algoritmos e Estruturas de Dados III

Lista sobre Árvores Binárias
Segundo Semestre de 2024

Prazo de Entrega: 11/outubro/2024, 23:59h
A entrega é **opcional**.
A nota da lista soma até 10 pontos na nota da primeira prova.

Complete os programas `tadArvBin.c` e `clienteArvBin.c` com as funções especificadas abaixo. O programa tadArvBin.c pode utilizar o TAD Fila (com itens do tipo Apontador) para fazer a inserção dos itens por nível.

Observação importante:

- Você **não pode** alterar a função `main` do arquivo `clienteArvBin.c`. Uma parte da correção será a execução de um `diff` da saída do seu programa com a solução.
- Você pode considerar que a entrada para os itens da árvore são valores no intervalo [1,999]

Entrega:
O trabalho é individual. A entrega deve ser feita por email, com o assunto "CI1057-2s2024 - Lista 1", para o endereço:

- carmemhara@ufpr.br

Sugiro que o email seja enviado da sua conta @ufpr.br. No semestre passado alguns alunos enviaram do endereço @inf.ufpr.br e houve problemas na entrega.

Forma de entrega:

- Deve ser enviado por email um único arquivo **GRR20xxx.tar.gz**:
  - ao ser descompactado, ele deve gerar um diretório chamado GRR20xxx
  - este diretório deve conter apenas os arquivos `tadArvBin.c` e `clienteArvBin.c`
- o trabalho será compilado com o comando:
  `gcc clienteArvBin.c tadArvBinInt.c tadFilaApontador.o -lm -o clienteArvBin`
- o trabalho será executado com a seguinte linha de comando:
  `./clienteArvBin < entrada.txt > saida.txt`
- a verificação do resultado será feita com:
  `diff saida.txt saida_solucao.txt`

Exercício 1: Função `insereArvLista` no arquivo tadArvBin.c
A função deve gerar uma árvore "degenerada", na qual todos os nodos tem apenas o filho esquerdo preenchido. Ou seja, a função gera uma árvore que tem o formato de uma lista.

Exercício 2: Função `insereArvNivel` no arquivo tadArvBin.c
A função deve gerar uma árvore balanceada, fazendo a inserção por nível. A árvore resultante é sempre uma árvore completa ou quase completa.

Exercício 3: Função `maiorParMaiorImpar` no arquivo clienteArvBin.c
A função retorna como resultado os parâmetros `maiorPar` e `maioImpar` preenchidos com o maior valor par e o maior valor impar dentre os itens armazenados na árvore.

Exercício 4: Função `ehDeBusca` no arquivo clienteArvBin.c
A função retorna 1 se a árvore binária é uma *árvore binária de busca* e 0, caso contrário.

Exercício 5: Função `espelha` no arquivo clienteArvBin.c
A função retorna a raiz da árvore modificada, de forma que as subárvores esquerda e direita são trocadas.

Exercício 6: função `paiMenor` no arquivo clienteArvBin.c
Altera os valores armazenados na árvore de forma que cada nodo contenha o menor valor dentre o nodo

corrente e os seus filhos. A alteração deve ser feita das folhas para a raiz. Observe que no final a raiz vai conter o menor valor armazenado na árvore.

Exercício 7: Função `novoPai` no arquivo clienteArvBin.c
Dobra a quantidade de nodos da árvore criando para cada nodo n um novo pai, da seguinte forma:

- se o nodo n contiver um valor par, ele será o filho esquerdo do seu novo nodo pai, contendo o dobro do seu valor;
- se o nodo n contiver um valor ímpar, ele será o filho direito do seu novo nodo pai, contendo o dobro do seu valor mais um.

Exemplo de Execução 1:

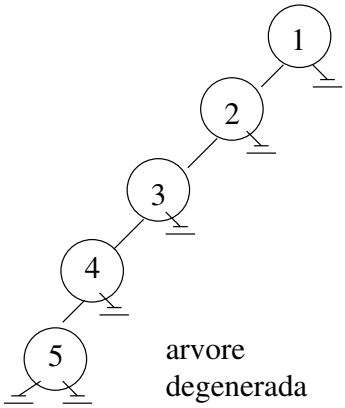
```
Insira uma sequencia de chaves. Finalize a sequencia com 0:
1 2 3 4 5 0
Arvore Degenerada
(1(2(3(4(5()())()())()())()))
Altura = 5
Nao eh uma arvore de busca
Insira uma sequencia de chaves. Finalize a sequencia com 0:
4 2 5 1 3 0
Arvore Balanceada
(4(2(1()())(3()()))(5()()))
Altura = 3
Eh uma arvore de busca
Maior par = 4
Maior impar = 5
Arvore espelhada
(4(5()())(2(3()())(1()())))
Arvore com pai menor
(1(5()())(4(3()())(2()())))
Arvore dobrada
(3()(1(11()(5()()))(8(4(7()(3()()))(4(2()())()))())))
```

Exemplos de Execução 2:

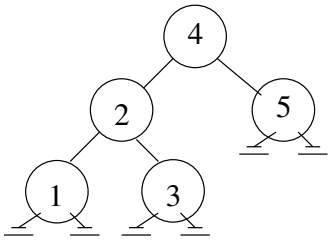
```
Insira uma sequencia de chaves. Finalize a sequencia com 0:
10 5 20 3 7 15 25 1 4 6 9 0
Arvore Degenerada
(10(5(20(3(7(15(25(1(4(6(9()())()())()())()())()())()())()())()())()))
Altura = 11
Nao eh uma arvore de busca
Insira uma sequencia de chaves. Finalize a sequencia com 0:
10 5 20 3 7 15 25 1 4 6 9 0
Arvore Balanceada
(10(5(3(1()())(4()()))(7(6()())(9()())))(20(15()())(25()())))
Altura = 4
Eh uma arvore de busca
Maior par = 20
Maior impar = 25
Arvore espelhada
(10(20(25()())(15()()))(5(7(9()())(6()()))(3(4()())(1()()))))
Arvore com pai menor
(1(15(25()())(20()()))(10(6(9()())(7()()))(5(4()())(3()()))))
Arvore dobrada
```

```
(3()(1(31()(15(51()(25()()))(40(20()())()))))(20(10(12(6(19()(9()()))
(15()(7()()))))())(11()(5(8(4()())())(7()(3()()))))))()))
```

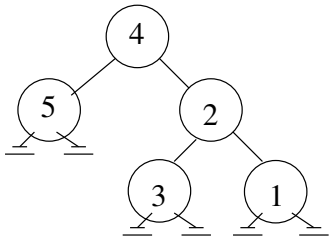
Árvores do Exemplo 1:



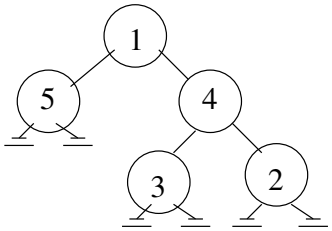
arvore degenerada



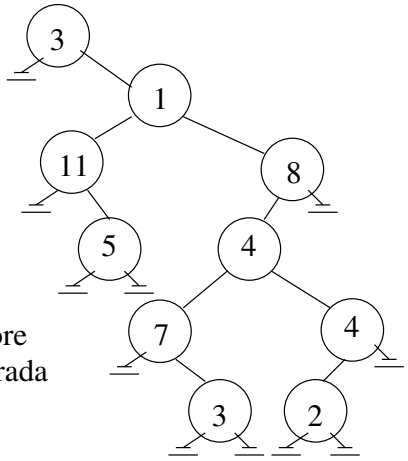
arvore balanceada



arvore espelhada



pai menor



arvore dobrada