ESO207A: Data Structures and Algorithms	
Homework *3*	*Due Date: 23rd Oct in class*

Instructions.

1. Start each problem on a new sheet. For each problem, Write your name, Roll No., the problem number, the date and the names of any students with whom you collaborated.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A clear description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks.*

Problem 1. Bipartite Graphs. A *bipartite* graph is an undirected graph $G = (V, E)$ whose vertices can be partitioned into two sets $V = V_1 \cup V_2$, where, V_1 and V_2 are disjoint subsets, and there are no edges between vertices in the same set (i.e., for any $u, v \in V_1$, there is no edge between u and v.

(a) A graph G is said to be colorable with k-colors, if each vertex is given one of the k colors and no two adjacent vertices are given the same color. (Adjacent vertices are given different colors). Show that a graph is bipartite iff it is 2-colorable.

(b) Show that an undirected graph is bipartite if and only if it contains no cycles of odd length.

(c) Give a linear time $O(|V|+|E|)$ algorithm to determine whether an undirected graph is bipartite or not.

(d) If an undirected graph has exactly one odd cycle, what is the minimum number of colors needed to color it.

Problem 2. Diameter of a tree A *tree* (also called a free tree in the Appendix of the CLRS book) is an undirected, acyclic and connected graph $T = (V, E)$. Design a linear time algorithm to compute the diameter of a given unweighted tree T, that is,

$$\text{diameter}(T) = \max_{u,v \in V} \delta(u, v)$$

where, $\delta(u, v)$ is the length of the shortest path from u to v. The algorithm should also maintain enough information to report a simple path whose length equals the diameter.

Problem 3. DFS Basics.

1. Give a counterexample to the conjecture that if a directed graph G contains a path from u to v and $u.d < v.d$ in a DFS of G, then, v is a descendant of u in the depth-first tree produced.
2. Show by an example that the DFS of a directed graph with a vertex u can end up as a solitary vertex u in the DFS forest of G, even though u has incoming and outgoing edges from it.

Problem 4. [Problem 22-2 of CLRS: Articulation points, bridges and biconnected components] This question concerns an undirected, connected graph $G = (V, E)$. An **articulation point** or a *cut-vertex* of G is a vertex whose removal disconnects G. A **bridge** of G is an edge whose removal disconnects G. A **biconnected component** of G is a maximal set of edges such that any two edges in the set lie on a simple cycle. Design an algorithm to determine the articulation points, bridges and biconnected components using DFS. Let $G_T = (V, E_T)$ be a depth first tree of G.

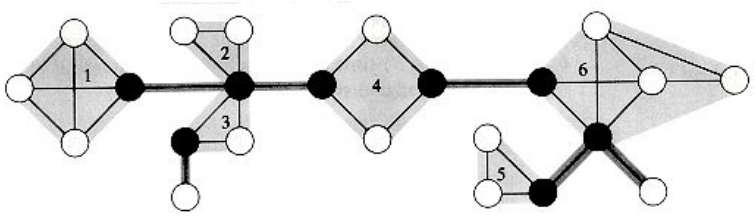

Figure 1: Shaded edge sets are biconnected components. Dark vertices are articulation points and darkened edges are bridges.

(a). Prove that the root of G_T is an articulation point of G iff it has two children in G_T.

(b). Let v be a non-root vertex of G_T. Show that v is an articulation point of G iff v has a child s such that there is no back edge from s or any descendant of s to a proper ancestor of v.

(c). Let

$$v.low = \min \begin{cases} v.d \ , \\ w.d : \ (u, w) \text{ is a back edge for some descendant } u \text{ of } v. \end{cases}$$

(d). Give an algorithm to compute all articulation points in time $O(|E|)$.

(e). Show that an edge of G is a bridge iff it does not lie on any simple cycle of G.

(f). Give an algorithm to compute all the bridges of G in time $O(|E|)$.

(g). Prove that the biconnected components of G *partition* the non-bridge edges of G.

(h). Give an $O(|E|)$ algorithm to label each non-bridge edge e of G with a positive integer $e.bcc$ such that $e.bcc = e'.bcc$ iff e and e' lie in the same biconnected component.