Conceptual Algorithmic Template for Deterministic Dynamic Programming

Suppose we have T stages and S states. We set up two-dimensional arrays **f** and **x** such that

f[t, i] holds the value of being in state i and time stage t (should be able to accept values of t up to $T + 1$)

x[t, i] holds the best decision to take in state i and time stage t

The optimization procedure may then be organized along the following general lines

Set up or read in problem data

Set up two-dimensional arrays for **x** and **f**

For each possible state i, set **f**[$T + 1$, i] to be the value of being in state i at the end of planning horizon. Depending on the problem and i, this value might be

- Zero (which means we don't need any code if we initialized the **f** array to zeroes)
- Infinity for disallowed ending states ($+\infty$ for minimization, $-\infty$ for maximization)
- Some "salvage" value (for example, what you could get by selling off excess inventory to a discounter at the end of the time horizon)

Loop over stages $t = T, T-1, \ldots, 1$ (backward!)

Loop over all possible states i

Set the current state value to be $+\infty$ for minimization, $-\infty$ for maximization

Determine which decision moves are possible from stage t, state i

Loop over all decisions d that are possible from this state

Evaluate the value of each decision d using the dynamic programing recursion formula. This typically means

Find the next state j implied by the action d
Compute all profits/costs for the current stage
Value of the decision d is (current profits/costs) + **f**[$t + 1$, j]

If the decision d improves on the best seen, record its value and the decision d

Store the corresponding optimal value for stage t, state i in **f**[t, i]
Store the corresponding best decision in **x**[t, i]

When done, the optimal solution value is in **f**[1, i_0], where i_0 is the initial state

To output the optimal sequence of decisions, start by setting i to the initial state, then trace forward through the optimal sequence of states as follows:

Loop over stages $t = 1, 2, \ldots, T$ (forward, this time)

Output the optimal decision **x**[t, i]

Overwrite i with the optimal state at time $t + 1$, computed from i and **x**[t, i]