Dynamic Programming Proofs

Dynamic programming proofs are can be straightforward and follow a standard pattern. Typically DP algorithms are based on a recurrence relation involving the optimal solution so the correctness proof will focus on justifying why the recurrence is correct.

Main Steps

Six steps to a dynamic programming algorithm proof of correctness.

Step 1: Define your sub-problem. Describe in English what your sub-problem means, whether it looks like $P(k)$ or $R(i,j)$ or anything else. For example, you might write "Let $S(k)$ be the largest number of items that can fit into a knapsack".

Step 2: Write a recurrence. Give a mathematical definition of your sub-problem in terms of "smaller" sub-problems.

Step 3: State your base cases. Sometimes only one or two or three bases cases are needed, and sometimes you'll need a lot (say $O(n)$). The latter case typically comes up when dealing with multi-variate sub-problems. You want to make sure that the base cases are enough to get your algorithm off the ground.

Step 4: Prove that your recurrence is correct. This is equivalent to arguing your inductive step in your proof of correctness. Go case by case and prove each case is correct.

Step 5: Present the algorithm/ Prove the Algorithm Evaluates the Recurrence. This often involves initializing the base cases and then using your recurrence to solve all the remaining sub-problems. You want to ensure that by filling in your table of sub-problems in the correct order, you can compute all the required solutions. Finally, generate the desired solution. Often this is the solution to one of your sub-problems, but not always.

Step 6: Prove the Algorithm is Correct. Having Shown that the recurrence has been evaluated correctly, your algorithm will probably conclude with something like "return C(A)" or "return L[m,n]". prove that this is the table value that you actually want.