E-fólio B | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Estrutura de dados e algoritmos fundamentais

CÓDIGO: 21046

DOCENTE: Paulo Shirley

A preencher pelo estudante

NOME: Hernâni Filipe Resendes Coelho

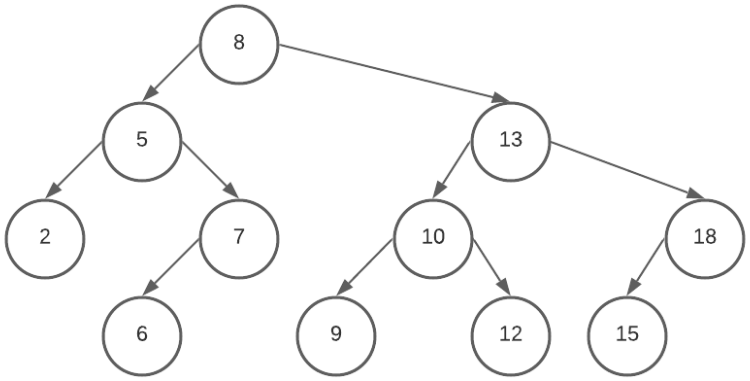
N.º DE ESTUDANTE: 1800045

CURSO: Licenciatura Engenharia Informática

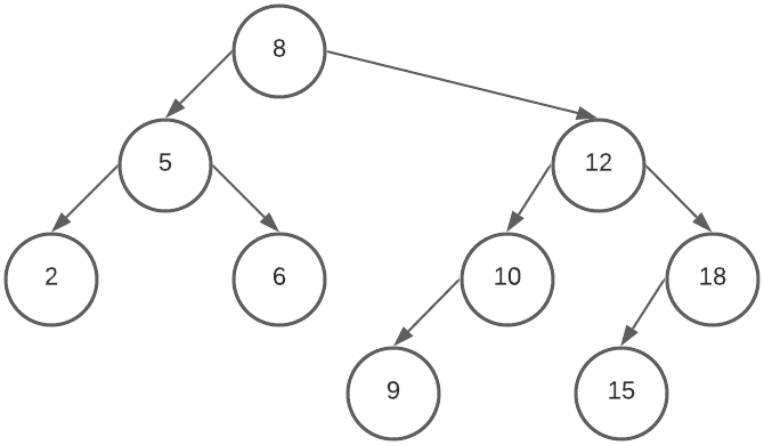
DATA DE ENTREGA: 27/05/2021

TRABALHO / RESOLUÇÃO:

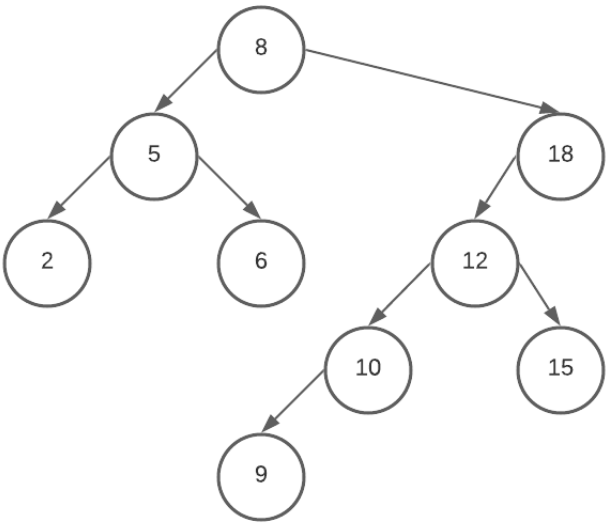
1.1.1 Inserir os itens 8, 5, 7, 13, 2, 10, 18, 15, 9, 6, 12



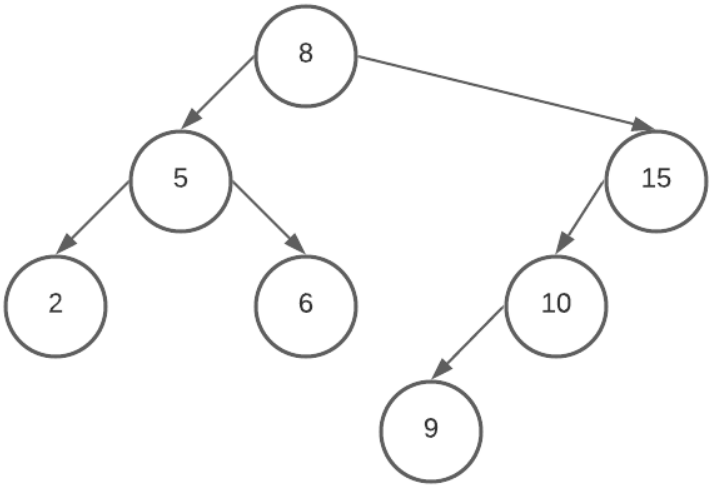
1.1.2 Remover os itens 7, 13



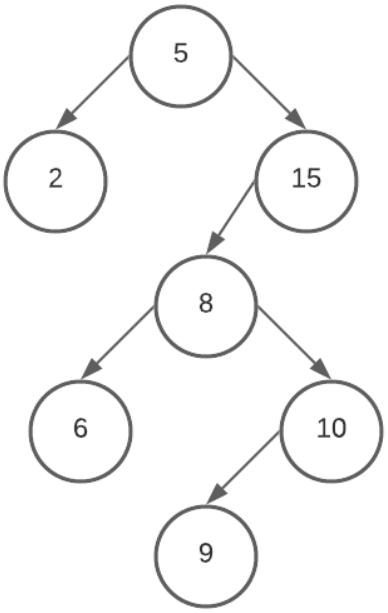
1.1.3 Efetuar uma rotação do item 18



1.1.4 Remover os itens 12, 18



1.1.5 Efetuar uma rotação dos itens 5, 15



1.2

Realizado no HackerRank com o seguinte email: 1800045@estudante.uab.pt com percentagem de sucesso de 100% nos case tests.

1.3

Após submeter o código na plataforma, reavaliei novamente o método para remover um item na árvore, e constatei que não implementei integralmente o algoritmo pretendido, delete by copying. Situação esta que já não pude corrigir após a submissão.

Como tal, passo a explicar a implementação pretendida, o algoritmo, delete by copying, utilizado para remover um item de uma determinada posição na árvore que tem o seguinte princípio de funcionamento:

- Se a árvore estiver vazia, não existe nada a remover como tal, não executará;

- O algoritmo recorre aos métodos “findAntecessor” e “findSucessor” para encontrar o antecessor e sucessor, de forma alternada, iniciando pelo antecessor.
- Se o item a remover for uma folha, então o “ramo” pai desta irá apontar para nullptr e será removida a folha;
- Se o item a remover for um nó sem uma das subárvores, ou seja, só com um filho, o seu descendente tomará o seu lugar e o endereço “antigo” deste nó será removido;
- Se o item a remover tiver dois filhos, então o item do seu antecessor/ sucessor será copiado para o nó e será removido o nó antecessor/sucessor. Caso este tenha um descendente, o pai do antecessor/sucessor passa apontar para ele.

Em suma, o método terá uma complexidade O(n) como caso mais desfavorável, pois em qualquer das situações acima, terá de atravessar a árvore até encontrar o nó que pretende remover. Apenas o caso de remover a raiz terá complexidade O(1), pois independentemente da altura da árvore, a raiz é sempre a referência e como tal terá tempo de execução constante para a remover.