CS2223

Final Exam

Name____________________________

Date: December 17, 1999

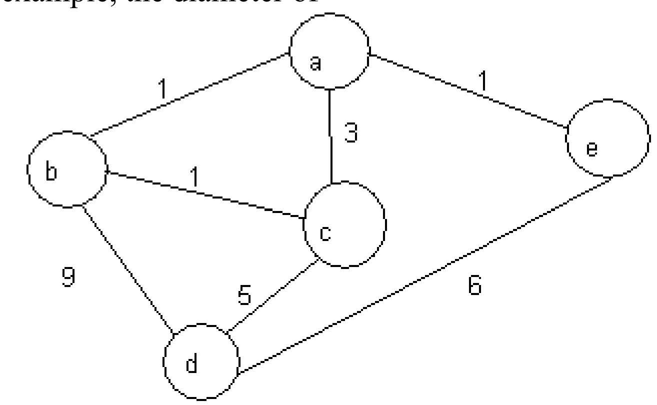
All documentation permitted

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, $0 \le m \le n-1$, such that $T[0] \le T[1] \le \ldots \le T[m]$ and $T[m+1] \le T[m+2] \le \ldots \le T[n-1]$. 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 $O\left(|A| * \left(|N| + |A|\right)\right)$.

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 σ to τ, 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 $length: A \to \mathbb{R}^+$, 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 $d: N \times N \to \mathbb{R}^+$ denotes the distance between a pair of nodes, then $diameter(G) = \max_{u,v \in N} \{d(u,v)\}$. If G is not connected, then $diameter(G) = \infty$. 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 $O(n^3)$.

CS2223

Solutions to Final Exam

1. We can use a divide-&-conquer algorithm:

   -Divide problem into subproblems $T[1..m]$ and $T[m+1..n]$ $\quad O(n)$

   -Recombine the subproblems by MERGing them $\quad O(n)$

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

```
m  = 0;
do m++ while ((T[m ]<=T[m+1]) && (m<n-1));          /*  Find m */
 if m <n -1  then MERGE(T[0..m ],T[m +1..n-1],T[1..n ])
```

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* ← true

   for each $a \in A$

       if not *connected*$(G/\{a\})$ **then** *biconnected* ← false

   return *biconnected*

where *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 σ. If node τ appears at depth k, then the shortest path from σ to τ 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 $O(n^2)$.