Árvores Binárias – Análise do uso

5. Quantidades de elementos X altura da árvore.

Em uma árvore binária, a raiz consegue armazenar 1 elemento.

Então uma árvore com um nível tem 1 elemento. Esse elemento pode ter até 2 filhos. Então, no segundo nível, a árvore pode ter 2 elementos, no terceiro nível pode conter até 4 elementos, no quarto nível 8 elementos, e assim sussessivamente. O nível n pode conter até $2^{(n-1)}$ elementos, como ilustrado na figura 1.

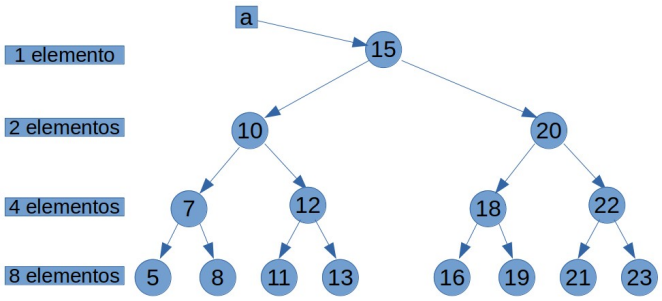

Figura 1. Número máximo de elementos em cada nível de uma árvore binária

E se somarmos a quantidade de elementos até o nível n, vamos obter 2^n-1.

Exemplo, uma árvore com 3 níveis contém 1 + 2 + 4 = 7 elementos. $2^3-1 = 8-1 = 7$.

Considere a altura da árvore como a quantidade de níveis da árvore.

Se tivermos uma árvore com 10 níveis, ela pode conter até $2^{10}-1 = 1024-1 = 1023$ elementos.

Já uma árvore com 20 níveis pode conter até $2^{20}-1 = (2^{10}*2^{10})-1$ ~= 1 milhão de elementos (1.048.575 elementos).

Tabela 1: Número máximo de elementos em cada nível e total em uma árvore binária.

Nível	Nº de elementos no nível n		Nº total de elementos até o nível n	
1	1	2^0	1	2^1-1
2	2	2^1	1+2=3	2^2-1
3	4	2^2	1+2+4=7	2^3-1
4	8	2^3	1+2+4+8=15	2^4-1
5	16	2^4	1+2+4+8+16=31	2^5-1
6	32	2^5	1+2+4+8+16+32=63	2^6-1
:	:	:	:	:
10	512	2^9	1023	2^{10}-1
:	:	:	:	:
n	2^{n-1}	2^{n-1}	2^n-1	2^n-1

Observe que o número de comparações necessário para buscar um elemento em uma árvore, no pior caso é justamente a altura da árvore. Já em uma lista linear, o número de comparações necessário para encontrar um elemento, no pior caso é exatamente o número de elementos da lista.

Então, se tivermos um conjundo de 1 milhão de elementos, no caso de árvore binária, no pior caso serão necessárias 20 comparações, pois uma árvore binária com 20 níveis comporta 1 milhão de elementos. Já no caso de uma lista linear com 1 milhão de elementos, serão necessárias, no pior caso, 1 milhão de comparações para encontrar um elemento na lista.

6. Indicações para uso de árvores binárias.

Os códigos de árvores binárias são mais complexos do que os códigos de listas lineares, incluindo as pilhas e filas. Considerando isso, vamos analisar em quais situações vale a pena implementar árvores binárias de pesquisa, no lugar de listas lineares.

1. Busca. Um dos nomes frequentemente usados para se referir a uma árvore binária é “árvore de busca binária” ou “árvore binária de busca”, especialmente porque é uma estrutura otimizada para a busca em grandes volumes de dados.

Mas para fazer uma busca de um elemento em um conjunto de dados, também é possível utilizar uma lista linear, seja ela ordenada ou não.

2. Ordenação. A árvore binária permite obter os dados em ordem crescente, independente da ordem de inserção. Mas uma lista ordenada também faz esse papel.

Então, se é possível usar listas, que são mais simples de implementar, porque não usar sempre uma lista ordenada? A resposta a essa questão precisa avaliar a quantidade de dados que precisam ser armazenados/ordenados.

Na lista ordenada, a quantidade de comparações necessárias para inserir um elemento no pior caso é igual à quantidade de elementos já inseridos na lista. Já na árvore binária (supondo que ela esteja balanceada), a quantidade de comparações necessárias para inserir um elemento na árvore cresce na ordem do logaritmo de base 2 da quantidade de elementos já inseridos na árvore, ou seja, a quantidade máxima de comparações depende da altura da árvore.

De modo geral, pode-se dizer que para pequenas quantidades de dados a complexidade da codificação da árvore não se justifica porque o acréscimo de custo computacional não é grande. Por exemplo, um computador pessoal atual pode resolver rapidamente (em questão de milissegundos) a ordenação dos dados dos alunos de uma turma, com no máximo 50 alunos.

Por outro lado, se for necessário trabalhar com a ordenação dos nomes dos alunos inscritos no ENEM, com 6 milhões de inscritos, uma lista linear será muito mais lenta que uma árvore binária, e portanto justifica plenamente sua implementação.

Outro parâmetro a ser utilizado para a decisão é a quantidade de vezes que a operação será realizada. Se for realizada uma única vez, pode valer a pena deixar o computador processando por mais tempo o algoritmo mais simples, do que investir recursos na implementação de um algoritmo mais sofisticado. Por outro lado, se a operação será realizada rotineiramente, vale a pena investir na implementação de um algoritmo melhor, e economizar tempo e energia ao longo do tempo de uso do sistema.

7. Balanceamento da árvore binária

A árvore binária de busca apresenta os seus melhores resultados quando está balanceada, ou seja, quando para todos os nodos da árvore a altura das sub-árvore da esquerda e da direita tem a mesma altura, ou no máximo a diferença de um nível de uma para outra.

A disposição dos elementos na árvore depende da sequência de inserção. Quando os elementos são inseridos aleatoriamente, a tendência é que os elementos fiquem razoavelmente bem

distribuídos, aproximando da distribuição de uma árvore balanceada. Entretanto, se a sequência de inserção for em ordem crescente (ou decrescente), cada elemento da árvore terá somente um filho da direita (ou da esquerda), sem filhos para o outro lado (figuras 2 e 3).

Isso é o que chamamos de árvore degenerada, e os algoritmos de busca e inserção de elementos passam a ter um comportamento igual ao de uma lista linear, em termos de quantidade de comparações para inserir ou buscar um elemento.

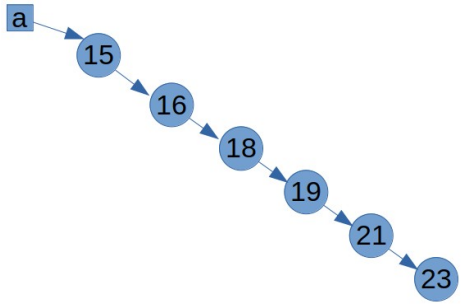

Figura 2. Árvore degenerada para a direita (inserção em ordem crescente)

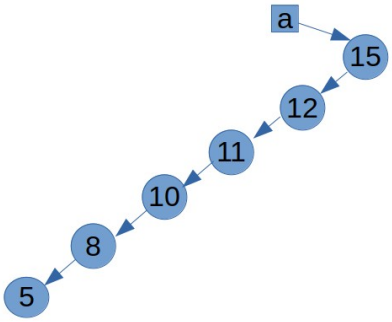

Figura 3. Árvore degenerada para a esquerda (inserção em ordem decrescente)

Para resolver esse problema existem algoritmos que mantém a árvore balanceada, alterando a disposição dos elementos na inserção e na retirada de elementos, sempre que for detectado que a árvore não está balanceada. Um exemplo de árvore binária que se mantém balanceada é a árvore AVL. O estudo da árvore AVL é deixado como exercício para quem tiver interesse.