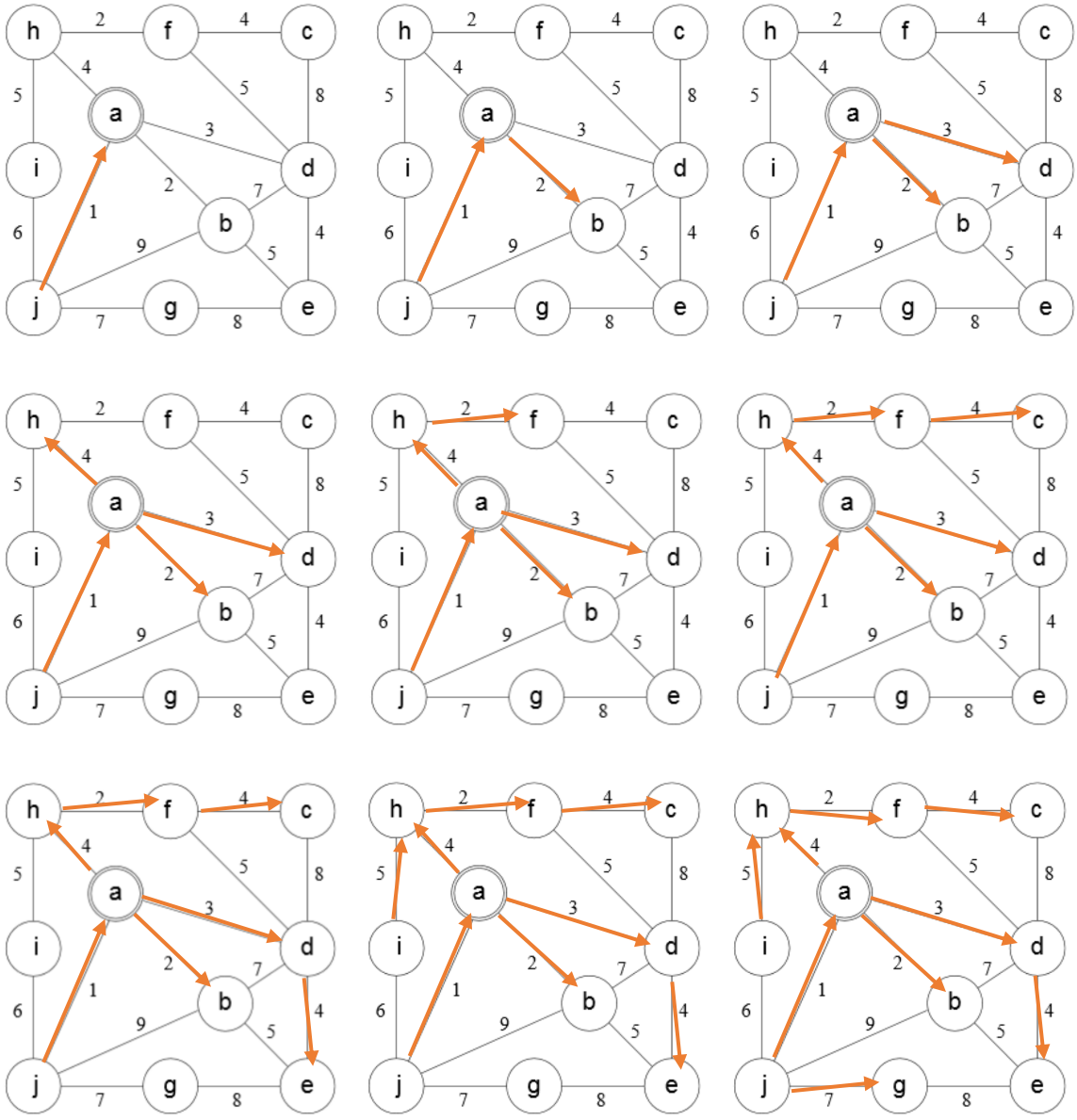
Casey Levy – CS 325 – HW 5

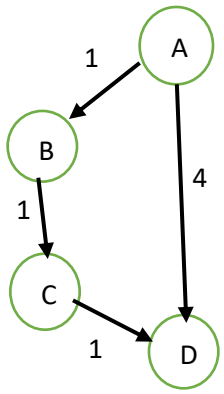
Problem 1



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

Total Weight: **32**

Problem 2



a) 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.

b) 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

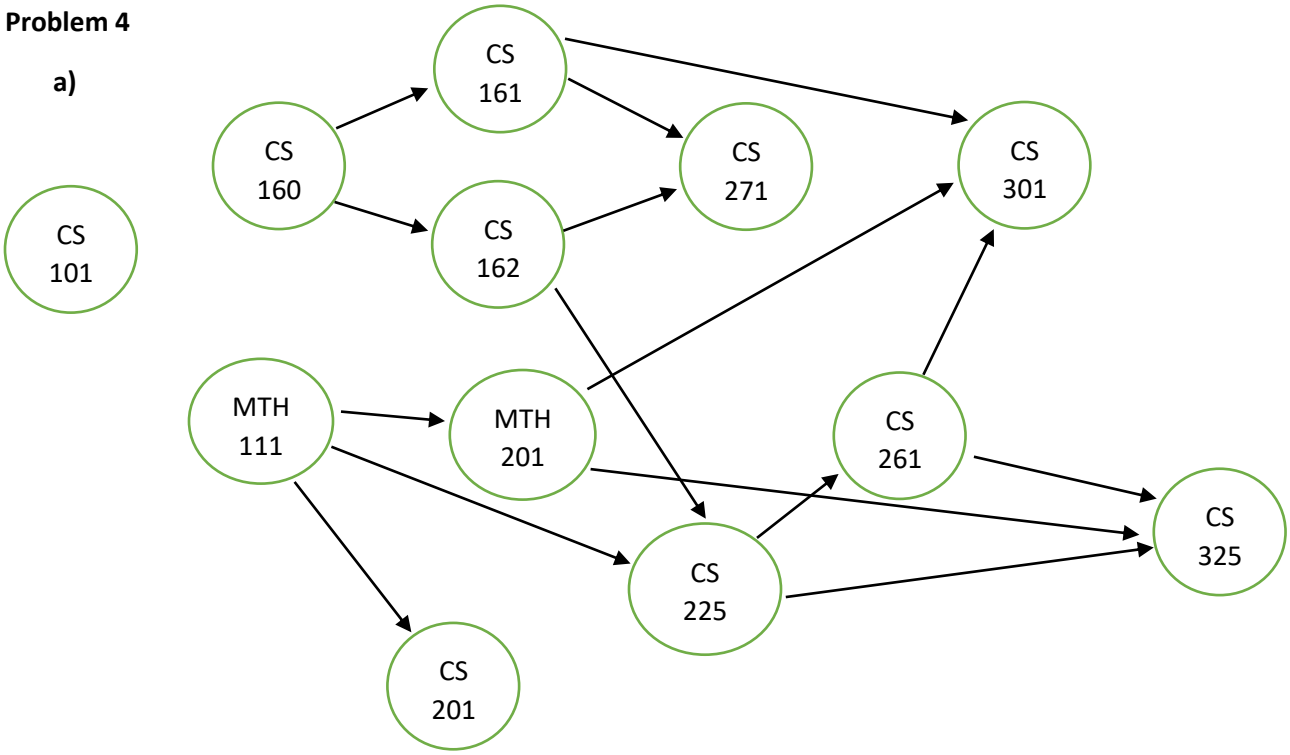
a) We could use a modified breadth-first search algorithm and modify it to skip over any edges that are less than W.

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.

b) Runtime for BFS: **O(V + E)**
Runtime if using a binary heap: **O(E log V)**

Problem 4

a)



b) Topological Sort: **CS 101, CS 160, CS 162, CS 271, MTH 111, CS 225, CS 201, MTH 201, CS 261, CS 301, CS 325**

c) 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**

d) Longest path: 4, 160 – 162 – 225 – 261 – 301.
This tells us that CS 301 has the most pre-requisites

Problem 5

a) 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.

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

b) Runtime: **O(V + E)**