1. (3 points) A digraph G is called a dominance-directed graph if for any pair of distinct vertices u and v of G, either u→v or v→u, but not both (here the notation u→v means there is an edge from u to v). To the right is an example of a dominance-directed graph.

In a dominance-directed graph, we define the **power** of a vertex, as being the total number of 1-step and 2-step connections to other vertices. Using the adjacency matrix and its square, calculate the power of each vertex and rank each team according to their vertex power:

![Graph H](image)

2. (2 points) Prove that in every simple graph there is a path from every vertex of odd degree to a vertex of odd degree.

3. (3 points) Argue that in a breadth-first search, the value \(d[u]\) assigned to a vertex \(u\) is independent of the order in which the vertices in each adjacency list are given. Using Figure 22.3 as an example, show that the breadth-first tree computed by BFS can depend on the ordering within adjacency lists.

4. (3 points) Suppose we are given directed acyclic graph \(G = (V, E)\) where \(V = \{v_1, v_2, \ldots, v_n\}\). Prove that if \(v_1, v_2, \ldots, v_n\) is a topological ordering of the vertices of \(V\), then the adjacency matrix corresponding to \(G\) is an upper triangular matrix (that is, all entries below the main diagonal are zero).

5. (3 points) NASA wants to link \(n\) stations spread over the country using communication channels. Each pair of stations has a different bandwidth available, which is known a priori. NASA wants to select \(n-1\) channels (the minimum possible) in such a way that all the stations are linked by the channels and the total bandwidth (defined as the sum of the individual bandwidths of the channels) is maximum. Give an efficient algorithm for this problem and determine its worst-case time complexity.

6. (3 points) 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.
7. (3 points) Give a simple example of a directed graph with negative weight edges for which Dijkstra’s algorithm produces incorrect answers. Why doesn’t the proof of Theorem 24.6 go through when negative-weight edges are allowed? NO NEGATIVE WEIGHT CYCLES!!

8. (5 points) Run the Floyd-Warshall algorithm on the weighted, directed graph of Figure 25.2. Show the matrix D (k) that results for each iteration of the outer loop.