Problem Set No. 5

1. Construct an example of a directed graph with a distinguished vertex \( s \) whose minimum spanning tree rooted at \( s \) is different from its shortest path tree rooted at \( s \).

2. For a connected undirected graph, a bottleneck minimum spanning tree is a spanning tree whose maximum edge cost is minimum.
   
   (a) Show that any minimum spanning tree is a bottleneck minimum spanning tree, but a bottleneck spanning tree need not be a minimum spanning tree.
   
   (b) Describe and analyze an \( O(m) \)-time algorithm to find a bottleneck minimum spanning tree. Hint: Use median-finding and graph contraction.

3. (CLR 23.1-6, p. 468) When an adjacency-matrix representation is used, most graph algorithms require time \( \Theta(V^2) \), but there are some exceptions. Show that determining whether a directed graph contains a sink—a vertex with in-degree \( |V| - 1 \) and out-degree 0—can be determined in time \( O(V) \), even if an adjacency-matrix representation is used.

4. (CLR 23.5-7, p. 494) A directed graph \( G = (V,E) \) is said to be semiconnected if, for any two vertices \( u,v \in V \), we have \( u \sim v \) or \( v \sim u \). Give an efficient algorithm to determine whether or not \( G \) is semiconnected. Prove that your algorithm is correct, and analyze its running time.

5. (Heuristic Search) Let \( G \) be a graph with two distinguished vertices \( s \) and \( t \) and an edge cost \( c(v,w) \) for each edge \((v,w)\). Assume that \( G \) has no negative cycles (though it may have negative edge costs). We wish to find a shortest path from \( s \) to \( t \) by heuristic search, using a distance estimate \( e(v) \) which is intended to be an easy-to-compute approximation to the actual distance from \( v \) to the destination \( t \). We use the labeling and scanning algorithm as described in class. To choose the next vertex to scan, we pick a vertex \( v \in L \) with minimum \( d(v) + e(v) \). Specifically, the algorithm is as follows:

   Initialize \( L = \{s\}, d(s) = 0, d(v) = \infty \) for \( v \neq s \).

   while \( L \neq \emptyset \) do begin
      delete from \( L \) a vertex \( v \) with \( d(v) + e(v) \) minimum;
      if \( v = t \) then stop else
for \((v, w)\) an edge do begin if \(d(v) + c(v, w) < d(w)\) then begin
\(d(w) = d(v) + c(v, w); p(w) = v;\)
if \(w \notin L\) then insert \(w\) into \(L\)
end end end

We call an estimate \(e\) **safe** if \(e(t) = 0\) and \(e(v) \leq c(v, w) + e(w)\) for every edge \((v, w)\).

(a) Prove that if \(e\) is a safe estimate, then \(e(v)\) is a lower bound on the distance from \(v\) to \(t\), for every vertex \(v\).

(b) Prove that if \(e\) is a safe estimate, the heuristic search algorithm will delete each vertex from \(L\) at most once, and will terminate with \(d(t)\) equal to the correct distance from \(s\) to \(t\), with the parent pointers from \(t\) indicating a shortest path from \(s\) to \(t\) (backwards).

(c) Prove that if \(e\) and \(f\) are two safe estimates such that \(e(v) \leq f(v)\) for every \(v\), then heuristic search run with \(f\) will delete no more vertices from \(L\) than heuristic search run with \(e\).

(d) Describe how to implement heuristic search so that the total running time is \(O(k \log k + l)\), where \(k\) is the number of vertices inserted into \(L\) and \(l\) is the total number of edges leading out of such vertices. (Assume \(e(v)\) is computable in \(O(1)\) time for any \(v\).) Hint: Use an \(F\)-heap. You will also need to **avoid** explicitly initializing \(d(v) = \infty\) for all vertices \(v \neq s\), since the number of such vertices may be much larger than \(k\). How can you do this?

6. (extra credit) Give a family of graphs (with some negative-cost edges) on which Dijkstra’s shortest path algorithm (shortest-first scanning) runs in exponential time.