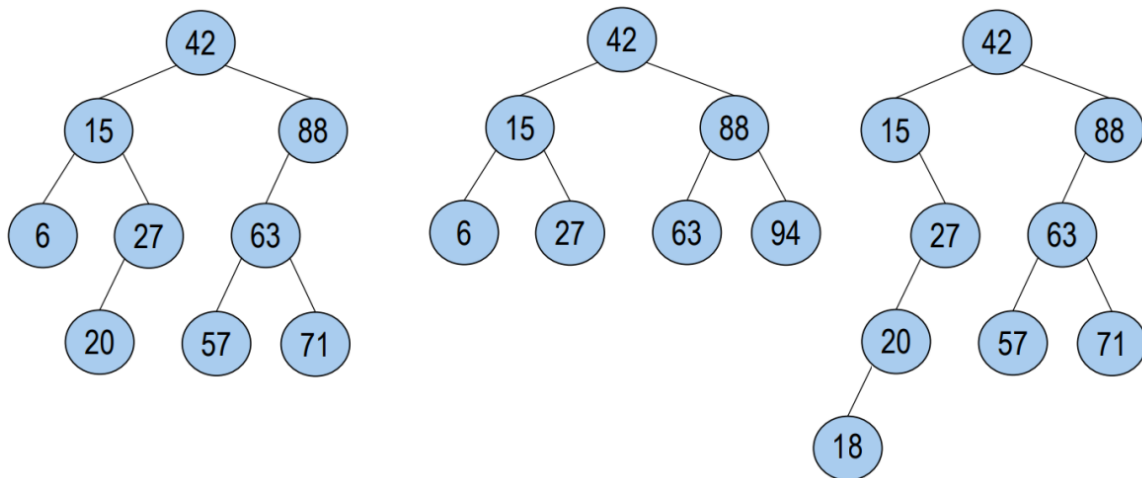


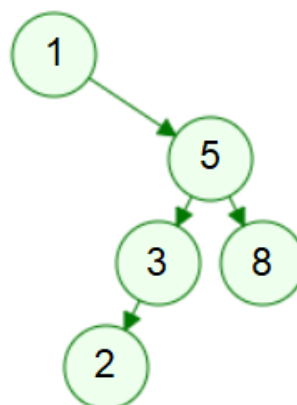
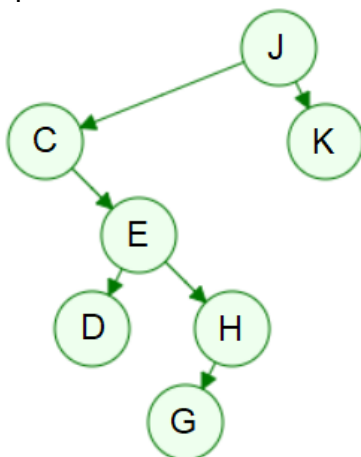
Lista de Exercícios – IBILCE UNESP

Estrutura de Dados I – Balanceamento e árvores AVL

- 1) Por que devemos, sempre que possível, balancear uma árvore binária de busca?
- 2) O que é uma árvore AVL e qual sua finalidade?
- 3) O que é fator de balanceamento e qual sua utilidade no contexto de AVL?
- 4) Verifique quais das árvores binárias de busca são AVL, e calcule o fator de balanceamento para cada um dos nós. Quais nós em cada uma das árvores encontram-se desbalanceados?



- 5) Descreva os passos básicos para:
 - (a) Rotação simples à esquerda e rotação simples à direita (considerar o caso particular de cada uma das rotações acima).
 - (b) Rotação dupla esquerda e rotação dupla direita.
- 6) Dada as seguintes ABBs. Aplique as rotações necessárias a fim de converter as ABBs para AVL.



- 7) Insira os números 34, 38, 50, 19, 13, 27, 21, 31 (nesta ordem) em uma árvore AVL. A cada nova inserção, calcule os fatores de balanceamento dos nós e, a partir disso, faça as rotações necessárias a fim de preservar o balanceamento

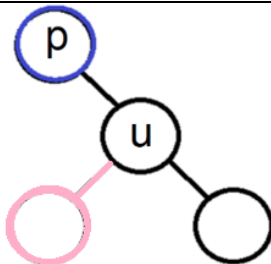
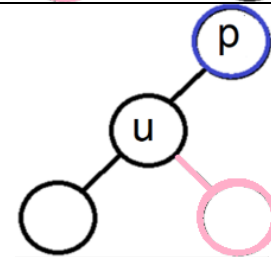
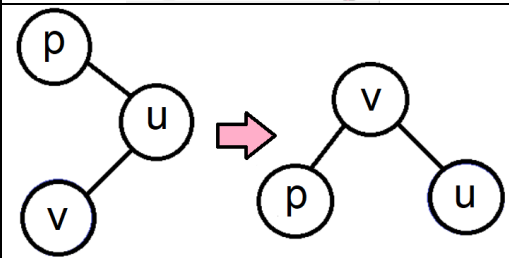
da árvore. Dica: ilustre as árvores intermediárias resultantes para facilitar a compreensão.

- 8) (a) insira os elementos {50, 1, 64, 12, 18, 66, 38, 95, 58, 59, 70, 68, 39} de forma a preservar a estrutura de uma AVL. Remova, nesta ordem, os elementos {95, 70, 50, 68} da ALV construída de modo a preservar o balanceamento.

- 9) Considere a struct a seguir, que representa um nó de uma AVL

```
//Representação de um nó de uma ALV
//-----
typedef struct no {
    int chave;
    struct *no esq;
    struct *no dir;
} no;
//-----
```

Escreva uma função que realize:

<p>(a) Rotação simples à esquerda. Protótipo: no *Rotacao_esquerda (no *p)</p>	
<p>(b) Rotação simples à direita. Protótipo: no *Rotacao_direita (no *p)</p>	
<p>(c) Rotação dupla esquerda. Protótipo: no *Rotacao_duplaEsq (no *p)</p>	
<p>(d) Rotação dupla direita. Protótipo: no *Rotacao_duplaDir (no *p)</p>	