Scapegoat Trees

Feito no ambito da disciplina de ED

Por:
Erica Wallberg 2017014841
João Gonçalves 2018014306

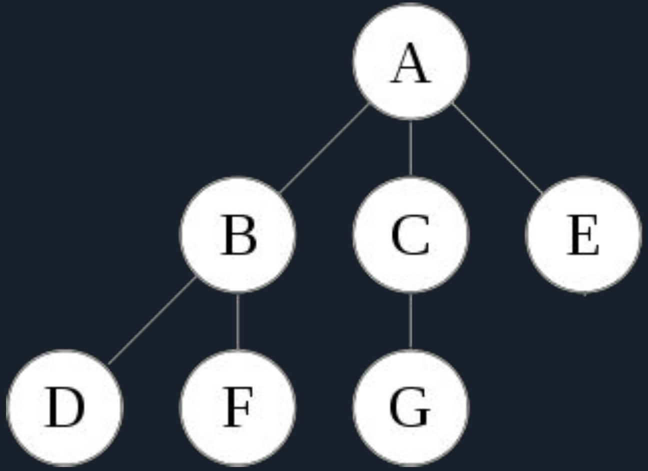
Árvores

Trees ou árvores são estruturas de dados.
Estas herdam características de topologia em árvore tal como o nome indica.

São estruturas eficientes ao tratamento computacional.

Uma árvore é formada por um conjunto de elementos que armazenam dados chamados nós.

Todas as árvores constituem uma raiz que se liga aos outros elementos denominados de ramos.

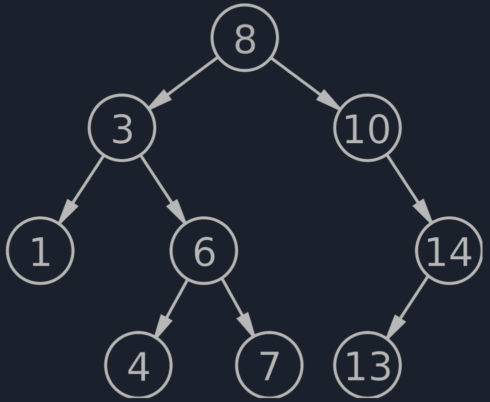


Árvores binárias de busca

Estas árvores estão estruturadas da seguinte forma:

- São constituídas por nós ligados em árvore.
- No lado esquerdo de cada nó pai tem-se um valor inferior ao do pai.
- No lado direito de cada nó pai tem-se um valor superior ao do pai.

As suas complexidades são:



Complexidade

Algoritmo	**Caso Médio**	**Pior Caso**
Busca	O(log n)	O(n)
Inserção	O(log n)	O(n)
Remoção	O(log n)	O(n)

Árvore binária de busca balanceada

Estas são árvores que automaticamente mantém a sua altura (número máximo de níveis abaixo da raiz) pequeno mesmo depois de sucessivas inserções e exclusões arbitrárias.

Uma árvore de pesquisa binária é considerada balanceada por peso se metade dos nós estiver à esquerda da raiz e a outra metade à direita. Um nó α-balanceado é definido como atendendo a um critério de equilíbrio de peso relaxado:

$$\text{size(left)} \leq \alpha * \text{size(node)}$$

$$\text{size(right)} \leq \alpha * \text{size(node)}$$

Árvore binária de busca balanceada

O tamanho (size) de cada nó pode ser definido recursivamente como:

```
//                                                                   //
private int tamanho(Nó root) {                                     //
   if (root == null)                                               //
       return 0;                                                     //
   return tamanho(root.esquerdo) + tamanho(root.direito) + 1;       //
}                                                                  //
//                                                                   //
```

Alpha (α)

$0.5 < \alpha < 1$

Árvore degenerada
(lista encadeada/ligada) → $\alpha = 1$

Árvores binárias
equilibradas → $\alpha = 0.5$

Um valor de α elevado resulta em menos balanços, tornando a inserção mais rápida, mas as pesquisas e exclusões mais lentas (vice versa para um α baixo).

Na prática, um α pode ser escolhido dependendo da frequência com que essas ações pretendem ser realizadas.

Equilíbrio de peso

Uma árvore de pesquisa balanceada por peso também deve ser α-altura balanceada, isto é:

$$\boxed{height(tree) \leq \lfloor \log 1/\alpha(size(tree)) \rfloor}$$

As Scapegoat Trees não têm garantia de manter o equilíbrio de peso em todos os momentos, mas são sempre vagamente equilibradas em altura.

$$\boxed{height(scapegoat\ tree) \leq \lfloor \log 1/\alpha(size(tree)) \rfloor + 1}$$

Violações desta condição de equilíbrio de altura podem ser detetadas no momento da inserção e implicam que uma violação da condição de equilíbrio de peso exista.

Árvores Scapegoat

O Scapegoat é um algoritmo de árvore binária de busca balanceada.

Árvores Scapegoat (bode expiatório) não têm custos de memória adicional por nó comparando com outros algoritmos de árvore binária de busca normais: um nó armazena apenas uma chave e dois ponteiros para os nós filhos. Isto torna as árvores scapegoat mais fáceis de implementar devido ao alinhamento de estrutura de dados, pode ser reduzido a sobrecarga de um nó até ⅓.

Em vez de pequenos incrementos de operações de rebalanceamento usados por maior parte dos algoritmos, árvores scapegoat raramente mas de forma cara escolhem uma “scapegoat” e reconstroem completamente a subárvore enraizada no scapegoat numa nova árvore binária completa.

Complexidades

Algoritmo	Média	Pior caso
Procurar	O (log n)	O (log n)
Inserir	O (log n)	O amortizado (log n)
Excluir	O (log n)	O amortizado (log n)

Nota: A análise amortizada considera as operações mais caras e menos caras juntas ao longo de toda a série de operações do algoritmo. Isso pode incluir a contabilização de diferentes tipos de entrada, comprimento da entrada e outros fatores que afetam seu desempenho.

Procura

A procura numa scapegoat tree é igual a todas as outras árvores balanceadas.

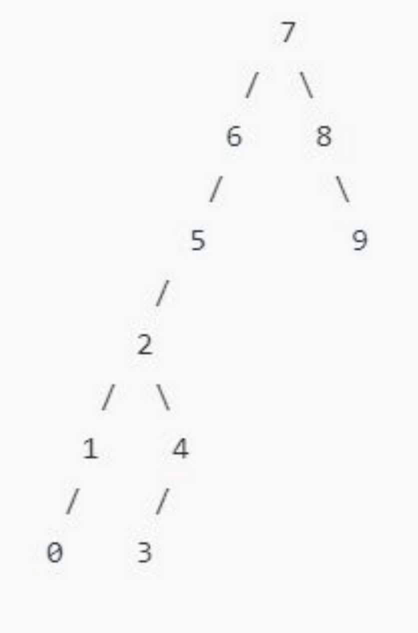
Inserção

- Scapegoat tree com:
  - 10 nós
  - Altura 5

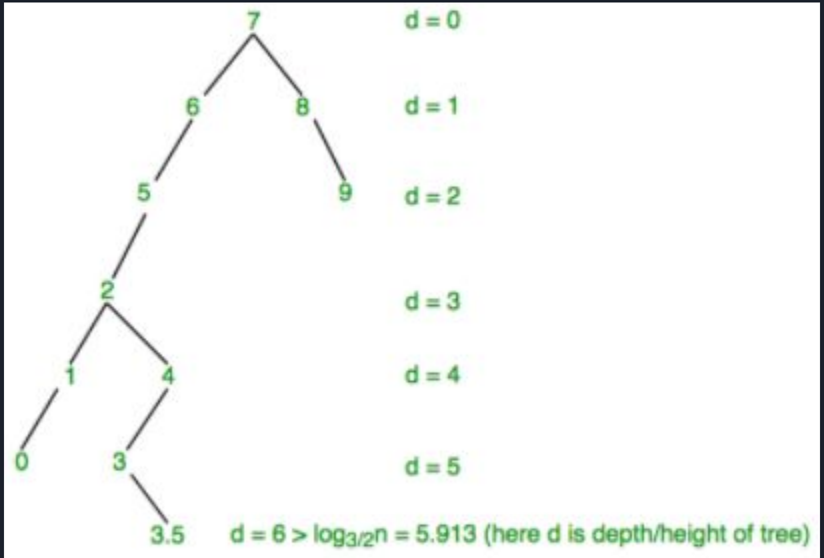
- Inicialmente, $d = 5 < \log_{3/2} n$ em que $n = 10$

- Alpha assumido como 2/3

Vamos inserir o valor 3.5



Inserção

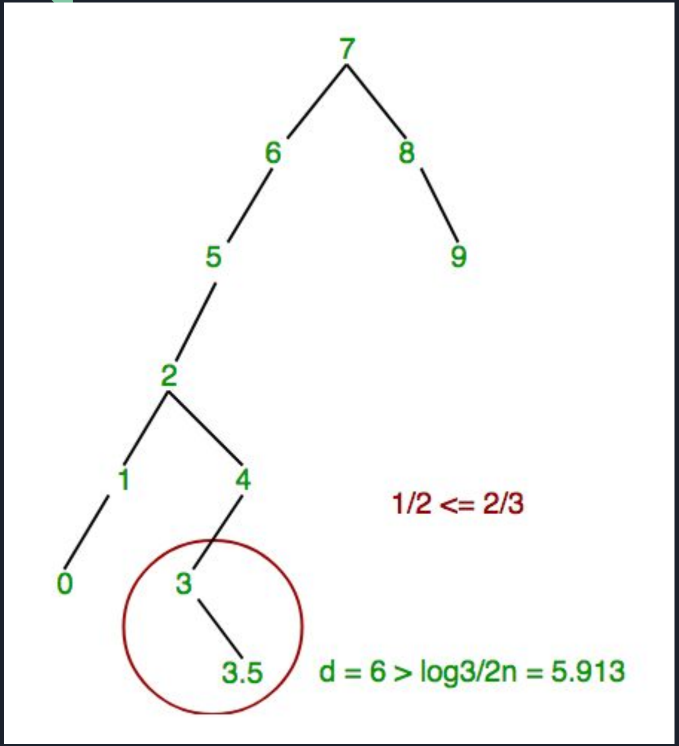


Sendo:
d => profundidade (6)
n => numero de nós (11)
d > $\log_{1/\alpha} n$ => condição de verificação de scapegoat

$$(6 > \log_{3/2} 11)$$

Iremos encontrar a scapegoat para resolver o problema de altura excessiva.

Inserção

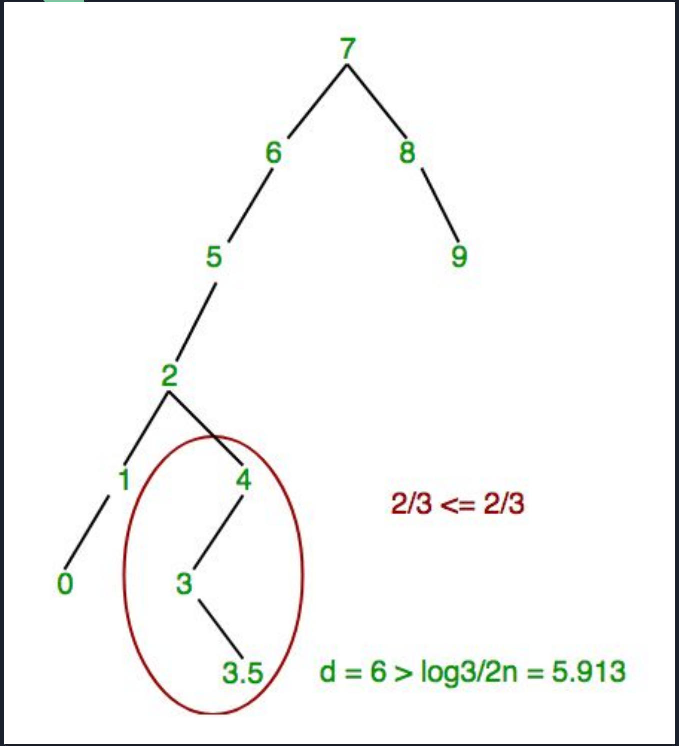


Verificar se size(3.5)/size(3) > ⅔

Visto que size(3.5) = 1 e size (3) = 2,
então size(3.5)/size(3) = ½
½ < ⅔

Sendo assim, este nó não é a scapegoat, subimos na árvore.

Inserção



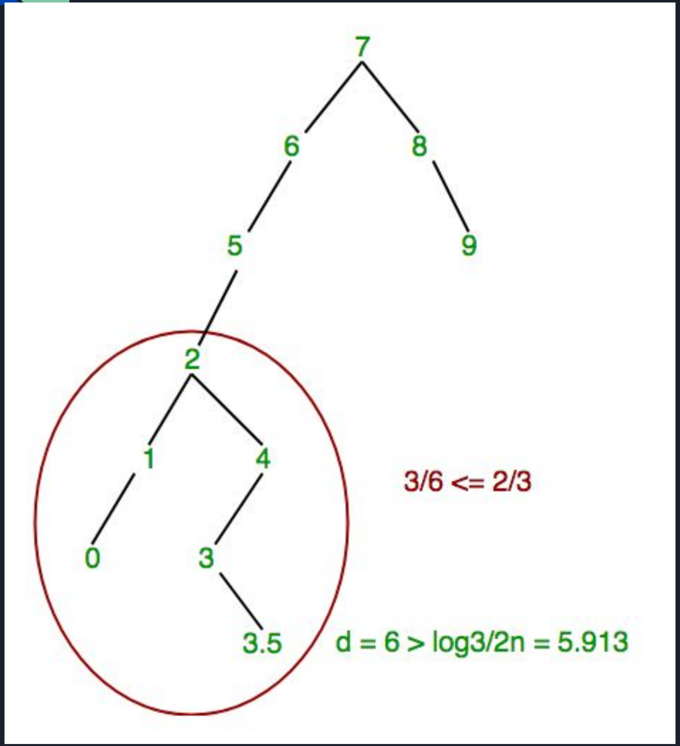
Visto que 3 não é a scapegoat, verificamos a mesma condição para o nó 4.

Como size(3) = 2 e size(4) = 3,
Então size(3)/size(4) = ⅔

⅔ =⅔
(tinha que ser maior do que ⅔)

Sendo assim, este não é a scapegoat e subimos no ramo

Inserção



Para esta fase,

Visto que size(4) = 3 e size(2) = 6

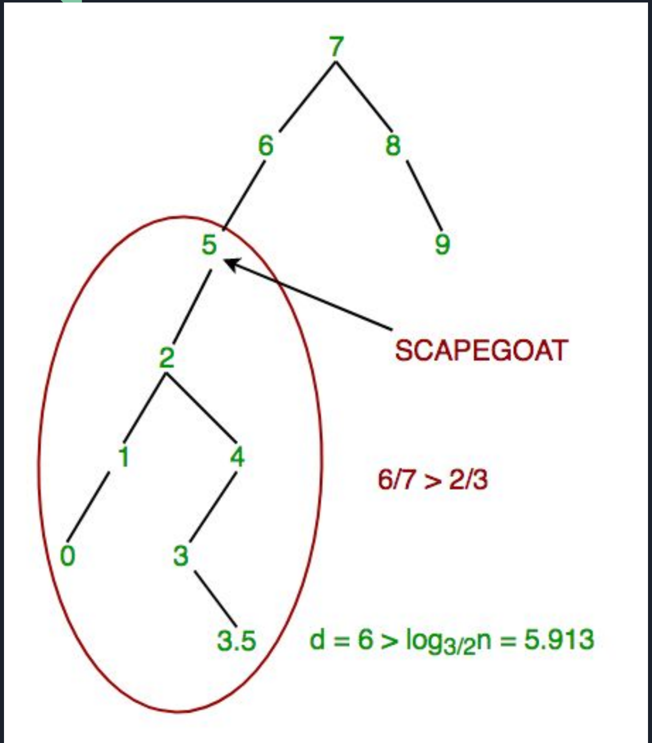
size(4)/size(2) = 3/6

$$3/6 < 2/3$$

(tinha de ser maior)

Mais uma vez, as condições não foram verificadas.
Subimos na árvore.

Inserção



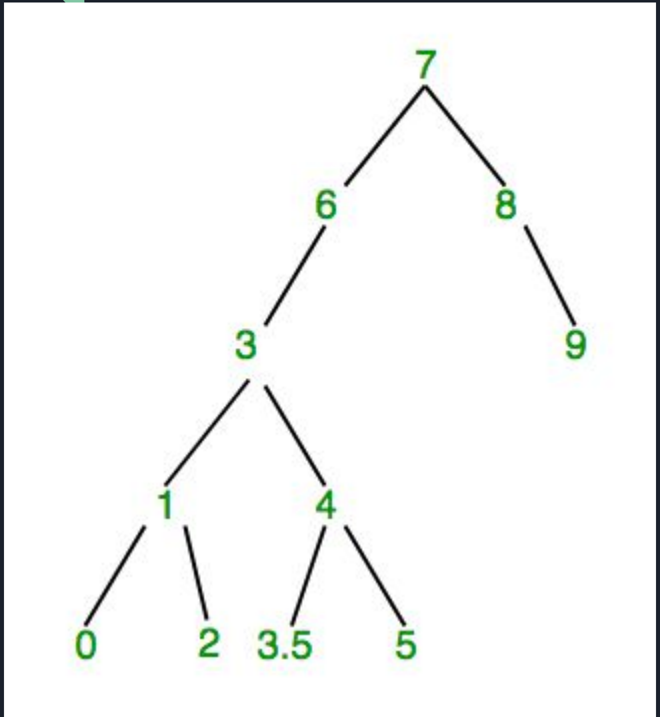
Nesta fase,

Visto que size(2) = 6 e size(5) = 7,
size(2)/size(5) = 6/7.

$6/7 > 2/3$

A condição é verificada.

Inserção



No final, após ter sido encontrada o scapegoat, a reconstrução será feita na subárvore enraizada na scapegoat, ou seja, em 5.

Apagar

As árvores de scapegoat são incomuns porque a exclusão é mais fácil do que a inserção.

Para permitir a exclusão, as árvores de scapegoat precisam armazenar um valor adicional com a estrutura de dados da árvore. Essa propriedade, que chamaremos de MaxNodeCount, simplesmente representa o NodeCount mais alto alcançado.

Ele é definido como NodeCount sempre que toda a árvore é balanceada e, após a inserção, é definido como max (MaxNodeCount, NodeCount). Para realizar uma exclusão, simplesmente removemos o nó como faria uma árvore de pesquisa binária simples, mas se *NodeCount ≤ α*MaxNodeCount*, reequilibramos toda a árvore em torno da raiz, lembrando-nos de definir MaxNodeCount como NodeCount.

Questões?