# Dynamic Programming 2

TJHSST Senior Computer Team

October 17, 2003

## 1 Introduction

*"It's the question, programmer. It's the question that drives us. It's the question that brought you here. You know the question, just as I did."*
*"What is DP?"*
-Adapted from "The Matrix"

You've surely heard of DP (Dynamic Programming), one of the more powerful algorithmic weapons in computer science. However, it is essential to wield this weapon wisely - that is, to be able to accurately recognize a problem as having a DP solution and find a recurrence in the problem. To do this, let's look over the idea that forms the foundation of DP.

## 2 The Basics

*"What are you trying to tell me? That I can write an $O(N^2)$ recursive solution for 2-dimensional knapsack?"*
*"No, programmer. I'm trying to tell you that when you're ready, you won't have to."*
-Adapted from "The Matrix"

The most basic element of a DP solution is a *recurrence*, which is when a function's value is defined in terms of previous values of the function. To look at the familiar Fibonacci sequence, if $F(n)$ is the $n$-th Fibonacci number, then

$$F(n) = F_(n-2) + F(n-1).$$

This is a very useful relationship, but we clearly need more information to calculate the sequence. After all, we never bother to define the numerical value of a single term. Therefore, the second element of any DP solution is a *base case*, an initial value for the function. Again returning to the Fibonacci sequence, our base case is

$$F(0) = 0, \quad F(1) = 1.$$

Now let's look at this in the context of another standard problem:

**Buy Low, Buy Lower [USACO Fall '98]**
Given a sequence $a_i$ of $N$ integers, find the longest subsequence of decreasing prices.

We define the function $F(x)$ to be the longest subsequence of decreasing prices ending on day $x$. Therefore, our base case is

$$F(1) = 1,$$

and our recurrence is

$$F(x) = \max_{i<x, a_i > a_x} (F(i) + 1).$$

# 3 Do It Yourself

> *"Unfortunately, no one can be told what DP is. You have to do it yourself."*
> -Adapted from "The Matrix"

Now's the chance for you to solve some tender and/or juicy DP problems. We'll just worry about the base case and recurrences, because you should be able to code up complete DP solutions from those alone.

**Money Systems [USACO Training Pages]**
Given $V$ different coin values, compute the number of ways to construct a value $N$, assuming an infinite supply of all coin values.

$F(x)$ = the number of ways to construct a value of $x$.
Base Case: $F(0) =$
Recurrence: $F(x) =$

**A Game [IOI '96]**
A game is played between two players, with $N$ integers laid out in a straight line. Players alternate turns, choosing a number from either the left or right side of the line. The number is then removed from the line and added to the player's score. Compute the optimal score for the first player, assuming both players play optimally.

$F(x, y)$ = the optimal score obtainable by the current player if the integers in positions $[x..y]$ remain.
(Hint): $T(x, y)$ = the total of the integers in positions $[x..y]$.
Base Case: $F(1, N) =$
Recurrence: $F(x, y) =$

*"Stop trying to solve it and solve it!*
-Adapted from "The Matrix"

**Subset Sums [USACO Training Pages]**
Compute the number of ways to partition the integers from 1 to $N$ into two sets such that the sum of both sets is the same.

$F(x, y)$ = the number of ways to partition the integers from 1 to $x$ into two sets such that the difference between the sums of the two sets is $y$.
Base Case: $F(1, 1) =$
Recurrence: $F(x, y) =$

**Big Barn [USACO Training Pages]**
Given an $NxN$ grid of 0s and 1s, find the largest square which contains no 1s.

$F(x, y)$ = the largest square whose top left corner is at $(x, y)$.
Base Case 1: If $(x, y)$ contains a 1, $F(x, y) =$
Base Case 2: If $(x, y)$ is on the bottom or right sides, $F(x, y) =$
Recurrence: $F(x, y) =$

*"You have to let it all go, programmer. Fear, doubt, and disbelief. Free your mind."*
-Adapted from "The Matrix"

# 4 Parting Words

No lecture can ever completely teach DP. Now that you have an understanding of the principles, you should try to find as many problems as possible that you can practice on. It is only through solving as many DP problems as possible that one can truly become a DP expert.

*"I can only show you the door. You're the one that has to walk through it"*
-Adapted from "The Matrix"