Ian Chadwick
U90596416
4/19/2021

EC330 Applied Algorithms and Data Structures for Engineers
Spring 2021
Homework 6 Written Portion Answers

1. Bipartite Graph [20 pt]
Describe a linear-time algorithm for determining whether a given undirected graph is bipartite. You can write your algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code. Justify your answer (make sure you describe the data structures that you use).

Given the input node, *n*, in the connected graph, *G*, a linear-time algorithm to determine whether an undirected graph is bipartite could be implemented by using a modified Breadth-First Search algorithm. The algorithm would work on an unconnected graph, but you would have to re-run the algorithm on the nodes that weren't part of the first group(s).

Essentially, each node in the graph will be colored white to start and then as it is visited it will be colored grey, then black as normal for BFS, and in addition, it will also be tagged either 1 or 2. The first node will be tagged 1, and any node that it is adjacent to will be tagged 2. If it is a 2 node, it will tag any adjacent nodes 1. If while emptying the que of nodes, a node ever points to a node of the same tag number, it is not bipartite and you can return false. Otherwise there will be two groups of nodes tagged either 1 or 2, with 1 nodes only adjacent to 2 nodes and 2 nodes only adjacent to 1 nodes. This means that the graph *is* Bipartite.

This will run in O(V+E) time which is linear.

The data structures used for this implementation would be an adjacency list graph, and a queue. The nodes of the graph will have a color and a tag number.

NOTE: Specifics of implementation would be modified slightly based on the graph type if necessary
Pseudocode continued on next page to avoid page break

isBipartite(G, n)

- color every node white and tag with 0 O(1)
- color *n* grey O(1)
- tag *n* with 1 O(1)
- create a queue *Q* O(1)
- *Q*.push(*n*) O(1)
- while(!*Q*.empty()) O(V+E)
    - *h* <- *Q*.front();
        - for each node adjacent to h
            - if (node is the same tag# as h)
                - return false. Graph is not bipartite.
            - if (node is white)
                - tag it the opposite tag# of h
                    - *(i.e. 1 if h is 2; 2 if h is 1)*
                - color node grey
                - *Q*.push(node) put it on the queue *Q*
    - color h black
    - Q.pop(h)
- return true
    - If after running while loop, you did not return false, graph is bipartite.