Heaps and graphs in different programming languages

Introduction

In this reading, the focus is on graphs and heaps. A graph is a non-linear data structure that can store information in a way that allows you to extract some interesting relations found in the data. Heaps can be defined as specialized graphs that place special importance on the topmost element.

Terminology

Graphs are comprised of nodes and edges. The node is where the data is stored, and the edge is a connection between two nodes. Unlike a tree, nodes do not have to be connected and can exist independently from the other nodes. An edge connects two nodes. An edge can be said to have a weight. This is a value that is stored in the connection that infers some information on the strength of the connection between the two nodes. A graph can be said to be directed, this means that the edges are focused (like a one-way street) or undirected (like a two-way street) and the connection infers information back and forth.

NetworkX

NetworkX is a Python-based external library that can be imported into the Python environment and utilized to model data. It is a lightweight third-party library that can work in any Python environment. Many featured graphs include directed, undirected, cyclic and acyclic. NetworkX also supports a wide range of graph-based algorithms, including:

- Distance metrics (the distance between two nodes)
- Centrality (how central a node is in relation to other nodes)
- And clique detection (relates to subsets within a graph)

A clique detection algorithm will analyze a graph to determine a range of subsets most likely to have strong internal and weak external relations. These are referred to as cliques due to their resemblance to social cliques. This can be a helpful way of identifying potentially related customers, given a sample of customer data and habits. NetworkX can be modeled using Python's matplotlib to give insight into the discoveries of the data. This is used as a back-end application by a programmer interrogating data for insights.

Heaps

Heaps are implemented differently in different languages but are essentially graphs with specific constraints. Heaps sort information in order so that they can quickly return min and max values. Thus, they employ a binary approach, and any implementation must have a maximum of two nodes. Depending on the implementation, a heap will have the largest or smallest value as the root. Finally, each branch of the heap will follow a sequential pattern.

A heap has an O(1) lookup time because it only returns one item. The highest or lowest value depends on whether it is a min or max heap. This means that once this value is popped, the following item is pushed onto the root node of the heap. This also impacts inserting onto a heap. When a new element is added, beginning at the root, it is compared to each node until the correct position is determined. The surrounding elements are then moved to ensure that it is placed in the appropriate position.

An alternative way that some languages like Kotlin or Javascript. which lack a built-in instance, represent heaps using an array list. The same properties you use with graphs are employed, though the actual implementation may differ slightly. The efficacy of heaps will depend on the underlying data structure used, so it is worth investigating how efficient your target language implementation is before using them.

Conclusion

In this reading, graphs and heaps have been reviewed. Heaps can be seen as specialized graphs, and a graph is made up of nodes and edges. There are some very specialized methods available for graphs, whose purpose is to allow us to infer some connection from the data that is stored there.