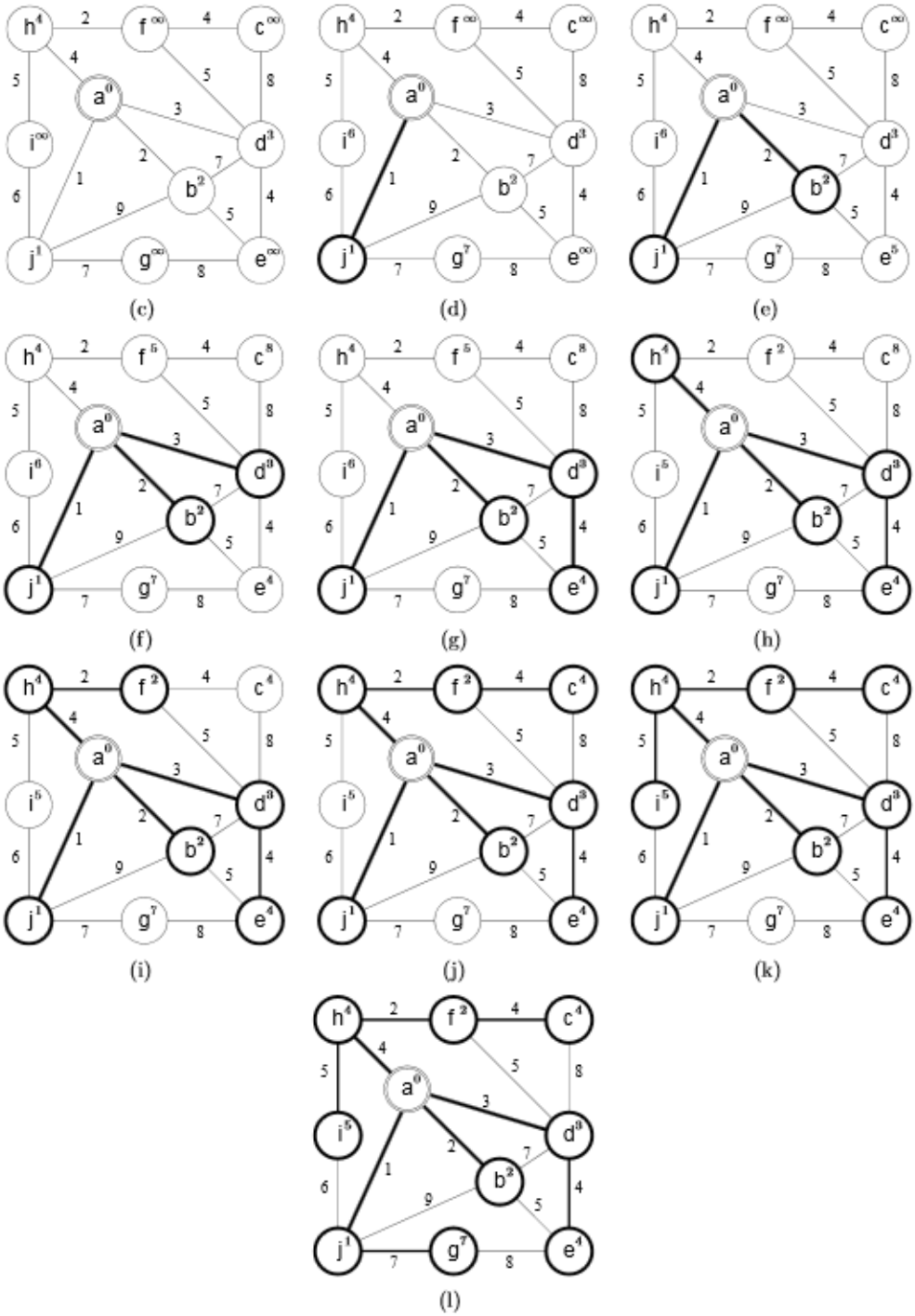
CS 325 – HW 5 - Solutions

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.



2. *(6 points)* Consider an undirected graph G=(V,E) with nonnegative edge weights $w(u,v) \geq 0$. Suppose that you have computed a minimum spanning tree G, and that you have also computed shortest paths to all vertices from vertex $s \in V$. 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.

NO it does not change

Proof: Let $T = \{e_1, e_2, ..e_{n-1}\}$ be the minimum spanning tree with total weight w(T) for the graph G(V,E) with weights w(e). Let the graph G'(V,E) be the graph G(V,E) but with weights w'=w(e)+1 for all e in E. I will show that a MST tree T for G is also a minimum spanning tree T' for G'.

Let $T' = T = \{e_1, e_2, ..e_{n-1}\}$ then the weight of T'

$w(T') = w'(e_1) + w'(e_2) + \ldots + w'(e_{n-1}) = (w(e_1) +1) + (w(e_2) +1) + \ldots + (w(e_{n-1})+1)$

$= w(e_1) +1+ w(e_2) + \ldots + w(e_{n-1}) +(n-1)$

$w(T') = w(T) + (n-1)$

Since there are (n-1) edges in the MST and 1 has been added to each edge the weight of the MST in T' must be (n-1) more than the weight of the MST in T for if $w(T') < w(T) +(n-1)$ this would imply that w(T) was not the MST of T. Therefore any MST in G is a MST in G'.

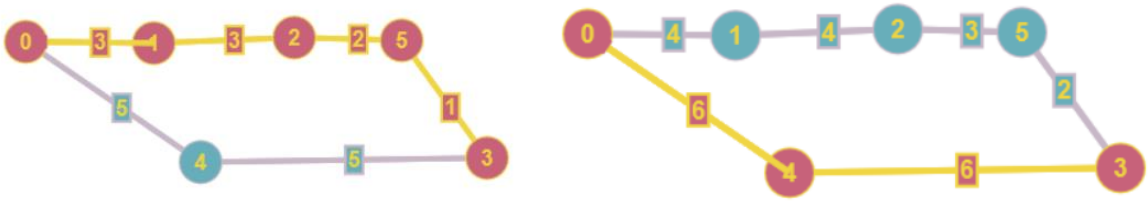
OR

Proof: That the MST does not change when each edge weight is increased by 1.

Suppose that the minimum spanning tree of the original graph was T. After each edge weight is increased by 1, the minimum spanning tree changes to T'. If T' and T are different there will be at least one edge $(u, v) \in T$ but $(u, v) \notin T'$. Suppose we add edge (u, v) to the tree T'. Let $T^* = T' + (u, v)$. Since (u, v) was not in T', (u, v) must be the longest edge in the cycle C formed in T*. But since (u, v) is the longest edge it cannot be in the MST T'. Since (u, v) is the longest edge when we decrease each edge weight by 1, (u, v) will still be the longest edge in cycle C formed in T. But the longest edge(u, v) cannot be contained in MST T. Therefore $(u, v) \notin T$ which is a contradiction. It implies that trees T and T' are the same.

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

YES. Give counterexample. See graph below SSP changes



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.

Perform a BFS of G starting at s and ignoring all edges with weight < W until edge t is reached. If t is not reached before the BFS terminates then there is no path from s to t with weight at least W.

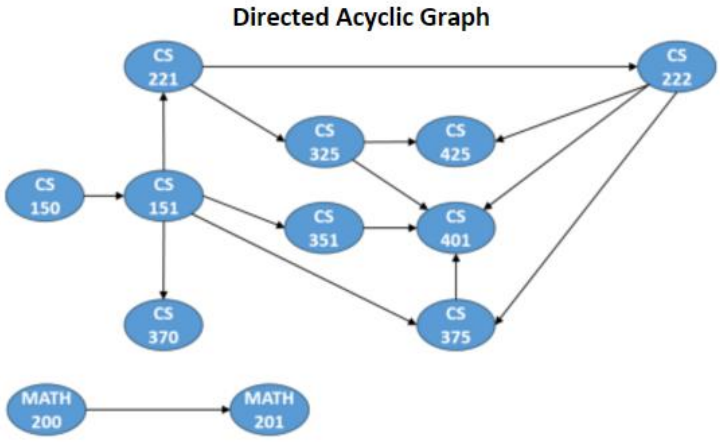
(b) What is the running time of youor algorithm.

O(E+V)

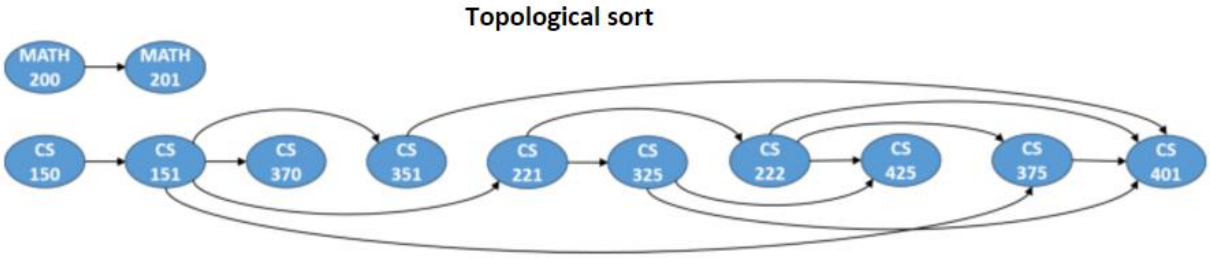
4. *(5 points)* Below is a list of courses and prerequisites for a factious CS degree.

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.



(b) Give a topological sort of the graph.



CS150 CS151 CS370 CS 351 CS221 CS325 CS222 CS425 CS375 CS 401 MATH 200 MATH 201 or

MATH 200 MATH 201 CS150 CS151 CS370 CS 351 CS221 CS325 CS222 CS425 CS375 CS 401

(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.

Again several correct variations.

1st Term: CS 150, MATH 200

2nd Term: CS 151, CS 201

3rd Term: CS 221, CS 351, CS 370

4th Term: CS 325, CS 222

5th Term: CS 425, CS 375

6th Term: CS 401

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

Algorithm to find the longest path in a DAG O(E+V)

1) Topologically sort the graph

2) Initialize the distances associated with vertices in the graph to 0. That is d(v)=0 for all v in G.

3) Start with the first vertex v in the topological sort.

- For each u in Adj[v] set d(u) = max { d(u), d(v)+1 }

4) Repeat step 4 with the next vertex in the topological sorted order until all vertices have been examined.

The length of the longest path is 5 which corresponds to the courses (1 point)

CS150, CS151, CS 221, CS222, CS375, CS401.

The length of the path+1 = 5+1 = 6 represented the minimum number of terms required to complete all courses with the given prerequisites. (Note: this is often called the Critical Path)

CS 325 – HW 5 - Solutions

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.

Several possibilities modifying BFS or DFS

Create a graph G where each vertex represents a wrestler and each edge represents a rivalry. The graph will contain n vertices and r edges.

Perform as many BFS's as needed to visit all vertices. Assign all wrestlers whose distance is even to be babyfaces and all wrestlers whose distance is odd to be heels. Then check each edge to verify that it goes between a babyface and a heel.

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

Depends on implementation. If using the algorithm described above

This solution would take O(n+r) time for the BFS, O(n) time to designate each wrestler as a babyface or heel, and O(r) time to check edges, which is O(n+r) time overall.

(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.