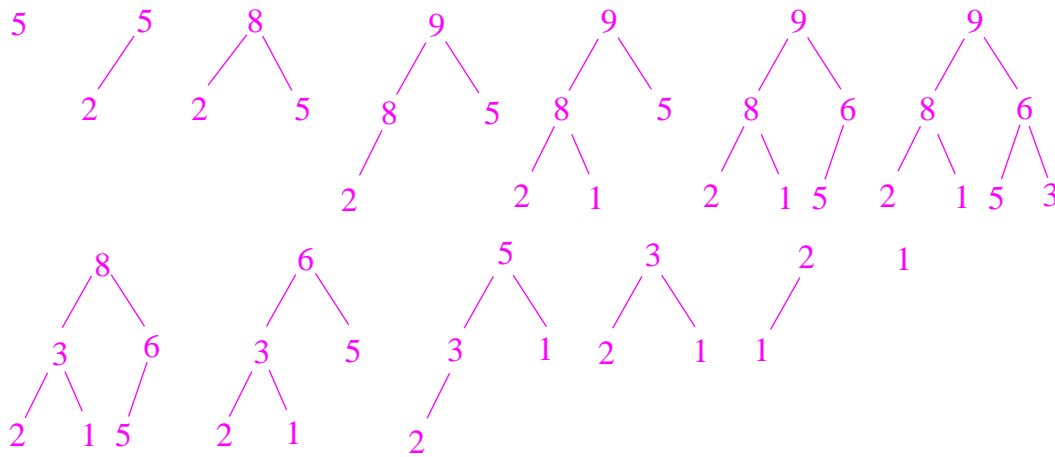


## COMP3600/COMP6466 in 2007 – Tutorial Three

### Question 1.

Insert the keys 5, 2, 8, 9, 1, 6, 3 into a max-heap one at a time, then remove the key in the root repeatedly until the heap is empty. What is the time complexity of sorting in this fashion?



The time is  $O(n \lg n)$ , which is not surprising since this is just heapsort.

### Question 2.

Almost all analysis of binary search trees assumes that the keys are distinct. Suppose the definition of the binary search tree ordering is changed to:

For each node, if the key is  $K$ , then:

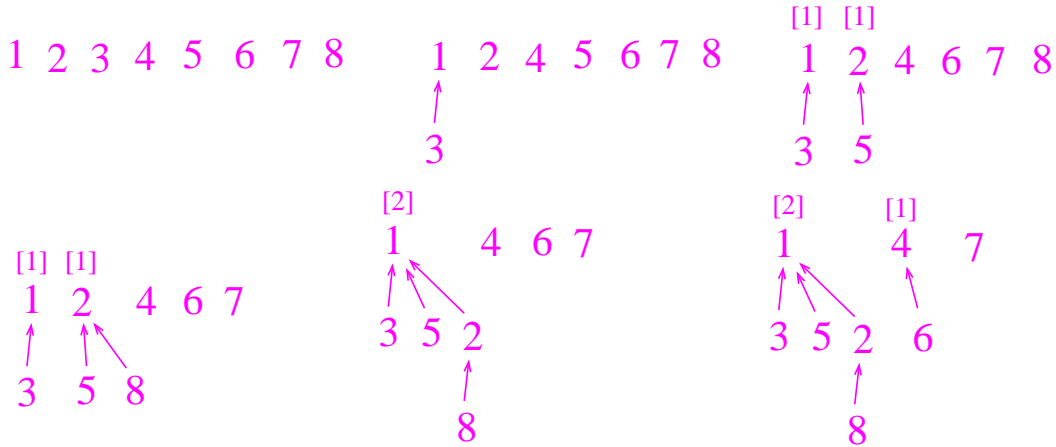
- \* the key in the left child (if any) is  $\leq K$
- \* the key in the right child (if any) is  $> K$

Redesign the insertion algorithm so that all insertions create a new node, even if the key is already present. Explain how to search in such a tree.

If the existing algorithm is followed, the nodes with a given key can be separated in the tree by other nodes, which is inconvenient. Try inserting keys 4, 3, 4, 3, 4 in that order. It is better to keep all the nodes with the same key in a single path. To do this: while inserting key  $K$ , if an existing node  $P$  with key  $K$  is encountered, insert the new node between  $P$  and its left child. For searching: once one node with the requested key is found, scan down to the left to collect all the others.

**Question 3.**

Apply the directed tree implementation of the disjoint sets data structure, using both heuristics, to find the components of the graph with 8 vertices and edges provided in this order: 1-3, 2-5, 2-8, 3-5, 4-6



The numbers in square brackets are ranks. There is a choice when merging two trees of the same rank, so other solutions are possible.

**Question 4.**

Given an undirected graph with  $n$  vertices and  $m$  edges, show how DFS can be used to find a cycle (or prove there isn't one) in  $O(n)$  time. Recall: DFS in general takes more time than that.

Run DFS until either a back edge is found (which gives a cycle) or the DFS finishes. Since there are at most  $n - 1$  tree edges, one of the first  $n$  edges examined must be a back edge unless there are no back edges at all.

**Question 5.**

Understand the splay tree data structure for the second assignment!