

# Árvore binária de busca

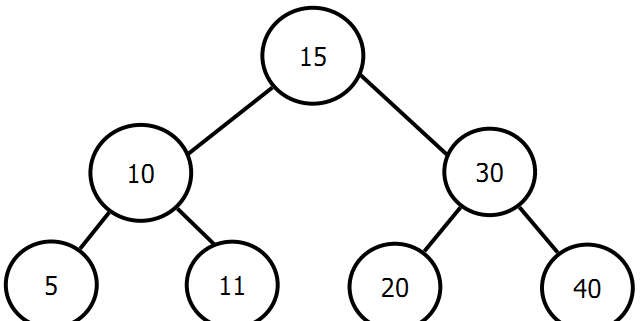
Outra solução é a Remoção por cópia.

A qual propõe que um nó com dois filhos a ser removido pode ser reduzido a uma das duas situações básicas: nó com apenas um filho e nó sem nenhum filho.

Isso é feito substituindo pela chave de seu sucessor imediato a chave que está sendo removida e em seguida removendo o nó que continha a chave do sucessor imediato.

# Árvore binária de busca

Obs.: o sucessor imediato de um nó é o nó mais à esquerda em sua subárvore à direita.

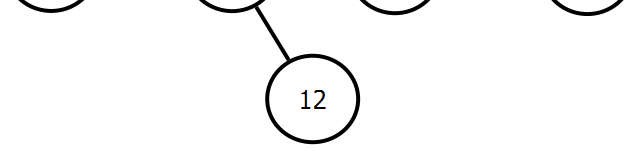
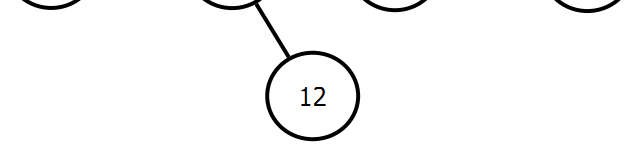
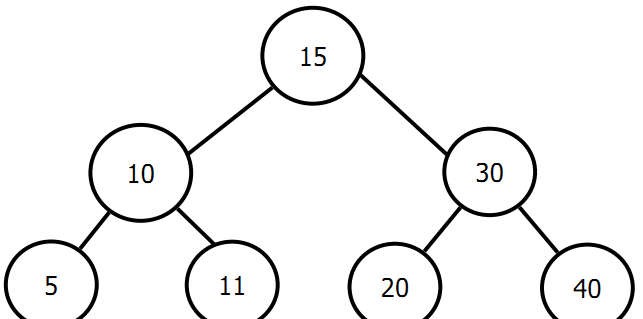


Vejamos

um

exemplo

:



Árvore Balanceada

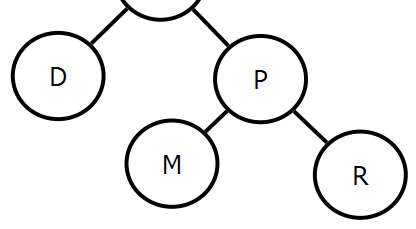
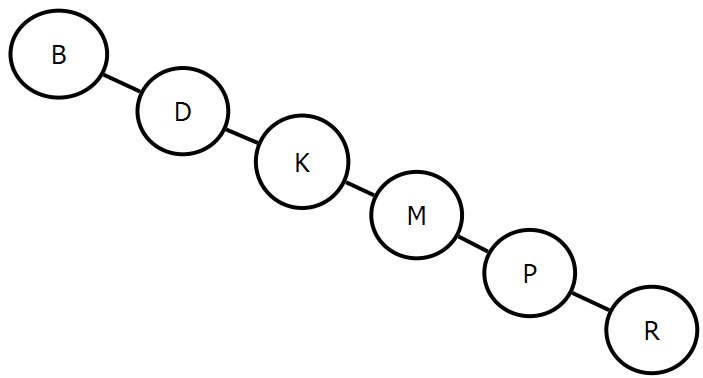
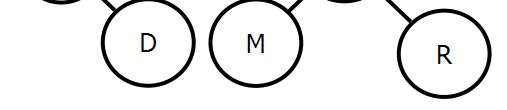
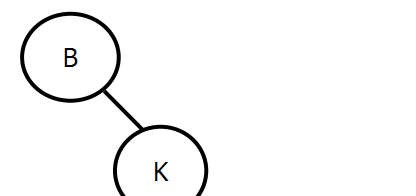
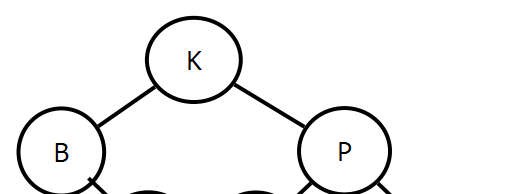
Além das árvores serem muito apropriadas para representar a estrutura hierárquica de um certo domínio, o processo de busca por um elemento em uma árvore tende a ser muito mais rápido do que em uma lista encadeada.

Contudo, para que efetivamente uma busca por um determinado elemento em uma árvore seja eficiente está, além de ser uma árvore binária de busca, deve ter seus nós adequadamente distribuídos.

**Árvore balanceada**

Observe as árvores a seguir:

(a)



(

b

)

(

c

)

Com uma análise das árvores apresentadas podemos concluir que uma grande gama de configurações de árvores binárias de busca é passível de ser obtida com um mesmo conjunto de nós e que o pior caso de uma busca por um nó está diretamente relacionado com a altura da árvore.

Sendo assim, podemos estabelecer a seguinte definição: Uma árvore binária é balanceada em altura ou simplesmente balanceada se a diferença na altura de ambas as subárvores de qualquer nó na 36árvore é zero ou um.

Uma definição complementar é a de árvore perfeitamente balanceada.

Uma árvore binária é perfeitamente balanceada quando, além de ser balanceada, todas as suas folhas encontram-se em um ou dois níveis.

Para facilitar a visualização, uma árvore que possui 10.000 nós pode ser configurada em uma árvore com altura igual a lg(10000) = ceil(13.289) = 14. Ou seja, qualquer elemento é passível de ser localizado com no máximo

14 comparações se a árvore for

37perfeitamente balanceada.

Neste ponto cabe a seguinte pergunta: Ao se analisar uma árvore pode-se determinar qual dentre os seus nós seria uma raiz adequada para torná-la perfeitamente balanceada, qual seria este nó?

O nó cuja sua chave (valor) representa a mediana das chaves presentes nos nós que compõem a árvore, para uma árvore com número impar de nós. Ou o nó cuja sua chave (valor) representa um dentre os dois valores mais próximos da mediana das chaves presentes nos nós que compõem a árvore, para uma árvore com número par de nós.

Vimos que uma estratégia baseada nesta observação que pode ser utilizada para 38balancear uma árvore é:



* Quando os dados chegarem armazene-os em um vetor;
* Após todos os dados terem sido armazenados no vetor, ordene-o;
* Agora determine como raiz o elemento do meio do vetor;
* O vetor consistirá agora em dois subvetores. O filho esquerdo da raiz será o nó com valor no meio do subvetor constituído do início do vetor até o elemento escolhido como raiz;
* Um procedimento similar é adotado para a

definição do filho direito da raiz;

* Este processo se repete até não existirem mais 39elementos a serem retirados do vetor.

O algoritmo apresentado possui um sério inconveniente, pois todos os dados precisam ser colocados em um vetor antes da árvore ser criada. O mesmo algoritmo também pode ser utilizado quando uma inserção ocorre sobre uma árvore já existente. Porém, com o mesmo inconveniente. Neste caso, os elementos da árvore devem ser retirados e colocado em um vetor, mesmo que a árvore binária de busca esteja desbalanceada, se for efetuado um percurso in-ordem elimina-se a necessidade de ordenar o vetor.

Posteriormente o novo elemento é inserido em sua posição adequada no vetor e a função definida anteriormente pode s~~e~~r



40utilizada.

Vimos que existem **formas mais eficientes de se balancear uma árvore**.

Um exemplo é o algoritmo desenvolvido por Colin Day e posteriormente melhorado por Quentin F. Stout e Bette L. Warren. Denominado Algoritmo DSW, devido aos nomes de seus idealizadores, baseia-se em percorrer uma árvore binária de busca tornando-a uma arvore degenerada (similar a uma lista encadeada) e posteriormente percorrê-la novamente tornando-a uma árvore perfeitamente balanceada.