Advanced Dynamic Programming Techniques

Jacob Steinhardt

The whole point of this lecture is to teach you about some advanced techniques in dynamic programming. If you are familiar with all of the basic DP techniques, then this should be an easy lecture. If you aren't, then you should be at the beginning lecture.

1 Technique 1: Keep Track of All Possible Transition States

Example problem: Given a 12 by 12 grid of squares, some of which are marked as unusable, determine the number of ways to select squares such that no two selected squares are adjacent (and no unusable squares are selected).

Solution: DP row-by-row, with the state being the row number and which squares you chose in the previous row.

2 Technique 2: Dynamic Greedy

Example problem: Given n cows, each with a certain weight and strength, arrange the cows in a tower such that the maximum stress of any cow is minimized, where stress of a cow is equal to the weight of the cows above her minus that cow's strength.

Solution: Look at two adjacent cows. When is it better to switch their positions? Use this to create a comparison operator with which to sort the cows.

Practice problem: In the above problem, what is the maximum number of cows we can have in a tower if we require that the weight a cow carries never exceeds her strength? Your algorithm should be $O(numcows * maxstrength)$ at worst. There is, in fact, an $O(numcows^2)$ solution that you should try to find.

3 Technique 3: Steady State Convergence

Example problem: You have a circular game board where each square has some integer value. The sum of the values on the board is negative. On each move, you can either stop playing and walk away with your current score, or roll a 6-sided die, in which case you go that many squares forward and your score increases by the value on the square you land on (which may be negative). If you play optimally, what is your expected winnings?

Solution: The equation $e(i) = v(i) + max(0, \frac{e(i+1)+...+e(i+6)}{6})$ holds for all i. If we start with random values for e_i and then repeatedly apply the above recursive equation, then our values will converge exponentially on the correct values.