Priority Queues and Heaps

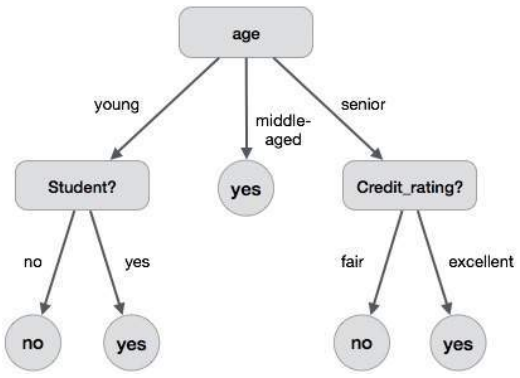
Sujit Chakrabarti

1 Real Life Example

They represent the natural flow of data or we can say that their structure conveys information Examples:

1. family tree

$$\begin{tikzcd}
 & & Prithvi \arrow[dl] \arrow[dr] & \\
 & Raj \arrow[dl] \arrow[d] \arrow[dr] & & Shashi \arrow[d] \\
Randhir & Rishi & Rajeev & Kunal
\end{tikzcd}$$



2. a decision tree

3. classification of species

$$\begin{tikzcd}
 & & Animal \arrow[dl] \arrow[dr] & \\
 & Mammal \arrow[dl] \arrow[d] & & Bird \arrow[d] \\
Land & Aquatic & & Flying
\end{tikzcd}$$

2 Tree Description

- root
- leaf and internal nodes
- parent and child

2.1 Properties

- Unique root
- Each node can only have one parent
- There can be no cycles in a tree

A few demonstrations or examples which do not qualify as a tree. May be done through an IVQ as well. We will give few structures to the student and ask them to identify which one of them is a tree.

3 Binary Tree

Now we can move on to a particular case of a tree called binary tree. We will learn about it because its easier to code. Also, you will learn later that the for basic data structure operations like, insertion deletion, searching etc. binary trees are more efficient than ternary trees.

Define a binary tree
What is a binary tree
Examples of a binary tree.
Show a visual representation of a binary tree structure
Properties of a Binary Tree:

1. A parent can have at most two children (one to the left and one to the right)

4 Traversals

If we want to visit a particular node, one can not visit it directly like arrays where we use the index of an element and access it. We need to traverse a particular path starting from the root node to reach a node in a tree. Now there can be multiple paths that can be used to access a particular node, lets see how one path can be more efficient than the other for a particular case.

Why do we need to traverse along a tree? How is tree traversal different from a linked list traversal Why the order in which we traverse a tree is important This can be explained with the help of a couple of demonstrations where one order is more efficient (in term of runtime) than the other. An example where bfs is more efficient than dfs can be used here. We need not to name the terms, but the example can be used. Now that the students know why the order of traversing is important, we can move onto some standard traversal techniques.

4.1 Pre Order Traversal

4.1.1 Application

Tree copy, polish notation

4.1.2 Code

4.1.3 Performance

4.2 Post order traversal

4.2.1 Application

4.2.2 Code

4.2.3 Performance

4.3 In Order Traversal

4.3.1 Application

4.3.2 Code

4.3.3 Performance

4.4 Breadth First Traversal

4.4.1 Application

4.4.2 Code

4.4.3 Performance

Having learnt various traversal techniques, students need to know that one is not necessarily better than the other. It all depends upon the type of application.

4.5 Additional Examples of all traversals (Optional)

5 Binary Search Trees (BST)

5.1 Introduction, Motivation

Arrays have an advantage in terms of searching, but linked list are better when it comes to insertion and deletion. Binary trees utilise the advantages of both using something called binary search trees.

5.2 Definition of BST

A visual representation of Binary Search Tree Binary Search Tree Properties (how it differs from binary trees) An IVQ where students need to identify a valid BST Few of the traversal techniques discussed above lead to some interesting results if applied to BST.

Linked list was O(n) when it comes to searching. Lets see how BST performs here.

5.3 Search

5.3.1 Algorithm

5.3.2 Code

5.3.3 Efficiency

5.3.4 Compare search in a BST to binary search

5.4 Insertion

5.4.1 Algorithm

5.4.2 Code

5.4.3 Efficiency

Linked list are better than arrays when it comes to insertion. BSTs have this advantage also with them.

5.5 Deletion

5.5.1 Algorithm

5.5.2 Code

5.5.3 Efficiency

5.5.4 Delete an element with no child

5.5.5 Delete an element with one child

5.5.6 Delete an element with 2 children

5.5.7 Efficiency

5.5.8 Code

5.6 Balanced BSTs

What is the height of the BST in an average case? What is the height of the BST in the worse case? (the BST will essentially be a linked list)