Rotação Simples 2 (LL)

Neste exercício você deve escrever parte da solução para árvores AVL!

- Elabore uma função (pode adaptar de outros exercícios que você resolveu no Code Bench) para fazer inserção em árvore binária de busca não balanceada (sem rotações).
- Após fazer a inserção de cada chave, você deve atualizar as alturas de todas as sub-árvores que participaram do processo de inserção.
- Cada nó da árvore pode ser representado como um dicionário que contém as chaves `'valor'`, `'esq'` e `'dir'` e `'h'`, significando, respectivamente, o valor da chave no nó, uma referência para a raiz da sub-árvore esquerda, uma referência para a raiz da sub-árvore direita e a altura do nó.
- Para este exercício, considere que a altura de uma sub-árvore vazia é 0. Então todos os nós folhas possuem altura 1.
- Elabore uma função chamada `rotacaoLL(T)` que recebe como entrada a raiz de uma sub-árvore `T` e aplica rotação à direita (da esquerda para a direita ou, simplesmente, LL). Sua função deve devolver a nova raiz dessa sub-árvore. Não se esqueça de atualizar a altura dos nós envolvidos na rotação.
- Escreva um programa principal que cria várias árvores não necessariamente balanceadas a partir de chaves lidas da entrada.
- Aplique rotação LL na raiz de cada árvore lida.
- Para cada árvore resultante, imprima a altura da árvore, o fator de balanceamento da raiz e o resultado dos percursos pré-ordem e pós-ordem.

Reforçando: não é necessário fazer rotações *durante a inserção*. Apenas ao final da construção da árvore é que você deverá aplicar a rotação. E, neste exercício, a rotação deve ser aplicada apenas na raiz.

Entrada

A entrada conterá vários casos de teste. Cada caso de teste é uma lista contendo uma sequência de chaves que devem ser inseridas em uma árvore inicialmente vazia. A entrada termina quando aparecer a lista vazia (ela não é um caso de teste).

Saída

Para cada caso de teste, imprima as seguintes informações sobre a árvore obtida depois da rotação:

```
Arvore {num}
h: {altura}
fb: {fb}
pre: {lista-pre}
pos: {lista-pos}
```

Substitua os especificadores pelo seguinte:

- `{num}`: o número do caso de teste (começa em 1 e incrementa a cada caso lido)
- `{altura}`: a altura da árvore (altura da raiz)
- `{fb}`: o fator de balanceamento da raiz
- `{lista-pre}`: a lista das chaves na ordem que aparecem durante o caminhamento pré-ordem
- `{lista-pos}`: a lista das chaves na ordem que aparecem durante o caminhamento pós-ordem

Imprima uma linha em branco após cada caso de teste.

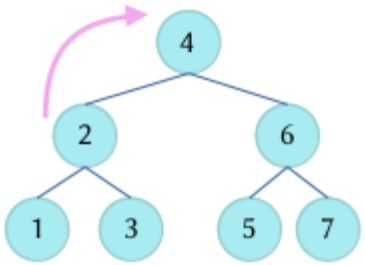
Dicas

Exemplo Comentado

Considere a seguinte entrada:

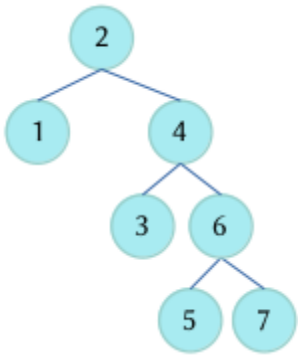
```
[4, 2, 6, 1, 3, 5, 7]
[]
```

A árvore construída a partir dessa entrada, bem como a rotação que será aplicada, está ilustrada na figura a seguir:



Observe que essa árvore já está balanceada. Em uma situação normal, nós **não** aplicaríamos rotação nela.

Entretanto, se aplicarmos a rotação LL na sua raiz, o resultado será a seguinte árvore:



A saída é obtida pelos percursos pré-ordem e pós-ordem:

```
Arvore 1
pre: 2 1 4 3 6 5 7
pos: 1 3 5 7 6 4 2
```

Exemplos de Entrada e Saída

Entrada	[3, 2, 1] [2, 3, 1] [3, 1, 5, 2, 4] []
Saída	Arvore 1 h: 2 fb: 0 pre: 2 1 3 pos: 1 3 2 Arvore 2 h: 3 fb: 2 pre: 1 2 3 pos: 3 2 1 Arvore 3 h: 4 fb: 3 pre: 1 3 2 5 4 pos: 2 4 5 3 1
Entrada	[5, 3, 6, 2, 4, 1] []
Saída	Arvore 1 h: 3 fb: 0 pre: 3 2 1 5 4 6 pos: 1 2 4 6 5 3