Clearly indicate (draw) the following spanning trees for the weighted graph in the problem sheets. There may be more than one correct answer.

(a) (2.5 out of 10) A depth-first spanning tree rooted at $s$.

Solution:

(b) (2.5 out of 10) A breadth-first spanning tree rooted at $s$.

Solution:
(c) (2.5 out of 10) A shortest path tree rooted at \( s \).

Solution:

(d) (2.5 out of 10) A minimum spanning tree.

Solution:
For each of the following graph problems, do the following.

- Either name or briefly describe the fastest algorithm seen in class to solve the problem.
- State the running time of the algorithm in terms of $V$ and $E$.

If an algorithm involves priority queues, feel free to state the running time assuming it uses a standard min-heap with $O(\log n)$ time operations. Naming or describing a correct algorithm that is slower than the best one we discussed for the problem is worth partial credit. You may assume the graph is connected for each part.

(a) (2 out of 10) Given a directed graph $G = (V, E)$ with non-negative edge weights $w : E \rightarrow \mathbb{R}_{\geq 0}$ and a vertex $s \in V$, compute a shortest path tree out of $s$.

**Solution:** We can use Dijkstra’s algorithm which runs in $O(E \log V)$ time.

(b) (2 out of 10) Given an undirected graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, compute a spanning tree of $G$ with minimum total edge weight.

**Solution:** We can use Kruskal’s algorithm which runs in $O(E \log V)$ time.

(c) (2 out of 10) Given a directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, compute $dist(u, v)$, the shortest path distance from $u$ to $v$, for all pairs of vertices $u$ and $v$.

**Solution:** We can use Floyd-Warshall which runs in $O(V^3)$ time.

(d) (2 out of 10) Given a directed graph $G = (V, E)$ and two vertices $s$ and $t$ in $V$, compute the minimum number of edges along any path from $s$ to $t$.

**Solution:** We can use breadth-first search which runs in $O(V + E)$ time.

(e) (2 out of 10) Given an undirected graph $G = (V, E)$, compute a spanning tree of $G$ with a minimum number of edges. (We did not directly discuss this problem in class, but one or more algorithms we did discuss will solve it anyway.)

**Solution:** All spanning trees have exactly $|V| - 1$ edges. We can use depth-first search (or any other fast variant of whatever-first search) which runs in $O(V + E)$ time.
For each of the following greedy algorithms for the minimum stabbing set problem, write **Correct** if the algorithm always constructs a minimum stabbing set or write **Wrong** if there is some input for which the algorithm does not produce a minimum stabbing set. You must clearly write exactly one of **Correct** or **Wrong** to get credit for each part.

(a) **(2 out of 10)** Choose the left endpoint $p$ of the interval that starts first, discard all intervals containing $p$, and recurse. (In other words, $p$ is the minimum value in $L[1..n]$.)

**Solution:** **Wrong**: Consider intervals $[0,2]$ and $[1,2]$. The greedy approach would take $p = 0$, requiring a second point to stab the second interval. A better solution is $\{2\}$. ■

(b) **(2 out of 10)** Choose the right endpoint $p$ of the interval that ends first, discard all intervals containing $p$, and recurse. (In other words, $p$ is the minimum value in $R[1..n]$.)

**Solution:** **Correct**: Given any minimum stabbing set, we can replace it’s leftmost point $q$ with $p$. No interval ends before $p$, so any interval stabbed by $q$ is still stabbed by $p$. ■

(c) **(2 out of 10)** Choose a point $p$ that is included in the maximum number of intervals, discard all intervals containing $p$, and recurse.

**Solution:** **Wrong**: Consider intervals $\{[0,1],[0,5],[0,6],[3,10],[4,10],[9,10]\}$. The greedy approach takes $p \in [4,6]$, requiring two more points to stab $[0,1]$ and $[9,10]$. A better solution is $\{0,10\}$. ■

(d) **(2 out of 10)** Choose the left endpoint $p$ of the interval that starts last, discard all intervals containing $p$, and recurse. (In other words, $p$ is the maximum value in $L[1..n]$.)

**Solution:** **Correct**: Given any minimum stabbing set, we can replace it’s rightmost point $q$ with $p$. No interval starts after $p$, so any interval stabbed by $q$ is still stabbed by $p$. ■

(e) **(2 out of 10)** Choose a point $p$ that is included in at least one interval, but is otherwise in the minimum number of intervals, discard all intervals containing $p$, and recurse.

**Solution:** **Wrong**: Consider intervals $[0,2]$ and $[1,2]$. The greedy approach would take $p \in [0,1)$, requiring a second point to stab the second interval. A better solution is $\{2\}$. ■
Suppose there are \( n \) students participating in the tournament and they play \( m \) games total. We want to make a consistent list of students (if possible) by topologically sorting the vertices of a graph.

1. \( \text{(2 out of 10)} \) What set should we use for the vertices of the graph? (Either define the set plainly in English, or provide a short algorithm for creating a useful vertex set.)

   **Solution:** The students are the vertices.

2. \( \text{(2 out of 10)} \) What set should we use for the edges of the graph? (Either define the set of edges plainly in English, or provide a short algorithm for creating a useful edge set.) Don’t forget to specify the direction of the edges.

   **Solution:** There is an edge from student/vertex \( i \) to student/vertex \( j \) if student \( i \) wins at least one game against student \( j \).

3. \( \text{(2 out of 10)} \) What must be true about the graph for a consistent list of students to exist?

   **Solution:** It must be acyclic (have no directed cycles).

4. \( \text{(2 out of 10)} \) Assuming a consistent list of students does exist, what will the topological sort return, and how will that help us order the students according to their ability to play Battleship?

   **Solution:** The topological sort returns an ordered list of vertices so edges only go forward in the list. This exact list is the list of students we seek.

5. \( \text{(2 out of 10)} \) In terms of \( n \) and \( m \), how long will it take to build the graph, run topological sort, and build the consistent list of students if a consistent list exists?

   **Solution:** There are \( n \) vertices and at most \( m \) edges in the graph so building it takes \( O(n + m) \) time. Topological sort takes \( O(V + E) \) time, so the **running time of the algorithm is** \( O(n + m) \).