`#include <senior week 6>`

Dynamic Programming

Dynamic Programming (often referred to as DP) is a problem solving technique where we break down the problem into smaller and smaller cases, until we reach a base case.

When is DP useful?

DP is usually useful when either:

1. Overlapping subproblems. When breaking down a problem, one may access the same subproblem multiple times; in this case, we can compute that subproblem only once.

2. Substructure. That is, we can break it down into multiple subproblems and combine them together efficiently to reach the original problem. For example, finding the largest element in a list; I can divide the list in half, find the largest of each half, and using this info I can figure out the largest element. However, this wouldn't work in some graph theory questions—DP pretty much only knows the position you are*, so repeating an edge can be a worry.

*We can cheat this a little by keeping track of multiple states. Will do next week or the week after

DP solutions usually have 3 components:

1. Base case(s). This is the simplest case, where we build up our dp. Also usually the easiest part.
2. State. We'll talk about this next week.
3. Transition. This is how we can transition from smaller cases to larger cases.

Example: Fibonacci

Recall the Fibonacci formula:

$$F_n = F_{n-1} + F_{n-2}$$

This was probably one of the examples used when you learned recursion. However, recursion is actually very slow for this problem, as it solves subproblems a lot.
This is where DP comes in. We can build up the Fibonacci sequence and have it as a list; that way, we only have to compute subproblems once.

Here, we can solve our base case easily: $F_0 = 0, F_1 = 1$. As for the transition, it's given to us from the formula.

```
int fib[]
fib[0] = 0; fib[1] = 1;
for (int i = 2; i < n; i ++){
      fib[i] = fib[i-1]+fib[i-2];
}
```

The best way to get a feeling for DP is to solve problems. Paper is cool (maybe).

These are all harder than the examples: Keep in mind the 3 components of DP (2 for now).
What is the base case?
How do I transition from smaller cases to larger cases?
CCC '07 J5 - Keep on Truckin' - DMOJ: Modern Online Judge
Educational DP Contest AtCoder B - Frog 2 - DMOJ: Modern Online Judge
CCC '12 S5 - Mouse Journey - DMOJ: Modern Online Judge (You'll need 2d for this one)

IOI '94 P1 - The Triangle - DMOJ: Modern Online Judge (solve this one after you solve the first 2)