EECS 340, Breakout Session Notes, November 11, 2019

1. As usual, begin by handing out the quiz and collect it quiz at 3:40.

2. Break the students into groups of 2 or 3 and assign them this problem: We want to identify if a graph has any cycles of odd length. That is, a path that starts at a vertex v and returns to that vertex visiting no vertex more than once, and the number of vertices on the cycle is odd. (You can motivate this by telling them that many NP-hard problems like Vertex Color, Independent Set, and Coloring, can be solved in polynomial time if the graph has no odd length cycles.)

3. Let them think for a minute, and then give them the hint that we can reduce this problem to Breadth First Search. Here is the BFS algorithm we did in lecture:

```
BFS(graph G)
1   mark all vertices as unvisited
2   set P[v] = null for all vertices v
3   for each v in G
4       if v is unvisited
5           BFS-Tree(G, v)

BFS-Tree(graph G, vertex s)
6   Q: an empty queue
7   mark s as visited
8   add s to Q
9   while Q is not empty
10      v → remove from Q
11      for each vertex w that is a neighbor of v
12          if w is unvisited then
13              mark w as visited
14              set P[w] = v
15              add w to the Q
```

4. The first idea they may come up with is to do a BFS from each vertex until it returns to the vertex. However, this will only give the smallest cycle. There could be an odd one that is longer.

5. A key point they should realize quickly (maybe with your help) is that any edge that is not part of the BFS tree must form a cycle. Where in the algorithm do we find a non-BFS tree edge? At line 12 if w is already visited. (They might thing that they can just then see how big the cycle with that edge is, but that still takes too long: up to m non-tree edges, and tracing the cycle is a minimum of $O(n)$. We want a $O(m)$ algorithm total.) You can try giving a simple graph of say 7 vertices, and trace a BFS. They may notice a pattern on how the non-tree edges connect in the tree.

6. Here is the proof you should help them to: Run a BFS, and keep track of the "level" of each vertex in the BFS tree. (Let L[v] = the level of the vertex, start L[s] = 0, and each time a vertex w is marked as visited, set L[w] = L[v]+1.) There are three possibilities

   - The non-tree edge connects vertices on the same tier
   - The non-tree edge connects vertices on adjacent tiers
   - The non-tree edge connects vertices on tiers further than 1 away

   The third is impossible. If vertex u connects to v more than 1 tier below, v would have been added when u was visited.

If a non-tree edge connects two vertices of the same tier we have an odd length cycle. Let the edge connect a and b, and let x be the common parent of a and b in the BFS-tree. Let k be the length of the path from x to a, and likewise from x to b, and we have a cycle of size $2k + 1$.

If a non-tree edge connects a and b on adjacent tiers, let x be the common parent, and let k be the length from x to a, and $k + 1$ the length from x to b, and we have a cycle of $2k + 2$.