Here is a list of what we have covered since the first exam.

- Three basic algorithm design techniques, divide and conquer, dynamic programming, and greedy,
- Recursive, top-down, and bottom-up implementations of dynamic programming algorithms.
- Dynamic programming algorithms for MCS, MNCS, LIS, LCS, Edit Distance.
- Dynamic programming algorithms for the integer knapsack problems.
- Greedy algorithm for the fractional knapsack problem.
- Trees and tree traversals (preorder, inorder, postorder, and level-order).
- Undirected graphs and their representations using adjacency list or adjacency matrix.
- DFS (both recursively and explicitly using a stack) and BFS (using a queue).
- Various spanning trees, e.g., DFS spanning tree, BFS spanning tree, MST.
- Prim's and Kruskal's algorithms for MST.

1. Sally is hosting an auction to sell n widgets. She receives m bids, each of the form

$$\text{“I want } k_i \text{ widgets for } d_i \text{ dollars”, for } i = 1..m.$$

   Explain how her optimization problem is really just a knapsack problem.

2. Let $S = \{ a, b, c, d, e, f, g \}$ be a collection of objects with weight-value pairs as follows:

```
a (4,11)
b (6,8)
c (5,10)
d (7,6)
e (3,12)
f (1,14)
g (8,4)
```

   Assume that we have a knapsack that can hold objects with total weight 18. Find the optimal solution to the 0-1 knapsack problem.

3. Let

$$D(i, j) = 1 + D(i - 1, j) + D(i, j - 2)$$

with

$$D(0, j) = D(i, 0) = D(i, -1) = 0.$$

Compute $D(6, 6)$ by filling in the following table in a top-down dynamic programming manner. That is, only fill in those table entries that the top-down algorithm needs filled in when computing $D(6, 6)$. (To save space, the following table does not contain row 0 nor columns 0 and -1.)

4. Let

$$C(i, j) = C(i - 1, j - 1) + C(i - 2, j - 2)$$

with

$$C(i, 0) = C(0, i) = 0 \quad \text{for } 0 \leq i \leq n$$

and

$$C(i, 1) = C(1, i) = i \quad \text{for } 1 \leq i \leq n.$$

(a) If we implement this as a top-down dynamic programming algorithm, show that computing $C(n, n)$ is $\Theta(n)$.

(b) If we implement this as a bottom-up dynamic programming algorithm, show that computing $C(n, n)$ is $\Theta(n^2)$.

(c) More specifically, show that computing $C(i, j)$ is $\Theta(ij)$ as a bottom-up algorithm, and computing $C(i, j)$ is $\Theta(\min(i, j))$ as a top-down algorithm.

5. Does this greedy strategy solve the Maximum Independent Subsequence problem?
   - At each step, find the maximum number that is independent from all the previously chosen numbers.
6. Draw a single binary tree T such that
   - Each node of T stores a single character.
   - A preorder traversal of T yields "examfun".
   - An inorder traversal of T yields "mafxuen".
7. (a) Give two different binary trees that yield the same preorder traversal.
   (b) Give two different binary trees that yield the same inorder traversal.
   (c) Give two different binary trees that yield the same postorder traversal.
8. In a binary tree with more than one node, is it possible that the preorder traversal of the tree visits the nodes in the same order as the postorder traversal? If so, give an example; otherwise argue why this cannot occur.
9. Given the preorder and inorder traversals of a binary tree, is it possible to reconstruct the tree? If so, sketch an algorithm to do it. If not, give a counterexample. Repeat the problem if you are given the preorder and postorder traversals.
10. Consider the undirected graph G with vertices
$$V = \{ a, b, c, d, e, f, g \}$$
and edges
$$E = \{ (a, b), (a, c), (a, d), (a, f), (a, g), (b, c), (b, g), (d, e), (d, f), (e, f) \}.$$
   (a) Draw a picture of the graph. Then draw an adjacency list representation of the graph where each vertex sees its neighbors in alphabetical order.
   (b) Draw the BFS spanning tree achieved by starting at vertex a and visiting neighbors in alphabetic order.
   (c) Draw the DFS spanning tree achieved by starting at vertex a and visiting neighbors in alphabetic order.
11. Suppose that G is a weighted graph that is not connected. Then G cannot have a minimum spanning tree.
   (a) What would be the result from applying Prim's algorithm to G?
   (b) What would be the result from applying Kruskal's algorithm to G?