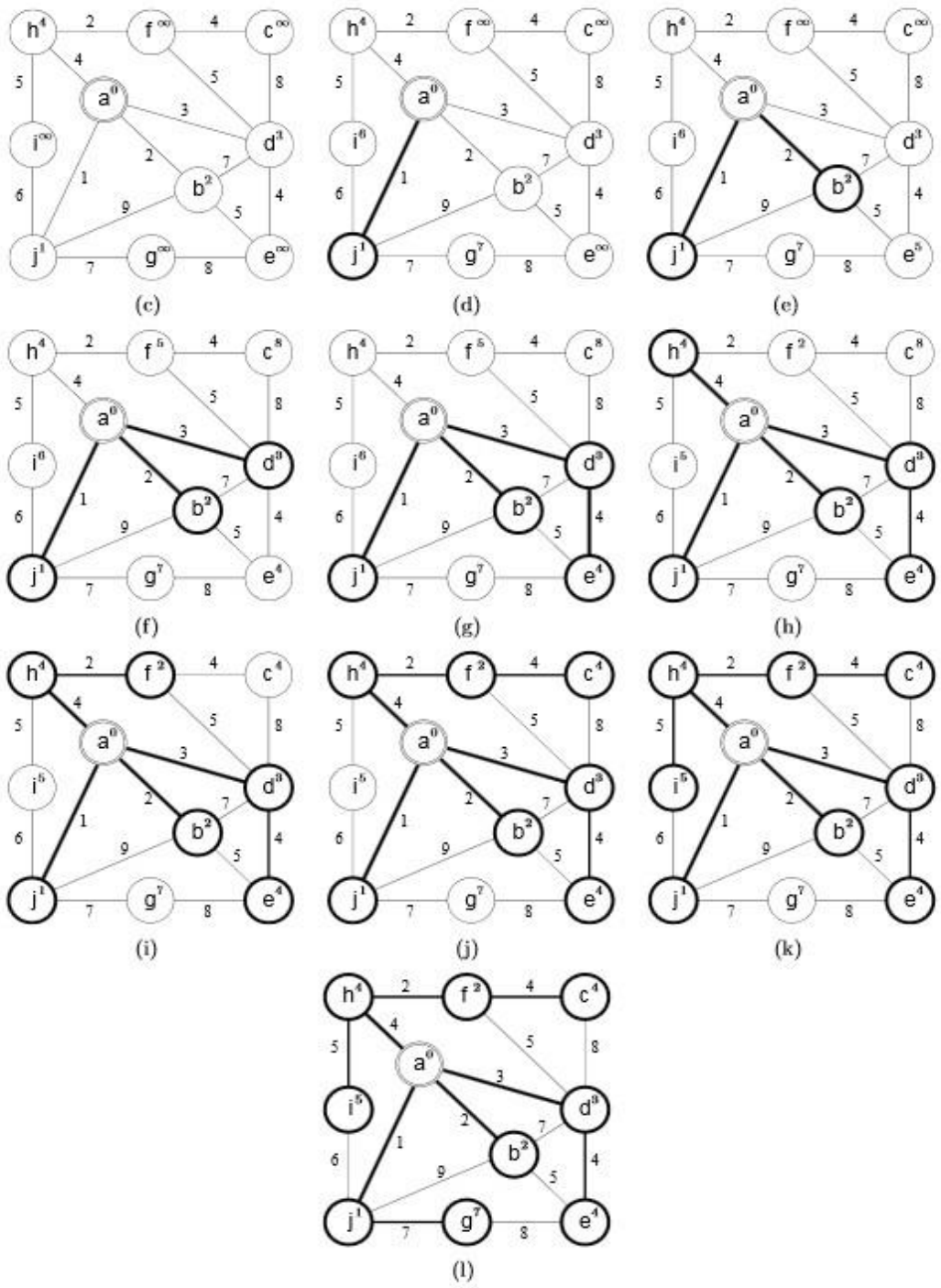
CS 325

Homework 5 Solutions

Problem 1 (3 points)



Problem 2 (6 points)

(a) 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.

Proof: 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'

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

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

$$= w(T) + (n-1)$$

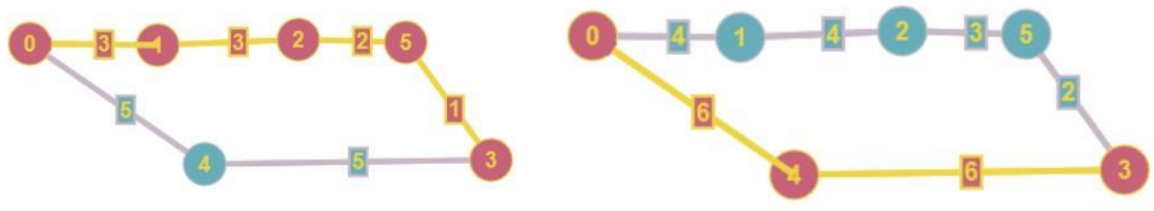
Since there are (n-1) edges in the MST and 1 has been added to each edge the weight of the MST in T' must be (n-1) more than the weight of the MST in T for if w(T') < w(T) +(n-1) this would imply that w(T) was not the MST of T. Therefore any MST in G is a MST in G'.

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) ∈ T but (u, v) ∉ 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) ∉ T which is a contradiction. It implies that trees T and T' are the same.

(b) YES. Give counterexample. See graph below SSP changes



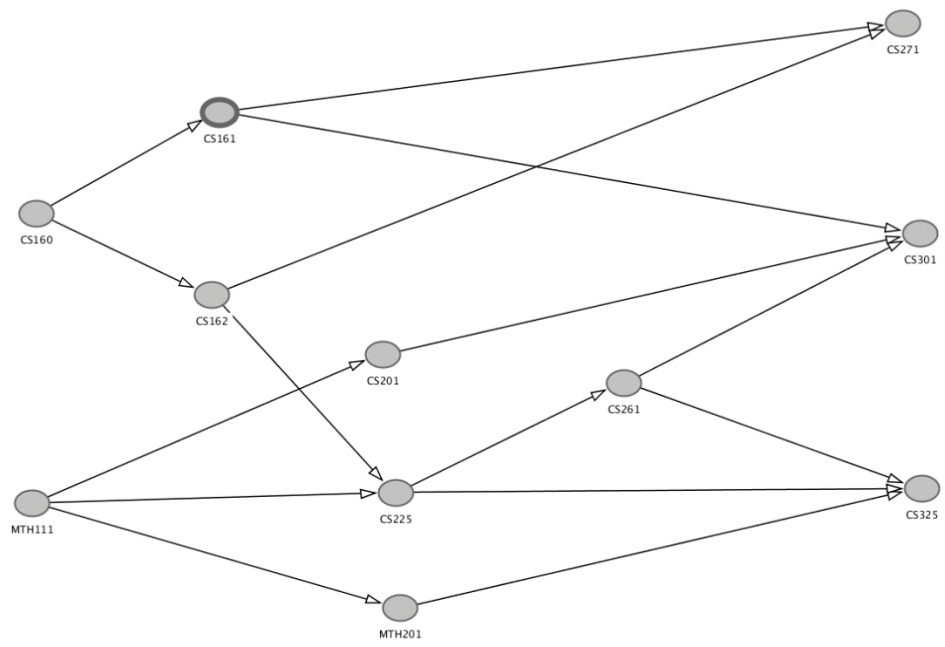
Problem 3 (4 points)

(a) 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) O(E+V)

Problem 4 (5 points)

(a)



(b) Several correct solutions, e.g.:

MTH111, MTH201, CS160, CS161, CS162, CS201, CS225, CS261, CS271, CS301, CS325

(c) Several correct solutions, e.g.:

1st term: MTH111, CS160

2nd term: MTH201, CS161, CS162, CS201

3rd term: CS225, CS271

4th term: CS261

5th term: CS301, CS325

(d) 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 4 (number of edges).

The length of the path plus one is 5 (number of vertices), and it represents the minimum number of terms required to complete all courses with the given prerequisites. (often called the Critical Path)

Problem 5 (12 points)

(a) 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 players whose distance is even to be babyfaces and all players whose distance is odd to be heels. Then check each edge to verify that it goes between a babyface and a heel.

(b) 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.