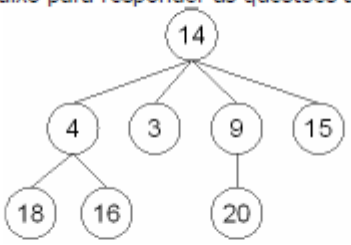
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
Bacharelado em Ciência da Computação / Engenharia da Computação
INF 01203 – Estruturas de Dados

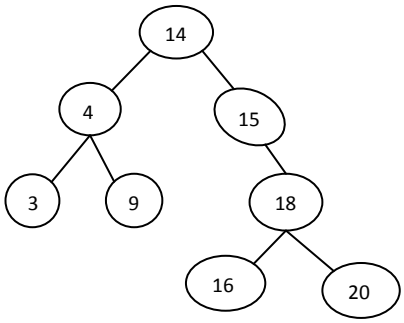
SEGUNDA AVALIAÇÃO

IDENTIFICAÇÃO
Nome: _____ **ID:** _____ **30/10/2008**

01. Considere a árvore n-ária abaixo para responder as questões apresentadas seguir:



a) (1,0) É possível transformar uma árvore de n-filhos em uma Arvore de Pesquisa Binaria (ABP) da seguinte forma. Percorre-se a arvore segundo algum critério de caminhamento (largura ou profundidade) e insere-se os elementos na arvore de pesquisa binária seguindo as regras de construção de uma arvore ABP. Desenhe uma arvore ABP seguindo caminhamento por níveis.



b) (0,5) A árvore resultante do exercício **(a)** é uma arvore AVL? Justifique sua resposta.

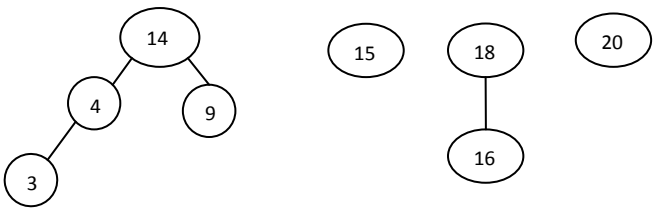
Não. Porque a condição para que uma árvore seja AVL é possuir nodos com fatores -1, 0 ou 1 e o nodo 15 possui fator -2.

c) (1,0) Mostre o caminhamento pré-fixado a direta e central-esquerda para a árvore resultante do exercício **(a)**.

Pré-fixado a direita: 14 -15 -18 -20 -16 -4 -9 -3

Central-esquerda: 3 -4 – 14 -15 -16 -18 -20

d) (1,0) Transforme a árvore resultante do exercício **(a)** em uma árvore n-aria seguindo as regras de transformação de árvore n-arias em arvore binária. Responda: é possível que a árvore resultante fique igual a árvore inicial da figura apresentada? Justifique sua resposta.

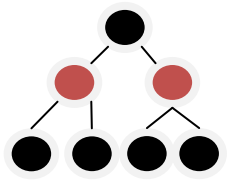


Não. A árvore possui filhos a direita e, de acordo com as regras de transformação resultaria uma floresta.

02 – (valor 1,5) – Para cada uma das afirmações sobre árvores rubro-negras, determine se é verdadeira ou falsa. Se você achar que é verdadeira, forneça uma justificativa. Se você achar que é falsa, forneça um contra-exemplo.

a) Um sub-árvore de uma árvore rubro-negra é também uma árvore rubro-negra.

Falsa. Sub-árvore com nodo raiz vermelho:

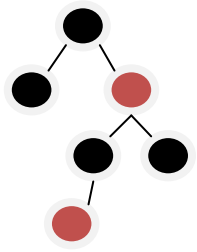


b) O irmão de um nodo externo ou é externo ou é vermelho.

Verdadeira. Se o nodo não for externo deve ser vermelho, caso contrário o caminho que passa por seu irmão terá um número maior de nodos pretos que o seu.

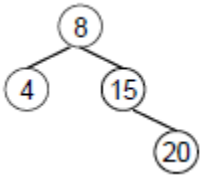
c) Toda árvore rubro-negra é também uma árvore AVL.

Falsa. A árvore pode ser rubro-negra mas não AVL.

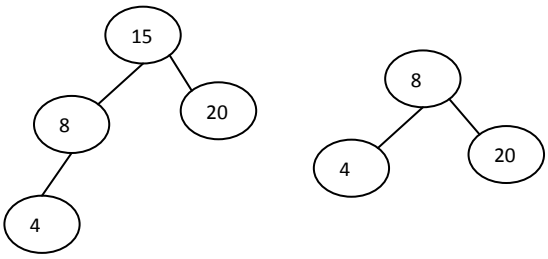


03 – (valor = 1,5) - Simule as operações solicitadas para a árvore splay abaixo. Redesenhe a árvore após cada rotação e indique os nomes das operações realizadas.

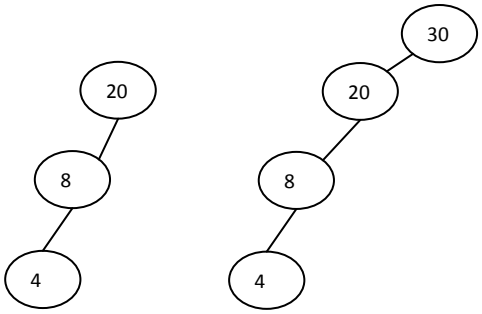
- remover 15
- inserir 30
- buscar 8
- inserir 25
- remover 10



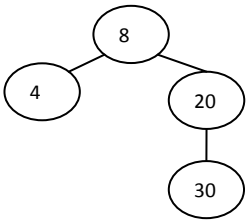
Remover 15: access15-> splay 15->zag -> remove 15 ->join



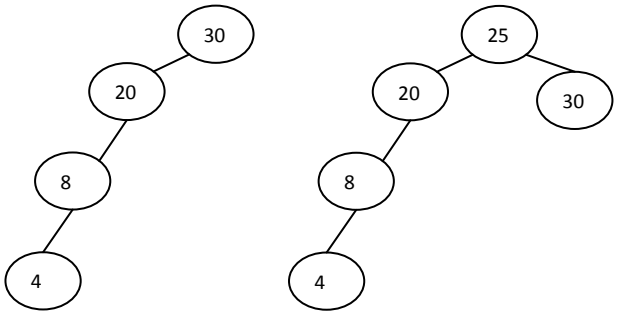
Insere 30:Access 30 -> splay 30 -> zag ->split -> insere 30



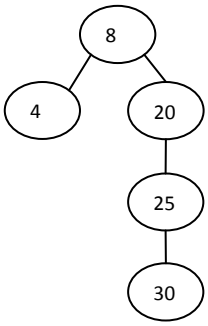
Buscar 8: splay -> zig-zig



Inseriri 25: Access 25 -> splay 30 -> zag-zag -> split



Remover 10: Access 10 -> aplay 8 -> zig-zig



04 – (valor = 1,5) - Faça uma função que receba um ponteiro para a raiz de uma ABP e retorne o número de nodos que ela possui.

```
int numNodos (Nodo* arv){
   if (arv == NULL){
      return 0;
  } else{
      return (numNodos (arv -> esq) + numNodos (arv -> dir) + 1);
   }
}
```

05 – (valor = 2,0) - Faça uma função que receba um ponteiro para a raiz de uma ABP e um valor de chave. A função deve retornar um ponteiro para o pai do nodo que contém a chave. Caso a chave não seja encontrada ou o nodo não tenha pai, a função deve retornar nulo. (2,0 pontos)

```
Nodo* pai (nodo* arv, int chave){
    Nodo* aux;
    if (arv == NULL || raiz -> chave == chave)
    return NULL;
      if(raiz -> chave > chave){
         if (raiz -> esq == NULL)
         return NULL;
         if (raiz -> esq -> chave == chave)
         return raiz;
         return pai (raiz -> esq, chave);
      }
      if (raiz -> chave < chave){
          if (raiz -> dir == NULL)
          return NULL;
          if (raiz -> dir -> chave == chave)
          return raiz;
          return pai (raiz -> dir, chave);
       }
}
```