This list of problems goes approximately from easier problems to harder problems. The first few problems are for warmup and easier than anything you might see on the exam, but don't panic if you can't do the last few problems.

Solutions to these problems, as with most DP problems, take the form of a recurrence relation, a short correctness proof of the recurrence relation (if it's not immediately obvious) and a running time analysis.

1. **Maximum value contiguous subsequence**: Given a sequence of n real numbers, $a_1, a_2, ..., a_n$, give an algorithm for finding a contiguous subsequence for which the value of the sum of the elements is maximized.

2. **Car Assembly** (see Figure 15.1 in CLRS): A factory has two assembly lines, each with n stations. A station is denoted by $S_{i,j}$ where i is either 1 or 2 and indicates the assembly line the station is on, and j indicates the number of the station. The time taken per station is denoted by $a_{i,j}$. A car chasis must pass through each of the n stations in order before exiting the factory. After it passes through station $S_{i,j}$ it will continue to station $S_{i,j+1}$ unless it decides to transfer to the other line. Continuing on the same line incurs no extra cost, but transfering from line i at station $j-1$ to station j on the other line takes time $t_{i,j}$. There is also an entry time e_i and exit time x_i which may be different for the two lines. Give an algorithm for computing the minimum time it will take to build a car chasis.

3. **Making change**: You are given n types of coins with values $v_1, ..., v_n$ and a cost C. You may assume $v_1 = 1$ so that it is always possible to make any cost. Give an algorithm for finding the smallest number of coins required to sum to C exactly.

   For example, assume you coins of values 1, 5, and 10. Then the smallest number of coins to make 26 is 4: 2 coins of value 10, 1 coin of value 5, and 1 coin of value 1.

4. **Box stacking**: You are given a set of boxes $\{b_1, ..., b_n\}$. Each box b_j has an associated width w_j, height h_j and depth d_j. Give an algorithm for creating the highest possible stack of boxes with the constraint that if box b_j is stacked on box b_i, the 2D base of b_i must be larger in both dimensions than the base of b_j. You can of course, rotate the boxes to decide which face is the base, but you can use each box only once.

   For example, given two boxes with $h_1 = 5, w_1 = 5, d_1 = 1$ and $h_2 = 4, w_2 = 5, h_2 = 2$, you should orient box 1 so that it has a base of 5x5 and a height of 1 and stack box 2 on top of it oriented so that it has a height of 5 for a total stack height of 6.

5. **Balanced Paritions**: Suppose you are given an array of n integers $\{a_1, ..., a_n\}$ between 0 and M. Give an algorithm for dividing these integers into two sets x and y such that $|\sum_{x_i \in x} x_i - \sum_{y_i \in y} y_i|$, the difference of the sum of the integers in each set, is minimized. For example, given the set $\{2, 3, 2, 7, 9\}$, you can divide it into $\{2, 2, 7\}$ (sums to 11) and $\{3, 9\}$ (sums to 12) for a difference of 1.

6. **Boolean parenthesizations**: You are given a boolean expression consisting of a string of the symbols TRUE, FALSE, AND, OR, and XOR. Give an algorithm for finding the number of ways to parenthesize the expression such that it will evaluate to TRUE. For example, there is only 1 way to parenthesize FALSE AND TRUE XOR TRUE such that it evaluates to TRUE: (FALSE AND TRUE) XOR TRUE. (XOR is exclusive or: F XOR F = F, T XOR F = T, F XOR T = T, T XOR T = F)

7. **Edit Distance**: Given two text strings $A = a_1a_2...a_n$ of length n and $B = b_1b_2...b_m$ of length m, you want to transform A into B with a minimum number of operations of the following types: delete a character from A, insert a character into A, or change some character in A into a new

character. The minimal number of such operations required to transform A into B is called the edit distance between A and B. Give an algorithm for finding the edit distance from A to B.

8. **Building Bridges**: Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a_1...a_n$ and n cities on the northern bank with x-coordinates $b_1...b_n$. The cities on each bank are also numbered 1 through n and these numbers do *not* correspond to the ordering of the x-coordinates. You can only build a bridge from a city on the south bank to a city on the north bank with the same number. No two bridges may cross each other. An example of a valid bridge building is shown in Figure 1. Give an algorithm for finding the maximum number of bridges that can be built.

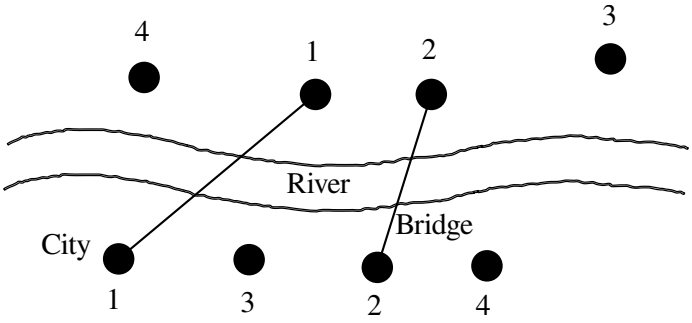

Figure 1: An example of a valid bridge building.

9. **Function Approximation**: Assume a function $f(x)$ generated a sequence of n points in the plane (x_i, y_i). Given an integer k, we choose a subset of $k+1$ points and order them according to increasing x-coordinate. This subset must include the first point (x_1, y_1) (smallest x-coordinate) and the last point (x_n, y_n) (largest x-coordinate). We define the function $g(x)$ as the straight line segments connecting each point to the next point in the set so $g(x)$ consists of k line segments. The error is then defined as

$$e = \sum_{i=1}^{n} (y_i - g(x_i))^2. \tag{1}$$

Give an algorithm that takes the n points and k as input and minimizes the error.

10. **Two-Person Traversal of a Sequence of Cities**: You are given an ordered sequence of n cities, and the distances between every pair of cities. Design an algorithm to partition the cities into two subsequences (not necessarily contiguous) such that person A visits all cities in the first subsequence (in order), person B visits all cities in the second subsequence (in order), and the sum of the total distances travelled by A and B is minimized. Assume that person A and person B start initially at the first city in their respective subsequences.

11. **Bin Packing**: You have n_1 items of size s_1, n_2 items of size s_2, and n_3 items of size s_3. Design an algorithm to pack all of these items into bins each of capacity C, with $C \geq s_3 \geq s_2 \geq s_1$, such that the total number of bins used is minimized.

12. **Scheduling**: Suppose you have one machine and n jobs, $a_1, ..., a_n$. Each job a_j has processing time t_j, profit p_j, and deadline d_j. The machine can only process one job at a time and that job must run uninterruptedly until completion. If job a_j is completed by deadline d_j, you receive profit p_j, but if it is completed after, you receive nothing. Assuming all processing times are integers between 1 and n and $d_j \geq t_j$ for all jobs, give an algorithm for computing the maximum profit you can make.

13. **Bitonic Tours**: You are given a set of points (x_i, y_i) in a two dimensional plane. You start at the point with the lowest x coordinate and must move strictly right until you reach the point with the highest x coordinate, after which you must move strictly left until you return to the starting point. Give an algorithm for determining the shortest distance you can travel.