Inserção em Árvores Binárias de Busca

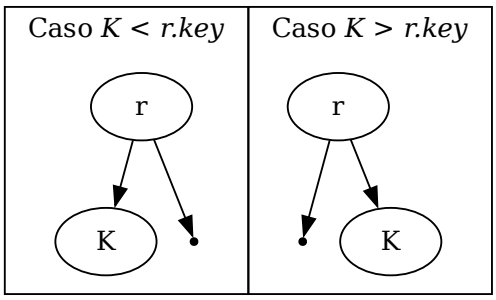
Algoritmos e Estruturas de Dados - 2025/02

Última atualização: 25/07/2025 11:02

Na última aula já discutimos ideias sobre a buscas em *ABB*s e vimos o que acontece quando o elemento buscado não está na árvore. Agora iremos inserir o valor K na árvore. Ainda não sabemos fazer isto, então vamos começar pensando em como fazer a inserção. Comecemos por organizar nosso raciocínio pela altura da árvore. Alguns casos são bem fáceis:

Árvore vazia: crie um nó com o $key = K$ e faça-o a nova raiz da árvore.

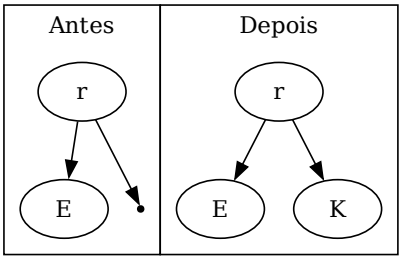
Árvores com altura $h = 0$: seja r esse único nó na árvore. Se $K < r.key$ então podemos só inserir na subárvore esquerda. Caso contrário ($K > r.key$), a inserção é na subárvore direita. Veja abaixo.



Caso *h=0*: após inserção da chave *K*

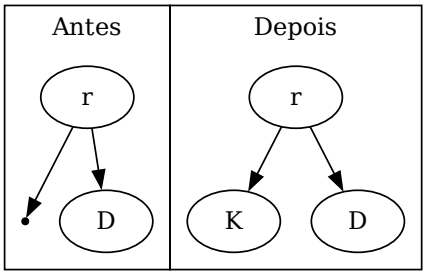
Árvores com altura $h = 1$: Vamos examinar algumas possibilidades abaixo:

1. só existe subárvore *esquerda* (como no exemplo abaixo) e $r.key < k$. Este caso é fácil!



Caso *h=1*: raiz só tem subárvore esquerda

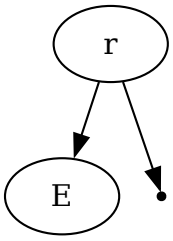
2. só existe subárvore **direita** e $r.key > K$. Este caso também é fácil!



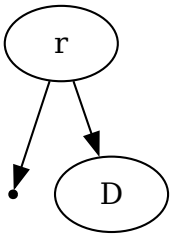
Caso *h=1*: raiz só tem subárvore direita

3. Existe subárvore esquerda e $r.key > K$. Ou seja, a busca iria para a esquerda em r, porém

não podemos inserir K diretamente pois já há um nó à esquerda de r.



4. Existe subárvore direita e $r.key < K$. Esse caso também não é óbvio. A busca iria para a direita em r, porém a mesma situação acima ocorre mas desta vez para a direita.



Os casos 1 e 2 já estão resolvidos e são fáceis: é possível inserir o nó diretamente no lugar vazio. Já os casos 3 e 4 parecem complicados mas não são. Árvores são estruturas recursivas: cada nó r contém uma ligação para uma subárvore esquerda (com todos valores $< r.key$) e uma subárvore direita (com todos valores $> r.key$).Nos casos 3 e 4 precisamos inserir na subárvore `E` e `D`, respectivamente. Ambas tem altura 1, então podemos usar o algoritmo visto mais acima!

Árvores com altura $h > 1$: Usar o argumento acima nos permite inserir em árvores de qualquer tamanho! Ou a inserção ocorreria em um espaço vazio (ou seja, um dos lados tem uma subárvore de altura 0) ou inserimos na subárvore correspondente (que tem altura $h - 1$).

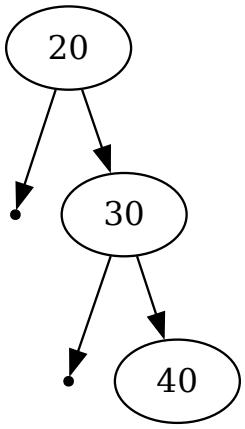
Esse algoritmo acaba?

Qual o número máximo de passos que esse algoritmo fará em uma árvore com N nós? E se dissermos que a altura dessa árvore é h, conseguimos uma estimativa mais justa?

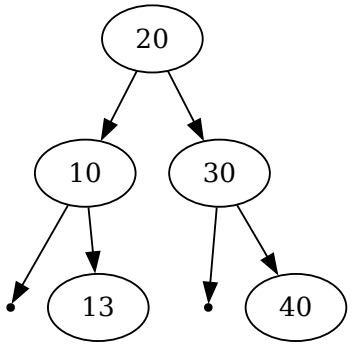
Exercitando e formalizando esse algoritmo

Vejamos alguns exemplos abaixo.

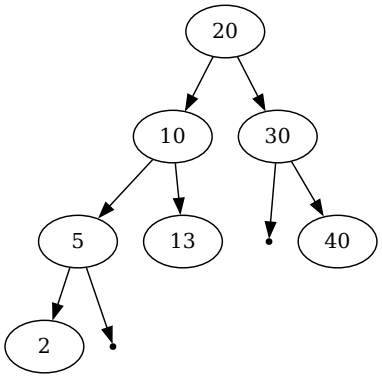
Exercício: Em qual lugar 27 seria inserido na árvore abaixo? Desenhe o novo nó no lugar correto.



Exercício: Em qual lugar 15 seria inserido na árvore abaixo? Desenhe o novo nó no lugar correto.



Exercício: Em qual lugar 15 seria inserido na árvore abaixo? Desenhe o novo nó no lugar correto.



Exercício: Desenhe as *ABB*s resultado da inserção dos seguintes valores.

- 1 5 -2 4 7 12 45 30 24
- 15 12 20 33 10 60 5 14 122
- -2 4 7 12 5 45 30 24 1

Algoritmo para inserção

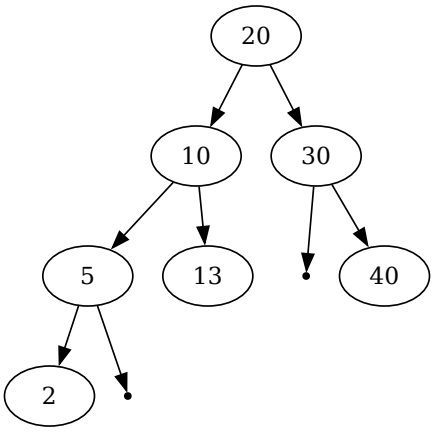
Vamos agora formalizar o algoritmo `INSERE(r, K)`. Esse algoritmo devolve a raiz da árvore `r`.

Algoritmo iterativo

Algoritmo recursivo

Ordem de inserção e altura da árvore

Exercício: Qual ordem de inserção nos permitiria reconstruir a árvore abaixo?



Exercício: Para a árvore acima, tente encontrar uma ordem de inserção que cria uma árvore de altura 3. Escreva-a abaixo.

Resposta:

Reflexões

Por fim, faça as seguintes reflexões e anote os resultados abaixo

1. Qual é a altura de uma ABB, no pior caso?
2. Em que situações o pior caso acontece?