EL9343 Homework 4

(Due Nov 13rd, 2020)

No late assignments accepted

All problem/exercise numbers are for the third edition of CLRS text book

---

1. Given an empty AVL tree and sequence {5, 1, 2, 4, 11, 17, 3, 7}, insert each element of the sequence to the AVL tree, one at a time, following the given order of the sequence. Draw all the 8 AVL trees after each insertion.

Solution:

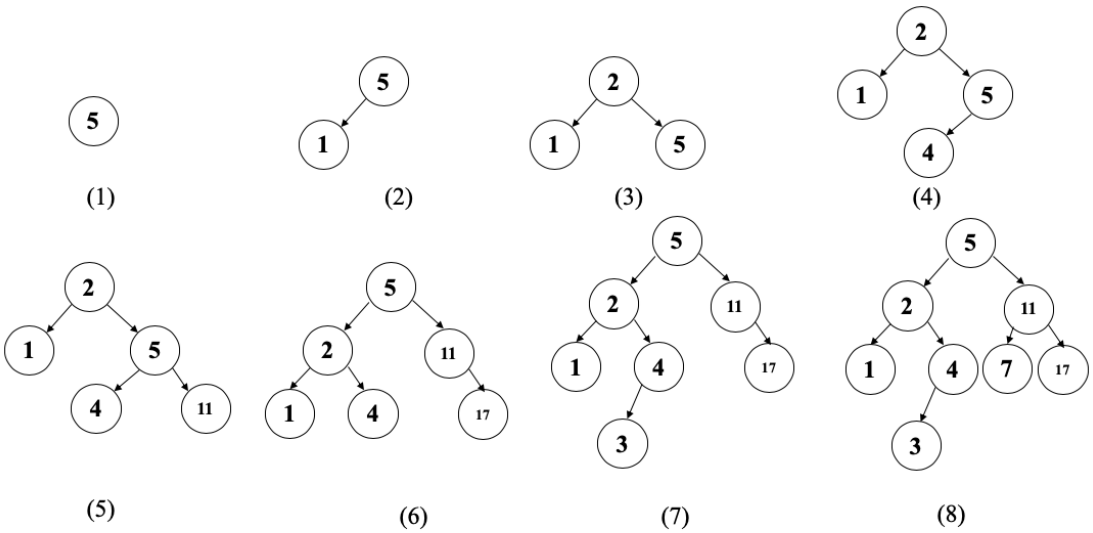


Figure 1: Insertion results for problem 1

2. What is the running time of DFS if the graph is given as an adjacency matrix? Justify your running time.

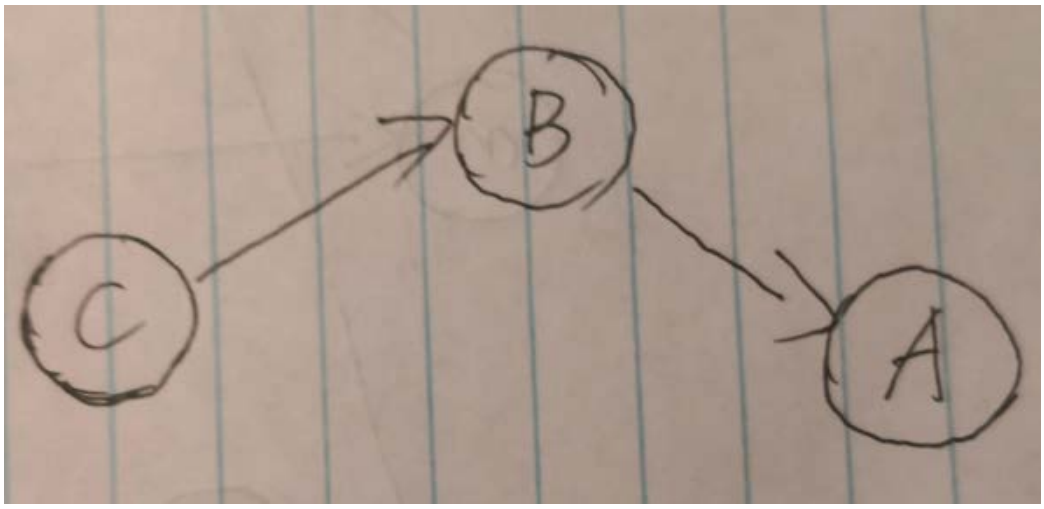
Solution:

If we use an adjacency matrix, the running time of DFS will be $O(V^2)$, where V is the nodes in a graph. In DFS, we will visit every node once and visit those nodes are adjacent to them. So we first will have the nodes we visit is $O(V)$, although there are only E edges, but once we visit a node (i.e. the ith node), we will visit the element in the ith row of the adjacency matrix. So totally, we will visit every element in the adjacency matrix (no matter it is 0 or 1), which is V^2. So the running time will be $O(V^2) + O(V) = O(V^2)$.

3. Explain how a vertex u of a directed graph can end up in a depth-first tree containing only u, even though u has both incoming and outgoing edges in G.

Solution:

For example, we have a directed graph G shown as below:



If we consider the visit order in alphabetical order, we will firstly visit A, and it will generate a DFS tree that only contain node A. Then, we will visit B, it also generates a DFS tree that only contain node B. Finally, we will visit C, it also generates a DFS tree that only contain node C. As we can see, Node B has both incoming and outcoming edges, but it ends up three trees only have one node, respectively. One of them is a tree that contains only B.

4. Write a method that takes any two node u and v in a tree T, and quickly determines if the node u in the tree is a $descendant$ or $ancester$ of node v.

Solution:

We can simply run DFS on a tree T and set the start node as the root, record the discover time and finish time of every node. Because T is a tree, the number of edges is $n-1$. So, the total running time is $O(V+E) = O(n+n-1) = O(n)$. As we have proved in the text book, given any 2 nodes u, v in the DFS tree, if $u.d < v.d < v.f < u.f$, u is the ancestor of v. Else if $v.d < u.d < u.f < v.f$, u is the descendant of v. Because we only need $O(n)$ time to complete DFS and store the discover time and finish time. Other operations are in constant time ($O(1)$), so the total running time is $O(n)$. More specific, we need $O(n)$ time to initialize and $O(1)$ time to determine the relationship between 2 nodes.

5. Problem 22-1 on page 621 of CLRS.

Solution:

- BFS on undirected graph:
  - (a) Since BFS first explores all white neighbors of a vertex, there cannot be any forward or back edges, all such edges would be in the tree.
  - (b) For each ordered tree edge (u, v), v is added through u, and while adding, we set $v.d = u.d + 1$.
  - (c) Let u be enqueued before v. v is then closer to the tail. From Lemma 22.3, we know that the difference between the distances of the tail and the head of the queue can at most be one, with tail's distance being greater. So we must have either $v.d = u.d$ or $v.d = u.d + 1$.
- BFS on directed graph:
  - (a) Similar to above.
  - (b) Similar to above.
  - (c) Holds for all edges.
  - (d) Since v was visited before u, we have $v.d \leq u.d$. Also, $v.d \geq 0$ for any v.

6. Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_\pi \subseteq E$ such that for each vertex $v \in V$, the unique simple path in the graph (V, E_π) from s to v is a shortest path in G, yet the set of edges E_π cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.

Solution:

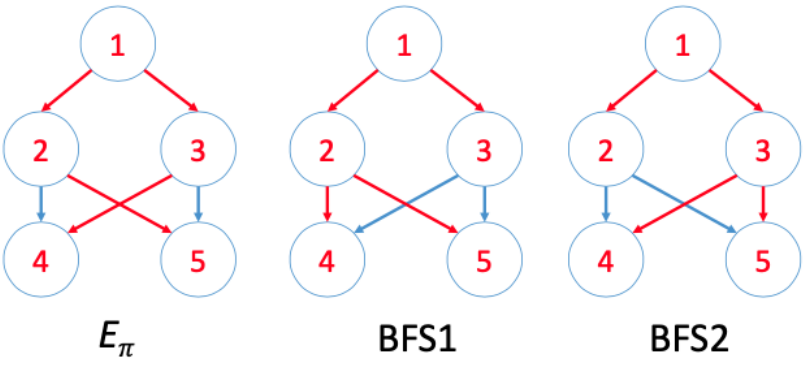


Figure 3: Solution for problem 6