Universidade Federal de Campina Grande
Centro de Engenharia Eletrica e Informática
Departamento de Sistemas e Computação
Graduação em Ciência da Computação

2ª prova. Implementação estendida de BST

Atividades necessárias antes de iniciar a prova:

1. Crie um projeto no Eplipse chamado LEDA, por exemplo (pode ser qualquer outro nome que lhe convier);
2. Descompacte o arquivo baixado (exceto o PDF) na pasta dos fontes (normalmente **src**) do seu projeto LEDA criado no seu workspace. O arquivo baixado tem a seguinte estrutura:
   - adt
   --bst
   ---BST.java (INTERFACE COM PEQUENAS MODIFICACOES)
   ---BSTNode.java (INTERFACE)
   ---BSTImpl.java (IMPLEMENTAÇÃO PREVIA DE BST COM PEQUENAS MODIFICACOES)
   --bst
   ---extended
   ----ExtendedBST.java (INTERFACE)
   ----ExtendedBSTImpl.java (IMPLEMENTACAO A SER PREENCHIDA PELO ALUNO)

   Algumas pequenas alteracoes foram feitas nas implementacoes anteriores:
   - BST possui um método getRoot()
   - O metodo search de BST agora retorna um BSTNode ao invés de um tipo K.

   Essas modificações já estáo implementadas em BSTImpl. Para reusar sua implementação anterior, copie os métodos dela para a BSTImpl fornecida (exceto o método search) ou então copie os métodos search e getRoot da BSTImpl fornecida para sua classe BSTImpl. Isso deve causar pequenos erros de compilação nos metodos que usam o search (ao invés de receber um K eles passam a receber um BSTNode, que contem um K interno).

3. No Eclipse, selecione a pasta dos fontes no projeto LEDA e faça um refresh (apertar F5). Note que deve aparecer um pacote adt.bst e adt.bst.extended em seus fontes contendo os arquivos acima.

Agora voce está pronto para começar a trabalhar nas seguintes atividades:

1. Observe a interface BST.java. Note que o método search agora retorna um BSTNode<K,V> ou NIL se ele não encontra a chave específica.
2. Observe também a existência implementação incompleta BSTImpl. Voce pode utilizar sua implementação prévia de BST.
3. Duas árvores são iguais se elas contém os mesmos nós, numa mesma distribuição. Implemente o método que diz se duas árvores são iguais ou não. A comparação deve ser feita de forma recursiva e pode utilizar apenas métodos da classe BSTNode.
4. Duas árvores binárias (BSTs) são SIMILARES se possuem a mesma distribuição de nós (independente dos valores nos mesmos). Em uma definição mais formal, duas BSTs são SIMILARES

se são ambas vazias, ou se são ambas não vazias e suas sub-árvores esquerdas são similares, e suas sub-árvores direitas também são similares. Implemente o métod que diz se duas árvores são similares ou não.

5. Uma árvore T contém uma sub-árvore ST se algum pedaço de T é exatamente igual a ST. Escreva um método na classe BSTImpl que diz se uma árvore contém outra dentro dela.

Instruções para o envio

Ao terminar a prova, faça os seguintes passos:

1. Compacte as pastas **adt.bst.** e **adt.bst.extended** que existe nos fontes de seu projeto LEDA (**src**). A compactação DEVE ser feita a partir do diretório raiz de seus fontes de forma a preservar a estrutura de pastas que refletem a estrutura dos pacotes (package) Java. Por exemplo, voce deve ter um arquivo compactado NOME_COMPLETO_DO_ALUNO.ZIP com a seguinte estrutura:
   - adt
   --bst
   ---BST.java
   ---BSTNode.java
   ---BSTImpl.java
   --bst
   ---extended
   ----ExtendedBST.java
   ----ExtendedBSTImpl.java

2. Envie esse arquivo com sua solução para o sistema de submissão informado pelo professor.

Observações finais:

- **A interpretação faz parte da prova.**
- **A prova é individual. A conversa entre alunos é proibida.**
- **É proibido coletar códigos prontos e adaptar. Implemente as questões. Isso é para seu aprendizado.**
- **Caso voce observe qualquer problema no sistema de submissão, contacte o professor imediatamente.**