CSE 101 Homework 2
Spring 2013
Algorithm analysis, correctness proofs, Graph representations, graph search
Due Wednesday, April 16
100 points total

Invariants The following algorithm, given a (not necessarily sorted) array of positive integers, $A[1..n]$, finds for each $1 \leq I \leq n$, the smallest value $J$ in the range $I \leq J \leq n$ with $A[J] > A[I]$ and returns that as $B[I]$, with $B[I] = n + 1$ if no such $J$ exists. The correctness proof will follow from a series of lemmas. Prove each lemma. (5 points each lemma)

NextLargest($A[1..n]$):
1. Initialize a stack of integers.
2. Push $n$.
4. FOR $I = n - 1$ downto 1 do:
   5. While Stack is non-empty and $A[I] \geq A[Top]$ do: Pop;
   6. IF Stack is empty THEN $B[I] \leftarrow n + 1$ ELSE $B[I] \leftarrow Top$;
   7. Push($I$)

Lemma 1: If the stack is non-empty, it is increasing from top to bottom.
Lemma 2: If the stack is non-empty, and has contents $J_1,...,J_k$ from top to bottom, then the corresponding array values are increasing from top to bottom, $A[J_1] < A[J_2] < A[J_3] < ...A[J_k]$.
Lemma 3: If $j > I$ and $j$ is not on the stack, then there is some $I \leq j' < j$ with $A[j'] \geq A[j]$.
Lemma 4: After the $I$’th iteration, $B[I]$ is the smallest $j \geq I$ with $A[j] > A[I]$, or is $n + 1$ if no such $j$ exists. (Hint: assume such a $j$ exists. Use Lemma 3 to prove that it is still on the stack when iteration $I$ starts. Then use Lemmas 1 and 2 to show that all the elements above it on the stack get deleted during this iteration.)

Time analysis Give a time analysis for the algorithm above. Be careful to explain and justify your answer. (20 points).

Triangle detection A triangle in an undirected graph $G = (V, E)$ is a triple of distinct nodes $u, v, w$ so that $\{u, v\}, \{v, w\}, \{u, w\}$ are all in $E$, i.e., there are direct edges in the graph between any two of the three nodes. Give an algorithm that, given an undirected graph in adjacency matrix representation, decides whether it has a triangle. Then give an algorithm
for the same problem, when the graph is given in adjacency list format. As always, you need to give an argument that your algorithm is correct (this could be short, in this case), and a time analysis (in terms of the number of nodes \( n = |V| \) and the number of edges, \( m = |E| \)). A good algorithm in either case has time \( O(nm + n^2) \).

(30 points, 15 points each representation. For each, 3 points correctness and correctness argument, 5 points for a correct time analysis of the given algorithm, and 7 points for efficiency (no points for efficiency unless you are better than the obvious \( O(n^3) \) algorithm, and full credit for matching the \( O(nm + n^2) \) bound.)

**Odd length paths** Consider the problem of, given a directed graph \( G \), and a node \( u \), list all the nodes \( v \) so that there is a path using an odd number of edges from \( u \) to \( v \). (Note that such a path might have to have a cycle, and need not be the shortest path.) Give an efficient algorithm (with correctness proof and time analysis) for this problem. My algorithm runs in time \( O(n + m) \). Hint: consider replacing each node with two nodes, one representing arrival via an even path, and the other via an odd path. (30 points, 10 for correctness and the correctness proof, 10 for a correct analysis of your algorithm, and 10 for efficiency, with full points for efficiency for the \( O(n + m) \) time above.)