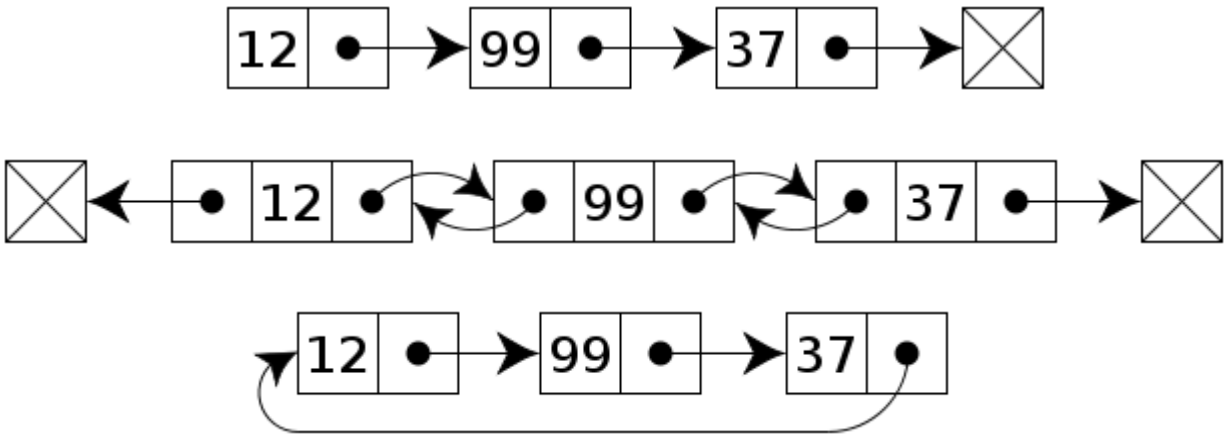
Estruturas de dados

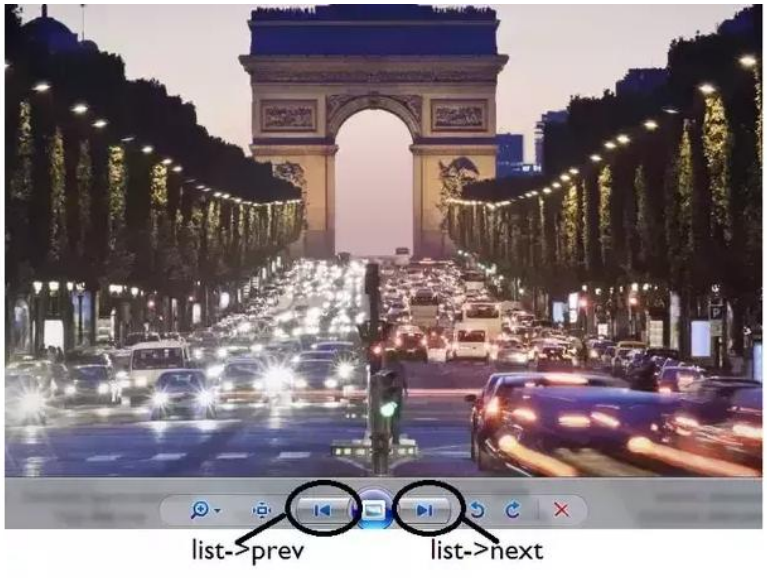
-Lista ligada
-Pilha
-Fila
-Árvore
-Hashtable

Lista ligada (Linked List)

-Se comporta como uma lista
-Dinâmica: é possível inserir e remover elementos
-O acesso dos dados é feito por vínculos entre elementos



Exemplos de lista ligada



Galeria de fotos

Fila de “Conga”

Propriedades e operações

-Lista simplesmente ligada:

No.proximo() retorna o próximo nó

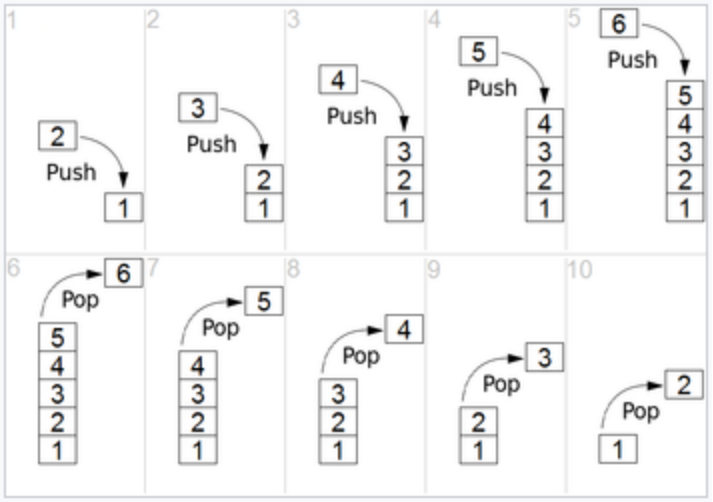
-Lista duplamente ligada:

No.proximo () retorna o próximo nó

No.anterior() retorna o nó anterior

Pilha (Stack)

-Como o nome já diz, os dados são empilhados
-O dado é posto em cima da pilha
-O dado é tirado de cima da pilha



Exemplos de pilha

-Ctrl+Z (Undo) e Ctrl+Y (Redo) nos programas de computador

-Chamada de rotinas

Porta guardanapo

Propriedades e operações

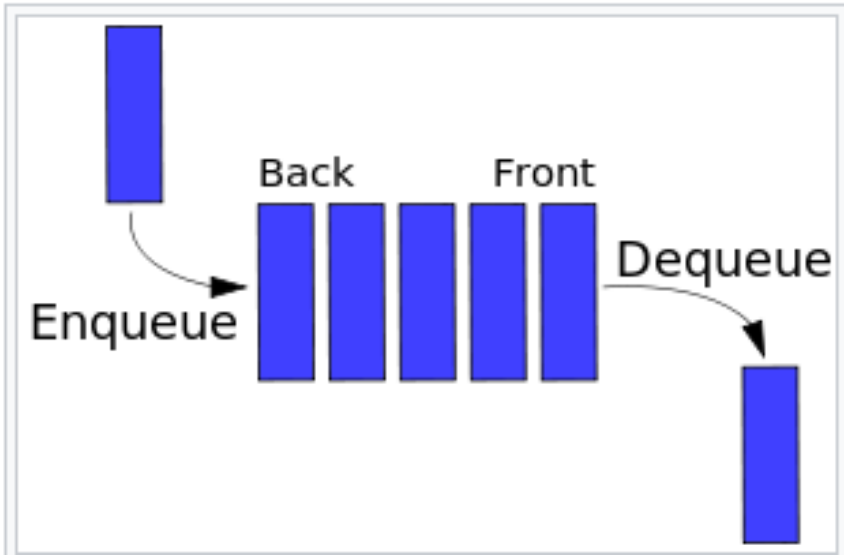
-push(dado): insere o dado (empurra)

-pop(dado): remove o dado e retorna seu valor

-comprimento(): retorna a quantidade de elementos

Fila (Queue)

-Os dados são enfileirados um atrás do outro
-O próximo a ser atendido é o primeiro da fila
-Quem chegar na fila fica por último



Exemplos de fila

-Uma fila de pessoas (...)
-Processos do sistema operacional

Porta papel

Pegador de pães

Propriedades e operações

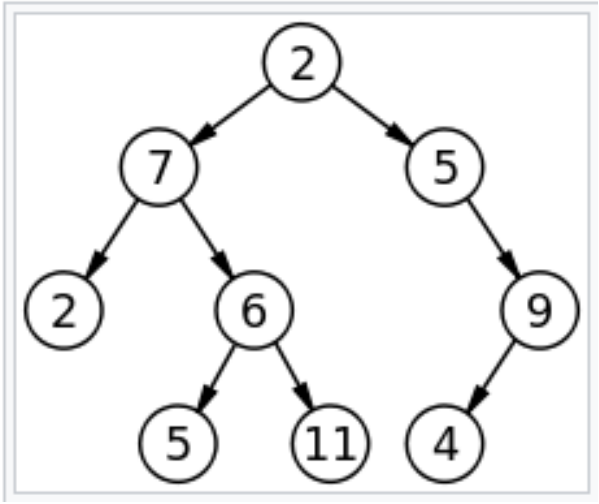
-enqueue(elemento): insere o elemento na fila (como se fosse alguém chegando)

-dequeue(): remove o elemento na última posição da fila e retorna seu valor

-comprimento(): retorna a quantidade de elementos da fila

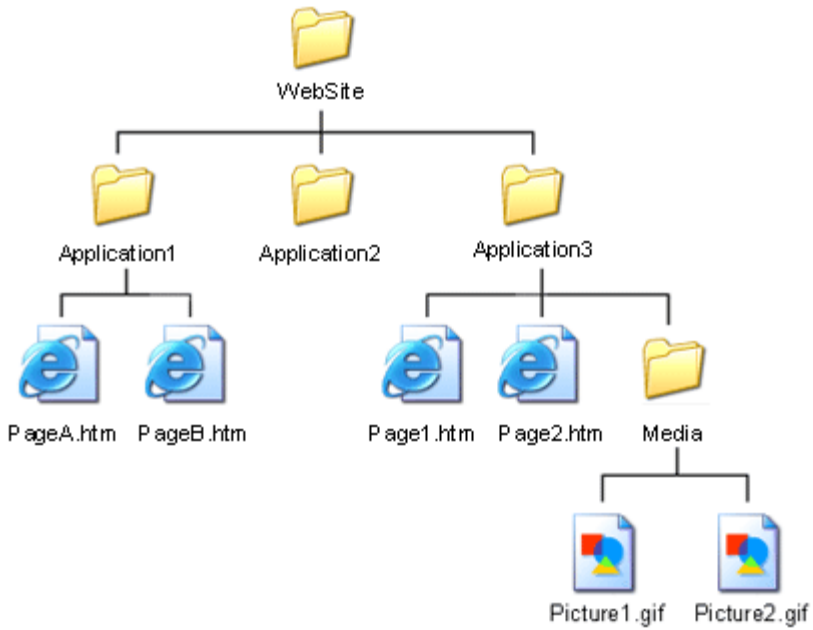
Árvore

-Os dados são organizados como uma árvore genealógica
-Os nós têm filhos, que por sua vez tem filhos, e etc.



Exemplo de árvore

Estrutura do diretório do Computador

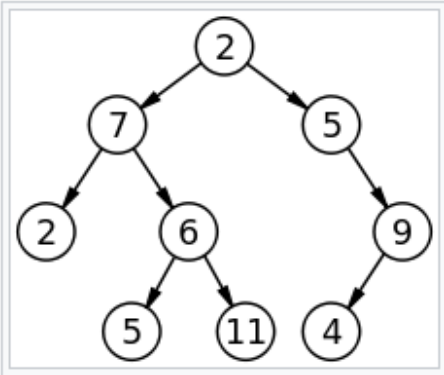


Propriedades

Raiz da árvore: “pai de todos”

Folhas da árvore: elementos sem filhos

Altura da árvore: é o número de “gerações” máximo da árvore



Exemplos de operações:

Node.firstChild(): retorna o primeiro nó filho

Node.lastChild(): retorna o primeiro último nó filho

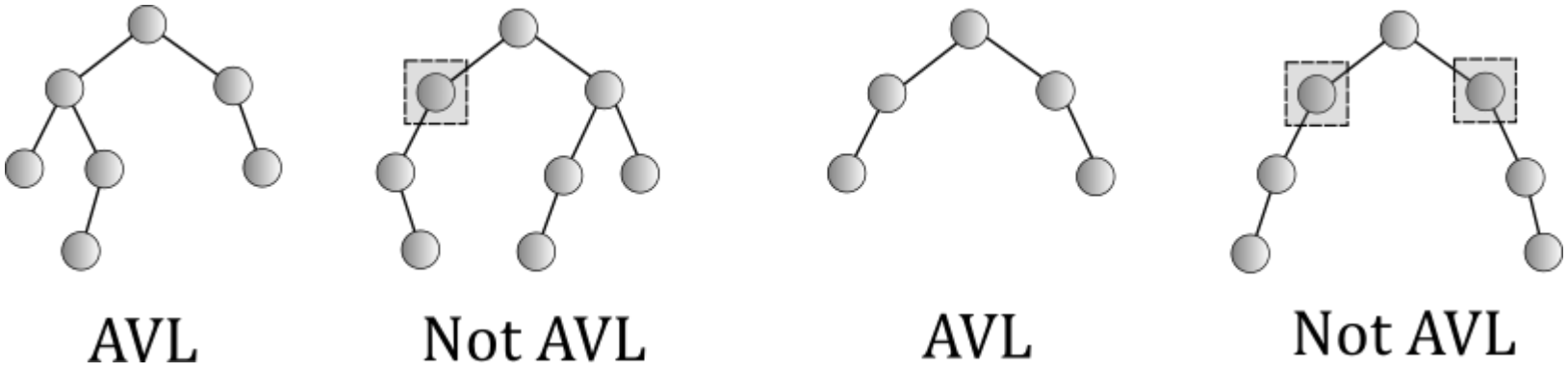
Node.children (): retorna uma lista com os nós filho

Árvores notáveis

-Árvore binária: quando o nó tem no máximo dois filhos.

-Árvore balanceada: intuitivamente, quando os nós estão bem distribuídos pela árvore, ou seja, grande parte dos nós têm dois filhos

-Árvore AVL: Uma árvore binária em que para cada nó, seus filhos constituam subárvores com uma altura que difira de no máximo 1.



Árvore binária de busca

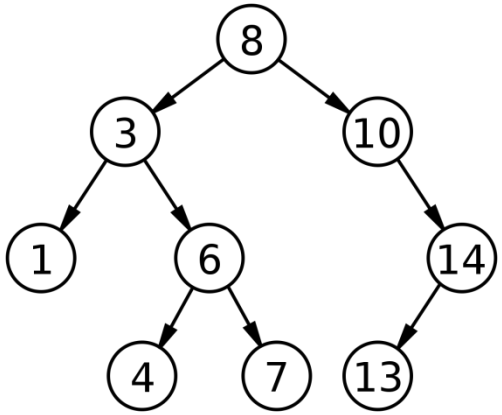
-A ideia é inserir dados em uma determinada ordem numa árvore binária

-Para cada nó é associado um número

-A regra da árvore binária é:

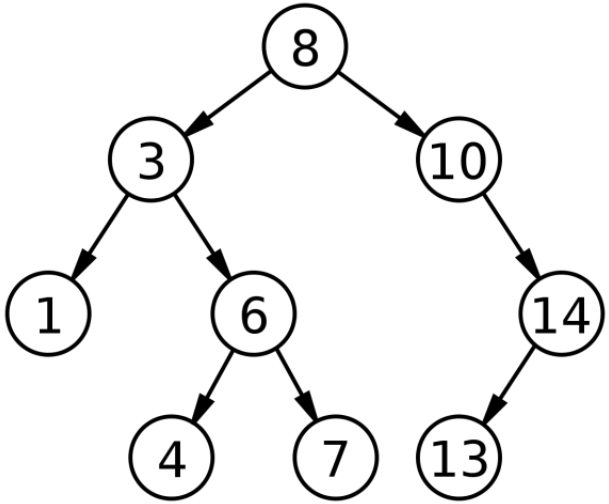
-O nó filho ESQUERDO é MENOR que o nó pai

-O nó filho DIREITO é MAIOR que o nó pai



Operações

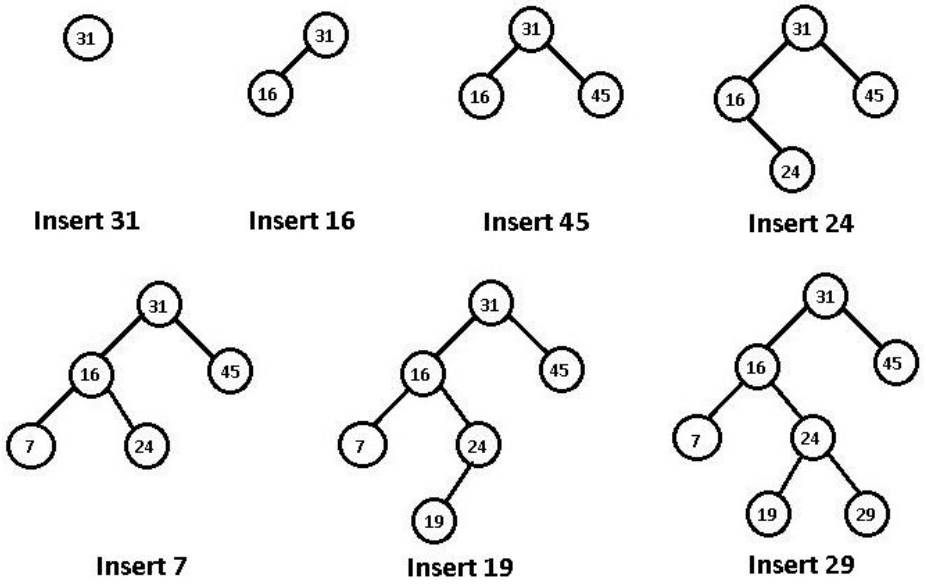
-Máximo: andar pela direita até chegar a uma folha
-Mínimo: andar pela esquerda até chegar a uma folha
-Busca: decidir se vai para direita ou esquerda, dado que o valor do próximo filho é maior ou menor do que o valor procurado.



Inserção

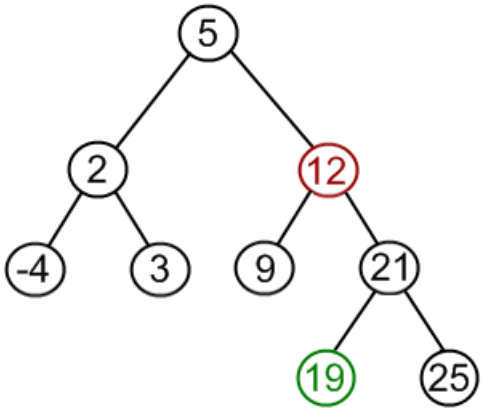
A função verifica e a chama recursivamente para inserir de um lado ou de outro dependendo do valor.

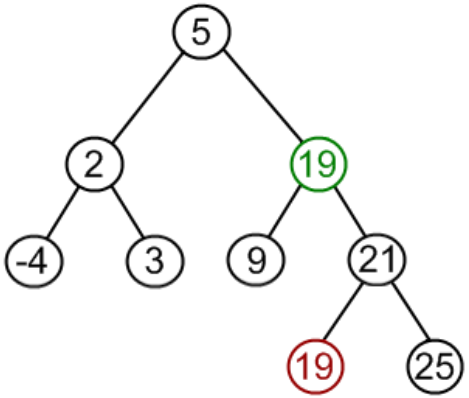
```
def binary_insert(root, node):
    if root is None:
        root = node
    else:
        if root.data > node.data:
            if root.l_child is None:
                root.l_child = node
            else:
                binary_insert(root.l_child, node)
        else:
            if root.r_child is None:
                root.r_child = node
            else:
                binary_insert(root.r_child, node)
```

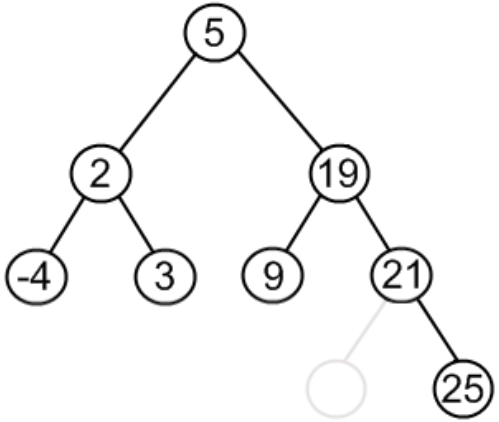


Remoção

3 casos:
-Nó sem filhos: apenas remover normalmente
-Nó com um filho: remover o nó e ligar o “filho” ao “vô”
-Nó com dois filhos:





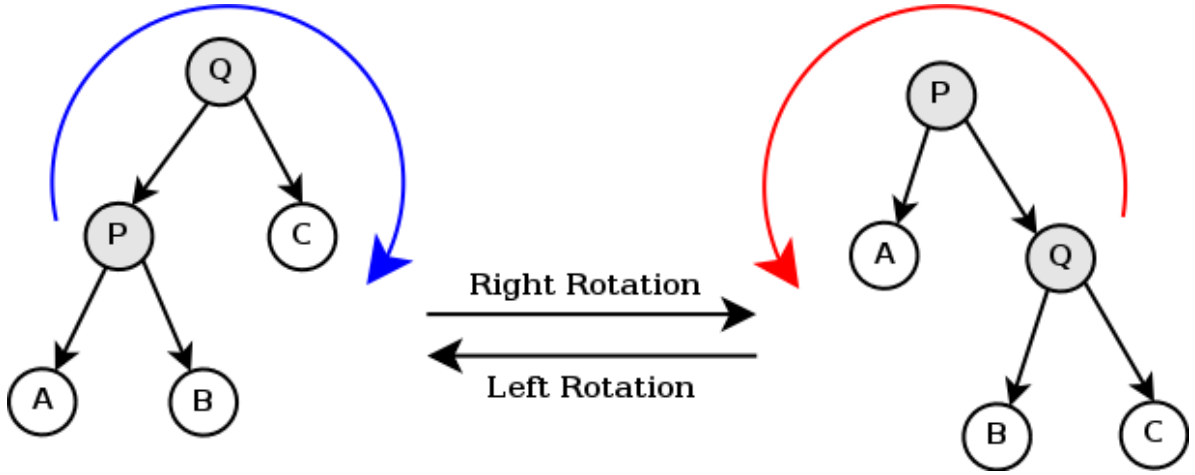


Rotação

Rotação é uma operação que preserva a estrutura da BST (Binary Search Tree) porém possibilita o balanceamento da árvore.

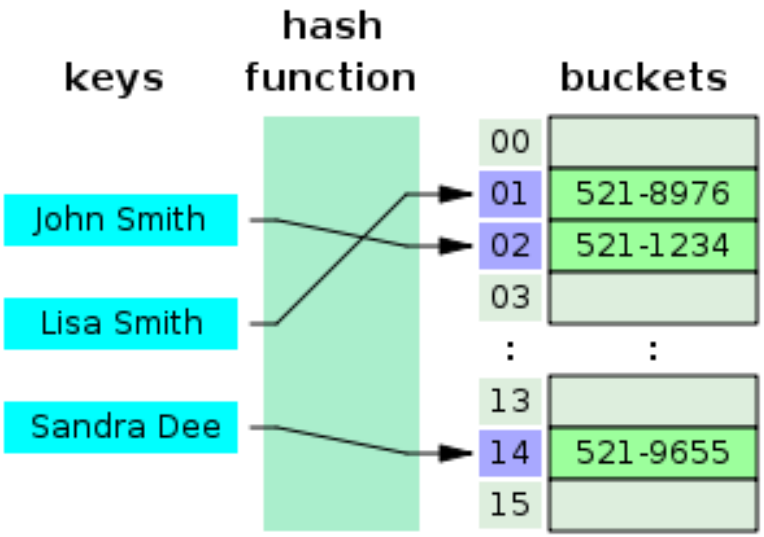
-Escolher um nó para rotacionar

-Escolher o lado de rotação



Hashtables

-Estrutura de dados com “buckets”
-Associa-se um valor arbitrário para cada dado que entra, e com isso é possível determinar em qual “bucket” ele vai entrar.
-Analogia: Buckets são caixinhas e os números são os “rótulos” ou “identificadores” do produto



Hashfunction

-O número “arbitrário” é chamado de Hash
-Função Hash: uma função que mapeia um valor: (um número ou string) para um outro número.
-A ideia de Hash é mapear inúmeros valores para um número reduzido de valores.

Exemplos de Hashfunction:
-Número de letras do nome
-Iniciais do nome
-Resto da divisão
-Número de bytes do dado na memória

Colisões

Por conta da natureza do Hash, dois dados diferentes podem ter o mesmo valor de hash

Exemplo: Escolhendo um hashfunction “número de letras” e tendo como input uma String:

Input1: Pietro hash=6
Input2: João hash=4
Input3: Maria hash=5
Input4: José hash=4

1	
2	
3	
4	João José
5	Maria
6	Pietro

Colisões

-Alternativa: utilizar outra Hashfunction
-Mesmo assim, dependendo do número de dados, irá ocorrer alguma colisão
-Alternativa permanente: utilizar uma estrutura de dados auxiliar dentro de cada bucket, como uma lista ligada

-Fator de carga: a razão entre número de dados e número de buckets. Para utilizar o máximo da hashtable é importante que o fator não seja muito grande.