Dynamic Programming

1) Greedy Method → decision is taken one time

2) Dynamic Programming → decision taken at each step

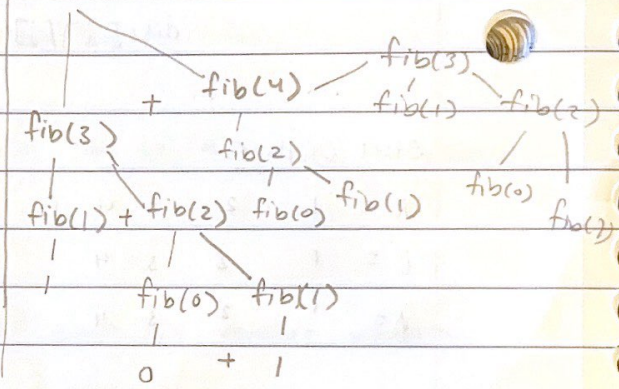
↑ principle of optimality

ex:

$$fib(n) = \begin{cases} 0 \\ 1 \\ fib(n-2) + fib(n-1) \text{ if } n > 1 \end{cases}$$

0, 1, 1, 2, 3, 5, 8, 13, 21

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



$T(N) = 2T(N-1) + 1$

$O(2^N)$

Reduce function calls of fib(1) & fib(2). Since they keep getting called increasing our big O.

F	0	1	-1	-1	-1	-1
	0	1	2	3	4	5

Start out at (-1) for all keep going through the fib(N) graph solving all of these to save time complexity when solving!.

F	0	1	1	2	3	5
	0	1	2	3	4	5

memoization has made it from $O(2^N)$ to $O(N)$. 15 calls to just 6.

memoization follows top down approach.

Tabulation

F	0	1	1	2	3	5
	0	1	2	3	4	5

$$fib(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ fib(n-2) + fib(n-1) & \text{if } n>1 \end{cases}$$

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

fib(5)? = 5 ✓

Table is generated. Using bottom up approach.

(Review slide Notes for Dynamic Programming!)

BOTH Tabulation and Memoization can be used, but most will be solved using Tabulation!