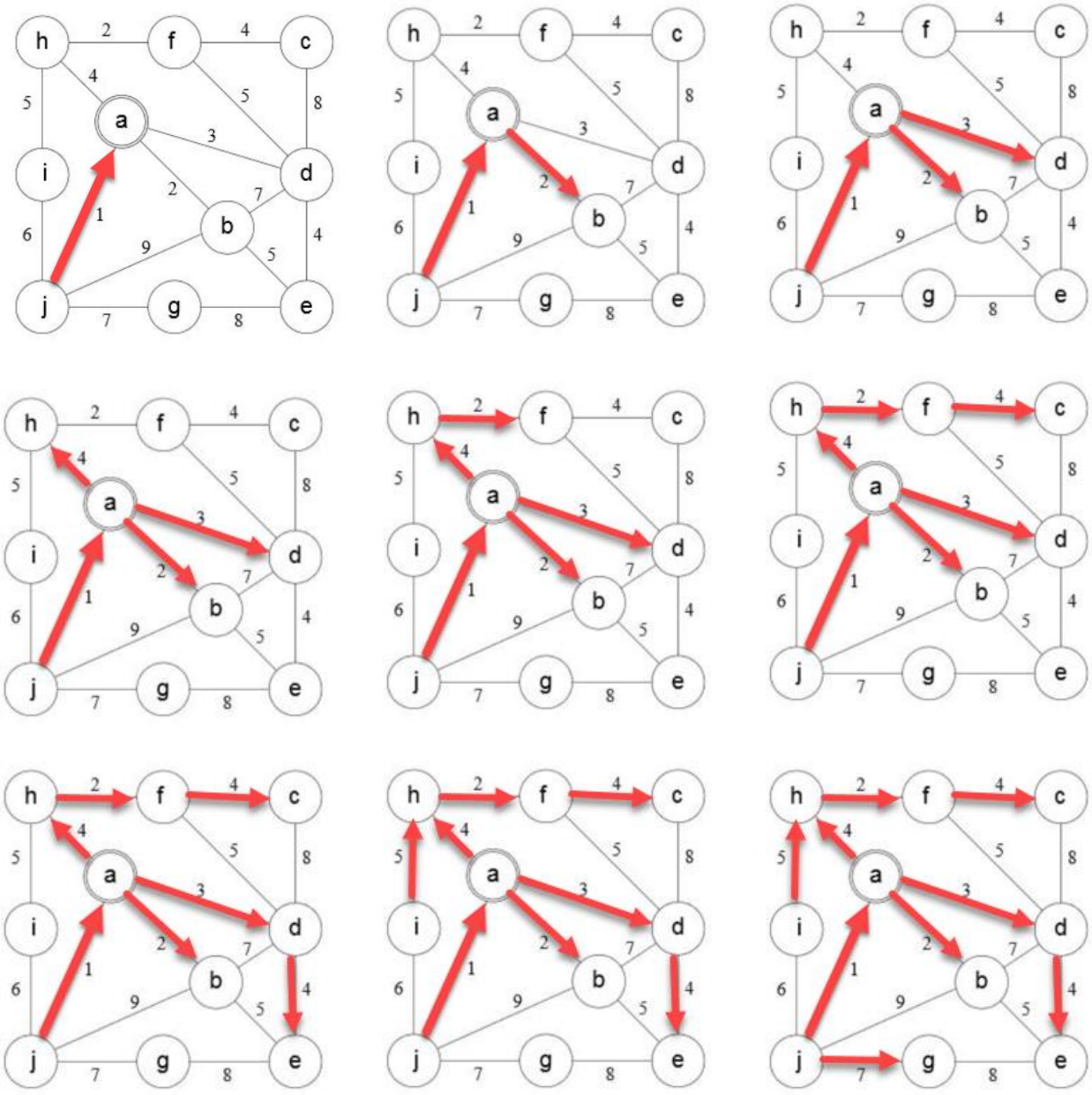
Bryan Rodriguez – Andrade

CS 325 F2020

Homework 5

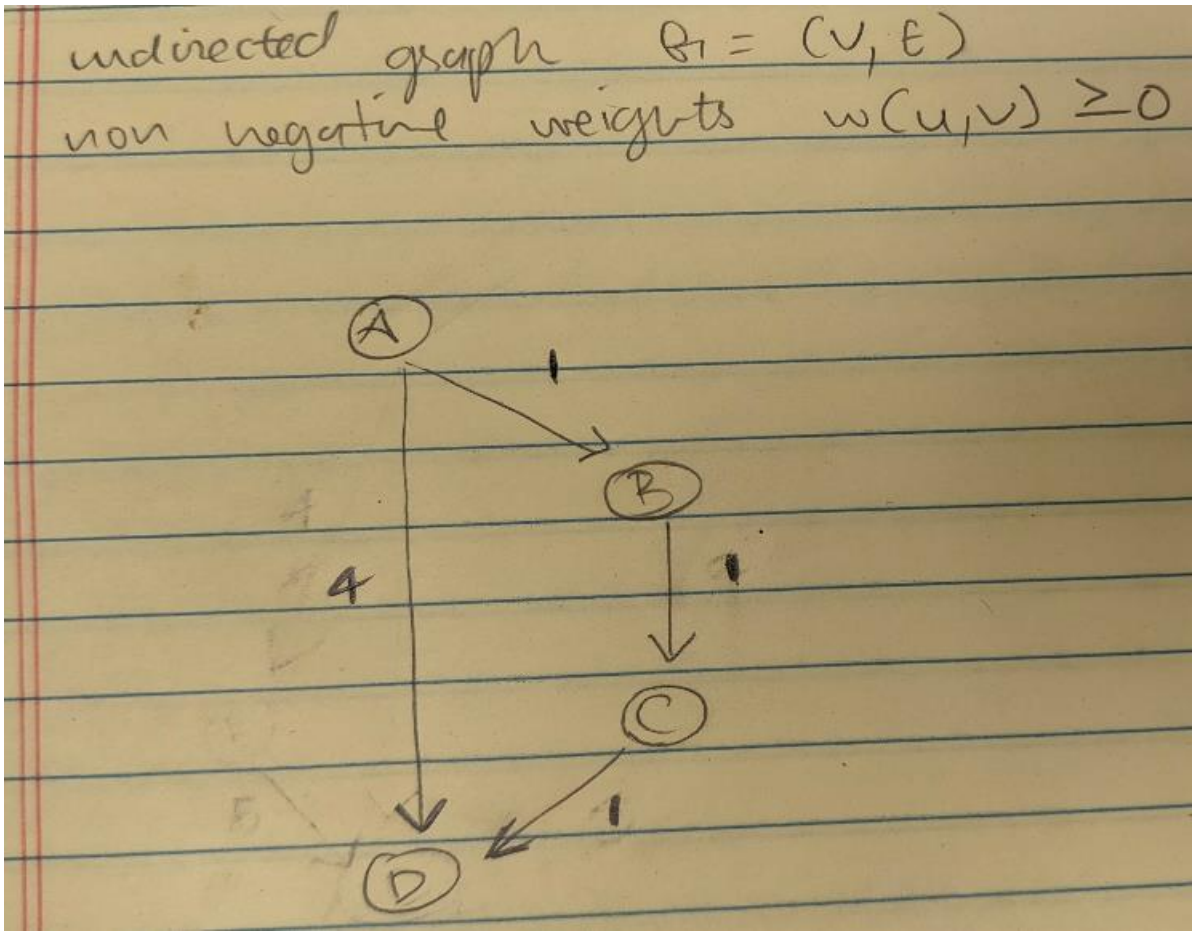
Problem 1

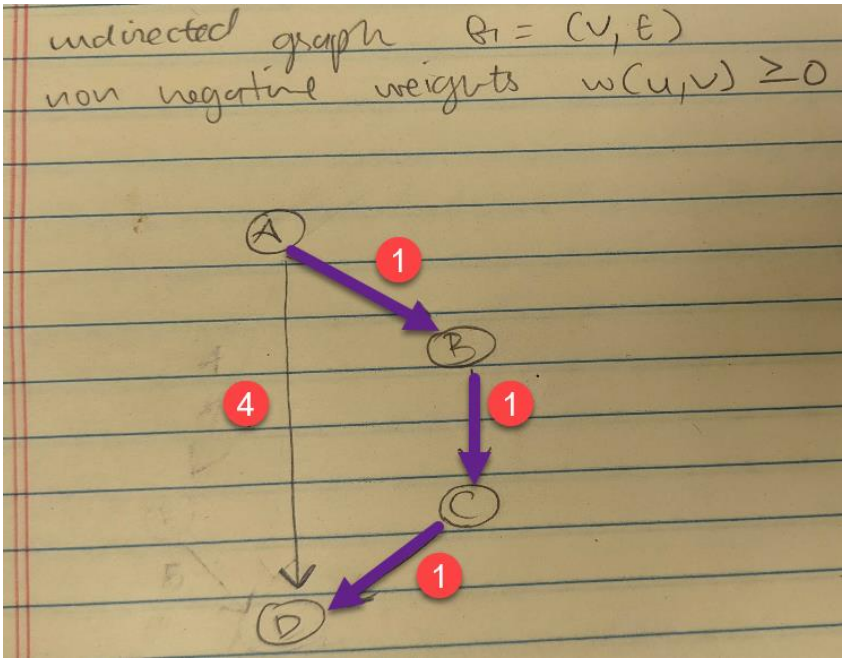


Primm's algorithm would go as follows: j->a->b->d->h->f->c->e->h->g

The total weight of the MST is 32.

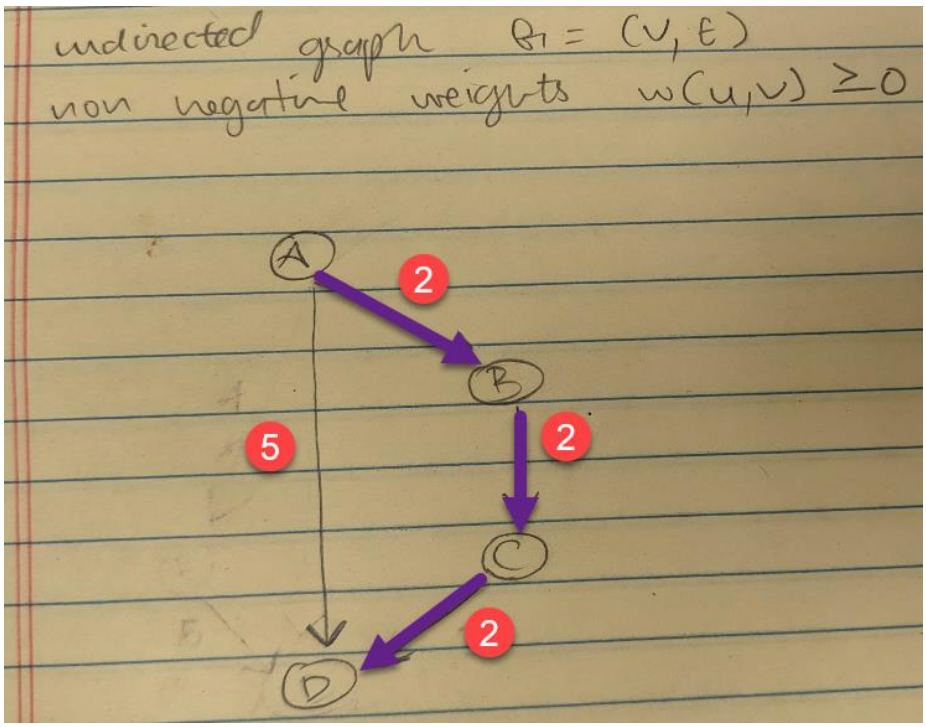
Problem 2

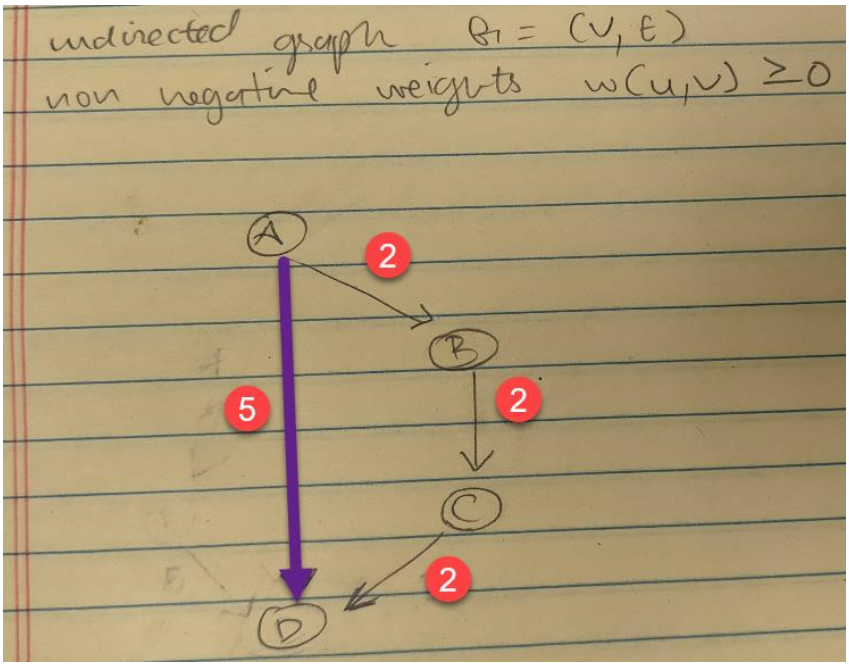

undirected graph G = (V, E)
non negative weights w(u,v) ≥ 0
A
B
C
D
1
4
1
1


undirected graph G = (V, E)
non negative weights w(u,v) ≥ 0
A
B
C
D
1
4
1
1

a) From above, the minimum spanning tree before all of the edges are increased by 1 is A->B->C->D, using Kruskal's the minimum spanning tree holds because the edges are increased because the smallest edge is always added regardless of connection. This show's that the minimum spanning tree would not change if each edge weight were increased by 1.

b) The shortest path would change in this scenario, from below, our original shortest path runs from A->B->C->D, however, when the weights are increased by 1 , the shortest path now is A->D





Problem 3

a) A breadth-first search algorithm can be used for this bottleneck, we would just need to modify it to skip any edges < W.
   a. Pseudocode:
      i. From the text we can use the BFS Pseudo:

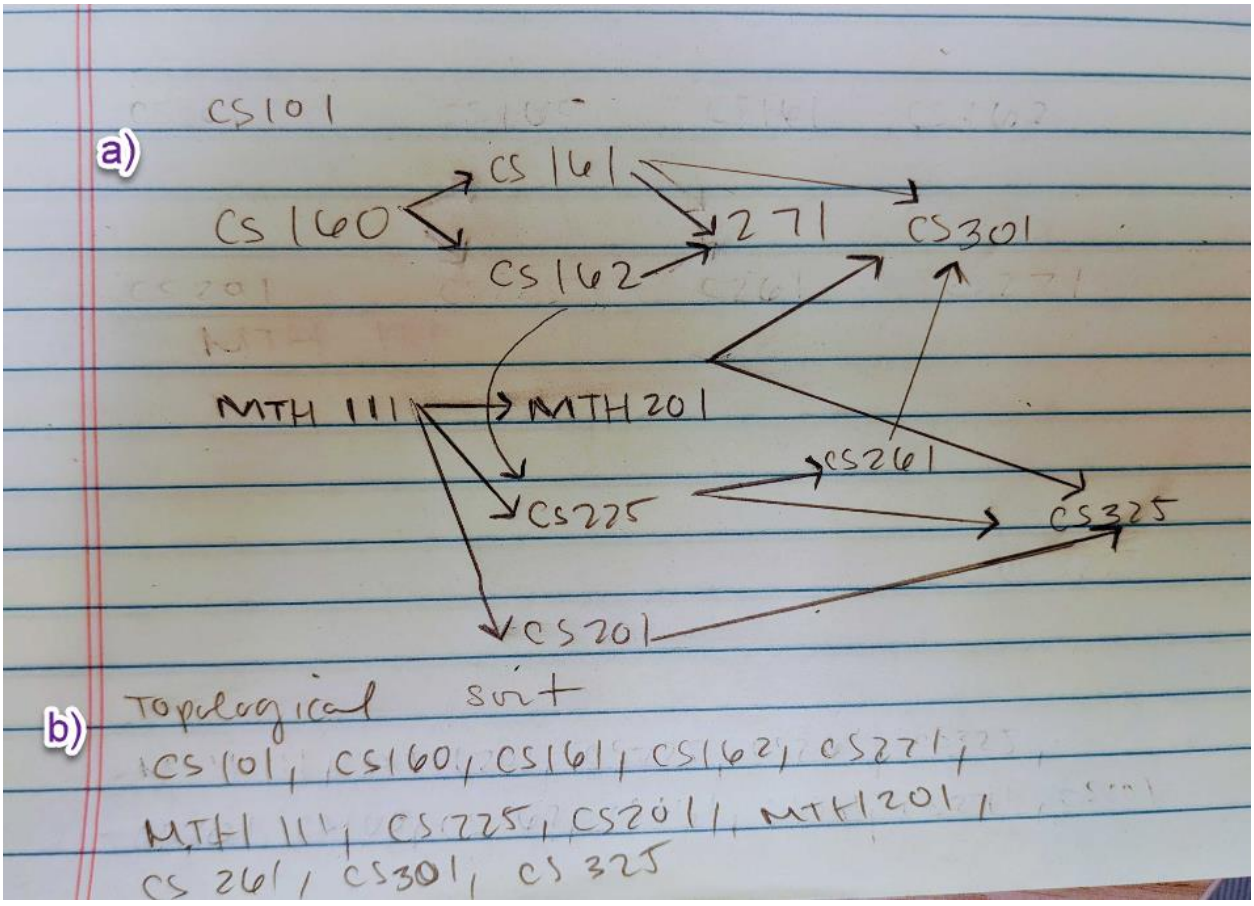
```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color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

      ii. The algorithm would be adjusted between line 14 and 15.

b) The runtime here is the same as BFS which is O(V+E)

Problem 4

Course	Prerequisite
CS 101	None
CS 160	None
CS 161	CS 160
CS 162	CS 160
CS 201	MTH 111
CS 225	MTH 111, CS 162
CS 261	CS 225
CS 271	CS 161, CS 162
CS 301	CS 161, CS 201, CS261
CS 325	MTH 201, CS 225, CS 261
MTH 111	None
MTH 201	MTH 111



The topological sort would be: 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 as long a there is no pre-req conflict:
   a. CS 101, MTH 111, CS 160
   b. MTH 201, CS 161, CS 162, CS 201
   c. CS 225, CS 271
   d. CS 261
   e. CS 301, CS 325

d) Following back CS 301 would give us: 160 → 162 → 225 → 261→ 301 so the longest path of this DAG is 4. This means that 301 has the highest number of pre-reqs associated to it.

Problem 5

a) This can be achieved using a BFS like the pseudo code linked above in problem 3.

```
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          <-- modified with: if weight >= W
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

Here instead of the modification at line 14 to account for W from problem 3 we would need to assign to “Babyfaces” at line 15/16 if v.d is even. If it is not then we would assign to “Heel’s”, furthermore we need to add a conditional if both u and v are on the same team then enter the print statement that returns “impossible”

b) Run time would be O(V+E)