1. (25 points) Suppose that given an integer constant \( n \) and an array \( T[n] \) of integers, you know that \( T \) is partially sorted in the sense that there exists an integer \( m, 0 \leq m \leq n-1 \), such that \( T[0] \leq T[1] \leq \ldots \leq T[m] \) and \( T[m+1] \leq T[m+2] \leq \ldots \leq T[n-1] \). However, you do not know the value of \( m \). Show a linear time upper bound on the worst-case complexity of sorting a partially sorted array \( T \).
2. (25 points) A graph is *arc-biconnected* if the removal of any single arc of the graph leaves a connected graph. Describe an algorithm to test if a graph $G=(N,A)$ is *arc-biconnected* whose worst case execution time is in $O(|A|*(|N|+|A|))$. 
3. (25 points) Describe an algorithm which accepts as input a graph $G=(N,A)$, a source node $\sigma \in N$ and a destination node $\tau \in N$, and returns a shortest path from $\sigma$ to $\tau$, where the length of a path is the number of arcs on the path. Your algorithm should execute in worst case time in $O(|N|+|A|)$. 
4. (25 points) For (undirected) graph $G=(N,A)$ with function $\text{length} : A \rightarrow \mathbb{R}^+$, the distance between any pair of nodes is the length of a shortest path between the nodes and the diameter of $G$ is the distance between two most distant nodes. That is, if $d : N \times N \rightarrow \mathbb{R}^+$ denotes the distance between a pair of nodes, then $\text{diameter}(G) = \max_{u,v \in N} \{d(u,v)\}$. If $G$ is not connected, then $\text{diameter}(G) = \infty$. For example, the diameter of

is 7 because the distance between $a$ and $d$ is 7, and they are the most distant vertices of $G$. Find an algorithm to compute the diameter of a graph in worst case time in $O(n^3)$. 
1. We can use a divide-&-conquer algorithm:
   - Divide problem into subproblems $T[1..m]$ and $T[m+1..n]$ $O(n)$
   - Recombine the subproblems by MERGing them $O(n)$

First locate $m$ (in time in $O(n)$), then merge the two sublists (in time in $O(n)$).

   \[
   \begin{align*}
   m &= 0; \\
   &\text{do } m++ \text{ while } ((T[m] \leq T[m+1]) \&\& (m < n-1)); \quad /* \text{Find m} */ \\
   &\text{if } m < n - 1 \text{ then MERGE(T[0..m],T[m+1..n-1],T[1..n])}
   \end{align*}
   \]

The while-loop requires time proportional to $m \in O(n)$ since it is executed at most $n$ times, and the body of the loop, as well as the test for entry into the loop, can each be executed in $O(1)$ time. MERGE requires time $O(n)$.

2. $biconnected \leftarrow \text{true}$
   \[\text{for each } a \in A \]
   \[\quad \text{if not connected}(G/\{a\}) \text{ then } biconnected \leftarrow \text{false}\]
   \[\text{return } biconnected\]

where $\text{connected}$ does a depth first search (or breadth first search) in time in $O(n+a)$ to check if its input is connected.

3. Do a breadth first search starting at node $\sigma$. If node $\tau$ appears at depth $k$, then the shortest path from $\sigma$ to $\tau$ has length $k$.

4. Run the Floyd-Warshall algorithm to solve the all pairs shortest path problem in worst case time in $O(n^3)$ and then find the distance between the most distant pair of nodes in time in $O(n^2)$. 