1. Simple-Quicksort uses the first element of the array as a pivot. Randomized-Quicksort uses a random array element as the pivot. In the worst-case both algorithms take \( O(n^2) \) time to sort an array.

(a) (2 pts) Give an example worst-case input for Simple-Quicksort. Show that Simple-Quicksort takes \( O(n^2) \) time to sort this input.

**ANSWER:** Suppose the input is 1, 2, 3, \ldots , n. Then each recursive call of Simple-Quicksort will pivot on the first remaining value and recurse on an empty array and the rest of the values taking \( O(n) + O(n-1) + \ldots + O(1) = O(n^2) \) time.

(b) (2 pts) What is the expected running time for Simple-Quicksort on your input from (a)?

**ANSWER:** The running time never changes, so the expected running time is \( O(n^2) \).

(c) (2 pts) What is the expected running time for Randomized-Quicksort on your input from (a)?

The input does not affect the running time because Randomized-Quicksort makes random choices, so the expected running time is \( O(n \log n) \).

(d) (2 pts) What is the worst-case running time for Randomized-Quicksort on your input from (a)?

The input does not affect the running time because Randomized-Quicksort makes random choices, so the worst-case running time is \( O(n^2) \).

(e) (2 pts) What is the worst-case input for Randomized-Quicksort?

This is a trick question—there is no worst-case input! Every input has an equally likely chance of taking \( O(n^2) \) time and every input has an expected running time of \( O(n \log n) \).
2. NP-complete problems are considered unlikely to be in P because they can each be reduced to Satisfiability and Satisfiability can be reduced to any problem in NP (so they can all be reduced to each other). It is often difficult to determine if any given problem is NP-complete and we require formal reductions to actually prove this. However, even though a general problem such as Satisfiability may be NP-hard, simplified versions of the problem may be in P. For example, 3-Satisfiability is NP-complete while 2-Satisfiability (satisfiability restricted to 2 variables per clause) is in P.

Consider the Hamiltonian Cycle problem: “Given a graph $G$, does $G$ have a simple cycle containing all of the vertices of $G$?” (Recall that a simple cycle contains no edge or vertex twice) This problem is NP-complete because it can be reduced from Vertex-Cover which can be reduced from 3-Satisfiability. Now consider each of the following modifications of Hamiltonian Cycle. Do you think it is NP-complete or in P? If you believe it is in P then give a brief description of a method that will answer the question in polynomial time. If you believe it is NP-complete then give a brief justification such as an NP-complete problem that reduces to this problem (formal reductions are technically required for this but outside of the scope of this course).

(a) (2 pts) Hamiltonian Cycle restricted to trees

**ANSWER:** This problem is in P. Trees are acyclic so we can simply answer “no” in constant time. In practice, given the general problem, a useful preprocessing step is to run the linear time cycle detection algorithm and answer “no” when we find no cycles at all.

(b) (2 pts) Hamiltonian Cycle restricted to directed acyclic graphs

**ANSWER:** This is in P and is similar to part (a). There are no cycles so we can answer “no” in constant time.

(c) (2 pts) Hamiltonian Cycle restricted to disconnected graphs

**ANSWER:** This is also in P. The graph is disconnected so there cannot be a cycle containing all of the vertices. Similar to part (a) and (b) we can return “no” in constant time.

(d) (2 pts) “Given a graph $G$ and a list of $k$ cycles in $G$, is one of these cycles a Hamiltonian Cycle?”

**ANSWER:** This is in P because the cycles are part of the input. We simply check each of the cycles to see if it is Hamiltonian taking $O(k \cdot n)$ time where $n$ is the number of vertices.

(e) (2 pts) The Travelling Salesman Problem: “Given a graph $G$ with weighted edges, what is the simple cycle of $G$ containing all of the vertices of $G$ that has maximum weight (if any such cycle exists)?”

**ANSWER:** This is NP-Complete because we can reduce Hamiltonian Cycle to it. Any Hamiltonian Cycle problem is just a Travelling Salesman Problem with all edge weights set to 1.
(f) (Bonus: 2 pts) Hamiltonian Cycle on hypergraphs. A hypergraph is a graph where the edges connect two or more vertices. A path through a hypergraph can follow an edge to any one of its endpoints.

**ANSWER:** This is NP-Complete because we can reduce Hamiltonian Cycle to it. Any Hamiltonian Cycle problem on a graph is a Hamiltonian Cycle problem on a hypergraph where each edge connects exactly 2 vertices.

(g) (Bonus: 2 pts) “Given a graph $G$ does the graph contain a cycle of length $k$?”
(Note: for this problem $k$ is a constant, not an input variable)

**ANSWER:** This problem is in P because $k$ is a constant. We can simply check all paths of length $k - 1$ from each of the $n$ vertices in $O(n \cdot n^{k-1} \cdot n) = O(n^{k+1})$ time. Such a constant is known as a fixed-parameter and is a useful tool for designing efficient algorithms to solve many NP-Complete problems.