Australian National University
Research School of Computer Science

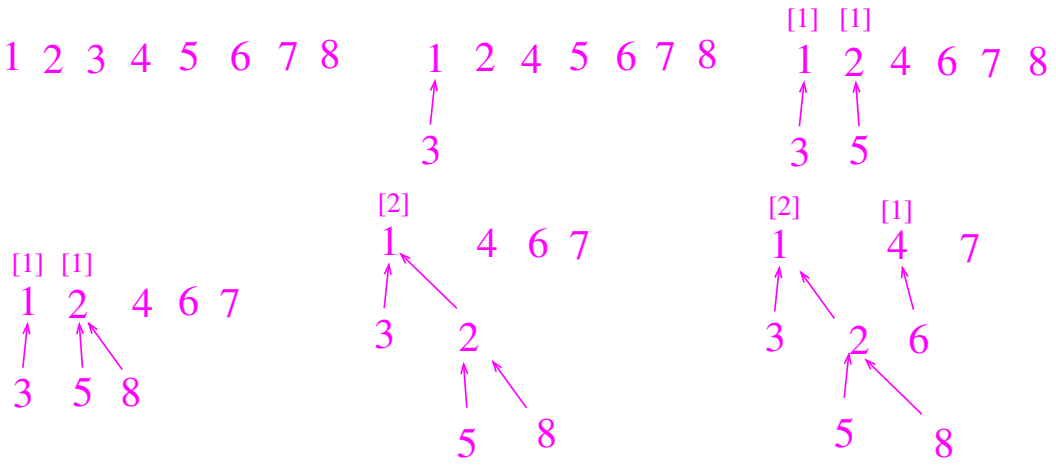
Computer Science COMP3600/COMP6466 in 2016 – Answer to Tutorial Five

Question 1.
(i) What are essential differences between the binary search tree and the red-black tree?
(ii) What is the left/right rotation? what's its purpose when applied in the red-black trees.

(i) The differences between the binary search tree and the red-black tree are (a) the height of the binary search tree is unbounded, which means its height may be $O(n)$, while the height of a red-black tree is always bounded by $O(\log n)$, which means each of 7 operations on it takes $O(\log n)$ time. (b) The insertion and deletion are relatively easy in binary search trees, while are much invloved in red-black trees as the 5 properties of the red-black tree properties must be maintained before and after the insertion or deletion.

(ii) The left/right rotation at a node is usually used to reduce the height difference between its left and right subtrees. However, such a transformation maintains the binary search tree property. The left/right rotation applied to the red-black tree is to maintain the properties of red-black tree properties, specially they are used when a node inserts to and deletes from a red-black tree.

Question 2.
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 3.

A depth-first search on a graph classifies the edges of the graph into tree, back, forward, and cross edges. A breadth-first search can also be used to classify the edges reachable from the source node of the search into the same four categories. Prove that in a breadth-first search of a directed graph, the following properties holds, where $d[v]$ is the distance from the starting node to v.

1. There are no forward edges.
2. For each tree edge $\langle u, v\rangle$, we have $d[v] = d[u] + 1$.
3. For each cross edge $\langle u, v\rangle$, we have $d[v] \leq d[u] + 1$.
4. For each back edge $\langle u, v\rangle$, we have $0 \leq d[v] \leq d[u]$.

(1) Assume that there is a forward edge $\langle u, v\rangle$ and u is an ancestor of v in the BFS tree. Then, when u is in the queue (a queue contains gray nodes only), the color of v has one of the three possibilities: white, gray, and black.

If v is white, then, the edge $\langle u, v\rangle$ is a tree edge when u is the head of the queue and u explores all its neighboring nodes, this contradicts that edge $\langle u, v\rangle$ is a forward edge.

If v is black, then, v has been visited before, and u has not yet been colored black, which means there is no relationship between u and v. This contradicts the fact that $\langle u, v\rangle$ is a forward edge and nodes u and v are the ancestor-descendent relationship.

Otherwise, v must be gray already. Now u explores its neighboring nodes including v, which means that there is no relationship between u and v. This contradicts the fact that edge $\langle u, v\rangle$ is a forward edge.

In summary, there is no forward edge in BFS search in a directed graph.

(2) Following the BFS tree construction, if $\langle u, v\rangle$ is a tree edge, then $d[v] = d[u] + 1$, because u is the parent of v in the tree.

(3) Consider a cross edge $\langle u, v\rangle$. Assume that $d[v] > d[u]+1$, i.e., $d[v] \geq d[u]+2$. When u becomes the head of the queue consisting of gray nodes, its neighboring nodes will be explored, then, $d[v] \leq d[u] + 1$, while we already knew that $d[v] \geq d[u] + 2$, which contradicts that the tree is a BFS tree. Thus, $d[v] \leq d[u] + 1$.

(4) Assume that $\langle u, v\rangle$ is a back edge, i.e.,, v is an ancestor of u in the BFS tree, then $d[v] \leq d[u]$ following the definition of the BFS tree. it is obvious that $d[v] > 0$.