CS 218, Spring 18

Entrance Quiz B

Name (first last) .................................................................

Student ID .................................................................

• This quiz is closed book, closed notes and 30 minutes long
• Read the questions carefully
• No electronic equipment allowed (cell phones, tablets, computers, . . . )
• Write legibly. What can’t be read will not be graded
• Use pseudocode (or English) to describe your algorithms
• Always remember to analyze the time complexity of your solution
• If you have a question about the meaning of a question, raise your hand
Problem 1. (42 points: 6 points if correct, 3 if unanswered, 0 if wrong)
Mark by true or false each of the following (no need to prove).

\[ 4^{\log_2 n} \in \Omega(n^2 \log n) \]  
\[ \log_3 3^{n^2} \in \Theta(n \log_2 2^n) \]  
\[ \sqrt{n} \log_3 n^2 \in O(n \log_3 n) \]

The following questions are on graphs; assume that \( n = |V| \) is the number of vertices, and \( m = |E| \) is the number of edges; DFS is “depth first search”; BFS is “breadth first search”; in DFS/BFS the the set of edges visited during the execution of these algorithms are called tree or discovery edges; non-tree edges are the others (also called back edges in DFS, cross edges in BFS)

Given the spanning tree \( T \) formed by the discovery (tree) edges of a DFS traversal of a connected undirected graph \( G \) started from node \( s \), for each vertex \( v \), the path on tree \( T \) is the shortest path between \( s \) and \( v \)

An edge \( e \) whose removal disconnects a graph is called a bridge; if DFS is run on a connected undirected graph \( G \), every bridge in \( G \) is a discovery (tree) edge in the DFS tree

For a connected undirected graph \( G \), the presence of a back (non-tree) edge in any DFS visit of \( G \) implies that \( G \) has a cycle

If one runs a BFS on a connected undirected graph, the number of cross (i.e., non-tree) edges is exactly \( m - n + 1 \)
Problem 2. (24 points: 8 points each)

For each of the concepts listed below write a precise (possibly formal) definition. Do not explain or comment about the corresponding algorithm, if any.

1. A transitive closure of an undirected graph $G = (V, E)$ is a graph $G = (V, E')$ where $(u, v) \in E'$ if there is a path from $u$ to $v$ in $G$.

2. The topological ordering of a directed acyclic graph $G = (V, E)$ is an ordering of its vertices, say $\{v_1, v_2, \ldots\}$, such that for every directed edge $(v_i, v_j) \in E$, we have that $i < j$.

3. A spanning tree of an undirected graph $G = (V, E)$ is any acyclic subgraph of $G$, i.e., $T = (V, E')$ such that $E' \subseteq E$ and $T$ acyclic.

4. A cycle in an undirected graph $G = (V, E)$ is a set of edges $\{(u_1, u_2), (u_2, u_3), \ldots, (u_{l-2}, u_{l-1}), (u_{l-1}, u_l)\}$ where $u_l = u_1$.

5. A binary heap is a nearly complete binary tree where all nodes are either greater than or equal to (or less than or equal) to each of its children.
Problem 3. (34 points)

Given a sorted (low to high) array of distinct integers $A = \{a_1, a_2, \ldots, a_n\}$, describe an $O(\log n)$ algorithm to determine whether there exists an index $i$ such that $a_i = i$. For example, in $\{-10, -3, 3, 5, 7\}$, $a_3 = 3$. In $\{2, 3, 4, 5, 6, 7\}$, there is no such $i$. Explain briefly how the algorithm works, and why your solution takes $O(\log n)$ time.

**Answer:** The key observation is that since $a_i$ are sorted and distinct, $a_i - i$ is sorted as well, so we can use binary search. In fact, the algorithm works exactly like binary search.

Algorithm `Search` ($A$ : array, $i$ : integer, $j$ : integer)

if ($i = j$) then return ($A[i] = i$)

let $k \leftarrow \lfloor (j - i) / 2 \rfloor$

if ($A[k] > k$) then return `Search`($A, i, k - 1$)

else return `Search`($A, k, j$)

We call the algorithm with `Search`($A, 1, n$). The complexity is $O(\log n)$ because it is modified binary search. Alternatively, one can solve $T(n) = T(n/2) + 1$. 