Rotação Simples 1 (RR)

Neste exercício você deverá elaborar parte da solução do algoritmo de inserção em árvores AVL!

- Escreva (ou reutilize de outros exercícios que você resolveu no Code Bench) uma função para fazer inserção em árvore binária de busca não balanceada (sem rotações).
- Cada nó da árvore pode ser representado como um dicionário que contém as chaves `'valor'`, `'esq'` e `'dir'`. Elas devem representar, respectivamente, o valor da chave desse nó, uma referência para a sub-árvore esquerda e uma referência para a sub-árvore direita.
- Escreva uma função chamada `rotacaoRR(T)` que recebe como entrada a raiz de uma sub-árvore `T` e aplica rotação à esquerda (da direita para a esquerda ou, simplesmente, RR) nessa raiz. Sua função deve devolver a nova raiz dessa sub-árvore.
- Escreva um programa principal que lê várias árvores não necessariamente balanceadas do teclado.
- Aplique rotação RR na raiz de cada árvore lida.
- Para cada árvore resultante, imprima os resultados 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 e Saída

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).

Como saída, imprima primeiro a frase `'Arvore {}'`, sendo que o especificador `'{}'` deve ser substituído pelo número da árvore que está sendo processada. Em seguida, imprima os percursos pré-ordem e pós-ordem da árvore obtida após a operação de rotação. Siga a formatação contida nos exemplos. Imprima uma linha em branco após cada caso de teste.

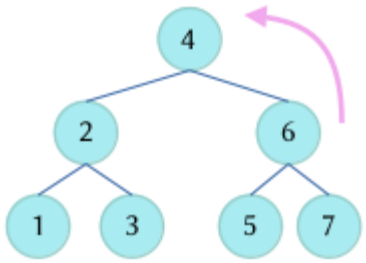
Dicas

Exemplo Comentado

Considere a seguinte entrada:

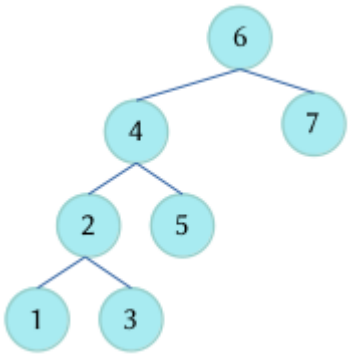
```
[4, 2, 6, 1, 3, 5, 7]
[]
```

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 RR na sua raiz, o resultado será a seguinte árvore:



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

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

Exemplos de Entrada e Saída

Entrada	[1, 2, 3] [2, 1, 3] [3, 2, 4, 1, 5] []
Saída	Arvore 1 pre: 2 1 3 pos: 1 3 2 Arvore 2 pre: 3 2 1 pos: 1 2 3 Arvore 3 pre: 4 3 2 1 5

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