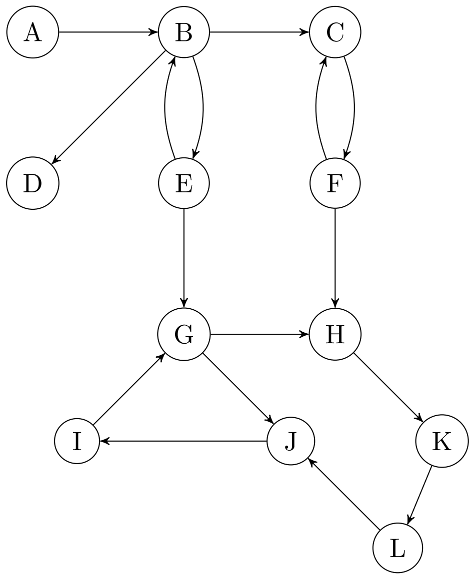
Recitation Problems - Com S 311

Week of Apr 8^{th} - Apr 13^{th}

1. A strongly connected component in a graph $G = (V, E)$ is a maximal subset $C \subseteq V$ such that for every pair of vertices $u, v \in C$, there is a directed path from u to v and a directed path from v to u.

   Consider the following graph:

   (a) Identify all the strongly connected components of the above graph.
   - {A}
   - {D}
   - {B, E}
   - {C, F}
   - {G, H, I, J, K, L}

   (b) Identify the vertices in the strongly connected component with the largest number of vertices.
   G, H, I, J, K, L

   (c) is there a vertex that can reach all other vertices in the above graph?
   Yes, vertex A.

2. Given a graph $G = (V, E)$, write an algorithm that checks whether there exists a vertex that can reach all other vertices.

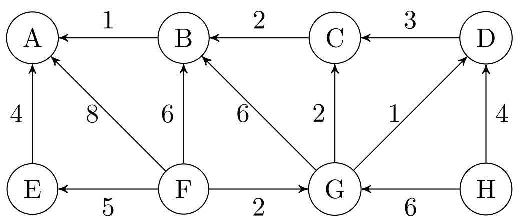
   Observation: If there is a vertex v that can reach all other vertices, then using DFS exploration of graph the vertex v must have the largest finish/end time.

   Proof sketch. Assume there is a vertex v which can reach all other vertices and there is a vertex u whose end-time after DFS exploration of the graph is such that $endTime(u) > endTime(v)$. This implies the exploration from v terminates before the termination of exploration from u. This is a contradiction as based on our assumption we know u can be reached from v.

Strategy: Do the DFS exploration of the graph and identify the vertex with largest finish/end time. This is a candidate vertex which **may be able to reach** all other vertices. So, we (reinitialize the vertex properties and) do DFS exploration starting from this vertex–if all vertices are explored then our algorithm returns true; otherwise our algorithm returns false.

Runtime: $O(V + E)$

3. Recall the Dijkstra's algorithm as presented in class. Consider the following weighted graph and write the valuation of d-value for each vertex at the end of each iteration of while-loop (that iterates until the priority queue is empty) in the algorithm. The source vertex is H.



(*Add more rows to the following table, if needed, for your solution.*)

Initial d-value	∞	∞	∞	∞	∞	∞	∞	0
Iteration	A	B	C	D	E	F	G	H
1	∞	∞	∞	4	∞	∞	6	0
2	∞	∞	7	4	∞	∞	6	0
3	∞	12	7	4	∞	∞	6	0
4	∞	9	7	4	∞	∞	6	0
5	10	9	7	4	∞	∞	6	0
6	10	9	7	4	∞	∞	6	0
7	10	9	7	4	∞	∞	6	0
8	10	9	7	4	∞	∞	6	0

The order in which the vertices are "added" to the priority queue are H, D, G, C, B, A. At the end of Dijkstra's algorithm, any vertex with an ∞ d-value is unreachable from the source H, in this case are vertices E and F.