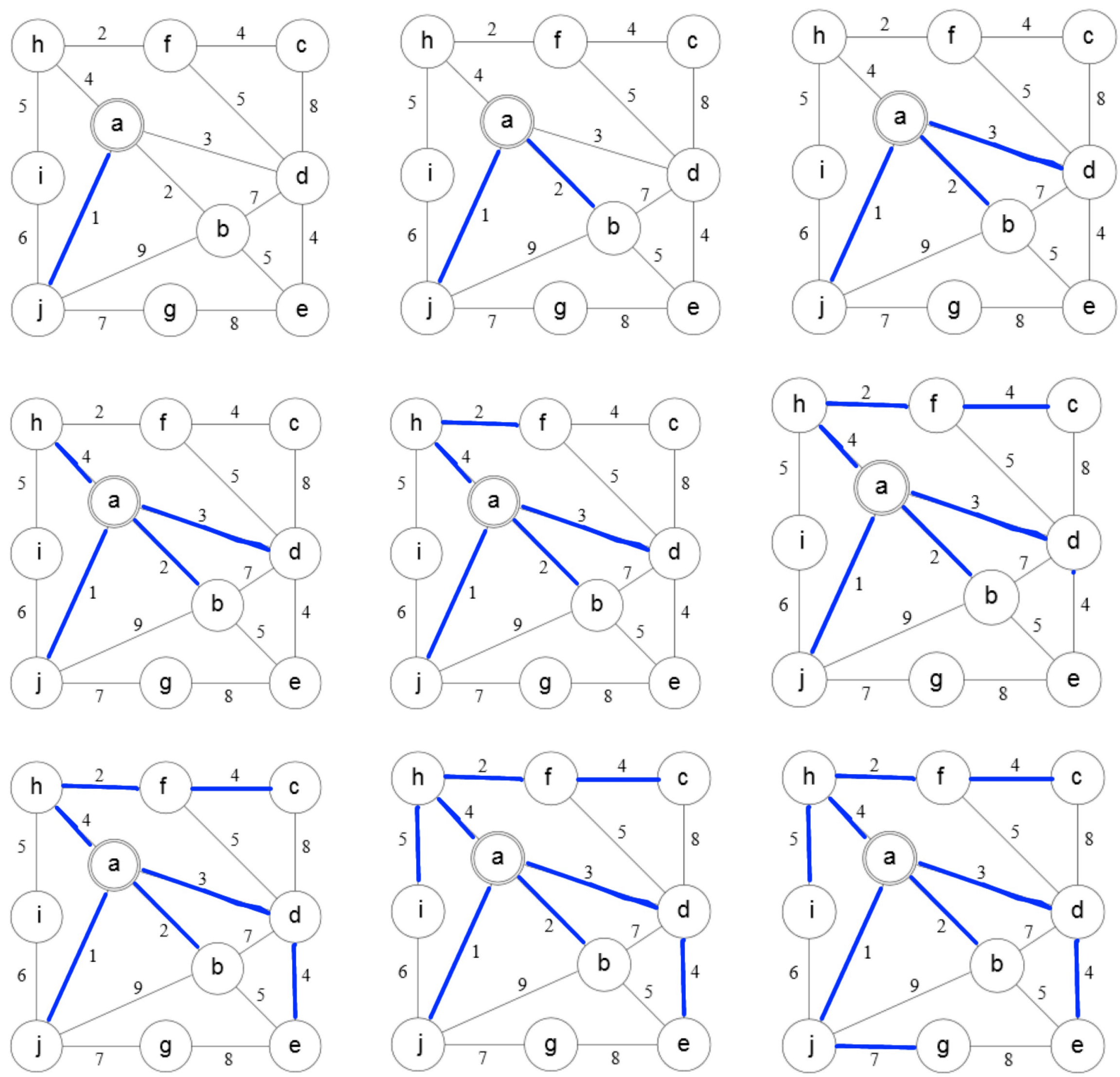
Thanhbinh Truong
CS325 Algorithms
May 13, 2018

Homework 5

Problem 1:

Weight of Minimum spanning tree = 32



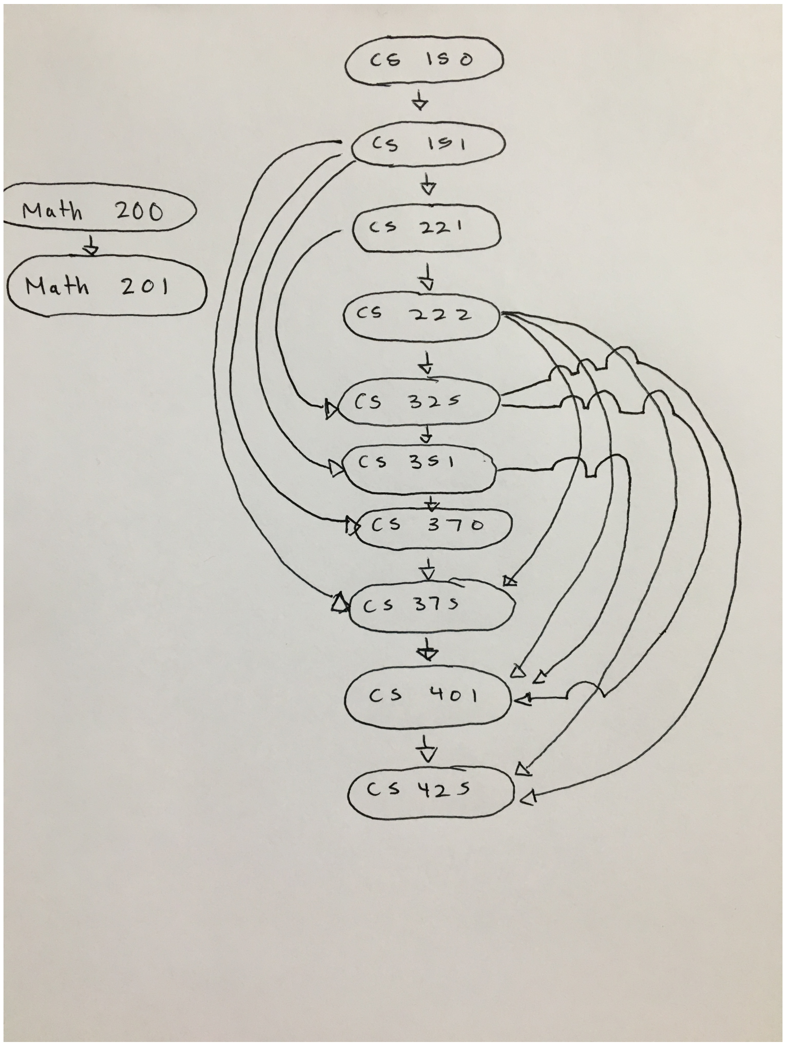
Problem 2:

a) No, the minimum spanning tree DOES NOT change. Using the Kruskal algorithm, it will build the MST through taking a certain path through the undirected graph. This path will consist of the lowest weights, and since the edge weights are distinct, the algorithm will follow the same pattern each time it is run. By adding 1 to each weight, it will not change the path the algorithm will take.
b) Now in this circumstance, the shortest path will change. This is a possibility because the length of a path is determined by the # of edges and the value of each of those edges. For example, you take a tree T and there is a path between points a and b that consist of 4 edges while holding a length of 5. Now, consider that there is another path that consist of 2 edges but has a length of 6. Obviously the length 4 path is the shortest and by increasing each edge by 1, the shortest path would now have a length of 9 and the length 5 would now only be at length 8. Therefore, this would be the new shortest path.

Problem 3:

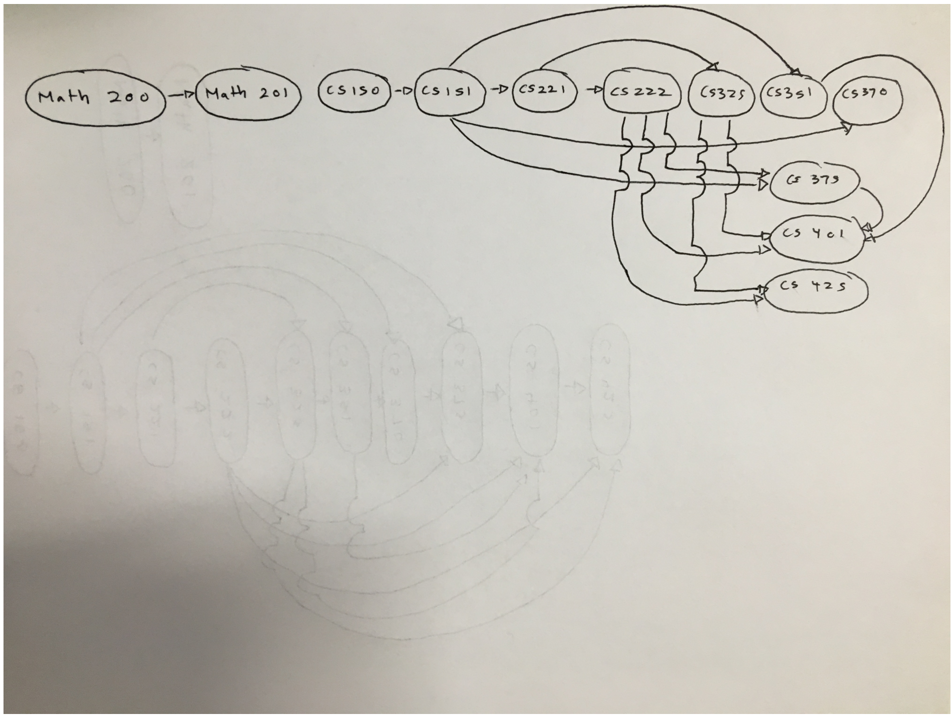
a) An efficient algorithm that solves this problem would be: A modified BFS. The modification would come through ignoring edges with a weight that holds a value less than W.
b) Running time: O(V+E)

Problem 4:

a)


b)

a. Math 200(21,24)
b. Math 201(22,23)
c. CS 150(1,20)
d. CS 151(2,19)
e. CS 221(3,18)
f. CS 222(4,17)
g. CS 325(9,10)
h. CS 351(11,12)
i. CS 370(13,14)
j. CS 375(15,16
k. CS 401(7,8)
l. CS 425(5,6)



c)

Term 1
Math 200
CS 150

Term 2
Math 201
CS 151

Term 3
CS 221
CS 351
CS 370

Term 4
CS 222
CS 325

Term 5
CS 375
CS 425

Term 6
CS 401

d) Longest path length in DAG: 5
   a. Length was found by looking through the graph and counting the length of each path.
   b. CS150, CS151, CS221, CS222, CS275, CS401 illustrates the longest path

5.

a)

```
g = input file graphed
BFS(g,first vertex)
	initiate vertices
	queue = first vertex

	while (queue != empty)
		node = queue.pop()
		for each neighbor in g.getNeighbors()
			if neighbor hasn't been visited
				visit
				mark visit
			if neighbor's name is even
				assign to babyface
			else
				assign to  heel
			if node and neighbor are on the same team
				designation can not work
If designation can not work
	print cannot work
Else
	print teams
```

b) Running time: O(n+r) due to BFS

c)