1. Let \( n \) be the number of integers in a list. An algorithm that does something cool with the numbers in the list consists of the following three for loops.

   for i in range(n):
       for j in range(n / 2):
           for k in range(n):
               # does something cool here that does not depend on \( n \)

   a. (2 points) Write a mathematical expression involving \( n \) that is the exact number of times something cool is executed.

   \[
   n \cdot \frac{n}{2} \cdot n = \frac{n^3}{2}
   \]

   b. (4 points) Write your expression as a \( O() \) function of \( n \).

   \( O(n^3) \)

   c. (4 points) Prove that your answer to b. is correct.

   \[
   \frac{n^3}{2} \leq cn^3
   \]

   \[
   \frac{1}{2} \leq c
   \]

   \[
   c = 1
   \]

   \[
   n > n_0 \quad n_0 = 1
   \]
2. (10 points) The activity selection problem can be solved with a greedy algorithm. What variation of this problem did we study that does not have a greedy solution?

Weighted activity selection problem

3. (10 points) You have written a correct dynamic programming algorithm with a top-down approach to solve a problem. However, you find that it is taking too much time for the size of problems are interested in. What can you do to reduce the computation time, without changing the top-down recursive nature of your algorithm?

Use memoization to remember values for smaller size problems whose values were previously computed.

4. (10 points) Can you select the fewest number of coins to total a given amount with a greedy algorithm? If your answer is “yes”, briefly describe the greedy algorithm. If your answer is “no”, give an example that cannot be solved with a greedy algorithm and explain it.

Yes. The cashier's algorithm. Select coins of largest value without exceeding the given amount recursively. Repeat with next smaller coins, until given amount is reached.

5. (5 points) Which algorithm did we discuss that can find a path in a graph from one node to another node when the edges can have positive or negative weights?

Bellman-Ford
6. (10 points) Show an example 0/1 Knapsack problem for which the greedy algorithm does not find the optimum solution. Start with a table of items that are sorted in decreasing order of value to weight ratios. Then explain what the greedy algorithm finds, and why it is not optimal.

<table>
<thead>
<tr>
<th>indices</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>weights</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>values</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Greedy:** Add item 1 to knapsack.
Cannot add another.
Value = 10

**Optimal Solution:** Add items 2 and 3.
Value = 18.

\[
\text{OPT}(i,j) = \begin{cases} 
\min \left( x_i \cdot y_j + \text{OPT}(i-1,j-1), \right. \\
\left. x_i \cdot y_j + \text{OPT}(i-1,j), \right. \\
\left. x_i \cdot y_j + \text{OPT}(i,j-1) \right)
\end{cases}
\]

7. (10 points) In the sequence alignment problem, explain in words what the three terms represent in the recurrence relation for the "score" of a particular solution.

1. Align characters in each sequence for an additional score of 1 if they don't match, 0 if they do match.

2. Align character in first string with a gap added to second string, for an additional score of 1.

3. Align character in second string with a gap added to first string, for an additional score of 1.
8. (10 points) Here is a table that shows the alignment score table that is filled in by following the sequence alignment algorithm discussed in class. Knowing how these scores are calculated, you can extract the minimum-score alignment. Write the resulting alignment as one four-character string over the other, with underscore characters added where a gap is inserted in the resulting alignment. Underneath each position in the alignment write the value that is added to the score. If more than one solution is possible, just provide one.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

 Alignment:

\[\text{ATCG}\]
\[\text{ATGT}\]
Score = 2

8. (10 points) Is the following set of codes for four characters a prefix code? Why or why not?

<table>
<thead>
<tr>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1011</td>
<td>0011</td>
<td>011</td>
<td>101</td>
</tr>
</tbody>
</table>

No. Code for A (101) is prefix of code for G (1011).
10. (10 points) What is the worst-case complexity for a branch-and-bound solution to a knapsack problem with \( n \) items?

- Must check all subsets of items.
- For \( n \) items, there are \( 2^n \) subsets.
- So \( O(2^n) \).

Could be reduced with careful choice of upper bound.

11. (5 points) Does this binary tree satisfy the min heap property? Why or why not?

No. Children of 31 must be \( \geq 31 \), but they are not.