1. **Short questions.**
   a. Explain why all comparison-based sorting algorithms must run in $\Omega(n \log n)$ time on an input of $n$ numbers.
   b. Suppose you have the option of using quicksort or bucket sort for sorting a set of numbers. When is it preferable to use bucket sort?
   c. Describe the *divide-and-conquer* approach for solving a problem.

2. **True or false.** If true, please provide a short proof. If false, please illustrate with a counterexample.
   a. Suppose we ran depth-first-search and breadth-first-search on the same graph $G$. The two algorithms always produce the *different* search trees.
   b. Let $e^*$ be the unique cheapest edge in $G$. The minimum spanning tree of $G$ always contains $e^*$.
   c. In the 0/1 knapsack problem where each object $i$ has a benefit $b_i$ and a weight $w_i$, and a knapsack that can carry a maximum weight of $W$, the greedy algorithm of iteratively choosing the object with the best benefit-to-weight ratio to place in the knapsack maximizes the total benefit of the items placed in the knapsack.

3. **Insertion Sort.**
   Given array $A[0 \ldots n-1]$, recall that insertion sort works in the following manner:
   
   ```
   for $i = 1$ to $n - 1$
     $temp \leftarrow A[i]$
     $j \leftarrow i.$
     while ($j > 0$ and $A[j-1] > temp$)
       $A[j] \leftarrow A[j-1]$
       $j \leftarrow j - 1$
     $A[j] \leftarrow temp$
   ```
   a. What is the worst case running time of insertion sort for the following types of inputs: (i) sorted input, (ii) reverse-ordered input, (iii) random input?
   b. Recall that if $i < j$ and $A[i] > A[j]$ then $(i, j)$ is an inversion of $A$. Modify insertion sort to count the number of inversions in $A$.

4. **Strongly connected components.** Given a directed graph $G = (V, E)$, recall that it is *strongly connected* if for every pair of vertices $u$ and $v$ in $V$, there is always a path from $u$ to $v$ and from $v$ to $u$. Not all directed graphs, however, are strongly connected.
connected. Nonetheless, every directed graph can be decomposed into strongly connected components (just like every simple graph can be decomposed into connected components). For example, the graph below has three strongly connected components: (a) the component consisting of node $s$ only, (b) the component consisting of node $t$ only, and (c) the component consisting of $a, b$ and $c$. Given a directed graph, describe an algorithm that would output all of the graph’s strongly connected components. What is its running time?

5. *Topological sorting and shortest paths.* (a) Topologically sort the graph below and (b) find the shortest path from $s$ to $t$ using one of the algorithms you’ve learned in class. Please illustrate intermediate steps!
6. **Floyd-Warshall’s algorithm.** Floyd-Warshall’s algorithm for all-pairs shortest paths was applied on a graph with 4 vertices and the matrix containing $d_{ij}^2$ for every pair of vertices $i$ and $j$ is the following

$$D^{(2)} = \begin{bmatrix}
0 & \infty & 3 & \infty \\
2 & 0 & 5 & \infty \\
9 & 7 & 0 & 1 \\
6 & \infty & 9 & 0
\end{bmatrix}.$$ 

Compute for the all-pairs shortest path distances in the graph. That is, find $d_{ij}^4$ for every pair of vertices $i$ and $j$.

7. **Minimum spanning trees with specified edges.** Suppose edge $e$ is a specified edge in graph $G$. Describe an algorithm that would find the cheapest spanning tree in $G$ that contains edge $e$. Argue the correctness of your algorithm. What is its running time?

8. **Graph Modeling.** A student needs to take a certain number of courses to graduate, and these courses have prerequisites that must be followed. Assume that all course are offered every semester, and the the student can take an unlimited number of courses. Given a list of courses and their prerequisites, we wish to compute a schedule for the student that requires the minimum number of semesters. How would you solve this problem using a shortest-path algorithm?