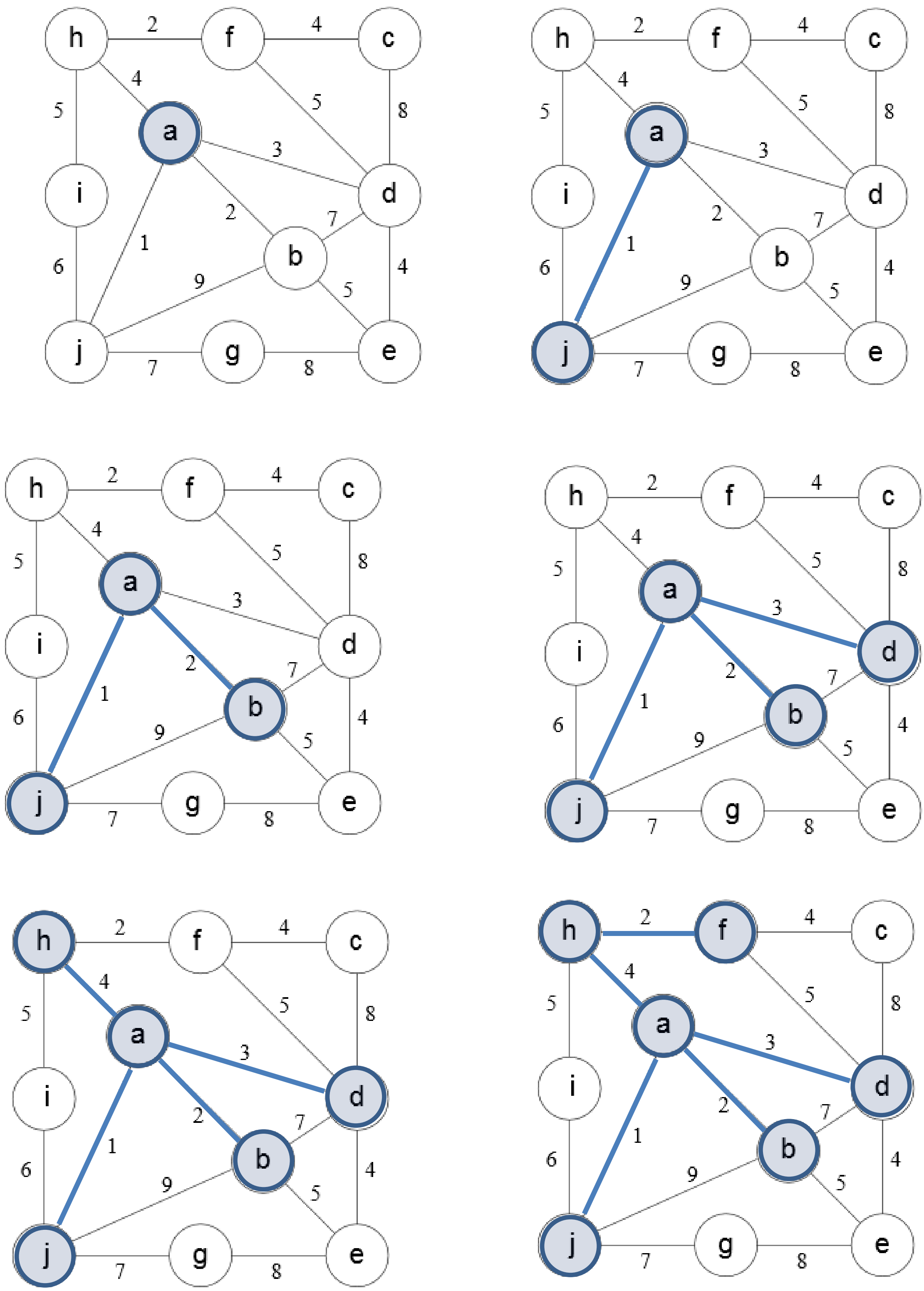
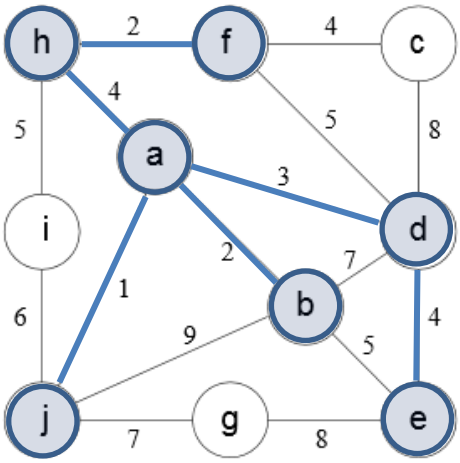
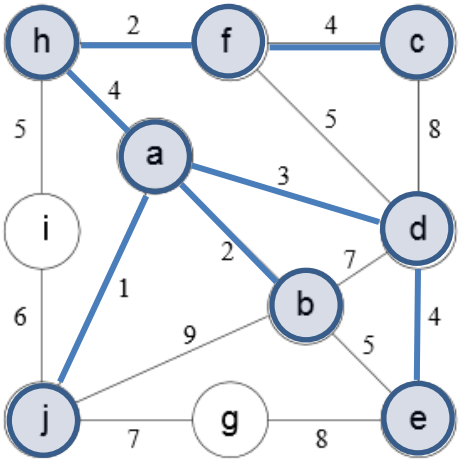
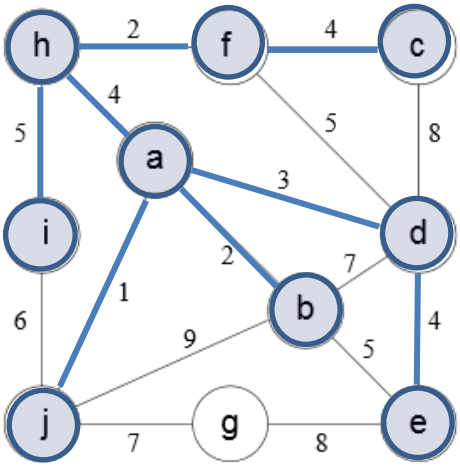
Brian Allison
11/5/2017
CS325
Assignment 5

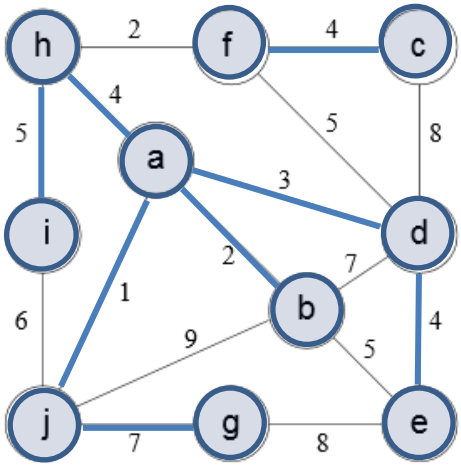
1. *(3 points)* Demonstrate Prim's algorithm on the graph below by showing the steps in subsequent graphs as shown in Figures 23.5 on page 635 of the text. What is the weight of the minimum spanning tree? Start at vertex a.
Ans: The weight is 32




h
f
c
a
i
d
b
j
g
e
2
4
4
5
5
8
3
2
7
1
6
9
4
5
7
8


h
f
c
a
i
d
b
j
g
e
2
4
4
5
5
8
3
2
7
1
6
9
4
5
7
8


h
f
c
a
i
d
b
j
g
e
2
4
4
5
5
8
3
2
7
1
6
9
4
5
7
8


h
f
c
a
i
d
b
j
g
e
2
4
4
5
5
8
3
2
7
1
6
9
4
5
7
8

2. *(6 points)* Consider an undirected graph G=(V,E) with nonnegative edge weights w(u,v)≥0. Suppose that you have computed a minimum spanning tree G, and that you have also computed shortest paths to all vertices from vertex s∈V. Now suppose each edge weight is increased by 1: the new weights w'(u,v) = w(u,v) + 1.

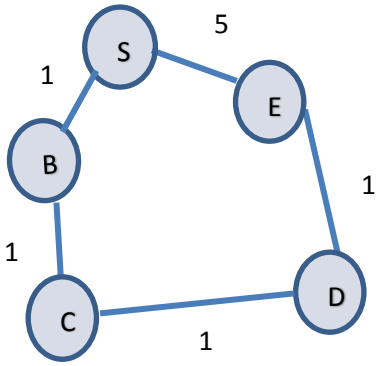
(a) Does the minimum spanning tree change? Give an example it changes or prove it cannot change.

Ans: For contradiction, suppose that the minimum spanning tree changes after increasing each edge weight by 1 and we have a new minimum spanning tree H. Let H have a total weight of m and let G (previous mst) have a weight of n. Based on the problem description, assume m ≠ n. By definition, a minimum spanning tree has the lowest possible total edge weight. Since H is our minimum spanning tree, it must have the lowest weight, so m < n. Now, let z be 1 * E (number of edges). Then we can see that, m – z < n – z (subtracting one from each edge)

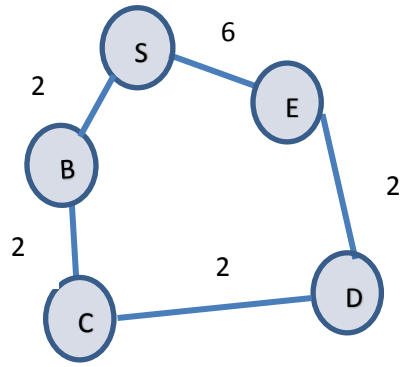
m – z is the weight of H before we added one to each edge, and n – z is the weight G before we added one to each edge. We already know that G was our minimum spanning tree before 1 was added to each edge. m – z < n – z is a contradiction then. Therefore, the minimum spanning tree does not change.

(b) Do the shortest paths change? Give an example where they change or prove they cannot change.

Ans: The shortest paths could change. Suppose graph G looks like this:



Clearly the shortest path from S to E travels along vertices S, B, C, D, E. However, if we increase the weight of each edge in the above graph by 1, it now looks like this:



Now, the shortest path from S to E travels along vertices S, E. Therefore, it is possible for the shortest paths to change when each edge increases by 1.

3. *(4 points)* In the bottleneck-path problem, you are given a graph G with edge weights, two vertices s and t and a particular weight W; your goal is to find a path from s to t in which every edge has at least weight W.

(a) Describe an efficient algorithm to solve this problem.

Ans: We can gaurrentee every edge is at least weight W by checking the weight of each path and not including paths that are less than W. From each vertex, we check to see if the adjacent vertices are t and if the path is at least W. If not, we add each adjacent vertex with an edge of at least W to a queue. Then we repeat the process from the top vertex in the queue until we reach s.

(b) What is the running time of youor algorithm.

Ans:

This is the same as a breadth first search, however we are not going down paths that are < W. Therefore, running time would be

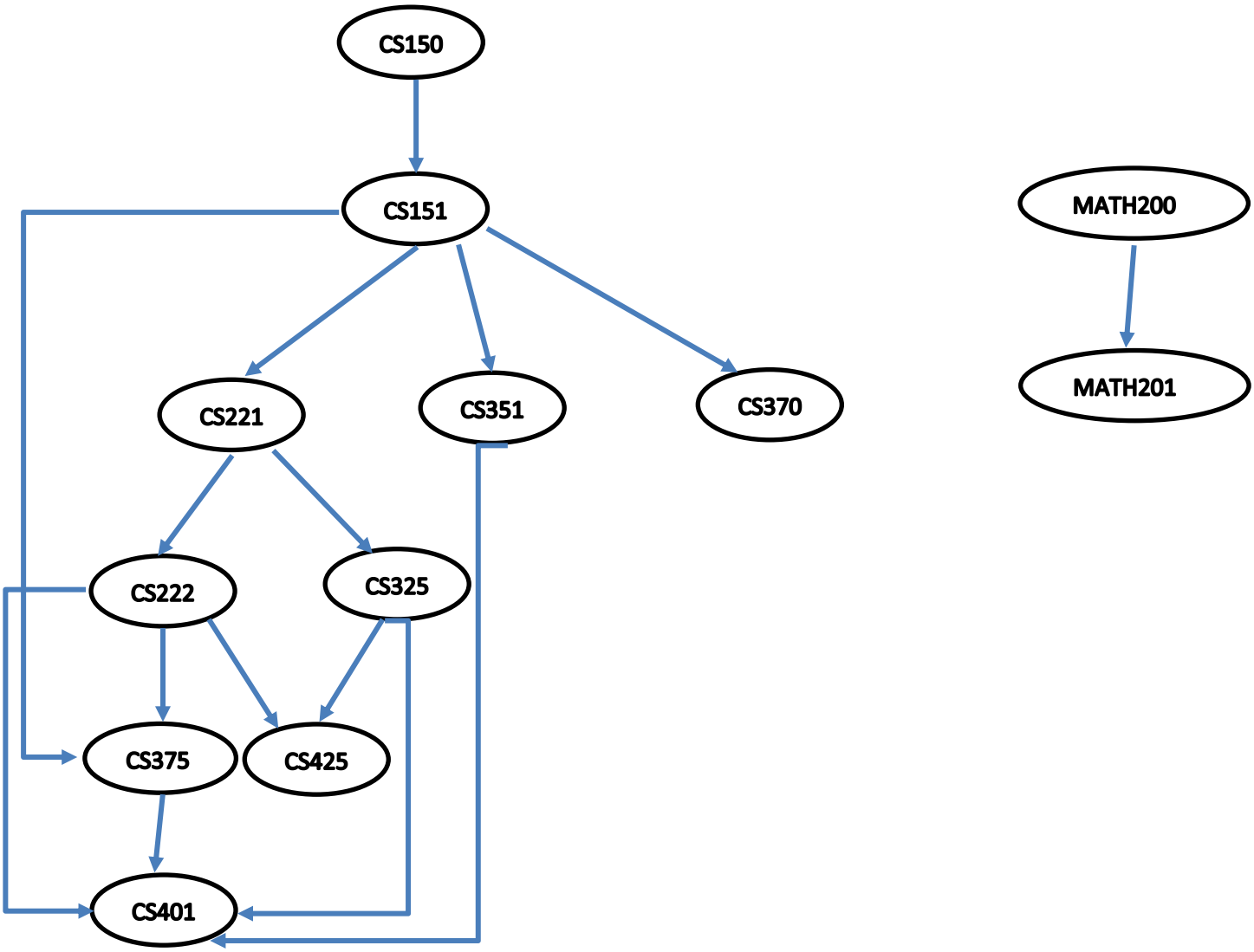
$\Theta(V + E)$

where V is the number of vertices and E is the number of edges.

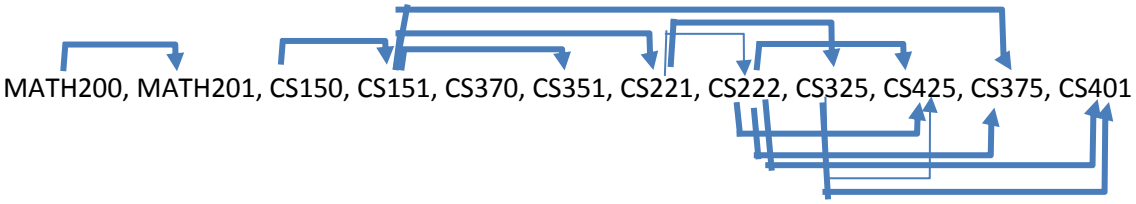
4. *(5 points)* Below is a list of courses and prerequisites for a factious CS degree.

Course	Prerequisite
CS 150	None
CS 151	CS 150
CS 221	CS 151
CS 222	CS 221
CS 325	CS 221
CS 351	CS 151
CS 370	CS 151
CS 375	CS 151, CS 222
CS 401	CS 375, CS 351, CS 325, CS 222
CS 425	CS 325, CS 222
MATH 200	None
MATH 201	MATH 200

(a) Draw a directed acyclic graph (DAG) that represents the precedence among the courses.


CS150
CS151
MATH200
MATH201
CS221
CS351
CS370
CS222
CS325
CS375
CS425
CS401

(b) Give a topological sort of the graph.



(c) If you are allowed to take multiple courses at one time as long as there is no prerequisite conflict, find an order in which all the classes can be taken in the fewest number of terms.

Ans:

Term 1: CS150, MATH200

Term 2: CS151, MATH201

Term 3: CS221, CS351, CS370

Term 4: CS222, CS325

Term 5: CS375, CS425

Term 6: CS401

(d) Determine the length of the longest path in the DAG. How did you find it? What does this represent?

Ans:
Longest length is 5
I chose the path with the highest number of consecutive edges, starting at CS150. The path is CS150 to CS401. This path represents the shortest possible number of terms (# of vertices in the path) that can be taken in order to complete the degree, due to the organization of the prerequisites for the courses and the inability to take a class before or while taking its prerequisite class.

5. *(12 points)* Suppose there are two types of professional wrestlers: “Babyfaces” (“good guys”) and “Heels” (“bad guys”). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n wrestlers and we have a list of r pairs of rivalries.

(a) Give pseudocode for an efficient algorithm that determines whether it is possible to designate some of the wrestlers as Babyfaces and the remainder as Heels such that each rivalry is between a Babyface and a Heel. If it is possible to perform such a designation, your algorithm should produce it.

Ans:

```
create wrestler_map (dictionary) where wrestler name dereferences wrestler object
        team variable in each wrestler object = -1 (no team assigned)
create rival_map (dictionary) where wrestler name dereferences list of rival name strings
for each e (string: name)  in rival map :   (controls for disjoint sets)
        if wrestler_map[e].team is -1:
                e is put into queue
                assign 0 (babyface) to wrestler_map[e].team
                while queue is not empty:
                        v = next item in queue
                        for j in rival_map[v]       (j represents a rival of v)
                                if wrestler_map[v].team == wrestler_map[j].team    (if rivals share rival)
                                        print “Team assignments are not possible. Two rivals share a rival”
                                else if wrestler_map[j].team is -1      (-1 means no team assigned)
                                        wrestler_map[j].team = (wrestler_map[v].team – 1) * -1
                                                ^babyfaces are team 0 and heels are team 1
                                                so above code ensures that rivals are on opposite teams
                                        add j to queue

                after team assignments are complete, for all i in wrestler_map
                        if wrestler_map[i].team == 0
                                add name to babyface team array
                        else
                                add name to heel team array
                        print “heels “, heel team array (all left over teamless wrestlers will also be heels)
                        print “babyfaces “, babyface team array
```

(b) What is the running time of your algorithm?

Ans:

The running time is the same as breadth first search. $\Theta(|V| + |E|)$.

This is because we are performing a breadth first search in order to assign wrestler teams and check if wrestlers who have already been assigned a team are rivals of another on the same

team. It takes E time to check the team assignment at each adjacent vertex, and we do this at every vertex. Therefore, It takes $\Theta(|V| + |E|)$ time to do the breadth first search.

(c) **Implemen**t: Babyfaces vs Heels.

Input: Input is read in from a file specified in the command line at run time. The file contains the number of wrestlers, n, followed by their names, the number of rivalries r and rivalries listed in pairs. *Note: The file only contains one list of rivalries*

Output: Results are outputted to the terminal.

- Yes, if possible followed by a list of the Babyface wrestlers and a list of the Heels .
- No, if impossible.

Sample Input file:

```
5
Ace
Duke
Jax
Biggs
Stone
6
Ace Duke
Ace Biggs
Jax Duke
Stone Biggs
Stone Duke
Biggs Jax
```

Sample Output:

```
Yes
Babyfaces: Ace Jax Stone
Heels: Biggs Duke
```

Submit a copy of your files including a README file that explains how to compile and run your code in a ZIP file to TEACH.