Dynamic Programming

The best way to get better at dynamic programming is a lot of practice. There are a lot of different types of dynamic programming problems and it gets easier to solve if you are familiar with patterns that you've noticed from other dynamic programming problems that you have solved. Googling "dynamic programming problems" gives plenty of resources with several practice problems to work through, along with past pset/exam dynamic programming problems.

Basics

Main steps of solving a dynamic programming:

1. Define the subproblems
2. Find how to combine solutions to subproblems to form solutions to larger subproblems
3. Choose an order to solve subproblems so the problems they depend on are solved before combining solutions

Make sure that these three steps are satisfied when giving a dynamic programming solution

Combining solutions

Often figuring out how to combine solutions helps lead to correct subproblem definitions. There are a few major ways to combine solutions

Prefix/Suffix - Start with solving smaller subproblems from beginning (or end) and expand subproblems towards the end (or the beginning). Examples: Assembly line, Crazy 8's, Piano fingering

Substring - Start with solving smaller subproblems substrings and expand outwards. We need to use this in problems where we can't take advantage of prefix/suffix structures. Examples: Multiplication parentheses, Two person coin game (Consider a row of n coins of values v(1) ... v(n), where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.)

Combination - If you're storing solutions in some multi-dimensional table, you often combine solutions from other cells. The order may be a combination of prefix/suffix/substring with each dimension of the table using a different structure. Examples: DNA alignment, Edit distance

Tree - Start with solving smaller subproblems at the leaves and expand upwards, level by level, until you solve the the subproblem at the root. Examples: Phylogeny, Vertex cover in tree, railroad gauge

Adding constraints

A common mistake is to define the subproblem without enough constraints. If you're having trouble working out a dynamic programming solution, look for other parameters that you might be able to fit into the subproblem to constrain it further. Extra constraints may make combining subproblems more straightforward and be the key to solving a dynamic programming problem. It may add more runtime, but sometimes we have to add constraints to the subproblems and a slower correct solution will often get more points than an incorrect solution anyway.

Examples: WIdget layout (added width constraint), Knapsack problem (added size constraint)

Running time

Calculating the runtime of a dynamic programming solution involves computing how many subproblems there are and how long it takes to solve each subproblem. The runtime is usually (Time per subproblem) $\times$ (Number of subproblems).