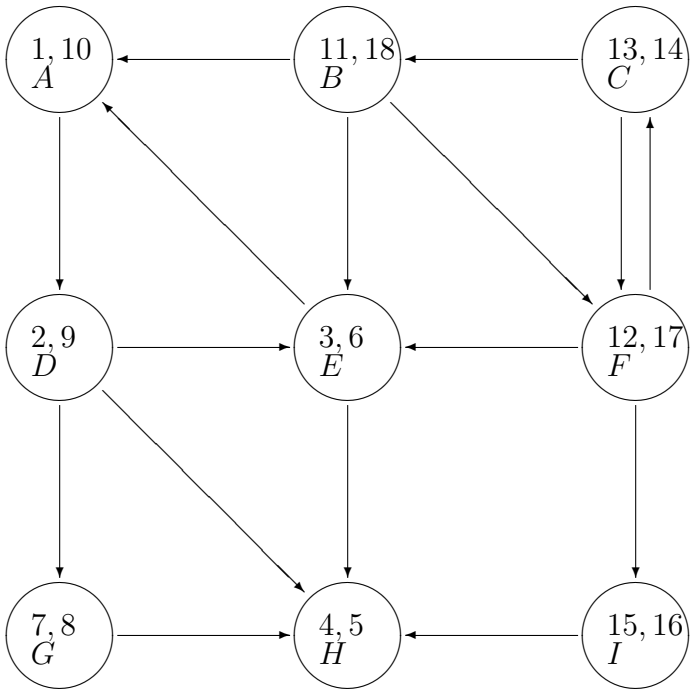
Quiz 2
CS 3510 (and 3511), Algorithms
Answer Key: March 12, 2009

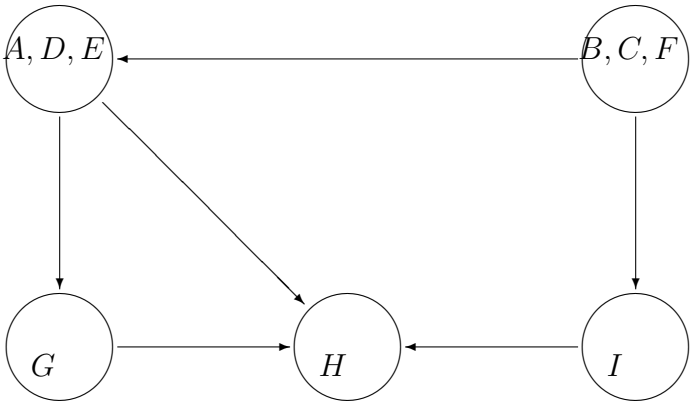
1. Do a Depth-First-Search of the graph below. Follow the convention of processing nodes in lexicographic (alphabetical) order. Show the pre- and post- numbers. (15 points)

   How many strongly connected components are there (no need to show how to find them)? What is the graph after you shrink them? (10 points)

   Given graph after a run of DFS algorithm will have pre and post times as shown below.

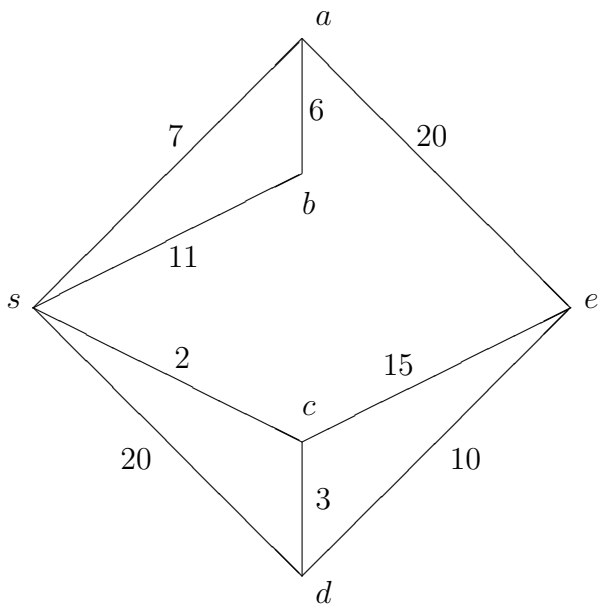


There are 5 strongly connected components of this graph. Below we draw a graph derived by shrinking these SCCs.



We can clearly see how we get these components from the previous figure, we group those nodes in a strongly connecteed component which have a path between every pair chosen from that group.

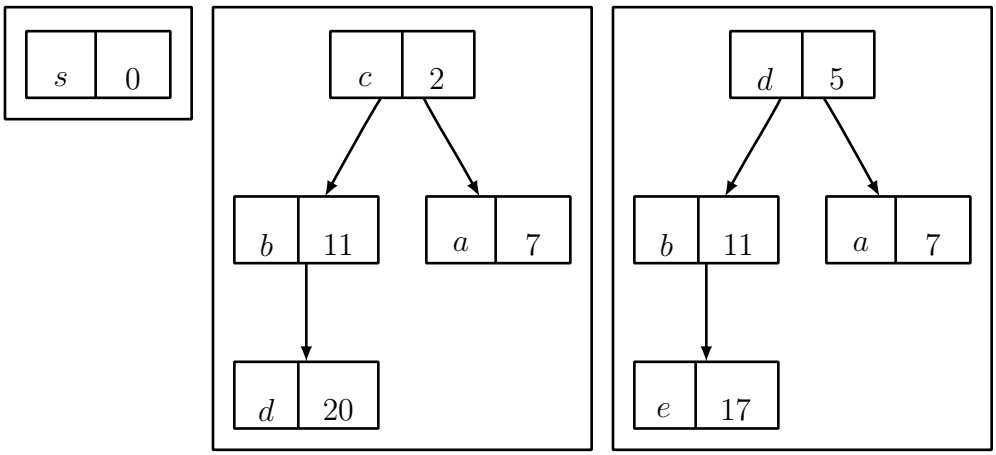
2. (a) If you run Dijkstra's algorithm on the following graph, what is the shortest path from s to each of the other vertices? (10 points)



Below we show states of the priority queue and at every step of algorithm and present corresponding estimated minimum distance of every vertex. Also, at every step we show a vertex with minimum distance from the source s which is taken out of the queue at the begining of the next step according to Dijkstra's algorithm.

Vertex	shortest path	Priority queue	Estimated shortest distances
-	-	$[s]$	[0]
s	0	$[cabd]$	[271120]
c	2	$[dabe]$	[571117]
d	5	$[abe]$	[71115]
a	7	$[be]$	[1115]
b	11	$[e]$	[15]
e	15	$[\phi]$	$[\phi]$

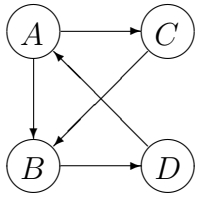
(b) If you use a binary heap for the priority queue, show what the heap will look like up to (and including) the time that vertex e is first added to the heap. As always, use alphabetic ordering to break ties. (15 points)



3. Is each of the following statements true or false? Explain briefly (prove or give a counterexample). (5 points each)

   (a) If the depth-first-search of a directed graph has a cross edge then the graph is not strongly connected.

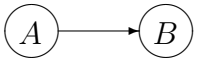
   False.
   counterexample:



   This graph is strongly connected and depth first search starting from A will lead to a cross edge.

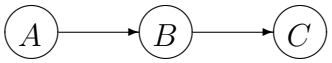
   (b) If a graph is not strongly connected, then its depth first search must have a cross edge.

   False.
   counterexample:



This graph is not strongly connected and also there is no cross edge.

(c) In a DAG (directed acyclic graph), the vertex with the highest pre-number must be a sink. False.
counterexample:



If in the above graph we start DFS at node C, which is the only sink, it will obviously not have the highest pre number.

(d) There is an efficient algorithm for finding the longest path in a graph with only positive weights. The answer to this question can be both true and false, but in each case you need to present the correct reason.
True.
For true your argument should be, make the edge capacities negative and find the shortest path using Dijkstra's. Note that here you disregard the possibility of positive cycles in the original graph.
False.
Here you should present the positive cycle argument by which you can say that determining the longest path now has no meaning, and there is no efficient algorithm to do so.

(e) There is an efficient algorithm for finding the longest path in a graph with only negative weights.
True.
Negate all the weights(so now all weights are positive) and now find the shortest path. This is a $O((|V| + |E|) \times \log |V|)$

4. The police department in the mythical city of Computopia has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. Furthermore, the city elections are coming up soon, and there is just enough time to run a linear time algorithm.

   a) Formulate this problem as a graph-theoretic problem and explain why this problem can indeed be solved by a linear-time algorithm. (10 points)

   Model the intersectios as nodes and one way streets as directed edges in the graph. Now the above problem can be modeled as the problem of determining the number of strongly connected components in the graph. If mayor's claim is correct there should be only one strongly connected component in the graph.
   Thus run DFS on reversed graph G^R to determine the post order and then run it again this time on the original graph G in by selecting nodes in decreasing order of post numbers. If we get only one connected component the mayor is right! This is a $O(|V|+|E|)$ algorithm since it just makes two runs of DFS which is itself an $O(|V|+|E|)$ algorithm.

b) Suppose now that the mayor needs to show that, even though her original claim was false, something weaker does hold: If you start navigating one-way streets from townhall, there is always a way to drive legally back to the townhall. Formulate this weaker property as a graph-theoretic problem and show that this can also be solved in linear time. (15 points) Now she should just run the previous algorithm and check the strongly connected component to which the townhall belongs. If this SCC has an outgoing edge to any of the other SCCs (we can say this because mayor was wrong and there are indeed multiple SCCs) then by taking a route that leads to this edge you can never come back to the townhall again. This can be checked in $O(|V| + |E|)$ again by running DFS starting at townhall, if in this search you encounter any node outside the SCC to which townhall belongs, then mayor's claim is not true.

Scrap