EC330 HW 7

Chase
Mattke
U18719879

1) Bipartite Graph - BFS approach ~

Let a graph G be bipartite, where the two sets of vertices are denoted as V_1 & V_2. In the following algorithm, we choose to label nodes respective to each set as red & black. We start at a vertex in V_1, per say, where we choose that this vertex & vertices in V_1 are red. Thus, all of its neighbors (that are in V_2 if G is bipartite) are labeled as black. To clarify further, each neighbor of these neighbors should be red. The pseudo-code can be written as follows:

```
bool  isBipartite (G) {   // N = any node
      color vertex N red, queue.push(N)
      while (!queue.empty())
            x = queue.pop()
        for all neighbors n of x (if no neighbors, exit for loop)
            if n is uncolored
                color it x's opposite color, queue.push(n)
            else
                if n is x's color, return 0
      return 1   // G is bipartite
}
```

The run-time of this algorithm $O(|V|+|E|)$ is due to the fact we do $O(1)$ work $|V|$ times placing/removing elements in the queue, & $O(1)$ work to check neighbor's colors, $2|E|$ times.

directed / undirected

2) Celebrity ~ parent that doesn't point to children

In the event a node u is pointed at by edges from node(s) v, but node u's edges don't point at node(s) v, node u is a celebrity.

Graph G has a Eulerian cycle when celebrity node u is strongly connected by node(s) v, and the in degree = the out degree for every vertex. An Eulerian walk exists when a walk can start & end at the same vertex. However, this last case, making the Eulerian walk, is impossible because upon visiting the celebrity node u, no other edges can be visited. By definition, no edge can be directed away from celebrity node u.

Graph G has a Hamiltonian cycle where every vertex can be visited. This is possible as long as we don't start at the celebrity node u.

3) Dijkstra's Algorithm ~ worst case w/ unordered LL

To implement Dijkstra's algorithm using an unordered linked list, we first assume the input is an adjacency list. The evaluation of operations is as follows:

insertion : $O|V|$

update : $O|E|$

find/delete minimum : $O|V|$

→ worst-case runtime of $O(|E| + |V|^2)$, and because $|E| < |V|^2$ for directed & undirected graphs, the worst-case runtime boils down to $O(|V|^2)$.