Homework 11

Yutong Huang (yxh589)

Problem 1

Proof. Let M be a deterministic Turing machine. Define M as follows:

1. On input $\langle G \rangle$, select a vertex u as the starting vertex
2. select an edge from u, (u, v), as the starting edge
3. traverse the graph through (u, v)
4. if M comes back to u through an edge other than (u, v) accept, otherwise repeat step 1 to 4 until all vertices are enumerated
5. reject.

Then M decides UCYCLE.
Case 1: G contains a cycle, then the graph traversal would eventually come across a vertex u and an edge that leads back to u that's different from $(u, v) \implies M$ accepts.
Case 2: G does not contain a cycle, then the traversal always comes back to u via $(u, v) \implies M$ rejects.

Since all vertices are enumerated in logspace, M also decides UCYCLE in logspace.

Therefore, UCYCLE $\in L$. □

Problem 2

Because **NL=coNL**, we can prove that BIPARTITE $\in$ **NL** by proving that $\overline{\text{BIPARTITE}} \in$ **NL**.

Proof. Let M be a non deterministic Turing machine. Define M as follows:

```
M(G: graph){
    int num <- 0;
    node start <- non deterministically choose a starting node from G;
    node next <- non deterministically choose a next node of start;
    while (num <= sizeOf(G.vertices)){
        if (next == start && num is odd){
            accept;
        }
        next <- non deterministically choose a next node of next;
        increment num;
    }
    reject;
}
```

It's clear that M decides $\overline{\text{BIPARTITE}}$ correctly.
Case 1: G does not have an odd cycle, then M will loop through all nodes and reject;
Case 2: G does have an odd cycle, then M will find the node that leads back to the starting node and accept.

And since M only needs to keep track of `num`, `start` and `next`, it can decides $\overline{\text{BIPARTITE}}$ in logspace.

Therefore $\overline{\text{BIPARTITE}} \in$ **NL** $\implies$ BIPARTITE $\in$ **NL** □

Problem 3

To prove that STRONGLY-CONNECTED is NL-complete, we need to prove that:

1. STRONGLY-CONNECTED $\in$ NL

   Proof. Since **NL=coNL**, we can prove this by showing that $\overline{\text{STRONGLY-CONNECTED}} \in$ NL.
   Construct a non deterministic Turing machine that decides $\overline{\text{STRONGLY-CONNECTED}}$:

   ```
   function M(G: graph){
       non deterministically select 2 nodes: a, b;
       if PATH(a,b) accepts {
           then there is a path from a to b, reject;
       } else {
           this is not a strongly connected graph, accept;
       }
   }
   ```

   Since this algorithm only needs to keep track of `a` and `b`, it only requires log space.

   Therefore $\overline{\text{STRONGLY-CONNECTED}} \in$ NL $\implies$ STRONGLY-CONNECTED $\in$ NL. ☐

2. $\forall L \in$ NL, L can be logspace reduced to STRONGLY-CONNECTED

   Proof. Since PATH is NL-complete, we can prove this by reducing PATH to STRONGLY-CONNECTED.
   Consider the following non deterministic Turing machine:

   ```
   function reduction(G: graph, s: vertex, t: vertex){
       Copy G onto output tape;
       foreach(node in G.vertices){
           write edge(node, s) to output tape;
           write edge(t, node) to output tape;
       }
   }
   ```

   Case 1: If there is a path from s to t, then the constructed graph is strongly connected because now every node can get to every other node via the path from s to t.
   Case 2: If there is no path from s to t, then the graph is not strongly connected, because the only additional edges in the constructed graph go into s and out of t, therefore no path from s to t is formed.

   This procedure only needs log space to store the node pointer.

   Therefore PATH is log space reducible to STRONGLY-CONNECTED $\implies$ STRONGLY-CONNECTED is NL-complete. ☐