COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Group Worksheet week 7 – 10.23.14
30 minutes

Problem #1: A connected graph is one in which every vertex is reachable from every other vertex. Prove that every connected undirected graph has a vertex whose removal (including all adjacent edges) will not disconnect the graph. Design an algorithm to find a vertex whose removal will not disconnect the graph. You may find it easier to construct a proof around an algorithm.

Problem #2: DFS – from recursion to iteration

We implemented DFS using the simple recursive routine shown below (this version of DFS stores paths unlike the simpler version).

```
private void dfs(Graph G, int v) {
    marked[v] = true;
    for (int w : G.adj(v)) {
        if (!marked[w]) {
            edgeTo[w] = v;
            dfs(G, w);
        }
    }
}
```

Rewrite this version w/o using recursion.

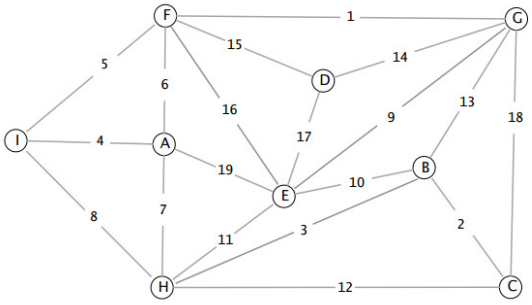
```
private void dfs_iterative(Graph G, int v) {
```

Problem #3: Odd length paths

Given a directed graph and a starting vertex u, give an algorithm for finding all vertices such that there is an odd-length path to those vertices from u. These paths may involve cycles. Your algorithm should complete in $O(E + V)$ time. If you're stuck, you might also try to come up with an algorithm that completes in $O(EV)$ time (which might be a little easier?)

Problem 4: Graph Problems

Consider the following weighted graph.



(a) Starting with vertex I, find the DFS sequence, if the heuristic "the smaller weight" is used to decide which node to process next.

(b) Device an algorithm, that finds the number of hops (minimum) from vertex I to any other connected vertex.

(c) Think about a possible algorithm, where this idea can be extended to finding the shortest path from vertex I to any other vertex (we will discuss this in detail soon)