Dynamic Programming

- Solve subproblems in a recursive manner.
- Memorize the answer to the subproblems.
- Optimal substructure.

**Fibonacci Sequence**

\[ f_1 = f_2 = 1 \quad f_n = f_{n-1} + f_{n-2} \quad 1, 1, 2, 3, 5, 8, 13, \ldots \]

**Input**: \( n \)

**Output**: \( f(n) \)

**Attempt 1**

```c
int fib(int n) {
    if (n==0) return 1;
    return fib(n-1) + fib(n-2);
}
```

```
Runtime = \Omega(f_n) \approx \Omega(1.618^n)
```

**Attempt 2**

```c
int fib(int n) {
    if (n>0) return f[n];
    int ans = fib(n-1) + fib(n-2);
    f[n] = ans;
    return ans;
}
```

```
Runtime = \Theta(n)
```

**Proof**: every time we run line 2-4 in \( \text{fib()} \), one entry in the array \( f \) changes from negative to positive. \( \Rightarrow \) \( O(n) \) time in total.

We call \( \text{fib()} \) \( O(n) \) times in total

Called once in \( \text{main()} \) and twice on line 2.