**Casey Levy – CS 325 – HW 5**

**Problem 1**

**Shape

Description automatically generated Shape

Description automatically generatedShape

Description automatically generated**

**Shape

Description automatically generatedShape

Description automatically generatedShape

Description automatically generated**

**Shape

Description automatically generatedShape

Description automatically generatedShape

Description automatically generated**

Order would be **j – a – b – d – h – f – c – e – h – g**

Total Weight: **32**

**Problem 2**

1

4

1

1

1. MST before all edges are increased by 1 is **A – B – C – D**. The MST holds if we use Kruskal’s since the lowest value edge is always added, no matter its connection. This proves the MST remains unchanged if each edge is +1.
2. The shortest path would change if all edges are increased by one. If this happens, all connections between A, B, C, and D would be 2 instead of 1, making the direct path from A to D the shortest path since 5 < 6.

**Problem 3**

1. We could use a modified breadth-first search algorithm and modify it to skip over any edges that are less than W.

We could also use a modified Prim’s Algorithm as well in a similar fashion. We can skip the min value extracted from Q if the value is < W.

Ex: If W is 7 and GetMin(Q) gives us 6, then nothing more is done but get the next min value in Q

Last step, we count the total number of E if E = n-1 where n is the number of vertices. If it isn’t, then no path is found.

1. Runtime for BFS: **O(V + E)**

Runtime if using a binary heap: **O(E log V)**

**Problem 4**

2. Topological Sort: **CS 101, CS 160, CS 162, CS 271, MTH 111, CS 225, CS 201, MTH 201, CS 261, CS 301, CS 325**
3. Taking multiple courses, no pre-req conflict:

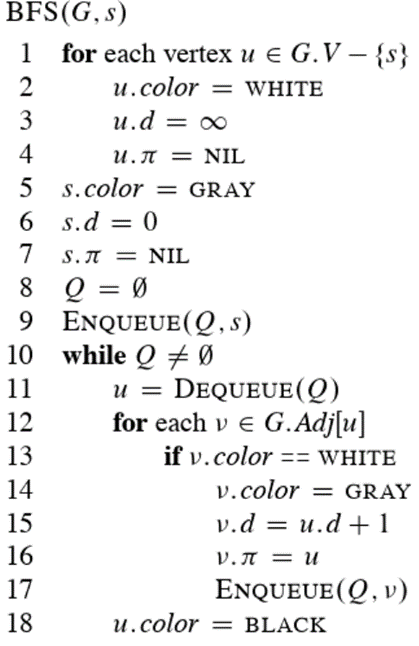
* **CS 101, MTH 111, CS 160**
* **MTH 201, CS 161, CS 162, CS 201**
* **CS 225, CS 271**
* **CS 261**
* **CS 301, CS 325**

1. Longest path: 4, 160 – 162 – 225 – 261 – 301.

This tells us that CS 301 has the most pre-requisites

**Problem 5**

1. A modified breadth-first search algorithm should be able to solve this problem. Below is the pseudocode taken from our textbook. “Babyfaces” would be assigned at lines 15/16 if ***v.d***is even. If it is odd, then “Heels” would be assigned instead. If both *u* and *v* are on the same team, then “impossible” would be returned.

****

1. Runtime: **O(V + E)**