CS 218, Spring 17

Entrance Quiz A

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

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

- This quiz is closed book, closed notes and 35 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

1

2

3

Total

/45

/25

/30

/100
Problem 1. (45 points: 5 points if correct, 2.5 if unanswered, 0 if wrong)
Mark by true or false each of the following (no need to prove).

$\sqrt{n} \log_2 n^3 \in O(n \log_2 n)$ ☑ True ☐ False

$9^{\log_3 n} \in \Omega(n^2 \log n)$ ☐ True ☑ False

$\log_2 2^n^2 \in \Theta(n \log_3 3^n)$ ☑ True ☐ False

RADIX-SORT runs in $O(dn)$ time for an array of $n$ $d$-digit integers ☑ True ☐ False

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)

A directed complete graph with $n$ nodes has exactly $n(n - 1)/2$ edges ☑ True ☐ False

Given the spanning tree $T$ formed by the discovery (tree) edges of a BFS 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 $v$ ☑ True ☐ False

An edge $e$ whose removal disconnects the graph is called a bridge; if BFS is run on a connected undirected graph $G$, it is a possible for a bridge in $G$ to be a cross (non-tree) edge ☑ True ☐ False

For a connected undirected graph $G$, the absence of back (non-tree) edges with respect to a DFS tree implies that $G$ is acyclic ☑ True ☐ False

If one runs a DFS on a connected undirected graph, the number of back (non-tree) edges is exactly $m - n + 1$ ☑ True ☐ False
Problem 2. (25 points: 5 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. 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 $i < j$

2. 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

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

4. 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

5. A transitive closure of a directed graph $G = (V, E)$ is a graph $G = (V, E')$ where $(u, v) \in E'$ if there is a directed path from $u$ to $v$ in $G$
Problem 3. (30 points)

You are given an unsorted array $A[1 \ldots n]$ of $n$ distinct integers. We say that $A[i]$ is a local maximum if $A[i]$ is bigger than its neighbors, that is, $A[i] > A[i-1]$ (if $i \neq 1$) and $A[i] > A[i+1]$ (if $i \neq n$). For instance in $A = \{3, 4, 1, 2, 5, 6, 0, 8, 9\}$, 4 is a local maximum, 6 is a local maximum, and 9 is a local maximum. Give an $O(\log n)$-time algorithm to find any of the local maximum in $A$. If there is more than one local maximum in $A$, we are OK with any one of them. Explain briefly how the algorithm works, and why it runs on $O(\log n)$ time.

You can assume that $n$ is a power of two. **Hint:** If $A[i]$ is not a local maximum because $A[i] < A[i+1]$, then must there be some local maximum $A[j]$ with $j > i$?

**Answer:** The answer to the hint is “yes”, because if $A[i] < A[i+1]$, then the first $j > i$ such that $A[j] > A[j+1]$ or $j = n$ will be a local maximum. Likewise, if $A[i] < A[i-1]$, then there must be a local maximum to the left of $i$. Thus, we can do binary search over \{1, 2, \ldots, n\} to find a local maximum. Here is pseudo-code:

```
Algorithm findPeak(A)
    if $n = 1$ then return $A[1]$
    $i \leftarrow \lfloor n/2 \rfloor$
    if $A[i] < A[i+1]$ then return findPeak($A[i+1 \ldots n]$)
    if $A[i] < A[i-1]$ then return findPeak($A[1 \ldots i-1]$)
    return $i$
```

Each recursive call takes constant time and cuts the problem size in half, so the total time is $O(\log n)$. 