• 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
Problem 1. (40 points: 4 points if correct, 2 if unanswered, 0 if wrong)
Mark by true or false each of the following (no need to prove).

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

An array sorted in decreasing order is always a max-heap

A max-heap is always a sorted array (in decreasing order)

The following T/F 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)

An undirected complete graph with \( n \) nodes has exactly \( n(n - 1)/2 \) edges

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: 6 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 $u_l = u_1$
Problem 3. (36 points)

Suppose you are given an array \( A = \{a_1, a_2, \ldots, a_n\} \) of \( n \) distinct integers. You are told that the sequence of values \( a_1, a_2, \ldots, a_n \) is unimodal, that is for some index \( p \in [1, n] \), the values in the array increase up to position \( p \) in \( A \), and then decrease the remainder of the way until position \( n \). Given the algorithm to find the position \( p \) in \( O(\log n) \) time. You can assume \( n \) to be a power of 2.

**Answer:** The algorithm works like a binary search. Compare the elements \( A[n/2], A[n/2−1] \) and \( A[n/2+1] \) to decide whether to search on the left, on the right, or whether we are done. More specifically

- if \( A[n/2−1] < A[n/2] < A[n/2+1] \), then search recursively in the entries \( A[n/2+1 \ldots n] \)


The algorithm has the same structure of binary search, its recurrence relation is \( T(n) = T(n/2) + O(1) \), which has solution \( O(\log n) \).