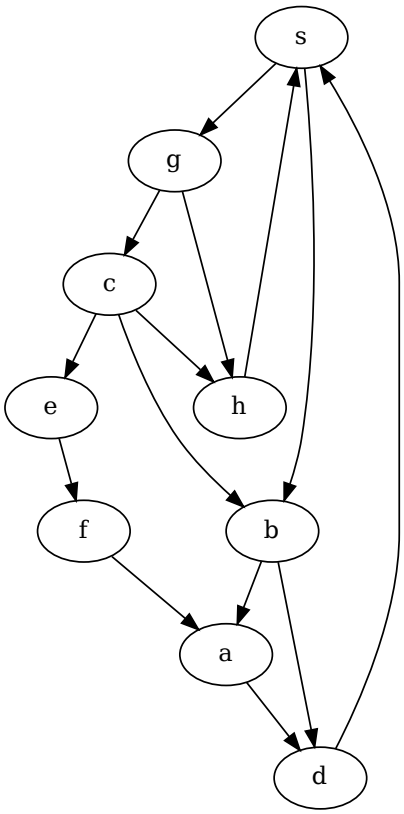
TD 3: DFS

1 Execute DFS

Execute the DFS algorithm on the directed graph below, starting from vertex s. You can assume that adjacency lists are ordered alphabetically. Show the discovery and finish times of all vertices, the tree calculated, and classify the arcs as tree, forward, backward, and cross.



2 Connected Components

Modify the DFS algorithm to identify the connected components of an undirected graph G. More precisely, your algorithm should take as input an undirected graph G and output an integer C equal to the number of connected components of G. Furthermore, it should compute for each $v \in V$ an attribute $v.cc$ such that for all $u, v \in V$ we have $u.cc = v.cc$ if and only if u, v are in the same connected component.

3 DAG Detection

A directed graph is called a DAG (Directed Acyclic Graph) if it contains no directed cycle as a subgraph. Show that we can decide if a given directed graph G is a DAG in linear time as follows: execute DFS on G and check whether a backward arc exists. Prove that G is a DAG if and only if no backward arc exists.

4 Unique Trees

Show that if $G = (V, E)$ is an undirected graph and $s \in V$, we have the following: if the trees produced by executing BFS and DFS on G, s are identical, then G contains no other edges except those of the tree output by the two algorithms.

5 Digraph Random Walks

The objective of this exercise is to understand why the random walk reachability algorithm we saw in class only works for **undirected** graphs. Give an example of a digraph G with n vertices and two vertices s, t so that the probability that a random walk which starts at s manages to reach t is as small as possible (asymptotically, as a function of n). Can you make your example work even on digraphs where all vertices have large outdegree?

Explain why your example has (essentially) the lowest probability possible. Use your argumentation to derive a randomized algorithm (perhaps with exponential expected running time) that decides if a given digraph G has a directed path from s to t.

6 Undirected random walk cover times

In class we claimed that in any n-vertex undirected graph, a random walk starting from any vertex will reach all other vertices in $O(n^3)$ steps in expectation. However, for some graphs it may actually be much faster to reach all vertices. Consider the following:

1. Show that if G is a clique K_n and s, t any two distinct vertices, if we start a random walk at s, the expected number of steps before we reach t is $\Theta(n)$.

2. Show that if G is a path P_n and s, t are its endpoints, if we start a random walk at s, the expected number of steps before we reach t is $\Theta(n^2)$.

3. Show that, despite the above, it is not correct to conclude that graphs with more edges make random walks faster. Construct a graph with n vertices, $\Omega(n^2)$ edges, such that the expected number of steps for a random walk starting at s to reach t is $\Omega(n^3)$.