CMPS 101
Final Review Problems

1. Let $G = (V, E)$ be a graph with n vertices, m edges, and k connected components.
   a. Show that if G is connected and acyclic, then $m = n - 1$. Use induction on either m or n.
   b. Show that if G is acyclic, then $m = n - k$. Use part (a).
   c. Show that if G is connected, then $m \geq n - 1$. Use induction on m.
   d. Show that in any graph G, $m \geq n - k$. Use part (c).

2. Let G be a digraph. Determine whether, at any point during a Depth First Search of G, there can exist an edge of the following kind.
   a. A tree edge that joins a white vertex to a gray vertex.
   b. A back edge that joins a black vertex to a white vertex.
   c. A forward edge that joins a gray vertex to a black vertex.
   d. A cross edge that joins a black vertex to a gray vertex.
   e. A tree edge that joins a gray vertex to a gray vertex.
   f. A forward edge that joins a black vertex to a black vertex.
   g. A cross edge that joins a white vertex to a black vertex.
   h. A back edge that joins a gray vertex to a white vertex.

3. a. State the parenthesis theorem.
   b. State the white path theorem.
   c. State the max-Heap property.
   d. State the min-Heap property.

4. Let G be a directed graph. Prove that if G contains a directed cycle, then DFS(G) produces a back edge. (Hint: use the white path theorem.)

5. Let T be a binary tree. Let $n(T)$ denote the number of nodes in T, and $h(T)$ denote the height of T. Show that $h(T) \geq \lfloor \lg(n(T)) \rfloor$. (Hint: You may use the following fact without proof. For any positive integer k, $\lfloor \lg(2k+1) \rfloor = \lfloor \lg(2k) \rfloor$.)

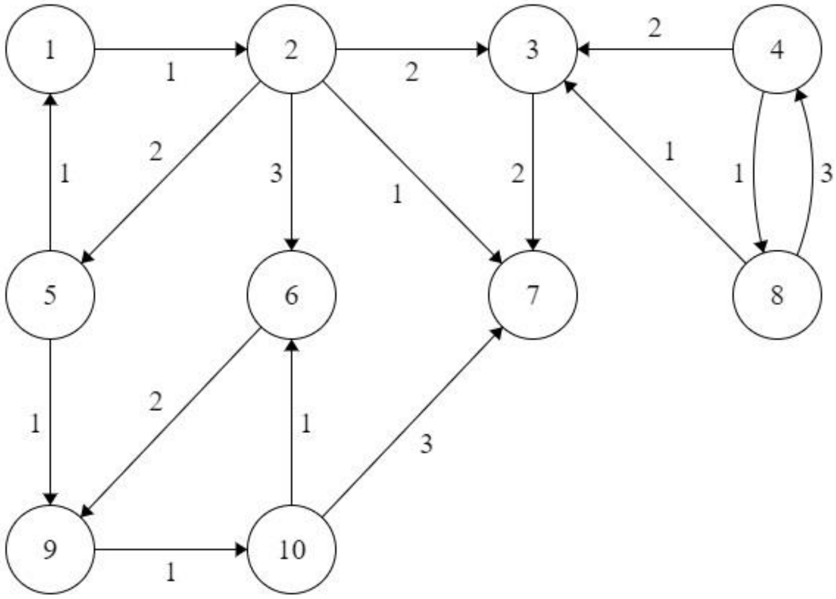
6. Re-write the algorithms Heapify, and HeapIncreaseKey from the point of view of a min-Heap, rather than a max-Heap. (In particular, HeapIncreaseKey should be renamed HeapDecreaseKey.)

7. Trace HeapSort on the following arrays. Show the state of both the array and ACBT after each swap.
   a. (9, 3, 5, 4, 8, 2, 5, 10, 12, 2, 7, 4)
   b. (5, 3, 7, 1, 10, 12, 19, 24, 5, 7, 2, 6)
   c. (9, 8, 7, 6, 5, 4, 3, 2, 1)

8. Let G be a directed graph, and let $s,\ x \in V(G)$. Suppose that after Initialize(G, s) is executed, some sequence of calls to Relax(,) results in $d[x]$ becoming finite. Show that G contains an s-x path of weight $d[x]$. (Use strong induction on the number of calls to Relax(,).)

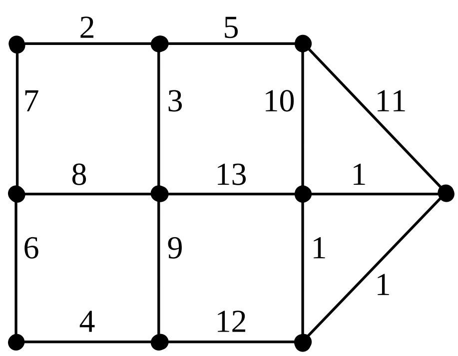
9. Let G be a directed graph, $s,\ x \in V(G)$, and suppose Initialize(G, s) is executed. Show that the inequality $\delta(s,\ x) \leq d[x]$ is maintained over *any* sequence of calls to Relax(,). (Use the result of problem 8.)

10. Perform Dijkstra(G, s) on the weighted digraph below. Trace the d[ ] and p[ ] values for each vertex after each call to Relax(,), and draw the resulting Shortest Paths tree.
    a. Use $s = 1$ as source vertex.
    b. Use $s = 5$ as source vertex.



11. Let G be a weighted connected graph (undirected) with *distinct* edge weights. Show that G contains a *unique* minimum weight spanning tree.

12. The following weighted graph contains three minimum weight spanning trees. Run the MWST algorithm of Kruskal on this graph to find two MWSTs. Find a third MWST by inspection.



13. Draw the Binary Search Tree resulting from inserting the keys: 5 8 3 4 6 1 9 2 7 (in that order) into an initially empty tree. Write pseudo-code for the following recursive algorithms, and write their output when run on this tree.
    a. InOrderTreeWalk()
    b. PreOrderTreeWalk()
    c. PostOrderTreeWalk()

14. State the Red-Black Tree Properties, then assign colors to the nodes in the above BST in such a way that it becomes a valid RBT. Note there is more than one way to do this. Find all such color assignments.

15. Let x be a node in a Red-Black Tree, and let $N(x)$ denote the number of internal nodes in the subtree rooted at x. Show that $N(x) \geq 2^{\mathrm{bh}(x)} - 1$. (Hint: use strong induction on $\mathrm{height}(x)$.)

16. Let T be a Red-Black Tree having n internal nodes, and height h. Show that $h \leq 2\, lg(\, n + 1)$. (Hint: use the result of the previous problem and RBT property (4).)

17. Insert the following keys (in order) into an initially empty Binary Search Tree: 11, 2, 13, 1, 3, 12, 4, 9, 7, 10, 6, 8, 5. Draw the resulting Binary Search Tree. Prove that it is not possible to assign colors Red and Black to the nodes of this tree in such a way that the Red-Black tree properties are satisfied. (Hint: use contradiction and the result of problem 16.)