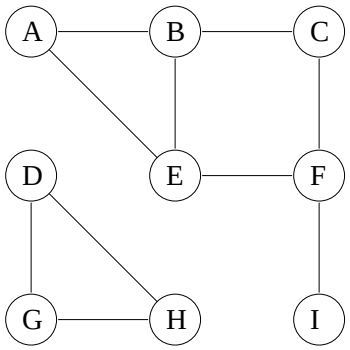
CS170 Discussion Section 4

September 19, 2012

Questions

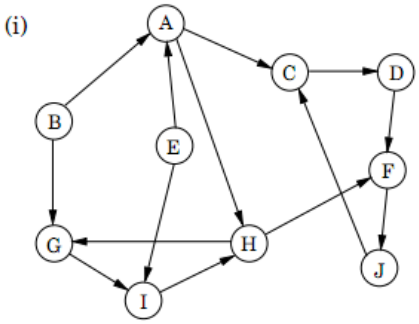
1. *Fun with DFS (Problem 3.1).*

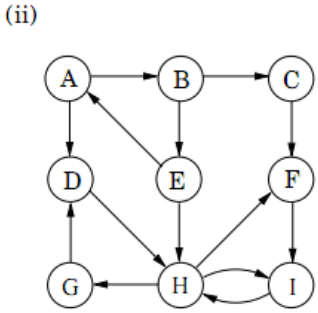
   Perform a depth-first search on the following graph; whenever there's a choice of vertices, pick the one that is alphabetically first. Classify each edge as a tree edge or back edge, and give the `pre` and `post` number of each vertex.



2. *More fun with DFS (Problem 3.4).*

   Run the strongly connected components algorithm on the following directed graphs G. When doing DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.





   In each case answer the following questions.

   (a) In what order are the strongly connected components (SCCs) found?

   (b) Which are source SCCs and which are sink SCCs?

   (c) Draw the 'metagraph' (each meta-note is an SCC of G).

   (d) What is the minimum number of edges you must add to this graph to make it strongly connected?

3. *Degrees (Problem 3.6).* In an undirected graph, the *degree* $d(u)$ of a vertex u is the number of neighbors u has, or equivalently, the number of edges incident upon it. In a directed graph, we distinguish between the *indegree* $d_{in}(u)$, which is the number of edges into u, and the *outdegree* $d_{out}(u)$, the number of edges leaving u.

   (a) Show that in an undirected graph, $\sum_{u \in V} d(u) = 2|E|$.

   (b) Use part (a) to show that in an undirected graph, there must be an even number of vertices whose degree is odd.

   (c) Does a similar statement hold for the number of vertices with odd indegree in a directed graph?

4. *Fun with algorithms (Problem 3.9).* For each node u in an undirected graph, let `twodegree`$[u]$ be the sum of the degrees of u's neighbors. Show how to compute the entire array of `twodegree`[.] values in linear time, given a graph in adjacency list format.

5. *How to graduate early (Problem 3.16).* Suppose a CS curriculum consists of n courses, all of them mandatory. The prerequisite graph G has a node for each course, and an edge from course v to course w if and only if v is a prerequisite for w. Find an algorithm that works directly with this graph representation, and computes the minimum number of semesters necessary to complete the curriculum (assume that a student can take any number of courses in one semester). The running time of your algorithm should be linear.

6. *Reachability (Problem 3.22).* Give an efficient algorithm which takes as input a directed graph $G = (V, E)$, and determines whether or not there is a vertex $s \in V$ from which all other vertices are reachable.