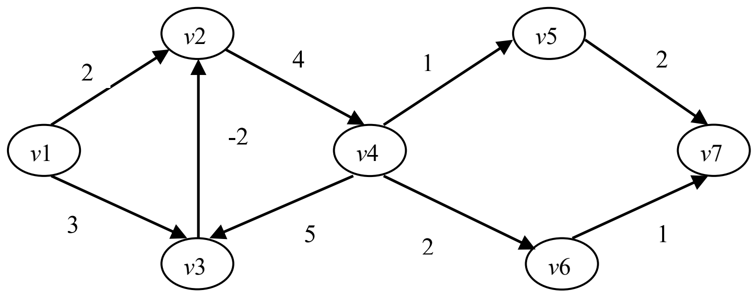
# Problem Set 9

**Q1** Show how to modify Dijkstra's algorithm to not only output the distance from v to each vertex in G, but also to output a tree T rooted at v such that the path in T from v to a vertex u is a shortest path in G from v to u.

**Q2** Simulate the execution of Dijkstra's algorithm to find the shortest path from *v*1 to *v*7 in the directed graph shown below.




Is there something going wrong? Explain.

**Q3** Bob loves foreign languages and wants to plan his course schedule for the following years. He is interested in the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:

- LA15: (none)
- LA16: LA15
- LA22: (none)
- LA31: LA15
- LA32: LA16, LA31
- LA126: LA22, LA32
- LA127: LA16
- LA141: LA22, LA16
- LA169: LA32

Find the sequence of courses that allows Bob to satisfy all the prerequisites. Describe your method briefly.

**Q4** Tamarindo University and many other schools worldwide are doing a joint project on multimedia. A computer network is built to connect these schools using communication links that form a tree. The schools decide to install a file server at one of them to share data. Since the transmission time on a link is dominated by the link setup and synchronization, the cost of a data transfer is proportional to the number of links used. Hence, it is desirable to choose a "central" location for the file server. Given a tree *T* and a node *v* of *T*, the *eccentricity* of *v* is the length of a longest path from *v* to any other node of *T*. A node of *T* with minimum eccentricity is called a *centre* of *T*.

1. Design an efficient algorithm that given an $n$-node tree *T*, computes a centre of *T*.
2. Is the centre unique? If not, how many distinct centres can a tree have?

*Hint:* Consider a tree *T* and the tree *T0* produced by pruning the leaves of *T*. What is the relation between the centres of *T* and *T0*?

**Q5** Consider the following greedy strategy for finding a shortest path from vertex *start* to vertex *goal* in a given connected graph.

1. Initialize *path* to *start*.
2. Initialize *VisitedVertices* to *{start}*.
3. If *start=goal*, return *path* and exit. Otherwise, continue.
4. Find the edge (*start, v*) of minimum weight such that *v* is adjacent to *start* and *v* is not in *VisitedVertices*.
5. Add *v* to *path*.
6. Add *v* to *VisitedVertices*.
7. Set *start* equal to *v* and go to step 3.

Does this greedy strategy always find a shortest path from *start* to *goal*? Either explain intuitively why it works, or give a counter-example.

**Q6** The time delay of a long distance call can be determined by multiplying a small fixed constant by the number of communication links on the telephone network between the caller and the callee. Suppose the telephone network of a company named Delstra is a tree. The engineers of Delstra want to compute the maximum possible time delay that may be experienced in a long-distance call. Given a tree, the diameter of T is the length (the number of edges) of a longest path between two nodes of T. Give an efficient algorithm for computing the diameter of T.

**Q7** Design an efficient algorithm for finding a longest path from a vertex u and a vertex v of an acyclic directed graph G where each edge has a weight. Specify the graph representation used and any auxiliary data structures used. Also analyse the time complexity of your algorithm.