Dynamic Programming Group Exercises

Adapted from material by Cole Frederick

Please work the following problems in groups of 2 or 3. Use additional paper as needed, and staple the sheets together before turning them in. **ONLY TURN IN 1 WORKSHEET/ANSWERS PER TEAM.**

1. Dynamic programming can often be used to solve optimization problems that have 2 characteristics:
   
   a. __________________________ substructure

   b. __________________________ subproblems

2. Apply these characteristics to the shortest simple (i.e. no cycles) path problem: given a set of graph nodes, if we know that a path $p$ from A to B is a shortest path from A to B, and we know that C is a node somewhere along that path: Use proof by contradiction to show that the sub-paths of $p$ from A to C and, from C to B are also shortest paths between their endpoints.

   Therefore all possible sub-paths of a shortest path are themselves shortest paths.

3. Is merge sort a candidate for dynamic programming?  ☐ Yes  ☐ No
4. We want to compute the product of several matrices: $A_1A_2...A_n$ using the fewest scalar multiplications possible. The problem is to create an optimal parenthesization of this product. As an example, if $A_1$ is $10\times 100$, $A_2$ is $100\times 5$, and $A_3$ is $5\times 50$, $A_1A_2$ takes $10\times 100\times 5 = 5000$ scalar multiplications, then multiplying by $A_3$ is another $10\times 5\times 20 = 2500$ scalar multiplications. These numbers are for the $((A_1A_2)A_3)$ parenthesizing. If we start with $A_2A_3$, this is $100\times 5\times 50 = 25,000$ scalar multiplications plus $10\times 100\times 50 = 50,000$ to multiply by $A_1$. This parenthesization is $(A_1(A_2A_3))$. For b-d, consider the subset $A_i...A_j$, which we will cut at $A_k$. Answer the following questions:

a. How many ways are there to parenthesize 4 matrices $A_1A_2A_3A_4$? ________________

b. What would a recursion statement be for an optimal parenthesization? (Use $P(n)$ for the parenthesization recurrence function name.)

$$P(n) = \{ \}$$

c. How many distinct subproblems are there? ________________________________

e. What is the most number of choices we need to check for each subproblem?

__________________________________________________________________________
5. For the rod-cutting problem in the book, once we made a choice of cutting a length $i$ from the total length $n$, the subproblem was finding the optimum solution for what was left, $(n-i)$. Here we had to consider all choices of $i$ from 1 to $n$. For the matrix chain multiply parenthesization problem, there are 2 subproblems once we decide where to split the product, say $k$, and we need to optimally parenthesize from $A_i$ to $A_k$, and from where we split to $A_{k+1}$ to $A_j$. Then we need to choose $k$ between $i$ and $j$.

   a. These examples demonstrate something we can say about the complexity of dynamic programs:

   complexity depends on the product of ________________________________

   b. and ______________________________________________________.


6. For the rod-cutting problem in the book, pseudo-code for a divide and conquer recursive, memorized dynamic programming and bottom-up dynamic programming algorithm were given. The problem is to cut a rod of length \(n\) so that the maximum profit is gained by selling pieces of various lengths (i). Draw the call graphs for the 3 versions of the solution given below. The prices/length are:

<table>
<thead>
<tr>
<th>length (i)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>price (p_i)</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

\[
\text{CUT-ROD}(p, n) \\
1 \quad \text{if } n == 0 \\
2 \quad \quad \text{return } 0 \\
3 \quad q = -\infty \\
4 \quad \text{for } i = 1 \text{ to } n \\
5 \quad \quad q = \max(q, p[i] + \text{CUT-ROD}(p, n - i)) \\
6 \quad \text{return } q
\]

(a) recursive solution

\[
\text{MEMOIZED-CUT-ROD-AUX}(p, n, r) \\
1 \quad \text{if } r[n] \geq 0 \\
2 \quad \quad \text{return } r[n] \\
3 \quad \text{if } n == 0 \\
4 \quad \quad q = 0 \\
5 \quad \text{else } q = -\infty \\
6 \quad \text{for } i = 1 \text{ to } n \\
7 \quad \quad q = \max(q, p[i] + \text{MEMOIZED-CUT-ROD-AUX}(p, n - i, r)) \\
8 \quad r[n] = q \\
9 \quad \text{return } q
\]

(b) memoized DP solution

\[
\text{BOTTOM-UP-CUT-ROD}(p, n) \\
1 \quad \text{let } r[0 \ldots n] \text{ be a new array} \\
2 \quad r[0] = 0 \\
3 \quad \text{for } j = 1 \text{ to } n \\
4 \quad \quad q = -\infty \\
5 \quad \quad \text{for } i = 1 \text{ to } j \\
6 \quad \quad \quad q = \max(q, p[i] + r[j - i]) \\
7 \quad r[j] = q \\
8 \quad \text{return } r[n]
\]

(c) bottom-up DP solution
7. Dynamic programming often works bottom-up

   a. to find optimal solutions to ________________________________

   b. and then finds an optimal solution by making a __________________________ among subproblems to use in the solution.

8. Consider the shortest simple path between 2 vertices in a graph. This problem can be solved by dynamic programming assuming an intermediate node between them and finding the shortest path from the first node to this intermediate and from the intermediate to the last node. The longest simple path problem by contrast, may have nodes that appear in longest paths both to and from the intermediate node.

   a. The longest simple path subproblems share __________________________, which causes conflicts,

   b. and thus the subproblems are not __________________________. Therefore dynamic programming cannot be used to solve this problem.