HW#4 Solution
Dr. Song, CSCI 323, Spring 2003
Department of Computer Science, Queens College

R-6.2

In a simple undirected graph, the number of edges m is bounded by

$$m \leq \frac{n(n-1)}{2} \leq n^2$$

Taking logarithm on both sides of the above equation, we have

$$\log m \leq 2 \log n \qquad \text{for } n \geq 1$$

That is $\log m$ is $O(\log n)$.

R-6.4

Build a directed graph out of the nine language courses in the following way:

1. Each course is a vertex;
2. If course LAX is a prerequisite of course LAY, then there is a directed edge from LAX to LAY;

Then we can perform topological sort on the digraph, which gives several topological orderings:

- LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, LA169
- LA22, LA15, LA16, LA127, LA22, LA31, LA32, LA126, LA141, LA169

each of which corresponds to a sequence of course that allows Bob to satisfy all prerequisites.

R-6.10

A topological ordering from the graph in G&T Figure 6.14(d) is:

$$ORD, BOS, JFK, MIA, DFW, SFO, LAX$$

C-6.2

Let $n_1, n_2, \cdots, n_i$ be the number of vertices in each connected component. Let $m_1, m_2, \cdots, m_i$ be the number of edges in each connected component. The total running time is $\Theta(n_1 + m_1 + n_2 + m_2 + \cdots n_i + m_i)$ which is $\Theta(n+m)$ where n is the total number of vertices and m is the total number of edges.

C-6.3

Proof. (by contradiction) Let T contain all discovery edges. Assume there is a cross edge from vertex v to u. Vertices u and x must have been discovered earlier than w by the depth-first-search strategy. If there is an edge between v and u, v should have been discovered earlier by u as a discovery edge of v. Then T would have included edge (v,u). However, this is a conflict with the assumption that edge (v,u) is not in T. So there is no cross edges outside T. That is all edges not in T are back edges. □

Algorithm 1 ConnectedComponent(G)

Input: an undirected graph G
Output: connected components of G
$i \leftarrow 1$
for each vertex v in G **do**
  if v is not visited **then**
    BFS(G, v) (or DFS(G, v)) and label each visited vertex with i
    $i \leftarrow i+1$
  end if
end for
$i \leftarrow i-1$
for each vertex v in G **do**
  $j \leftarrow$ the label of v
  put v into the j-th connected component C_j of G
end for
return $C_1, C_2, \cdots, C_i$

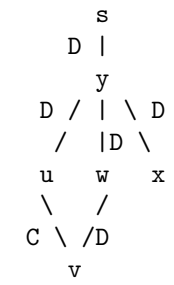

Figure 1: An assumed cross edge after DFS for C-6.3.

R-7.1

Ignore.

C-7.2

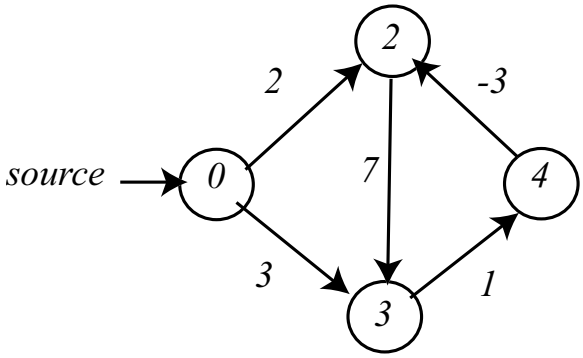

Figure 2: The value (2) of the top node produced by the Dijikstra's algorithm is not the distance from the source to the node, which is supposed to be 1.