Data Stuctures: Assignment 2

Josh Wainwright
UID:1079596

1 Heapsort

Heap sort is an algorithm that sorts a set of elements using a priority collection. A priority collection allows a set of elements to be arranged such that the element with the highest priority is available directly with complexity O(1).

1. Add all elements to be sorted onto an initially empty priority collection.
2. Remove each element "in order".
3. Since the priority collection presents the highest priority element, simply by maintaining the validity of the data structure, the next largest element will always be the root node.
4. So to remove "in order", simply take off the root node and verify that the collection is valid.

Adding elements to the heap involves the steps below.

1. Place the new element in the last position available on the final level of the tree structure. If the final row is full, start a new row from the left.
2. Compare the new node with its parent.
3. If the parent has a higher priority (is larger), the element is in the correct position and the addition is complete.
4. If the parent has a lower priority (is smaller) replace the parent with the newly added node.
5. Repeat steps 2–4.

The priority collection methods that would be required for the heapsort algorithm are simply;

- `add(element,priority)`, for creating the structure, and
- `high_priority()`, which returns the highest priority element for removing the elements in sorted order.

2 Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm locates the shortest path from a starting node to any given destination node across a graph. It does this by finding the shortest path for every node in the graph.

A priority collection can be used when calculating the distance to the neighbours of the elements that have already been considered. Since the shortest path is wanted, the paths with the shortest distance have the highest priority and so, when the algorithm has completed, the shortest path is found by removing the nodes from the collection by priority. This will give the path from the destination to the starting node with the shortest path. Thus, the reverse of this path is the shortest path from start to destination.

The methods required to implement Dijkstra's algorithm would be;

- `add(element,priority)` for adding a new node to the currently stored path,
- `high_priority()` for retreiving the shortest path still in the collection and
- `change_priority(element, new_priotirty)` which is used to update the path length to a node when a shorter one is found.