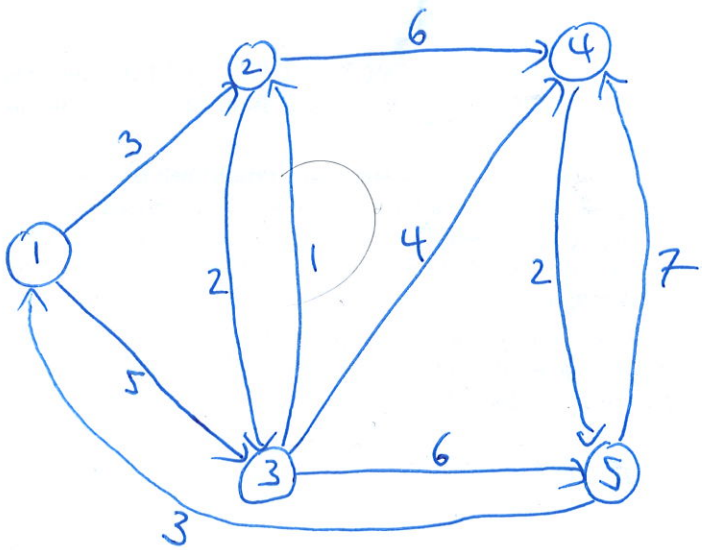
ELL781: Major Test

November 24, 2018

Maximum Marks: 24

1. Show that if $T(n) = T(2n/5) + T(3n/5) + cn$ (c is a constant), then $T(n)$ is $\Omega(n \log n)$, by:
   (a) Taking the given solution to be a guess and showing its correctness. **[2]**
   (b) Getting to a closed form via substitution (draw the recursion tree). **[2]**

2. Consider the below directed, weighted graph.

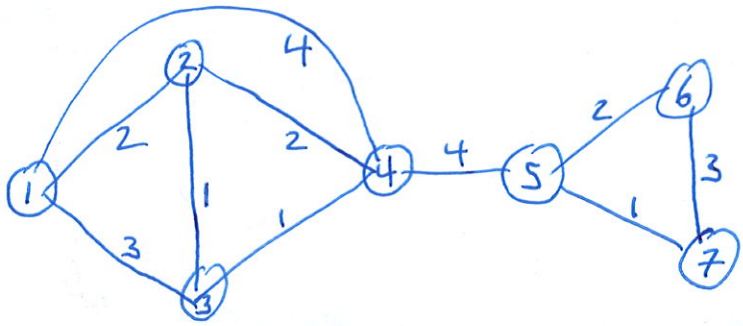


   (a) Show the execution of Dijkstra's algorithm on this graph, taking the node labelled 1 as the source node. As was done in class, show a full table with rows corresponding to iterations, and columns showing for each iteration:

   - the set S of vertices to which shortest paths have been found,
   - the specific vertex w which gets added to S in that iteration,
   - the entries of the array D, whch stores the cost of the current shortest path to each vertex, and
   - the entries of the array P, which stores the preceding vertex on the current shortest path to each vertex. **[5]**

   (b) As discussed in class, the actual sequence of vertices on the shortest path to the j^{th} vertex can be obtained by calling a function of the form $path(j, P)$, where P is the array of preceding vertices as above. For the particular P just obtained, show the sequence of recursive calls and the output returned by the call $path(5, P)$. **[2]**

3. Consider the below graph.



Obtain (showing the full sequence of edges added) spanning trees of this graph via the following algorithms (in all cases where there is a choice, assume that vertices/edges are examined in numerical order). Report the total cost of each spanning tree.

(a) An MST via Prim's algorithm. **[2]**

(b) An MST via Kruskal's algorithm. **[2]**

(c) DFS, starting at vertex 4. **[1.5]**

(d) BFS, starting at vertex 1. **[1.5]**

4. Recall the algorithm APPROX-VERTEX-COVER, which maintains a set of uncovered edges, and at each iteration picks one edge from this set at random and adds both its endpoints to the cover set.

(a) A *matching* is a set of edges in a graph such that no two edges share a common vertex. A *maximal matching* is a matching to which no further edges can be added (whilst maintaining the matching property). Prove that, for any input graph, the set of edges picked by APPROX-VERTEX-COVER (which we had denoted as A in class) forms a maximal matching of the given graph. **[3]**

(b) Give an example of a graph for which APPROX-VERTEX-COVER can never give the optimal solution, no matter what sequence of edges it may happen to pick. Explain/prove why this is so for your answer. **[3]**