**4.1 Heap Sort Background information**

Unlike Merge sort or Quick Sort(uses divide and conquer method), Heap Sort is a comparison based sorting algorithm with a tree data structure as reference. Quick Sort’s has an average, best and worst time complexity of O(nlog(n)), and operates by divided the array into sorted and unsorted parts and focus on reducing the unsorted areas. The core concept to perform Heap Sort is converting the data set into a tree data structure, and perform swaps with data to achieve a balance binary tree binary as the final product. In a max heap sort, each integers represents a node(parent) within a tree, and any number smaller than the node is linked to the left and right node(child). A parent has a maximum of 2 node, with right node is larger than the left node. Please refer figure 4.1 for a binary tree reference.

**Advantage of Heap Sort**

* Has best, worst and average time complexity of O(nlog(n))
* Heap Sort has consistent performance given it equal performance in best, worst and average time complexity
* Memory uses is minimal compared to other sorting algorithm(in-place sorting algorithm).

**Disadvantage of Heap Sort**

* Additional space and process requires to read the sorted data, as data set is rearranged as a tree type data structure.
* May occur a mistake when sorting multiple equal elements(not a stable sort).

**4.2 Heap Sort Version**

g

**4.3 Methodology of Heap Sort**

1. Convert the data set into a heap
2. Check current node(first element) with the child element,
   1. If node is smaller than either child element, perform swap with the largest element
   2. Repeat step 2 with current node until both child element are smaller the the parent node
3. Repeat step 2 for every element in lower half of the data set
4. Perform swap with the last element and the first element
5. Repeat only step 2 with first element and return step 4

**4.4 Data Collection for Heap Sort**

g

**4.4.1 Dat Analysis for Heap Sort**

g

**4.5 Evaluation for Heap Sort**

<https://www.geeksforgeeks.org/building-heap-from-array/>

<https://courses.cs.washington.edu/courses/cse373/18wi/files/slides/lecture-14-ann.pdf>

<https://www.geeksforgeeks.org/heap-sort/>

<https://www.happycoders.eu/algorithms/heapsort/>

<https://courses.cs.washington.edu/courses/cse373/18wi/files/slides/lecture-14-6up.pdf>