Universidade Federal do Rio de Janeiro

Trabalho de Organização de Dados 2

Grupo:

DANILO SILVA DE CARVALHO – 107390588
ROBERTA SANTOS LOPES – 107362886
TAÍSA LOPES MARTINS – 107362828
VINICIUS BASTOS BITTENCOURT – 107362983

Árvore AVL

Motivação:

Por que foi criada essa estrutura?

Vamos analisar uma estrutura de dados muito comum: a Árvore Binária de Busca

Árvore Binária de Busca

Estrutura de dados onde todos os nós são chaves, todos nós à esquerda contêm uma sub-árvore com os valores menores ao nó raiz da sub-árvore e todos os nós da sub-árvore à direita contêm somente valores maiores ao nó raiz desta última sub-árvore.

Árvore Binária de Busca

Tal fato nos permite fazer buscas, inserções e remoções e exibir os resultados de forma ordenada com muita flexibilidade.

Árvore Binária de Busca

Exemplo:

Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.

Árvore Binária de Busca

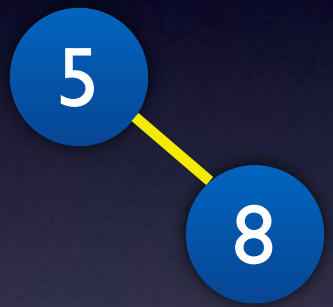
Exemplo:

Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.

Árvore Binária de Busca

Exemplo:

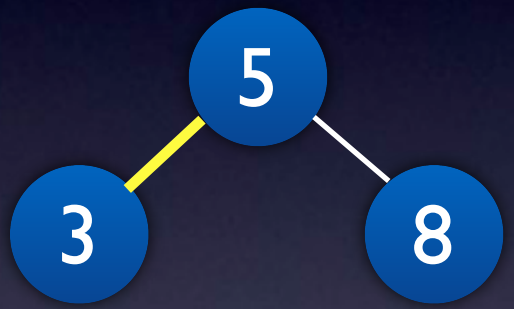
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

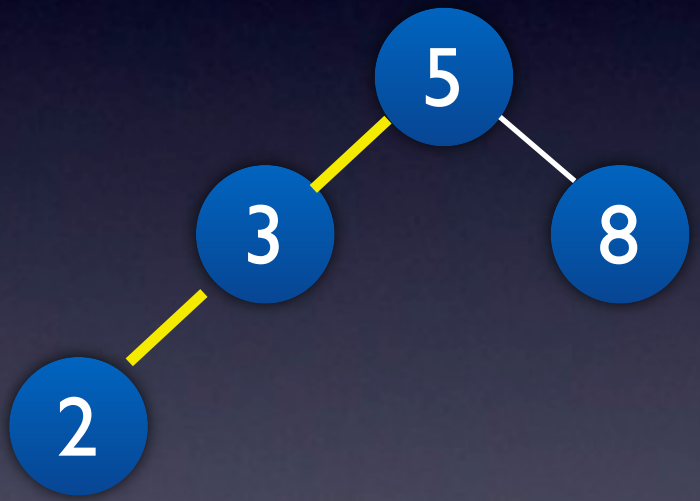
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

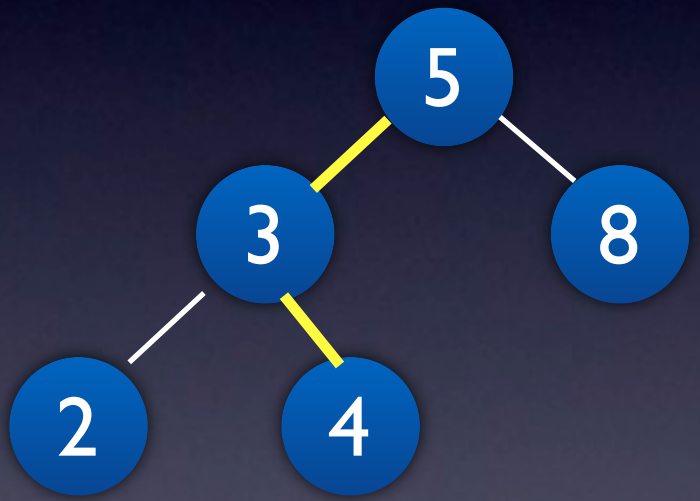
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

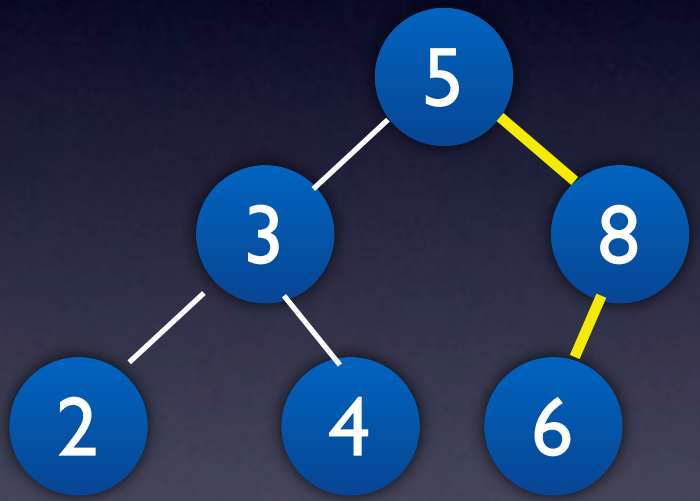
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

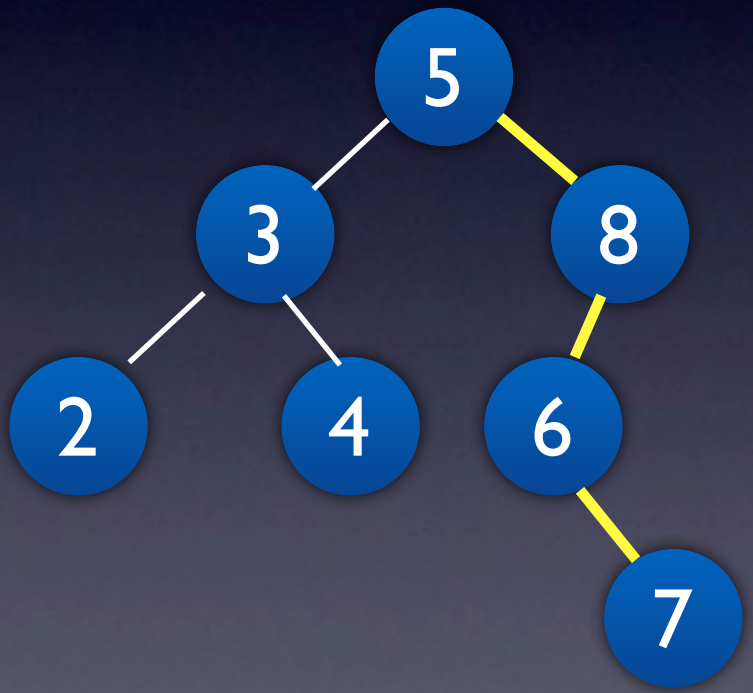
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

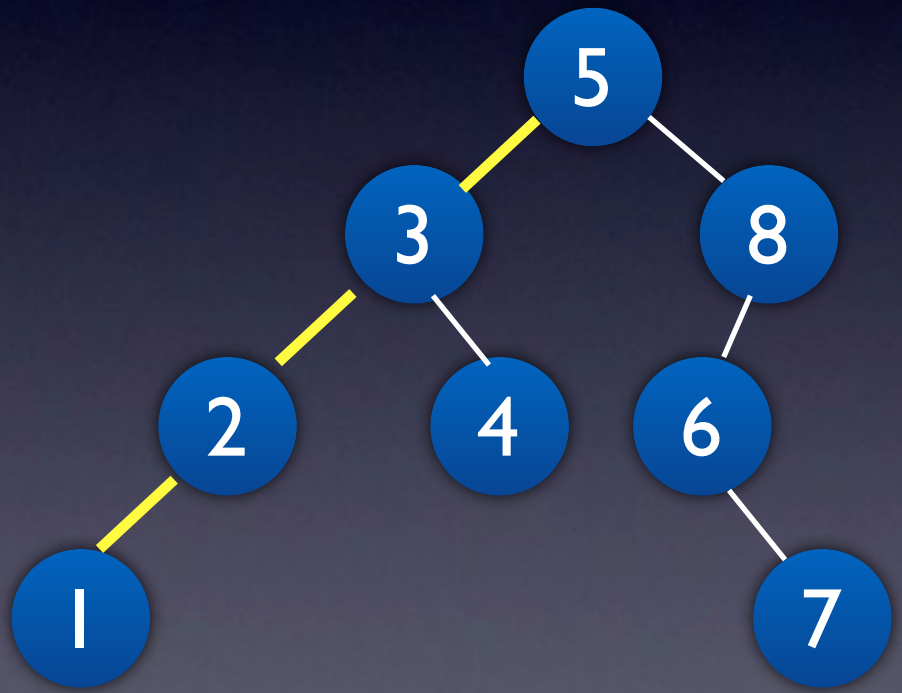
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

Exemplo:

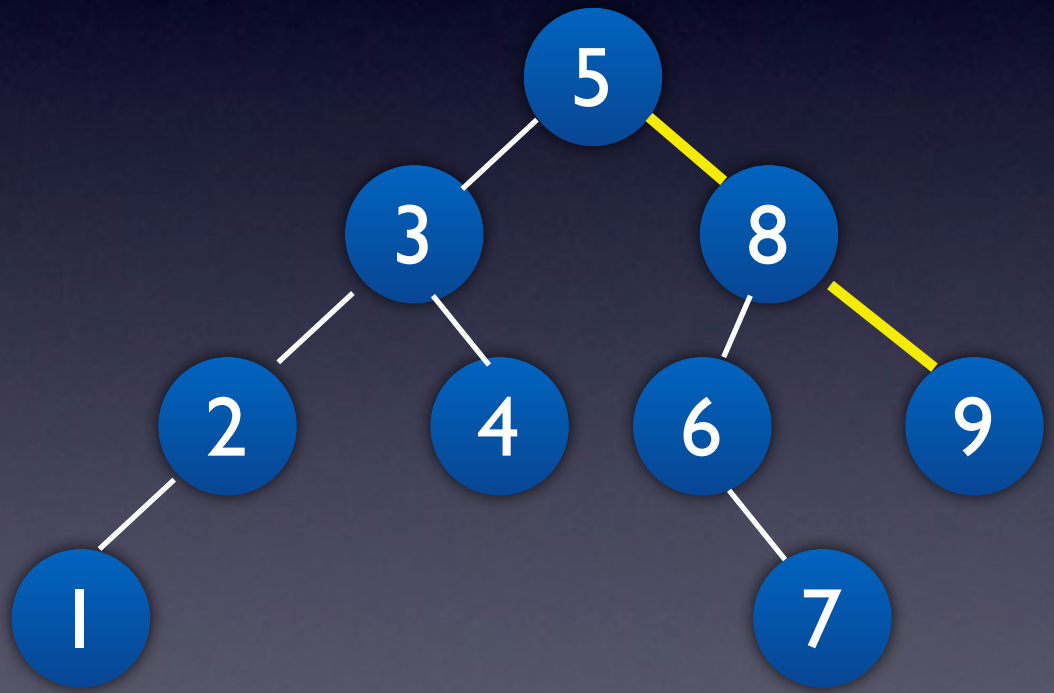
Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



Árvore Binária de Busca

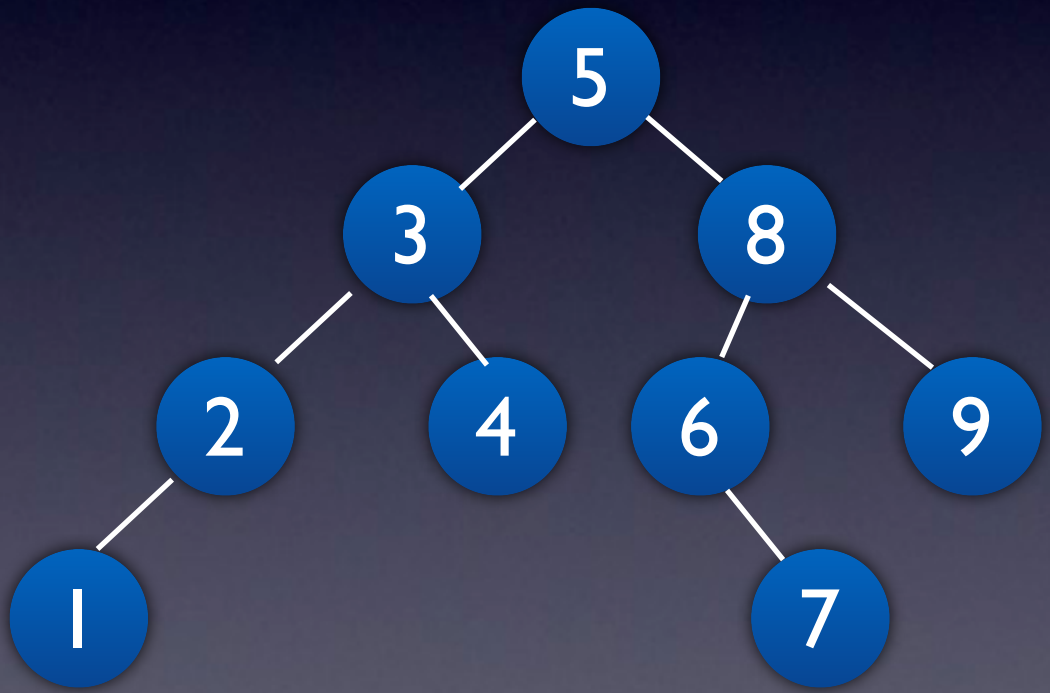
Exemplo:

Vamos inserir os números 5, 8, 3, 2, 4, 6, 7, 1, 9, nessa ordem.



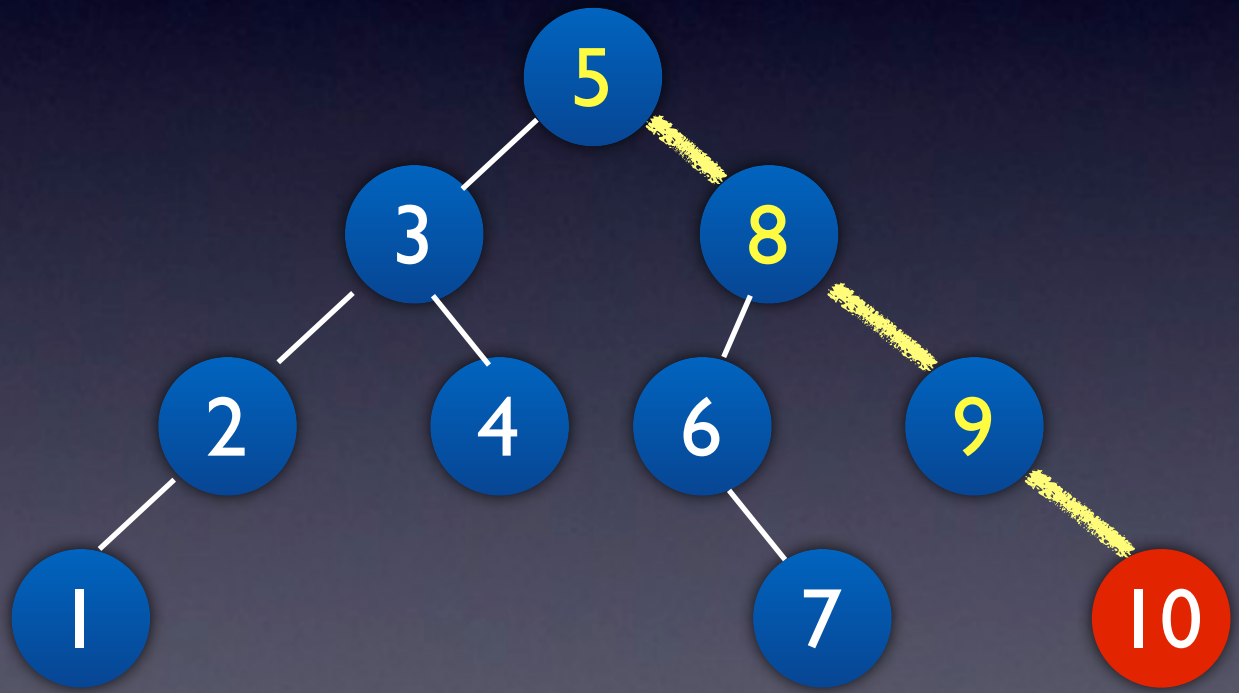
Árvore Binária de Busca

Exemplo:
Vamos agora inserir a chave de valor 10.



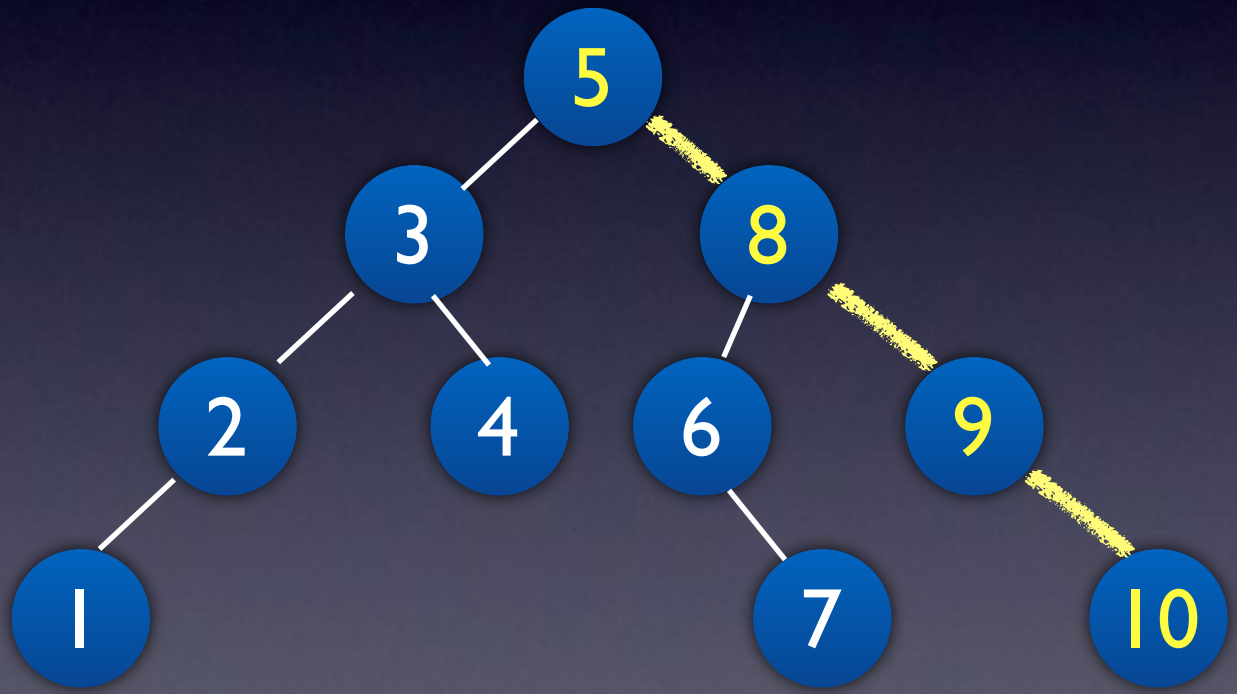
Árvore Binária de Busca

Exemplo:
Vamos agora inserir a chave de valor 10.



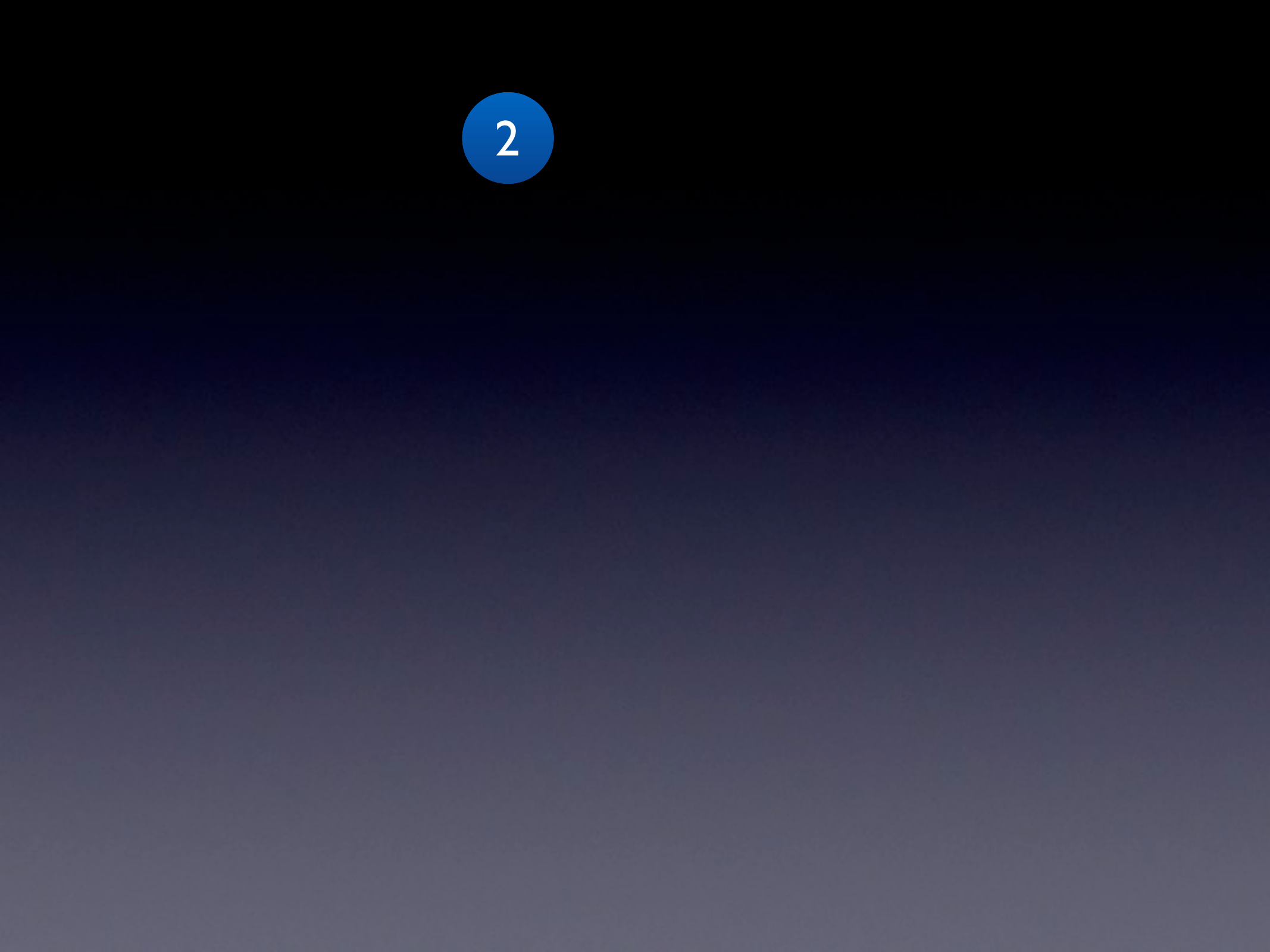
Árvore Binária de Busca

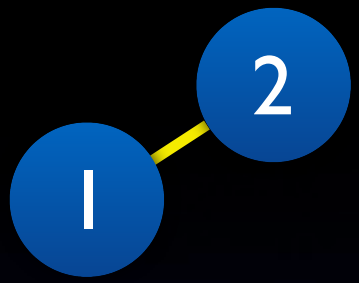
Note que tivemos que fazer um pequeno percurso, com poucas comparações e a inserção ocorreu de forma bem rápida.

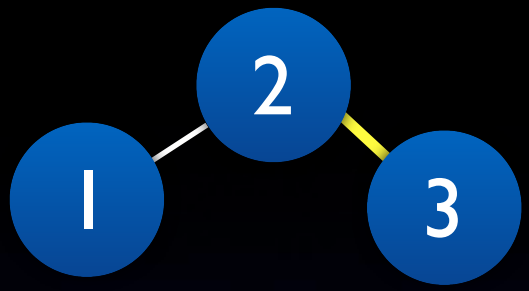


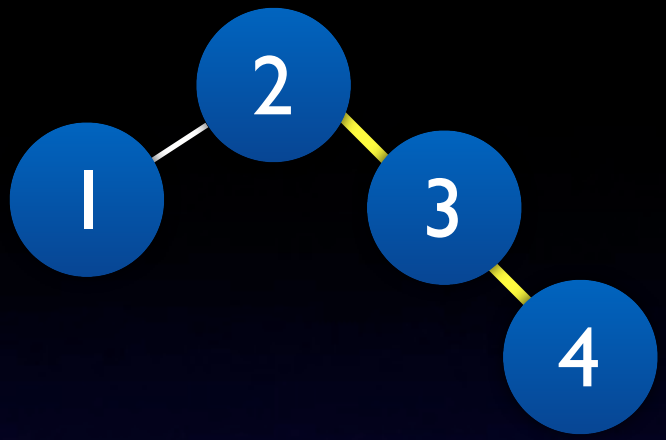
Árvore Binária de Busca

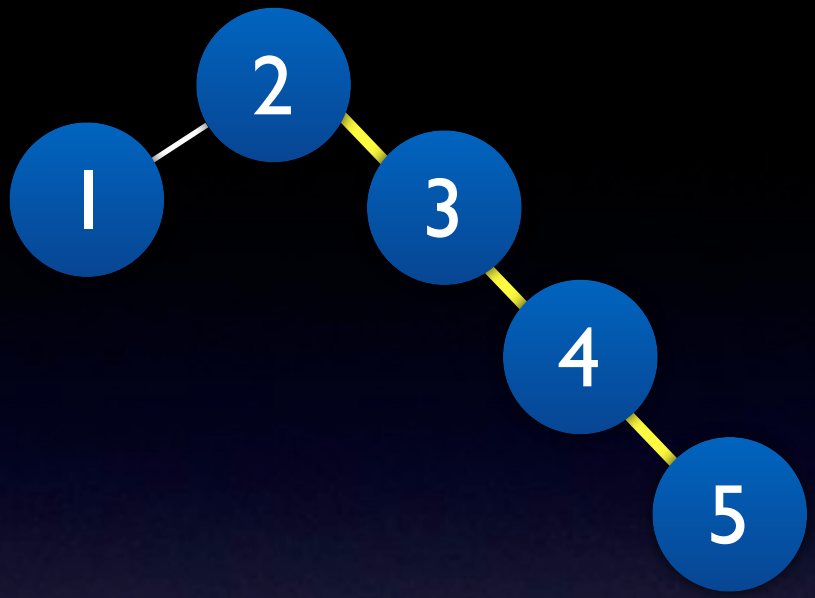
Agora vamos inserir os mesmos números, porém em outra ordem: 2,1,3,4,5,6,7,8,9

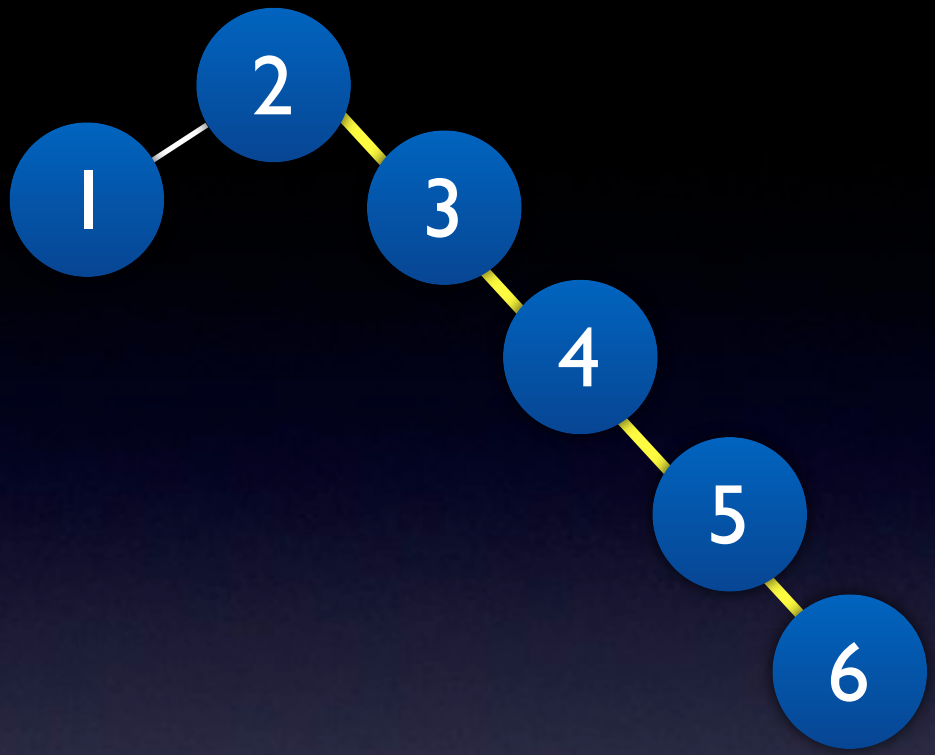

2

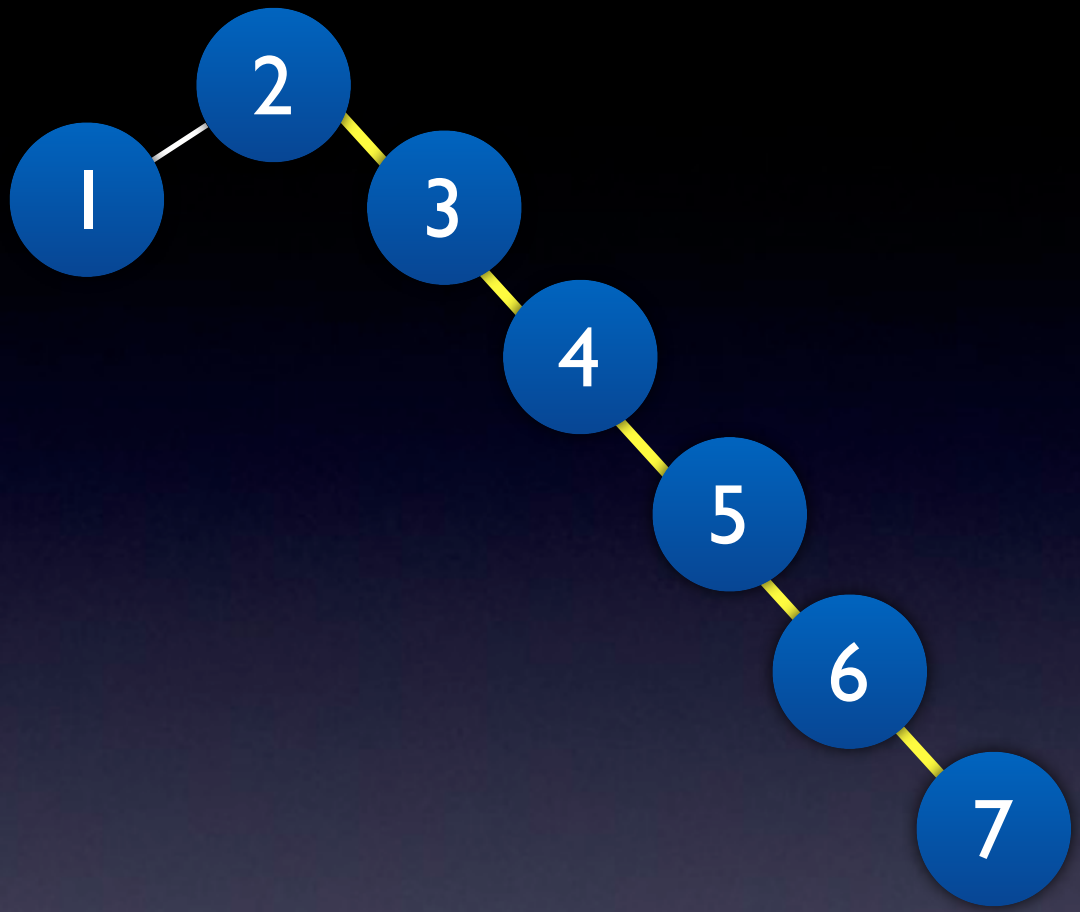

2
1

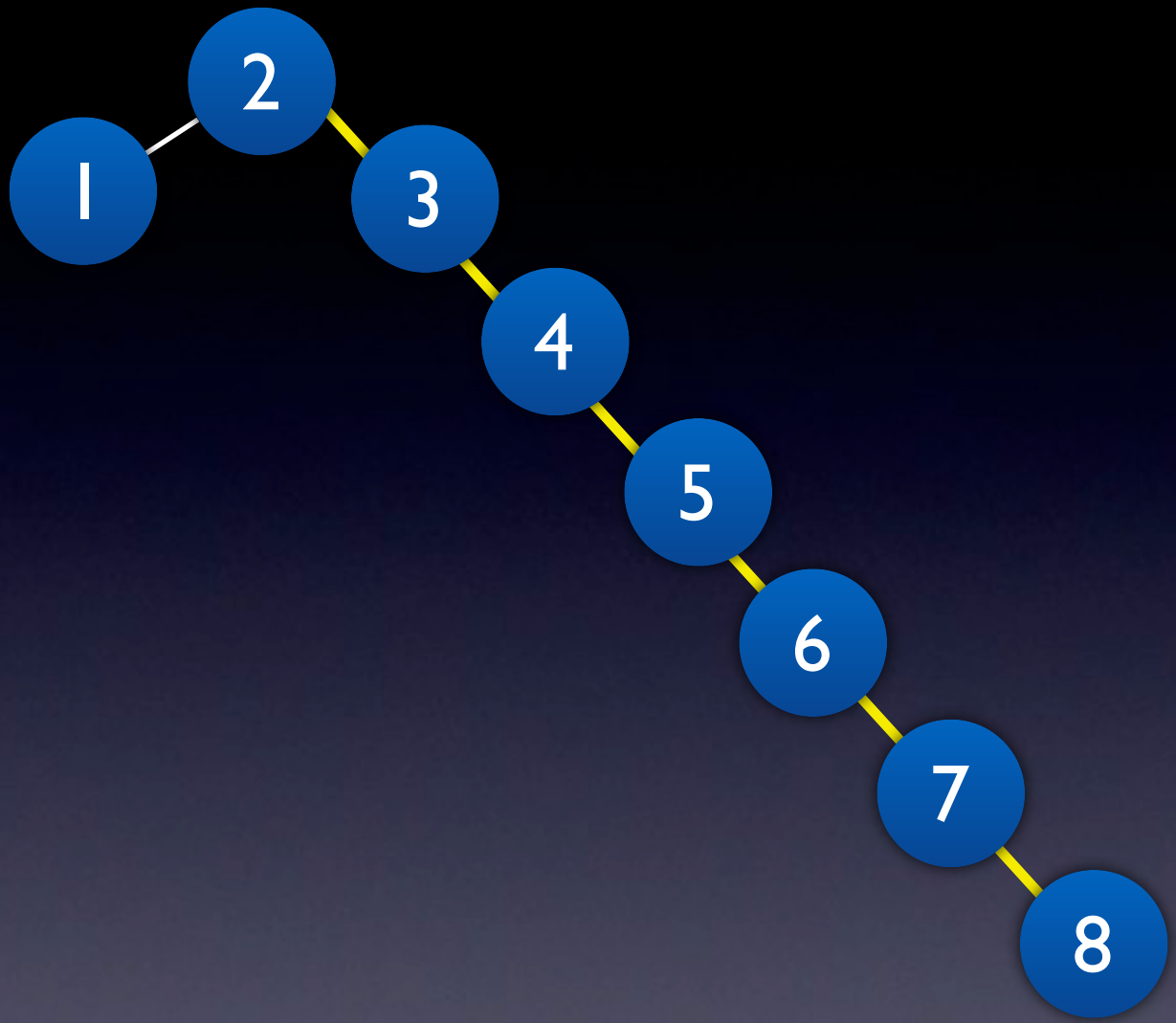

2
1
3

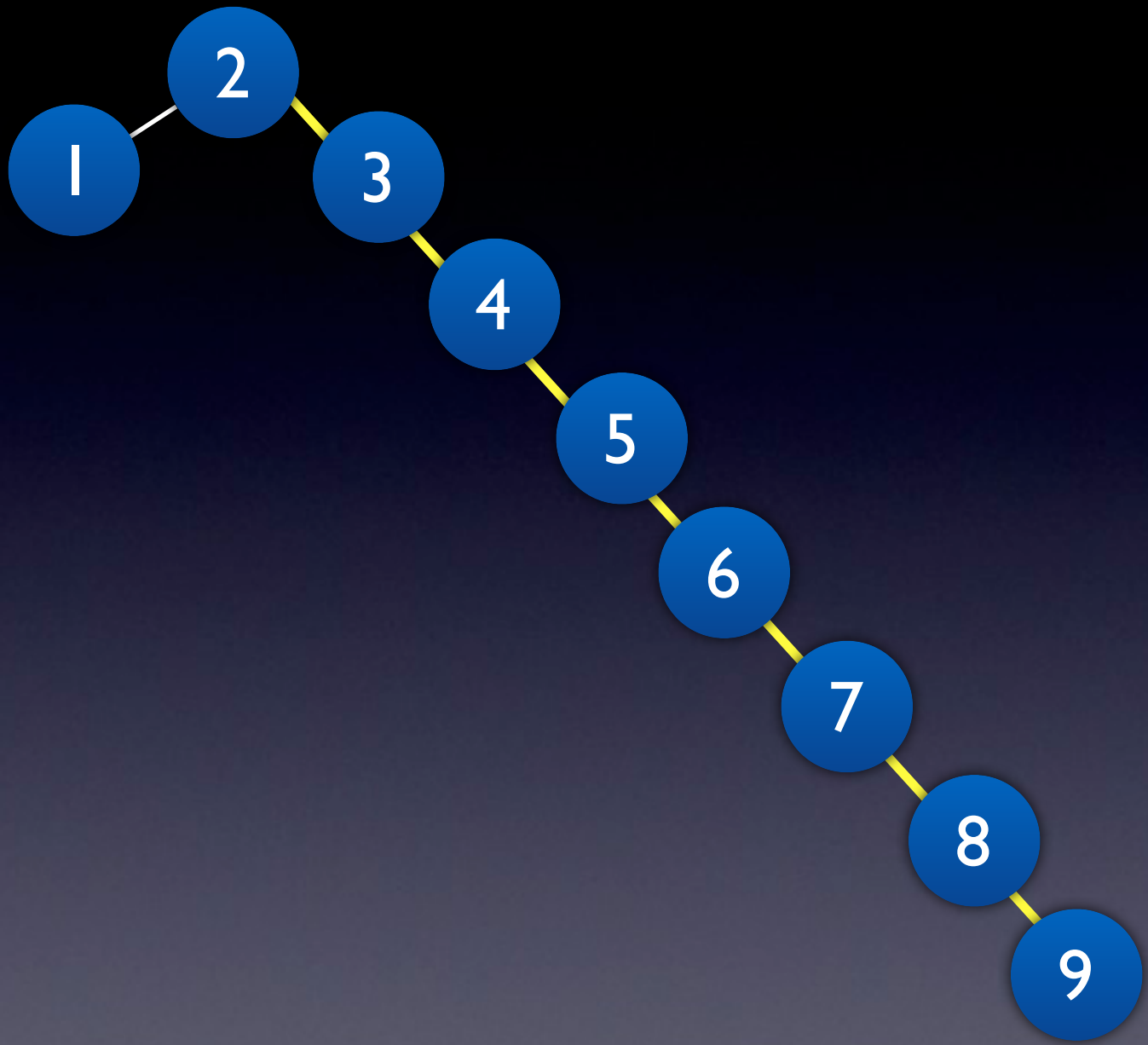

2
1
3
4

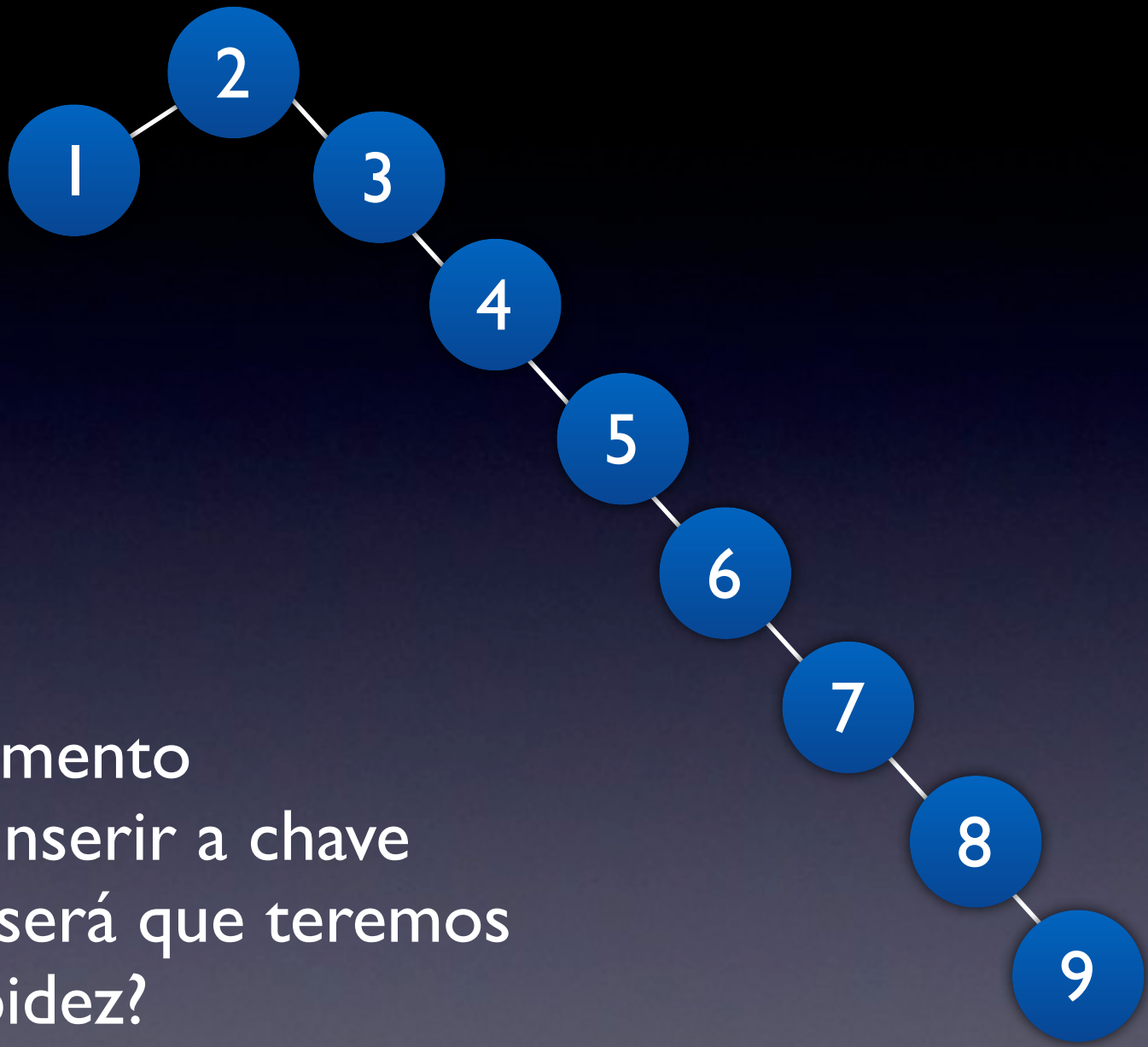

2
1
3
4
5


2
1
3
4
5
6

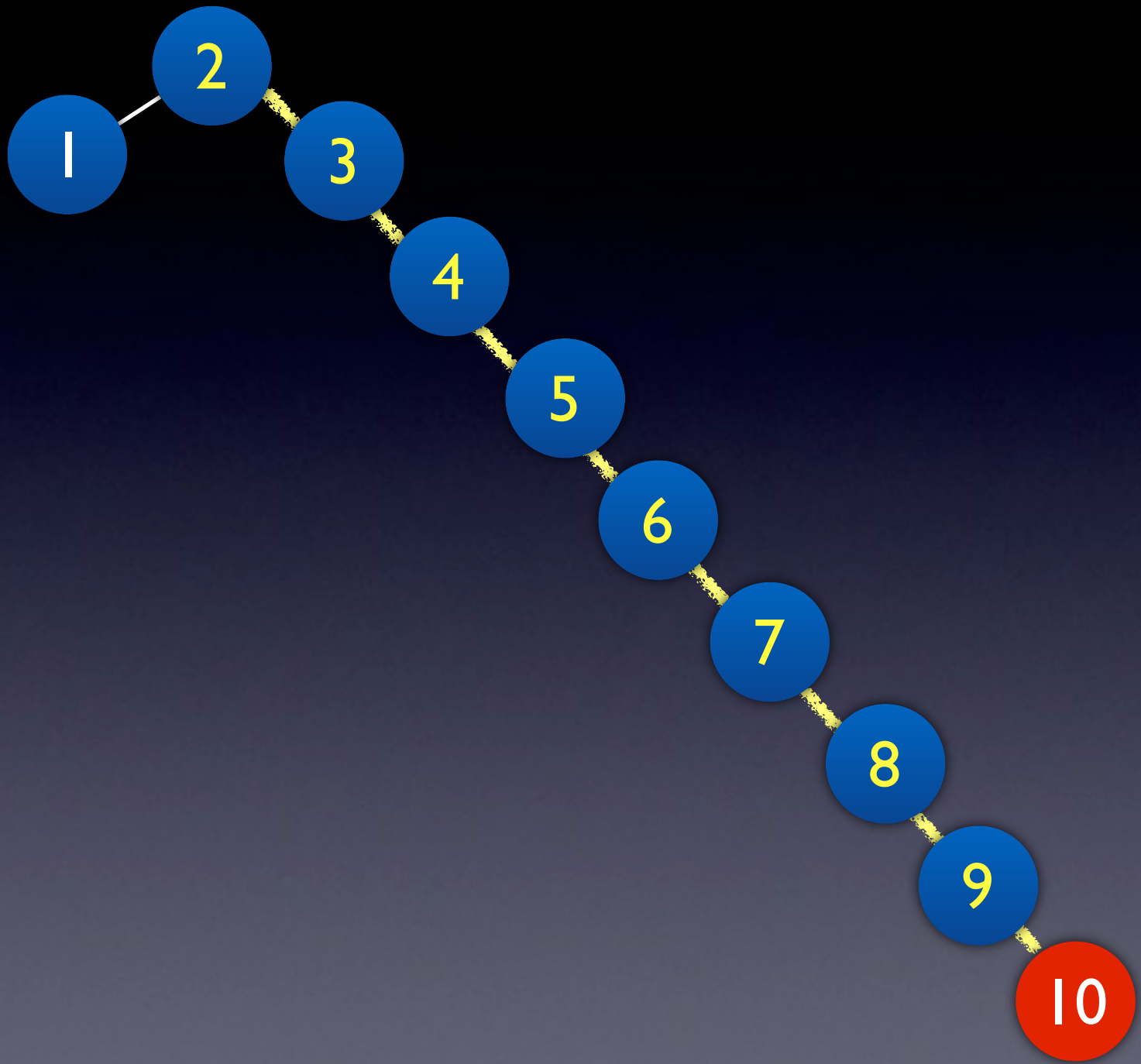

2
1
3
4
5
6
7

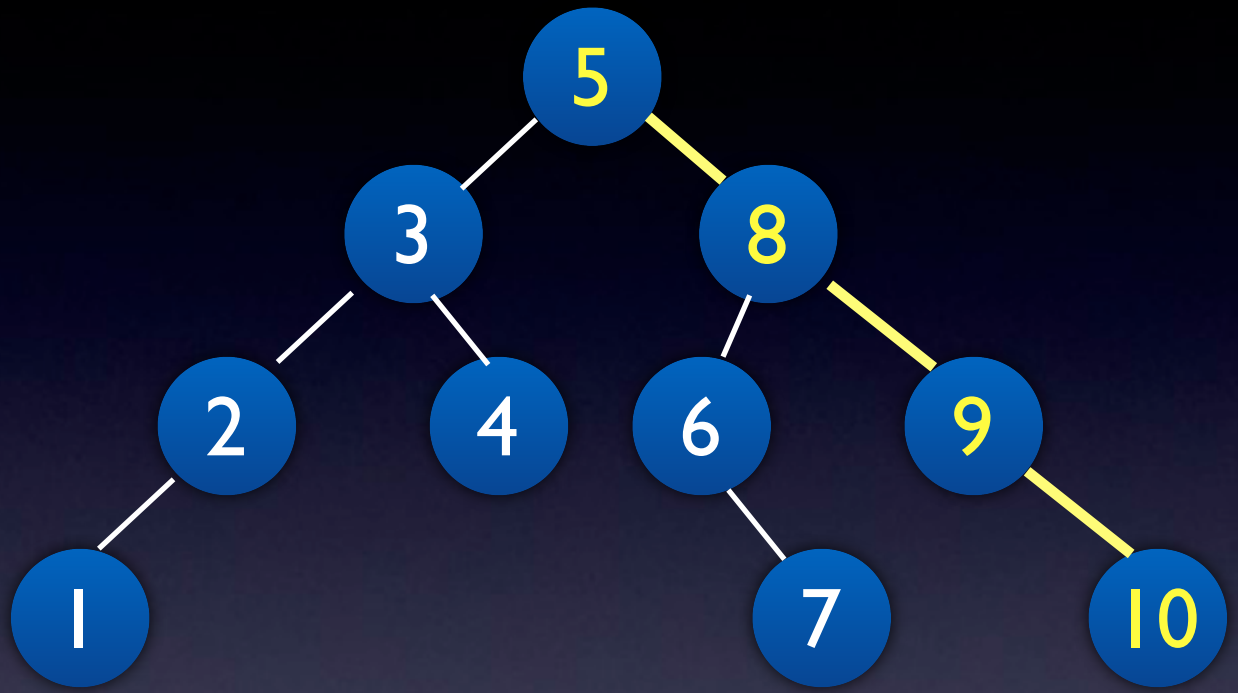

2
1
3
4
5
6
7
8


2
1
3
4
5
6
7
8
9



Se nesse momento
desejarmos inserir a chave
de valor 10, será que teremos
a mesma rapidez?


2
1
3
4
5
6
7
8
9
10



Nesta árvore, que está balanceada, tivemos que dar poucos passos para chegar na posição desejada.

Nesta outra, que não está balanceada, tivemos que fazer um percurso muito maior.

Conclusão:

Árvores binárias são mais eficientes quando balanceadas.

Problema:

Como garantir o balanceamento de uma árvore?

É aí que surge a nossa incrível árvore AVL!!!

O termo AVL vem de seus criadores, Adelson Velsky e Landis.

Definição

É uma árvore binária de busca auto-balanceada onde as alturas das duas sub-árvores a partir de cada nó difere no máximo em uma unidade.

As operações de busca, inserção e eliminação de elementos possuem complexidade O(log n).

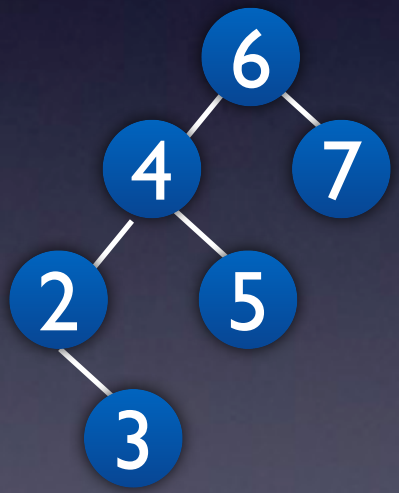
Definição

Ao inserir ou eliminar elementos, a árvore terá de ser verificada para que, caso necessário, seja feito um rebalanceamento, que ocorre por meio de rotações.

Vamos ver um pouco da implementação em Java...

Entendendo as rotações

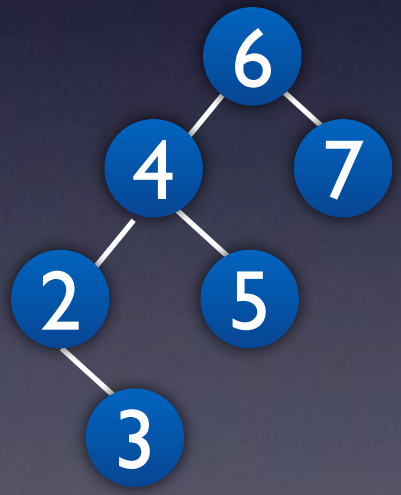
Vejamos essa árvore:



Entendendo as rotações

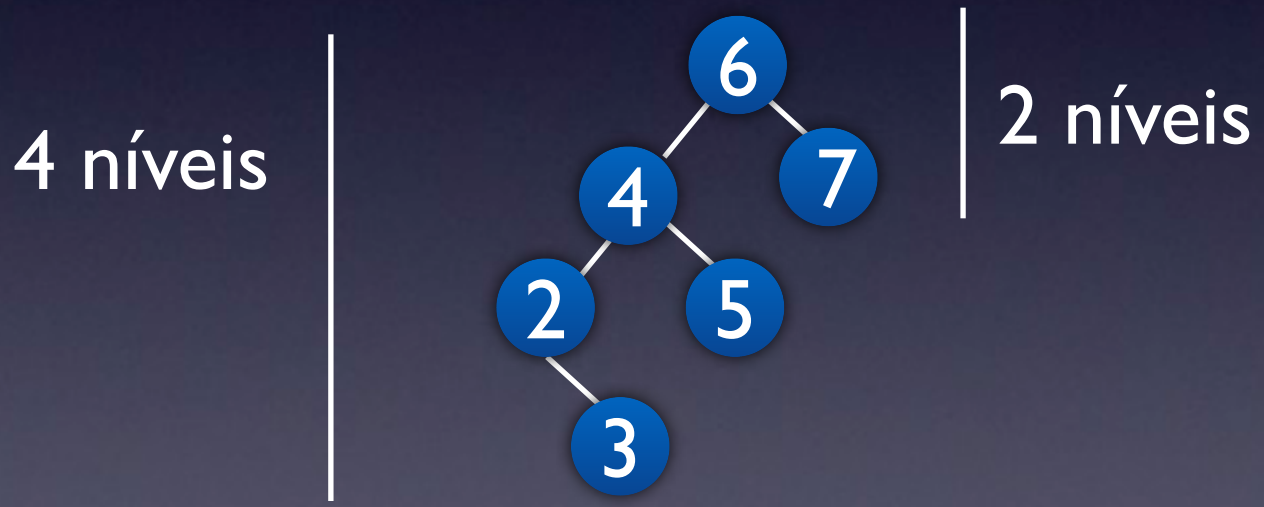
Rotação Simples

A altura da sub-árvore de raiz 4 difere de mais de 1 unidade da de chave 7.



Entendendo as rotações

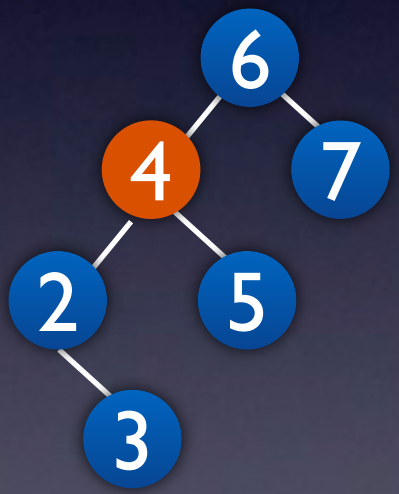
Como a altura da sub-árvore à esquerda de 6 difere de 2 em relação à sua sub-árvore direita, rotaciona-se em torno do 4.



lado esquerdo desbalanceado -> rotação para direita

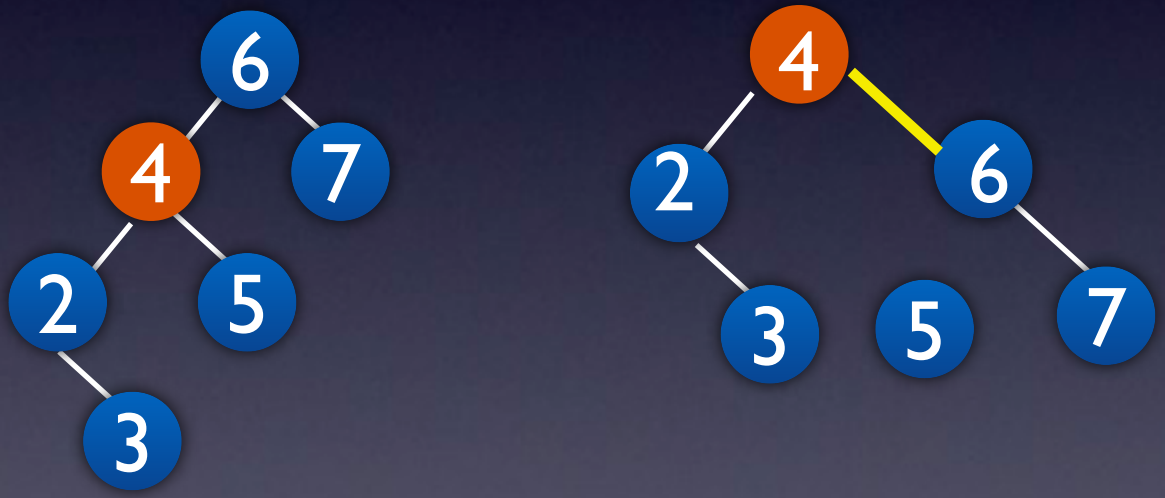
Entendendo as rotações

4 passa a ser a nova raiz



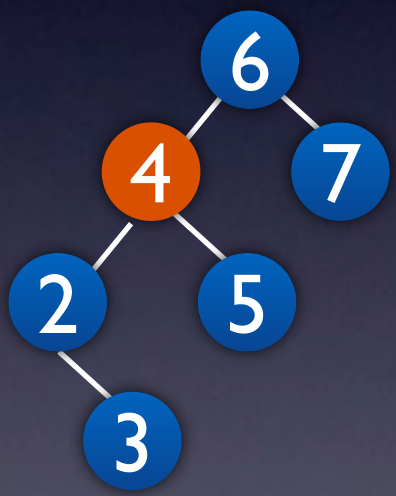
Entendendo as rotações

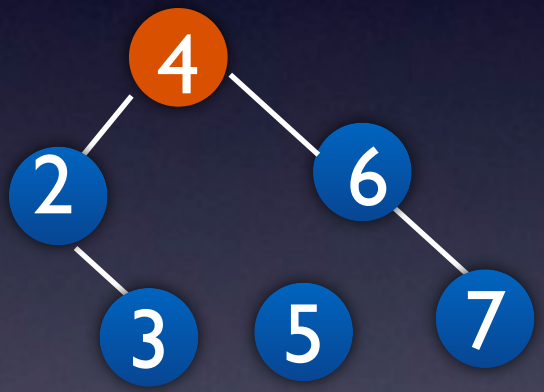
sub-árvore enraizada em 6 vira filho direito de 4

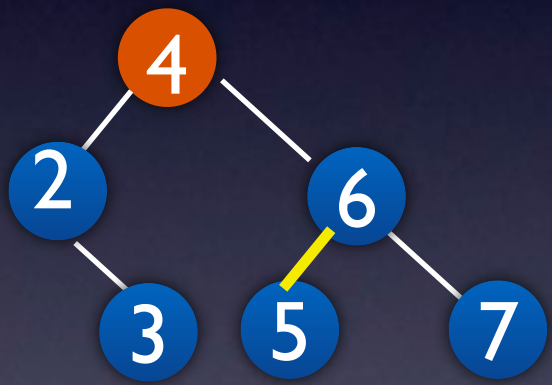


Entendendo as rotações

5 passa a ser filho esquerdo de 6

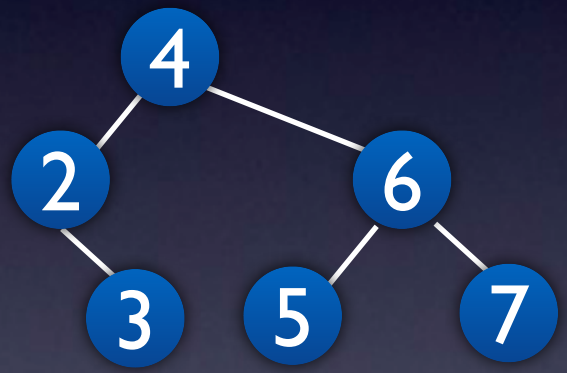






Entendendo as rotações

Resultado Final



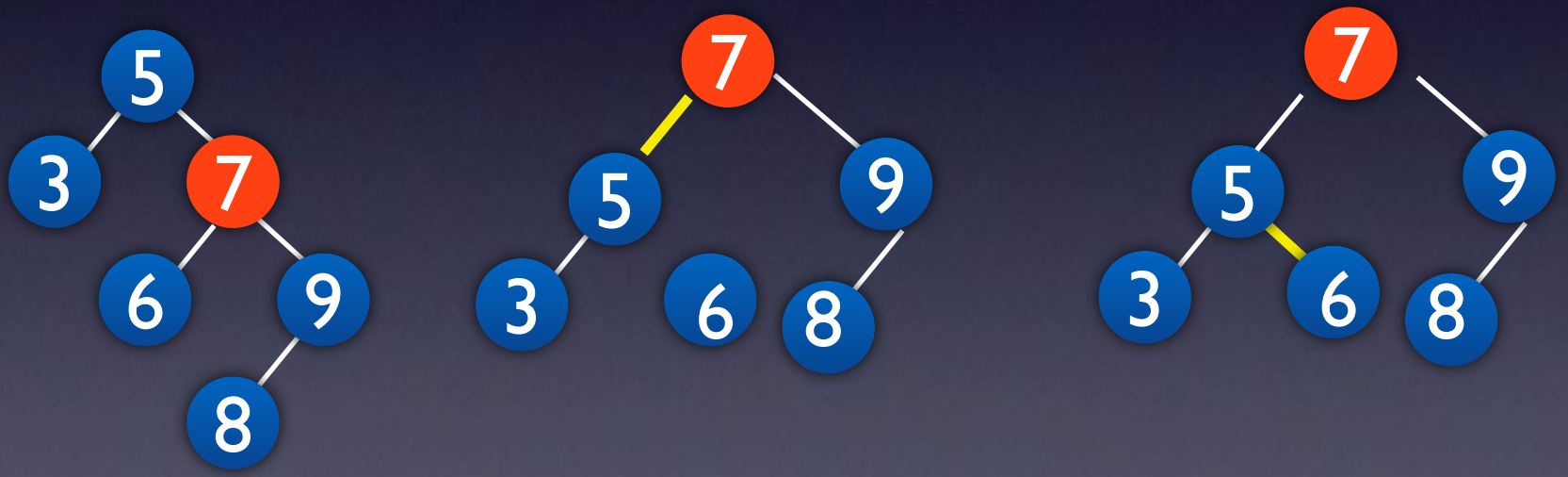
Balanceada!

Entendendo as rotações

Uma rotação à esquerda segue o mesmo algoritmo porém invertendo direita com esquerda.

Entendendo as rotações

A altura da sub-árvore de raiz 7 difere de mais de 1 unidade da de chave 3.



Entendendo as rotações

Generalizando...

Rotação à Direita

Seja Y o filho à esquerda de X
Torne X o filho à direita de Y
Torne o filho à direita de Y o filho à esquerda de X.

Rotação à Esquerda

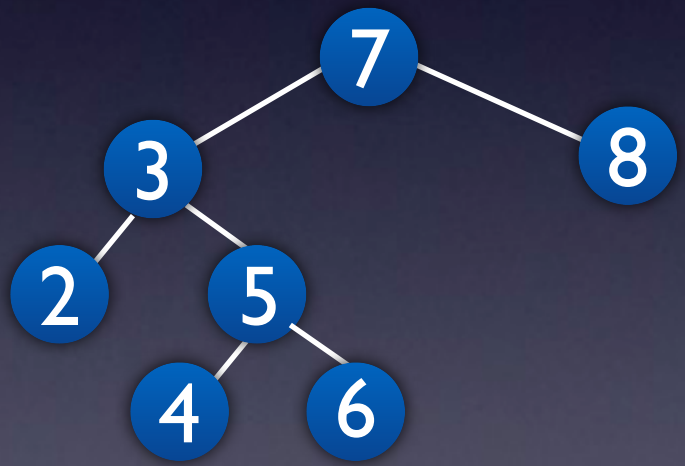
Seja Y o filho à direita de X
Torne X filho à esquerda de Y
Torne o filho à esquerda de Y o filho à direita de X.

Entendendo as rotações

Mas nem sempre apenas uma rotação é suficiente para balancear a árvore...

Entendendo as rotações

Como balancear essa árvore ?



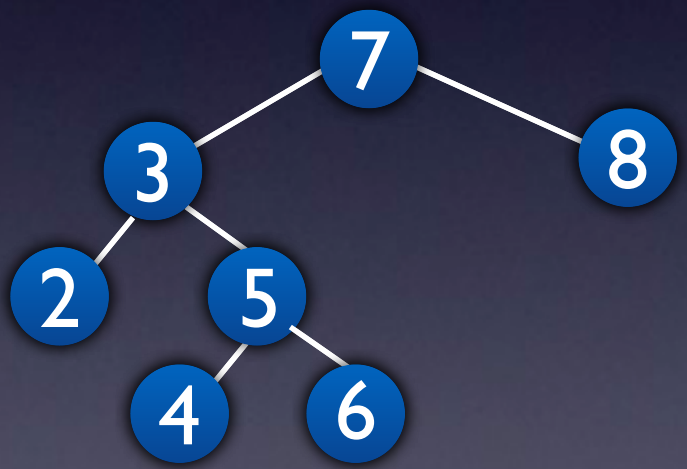
Entendendo as rotações

Rotação Dupla

Se um nó estiver desbalanceado e seu filho estiver inclinado no sentido inverso ao pai, teremos a ocorrência de uma rotação dupla.

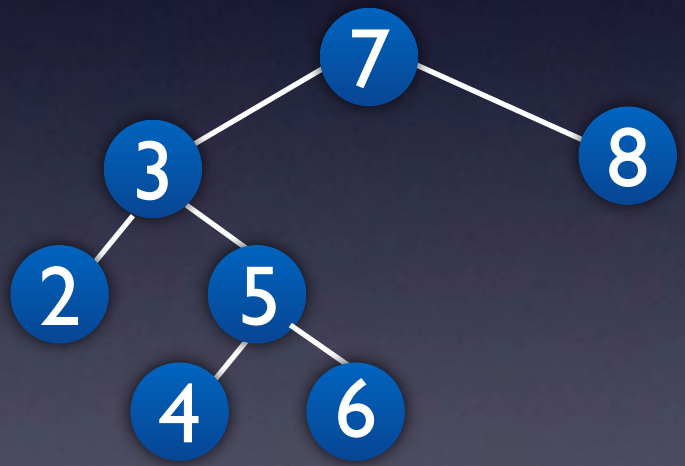
Na verdade, trata-se apenas de duas rotações simples seguidas.

Entendendo as rotações



Entendendo as rotações

Se um nó estiver desbalanceado e seu filho estiver inclinado no sentido inverso ao pai, teremos a ocorrência de uma rotação dupla.



Entendendo as rotações

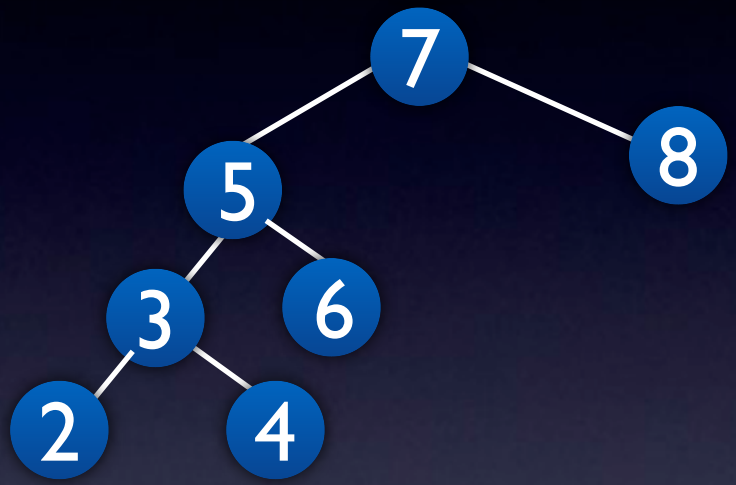
Se um nó estiver desbalanceado e seu filho estiver inclinado no sentido inverso ao pai, teremos a ocorrência de uma rotação dupla.

Entendendo as rotações

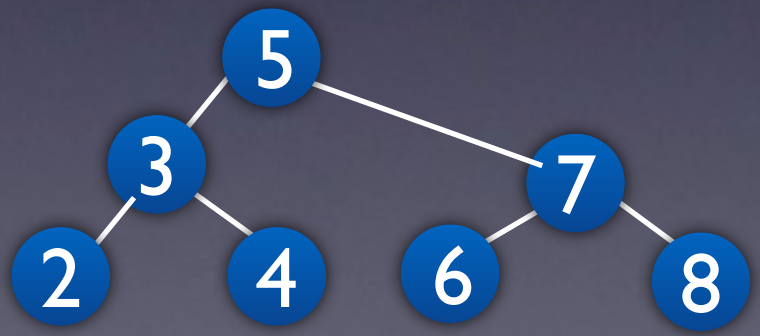
Se um nó estiver desbalanceado e seu filho estiver inclinado no sentido inverso ao pai, teremos a ocorrência de uma rotação dupla.

seu filho, o 6, está no sentido inverso do 7, seu pai

Rotação Dupla à direita



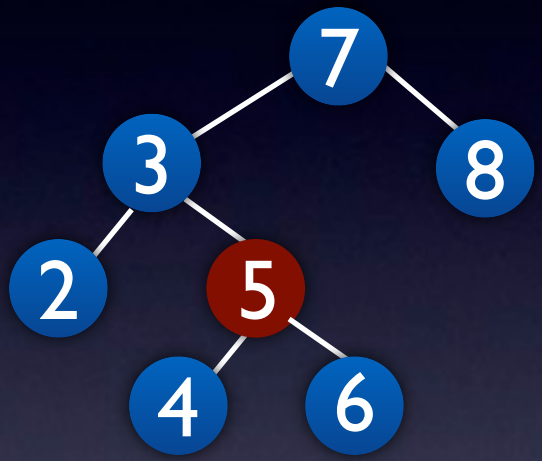
Rotação Simples à esquerda



Rotação Simples à direita

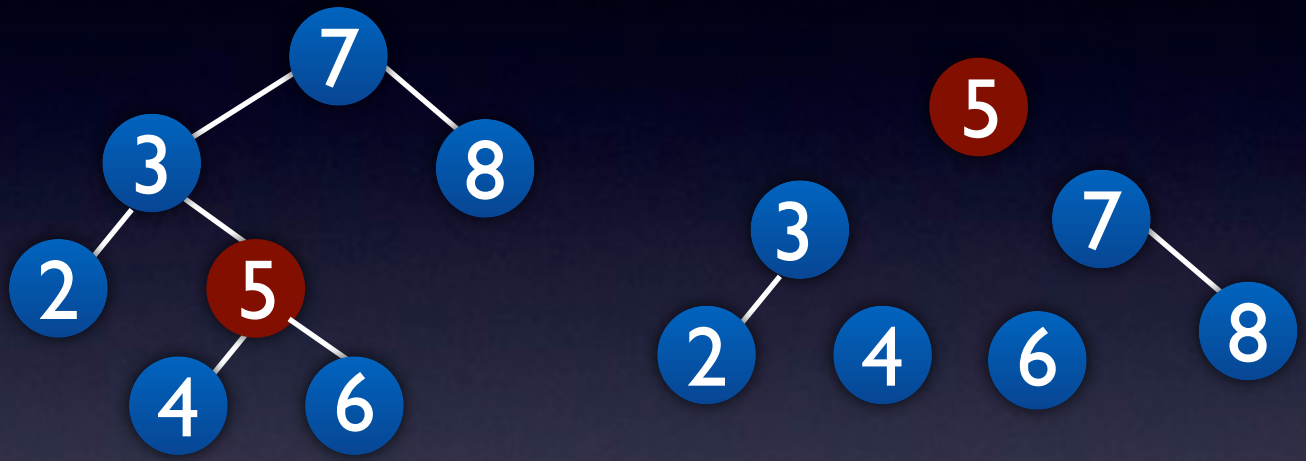
Rotação Dupla à direita

5 será a nova raiz da sub-árvore enraizada em 7



Rotação Dupla à direita

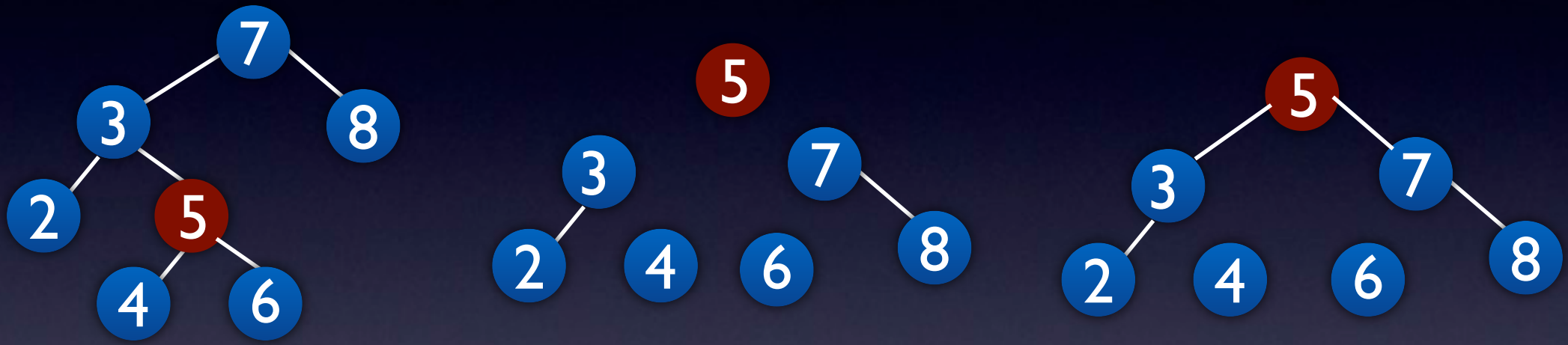
Como a nova raiz passa a ser o 5, o nó 3 passa a não ser mais filho de 7



Além disso, as arestas de 4 e 6 com 5 são desfeitas

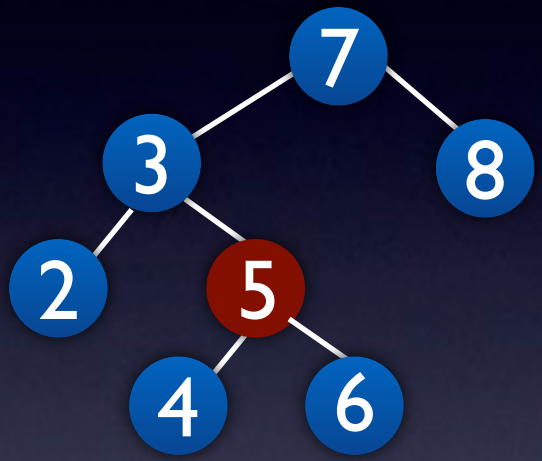
Rotação Dupla à direita

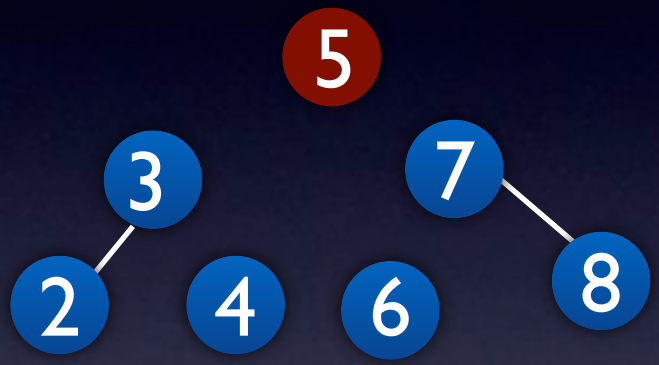
As sub-árvores de raizes 3 e 7 se ligam a 5

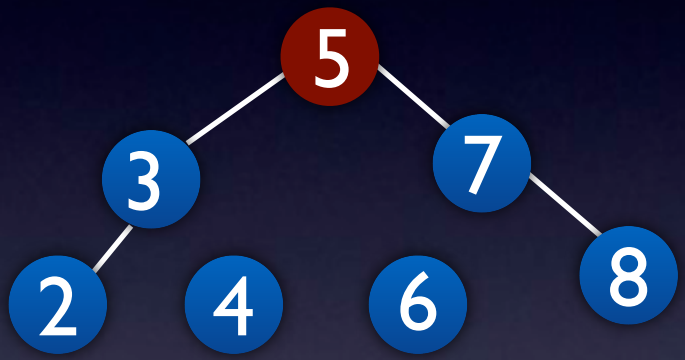


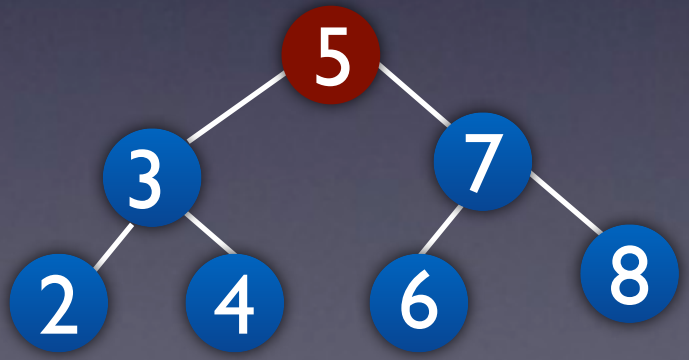
Rotação Dupla à direita

4 se liga a 3 e 6 se liga a 7









Árvore Balanceada!

Vamos ver agora as rotações em Java...

Inserção

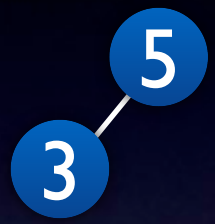
A inserção em uma árvore AVL ocorre da mesma forma que a inserção na árvore binária, porém, levando em conta o balanceamento da árvore, ou seja, realizando as operações de rotação quando necessárias.

Exemplos

Insere 5

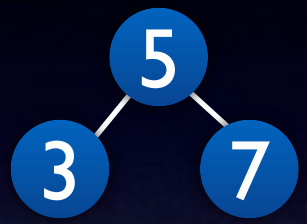
Exemplos

Insere 3 - balanço correto - difere 1



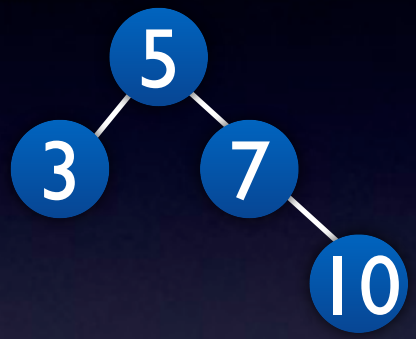
Exemplos

Insere 7 - balanço correto - difere 0



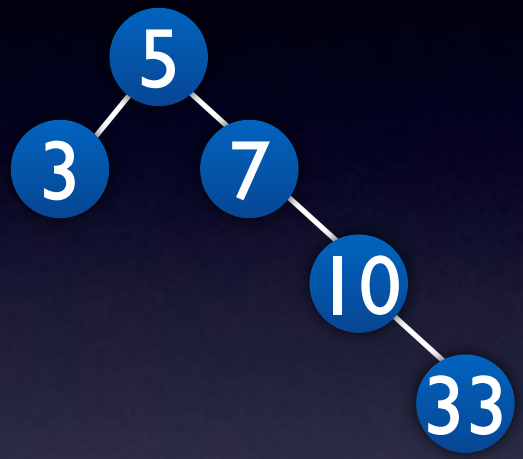
Exemplos

Insere 10 - balanço correto - difere 1



Exemplos

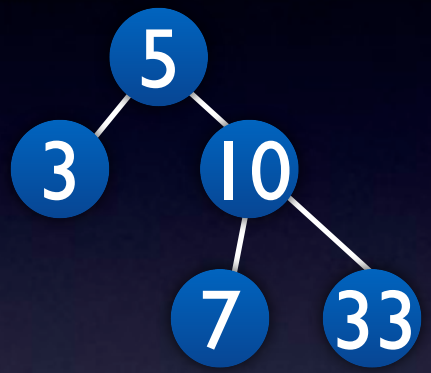
Insere 33 - balanço errado - difere 2



Neste caso, realizar rotação simples à esquerda

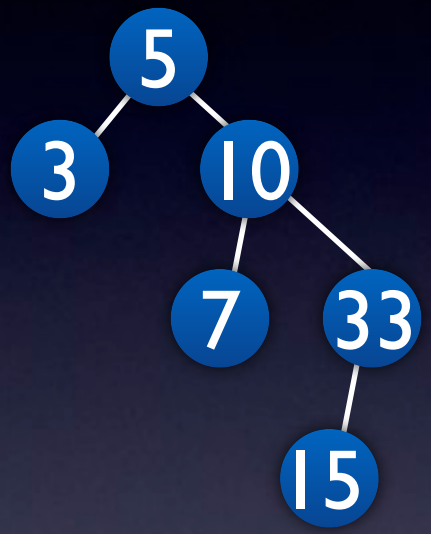
Exemplos

balanço correto - difere 1



Exemplos

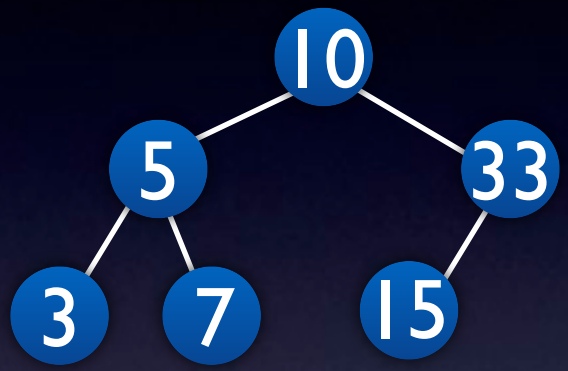
Insere 15 - balanço incorreto - difere 2



Neste caso, realizar rotação simples à esquerda

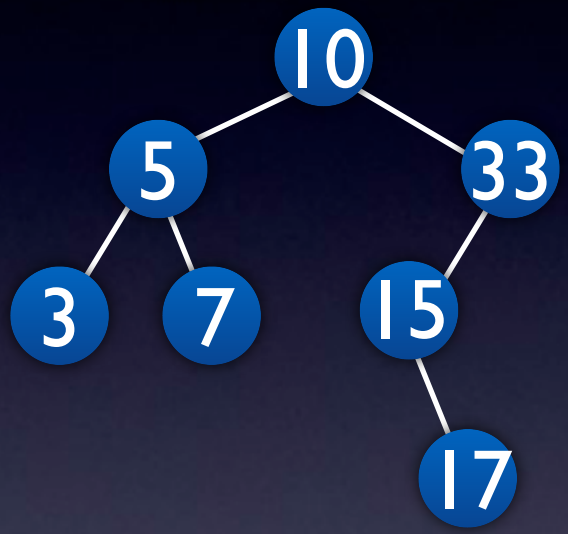
Exemplos

Agora o balanço está correto - difere 1.



Exemplos

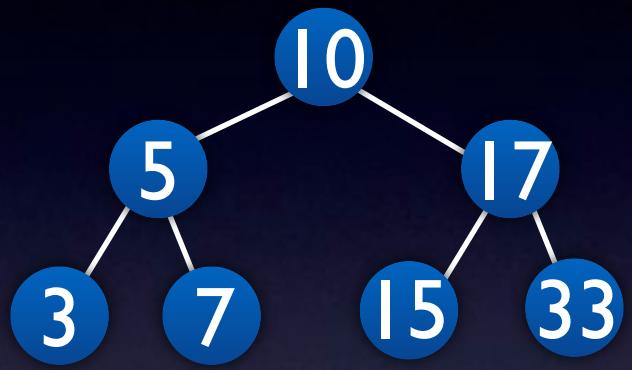
Insere 17 - balanço incorreto - difere 2



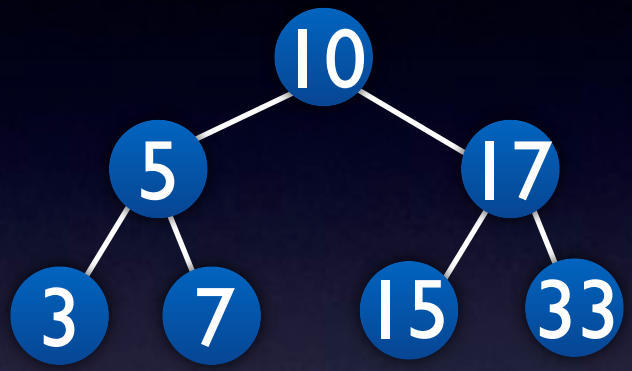
Neste caso, realizar rotação dupla à direita

Exemplos

balanceada - diferença de zero nas alturas



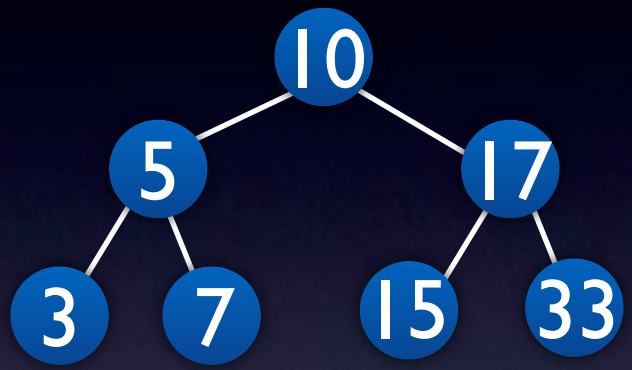
Exemplos



Isto é uma AVL balanceada.

Vamos ver o algoritmo de inserção em nosso programa em Java...

Continuando...



Agora vamos brincar com a remoção...

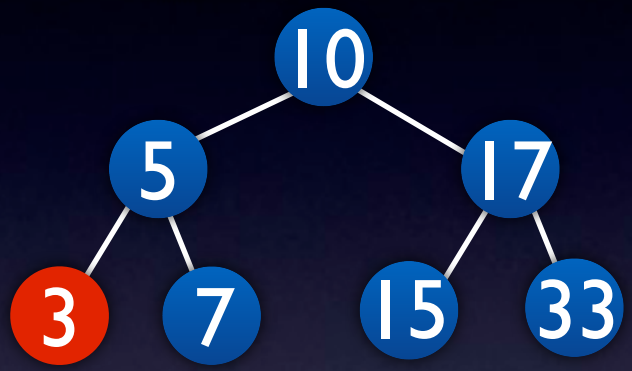
Remoção

Remover um nó de uma árvore AVL consiste em remover diretamente o nó caso ele seja uma folha e verificar o balanceamento realizando as rotações quando necessárias.

Caso o nó não seja uma folha troca-se ele pelo maior nó menor que ele (uma folha) e então prossegue como explicado acima.

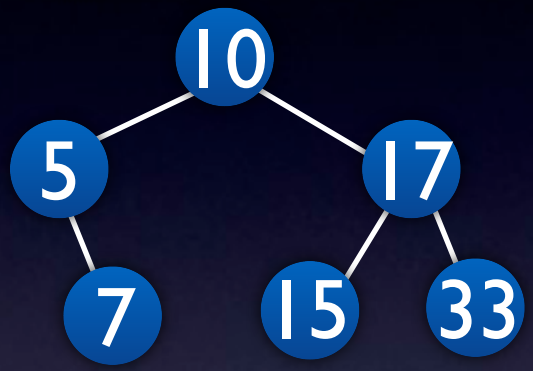
Exemplos

Remove 3



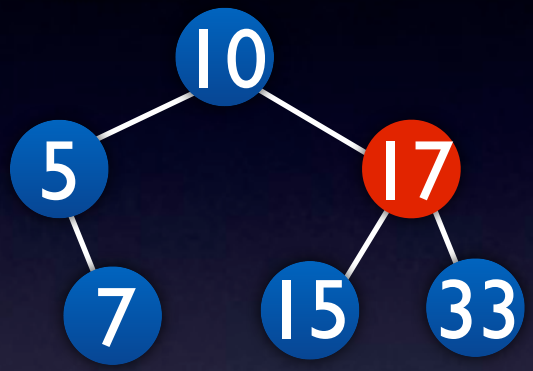
Exemplos

OK, balanceada (fácil)



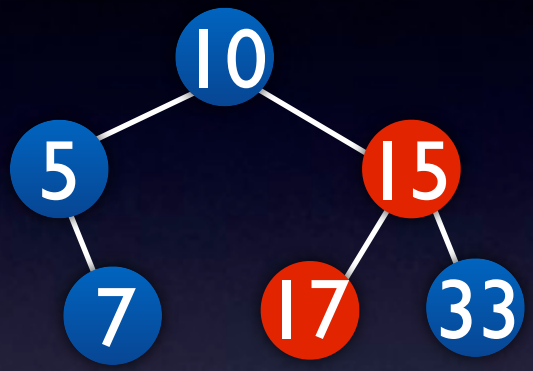
Exemplos

Agora vamos remover o 17...



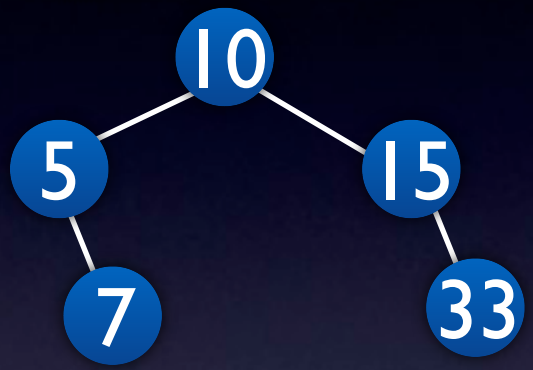
Exemplos

Agora vamos remover o 17...



Exemplos

Foi trocado pela chave mais à direita do seu filho esquerdo.

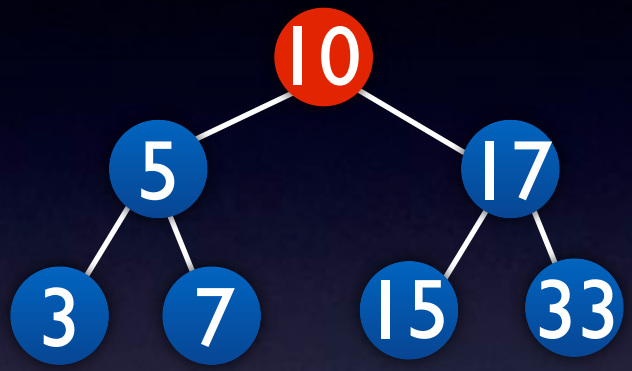


Vamos complicar um pouco...

Voltando a árvore balanceada antes das remoções...

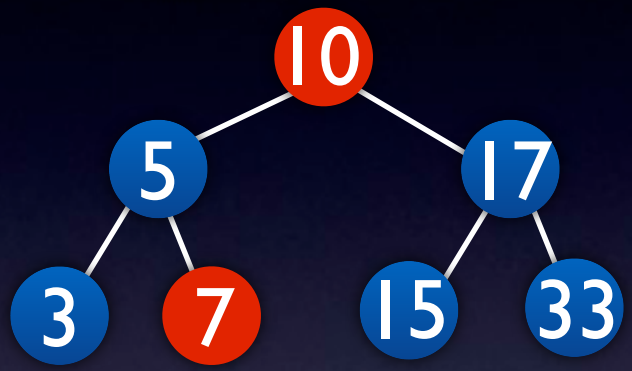
Exemplos

Remover o 10!



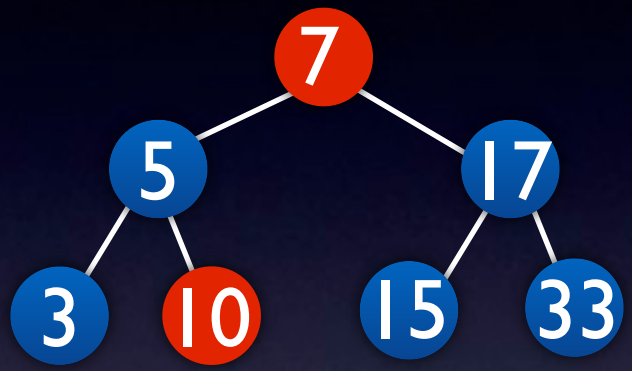
Exemplos

7 é o nó mais à direita do filho esquerdo de 10



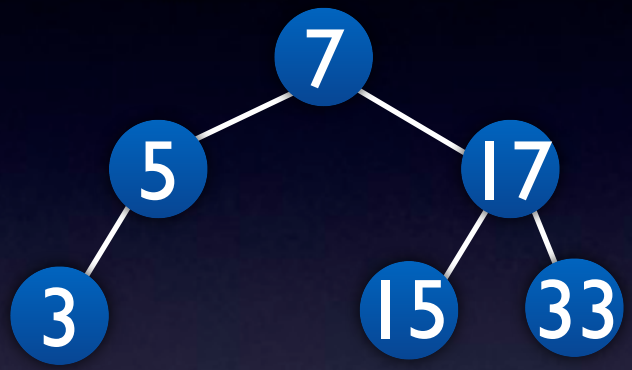
Exemplos

Invertemos as posições e excluimos a folha 10



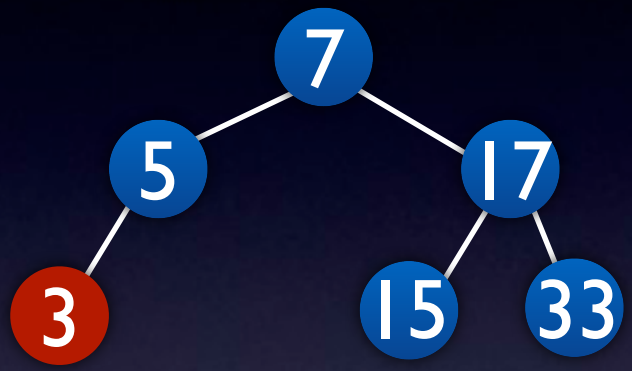
Exemplos

7 vira raiz e a árvore está balanceada.



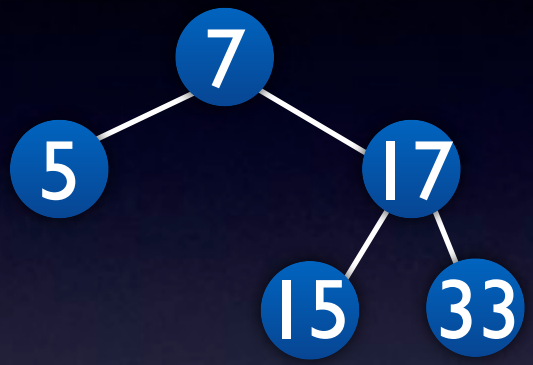
Exemplos

Agora vamos remover o 3



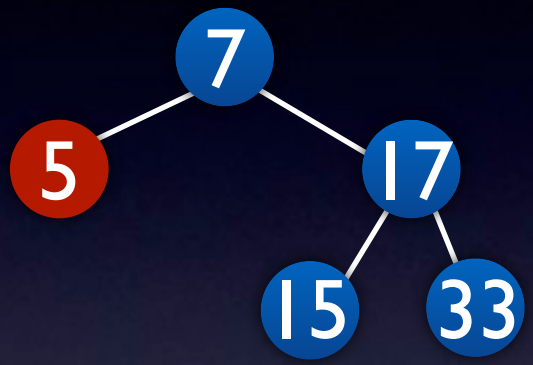
Exemplos

Balanceada - difere de 1



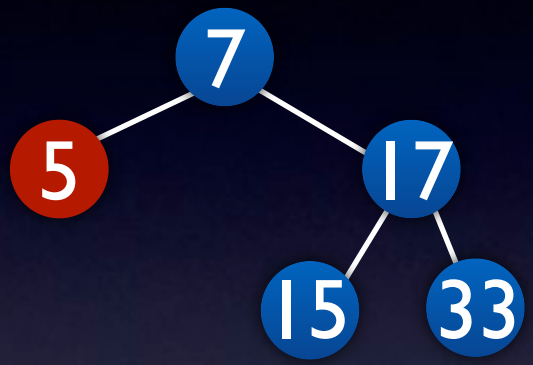
Exemplos

Agora vamos remover o 5



Exemplos

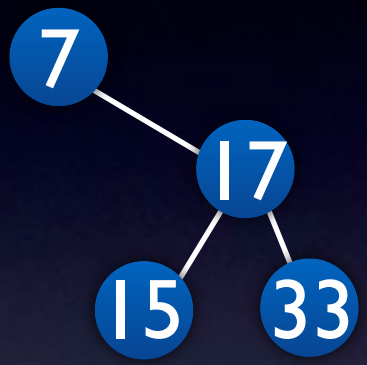
Agora vamos remover o 5



Note que agora teremos que fazer rotações, pois ficará desbalanceada.

Exemplos

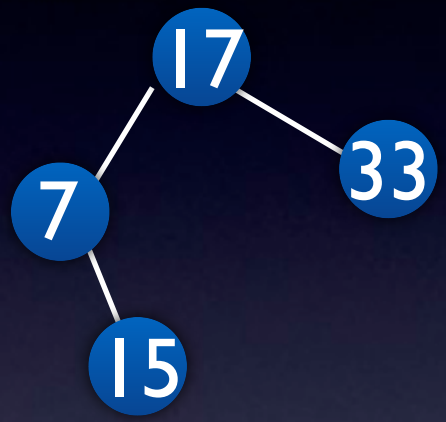
Balanceamento errado - difere 2



Faremos uma rotação simples à esquerda.

Exemplos

Balanceamento correto - difere 1



Agora está novamente balanceada.

E por fim, o algoritmo de remoção...

O desempenho das operações numa árvore pode ser medido pela sua altura.

Árvore	Altura Máxima
A.Bin.Busca	n
Rubro-Negra	$2 * \log_2(n)$
AVL	$1.44 * \log_2(n)$

Árvore	Complexidade
A.Bin.Busca	O(n)
Rubro-Negra	$O(\log_2(n))$
AVL	O(log2(n))

Vantagens

Árvores AVL são mais rápidas nas operações de busca.

Desvantagens

Perde-se tempo quando necessitamos fazer um rebalanceamento. Isso pode ocorrer nas inserções e remoções.

Conclusão

Árvores AVL são bastante eficientes.

Vamos agora ver mais exemplos no programa que desenvolvemos...