



www.python.pro.br

Roteiro - Árvores

Definição e Justificativa

Visualização

Nomenclatura

Árvores Binárias

Árvores com Objetos

Exercícios: Busca em Largura e Profundidade *

Definição e Justificativa

Estrutura composta por nós

Cada nó pode possuir nós filhos

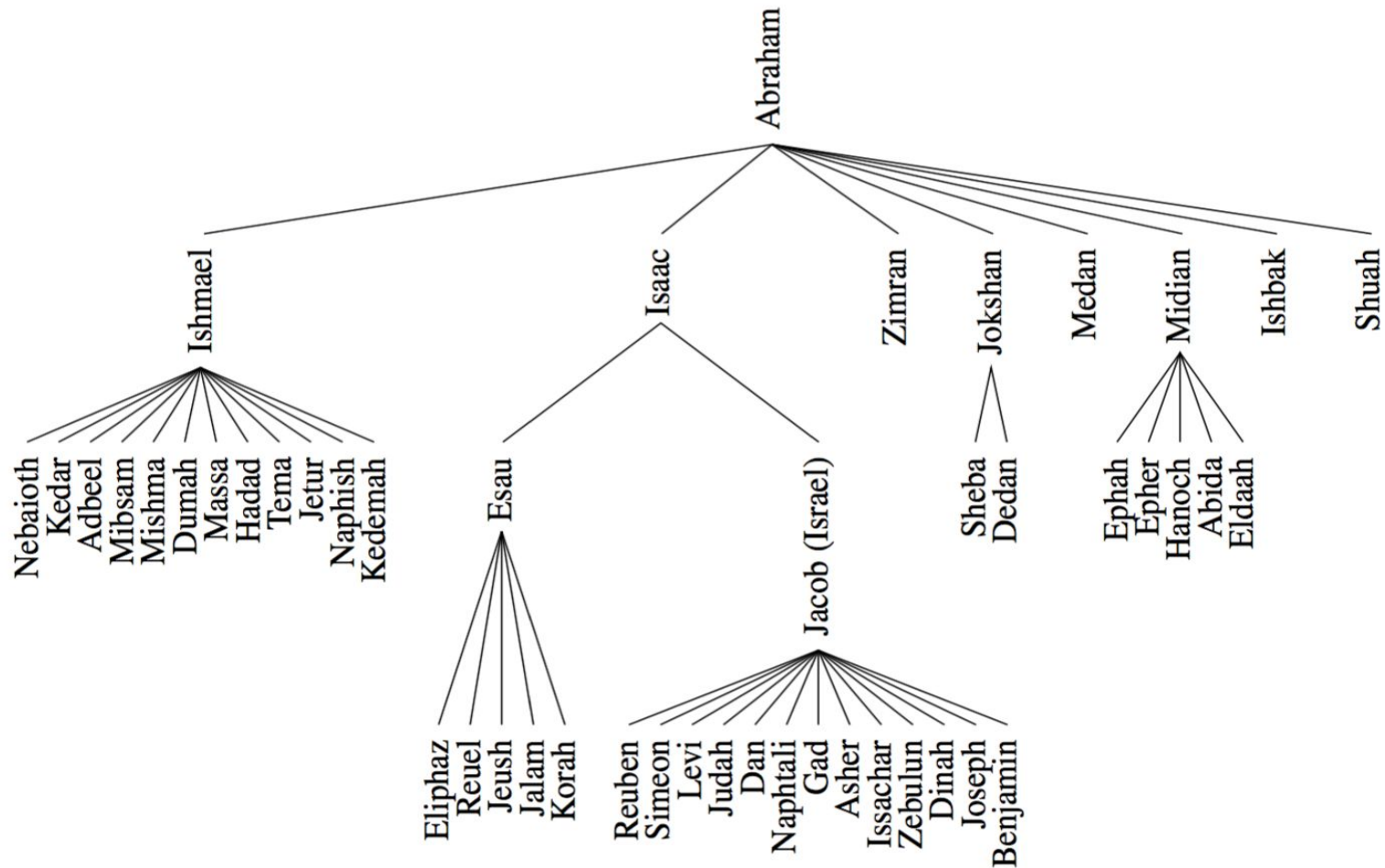
Cada nó possui apenas um pai, a exceção do nó raiz, que não possui pai

Estrutura de Dados não linear

Organização Hierárquica

Usado em sistemas de arquivos *

Visualização: Árvore Genealógica



Nomenclatura

Raiz (root)

Folhas (leaves)

Pai (parent)

Irmão (sibling)

Filho (child-children)

Folhas (external, leaves, leaf)

Interno (internal)

Altura (height) *

Árvores Binárias

Árvores cujos nós possuem no máximo 2 filhos

Árvore Balanceada

Representação em vetor

Fórmula para filhos:

Esquerdo: $(i+1)*2-1$

Direito: $(i+1)*2$

Pai: $\text{floor}((i-1)//2)$

*

Árvores Binárias: Exemplos

Árvore Binária de Busca (BST)

Todos descendentes à esquerda são menores

Todos descendentes à direita são maiores

Usos: Busca binária

Heap

Todos filhos são menores

Usos: achar mínimo, heapsort *

Árvores com Objetos

Utilizar estrutura Noh

Atributo pai

Versão 1:

Atributo filho_esquerdo

Atributo irmao_direito

Versão 2:

Atributos filhos

Versão 1 versus 2 *

Exercício - Travessia em Profundidade

Implementar métodos de Noh e Arvore de acordo com teste: <http://bit.ly/23rLzI6>

Profundidade: Para cada nó, iniciando da raiz, fazer busca em profundidade em filho_esquerdo, depois irmao_direito do filho, imprimindo valores

Analisar complexidade de tempo e espaço*

Exercício - Travessia em Largura

Implementar métodos de Noh e Arvore de acordo com teste: <http://bit.ly/1T1AF3u>

Largura: Para cada nó, iniciando da raiz, Imprimir seu valor. Enfileirar filhos e repetir procedimento para cada um deles.

Analisar complexidade de tempo e espaço*

Obrigado

renzo@python.pro.br
@renzoprobr

