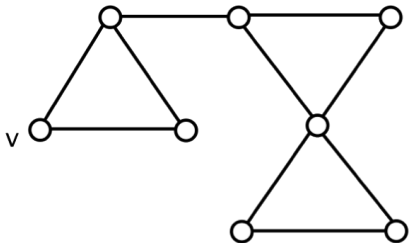
Algorithmics - Tutorial Sheet 5

Graphs and graph algorithms

---

1. **[Work in pairs, switch roles for the two tasks]** Construct (*a*) a depth-first spanning tree and (*b*) a breadth-first spanning tree of the graph shown below.



   In each case, start the search/travel at vertex v.

2. Describe an algorithm based on depth-first (or breadth-first) search to determine whether a given graph is bipartite.

   Recall: a graph $G = (V, E)$ is bipartite if its vertex set V can be partitioned into two disjoint subsets $V = U \cup W$ such that each edge of the graph connects a vertex in U to a vertex in W.

3. Recall, a *Eulerian cycle* in an undirected graph is a cycle that includes every edge of the graph exactly once. It can be shown that a graph G contains an Eulerian cycle if and only if G is connected and every vertex has even degree (i.e. is adjacent to an even number of vertices). Such a graph is called an *Eulerian graph*.

   Describe how depth-first search can be adapted to find an Eulerian cycle in an Eulerian graph.

   (A *Hamiltonian cycle* in an undirected graph is a cycle that includes every vertex of the graph exactly once. A graph containing a Hamiltonian cycle is called a Hamiltonian graph. Despite the superficial similarity of this concept to that of an Eulerian graph, there is no known efficient algorithm to determine whether a given graph is Hamiltonian.)

4. Describe in detail how depth-first search can be used to check for deadlock, i.e. to determine whether a given directed graph, represented by adjacency lists, contains a cycle. What is the complexity of your algorithm?