14. Dynamic Programming

Wednesday, June 8, 2022 8:33 PM

- Optimization technique using caching
  - Solve Problem by dividing into subproblems, solving each subproblem once and storing their solution in case the same subproblem occurs
  - Divide & conquer + Memoization

Memoization == caching

- Storing values to use later on
- caching is just a way to speed up programs by holding data in an easily accessible box
- analogous to bringing a backpack to school, so you don't have to go home to retrieve items

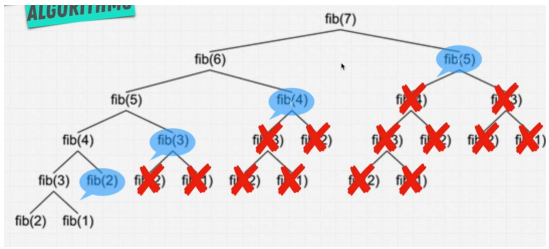
Memoization

```
if ( input in cache)
    return cache[n]
else
    calculation
    cache[n] = ...
    return cache[n]
```

Simply remembering a solution to a subproblem and if subproblem return remembered solution

D.P.

1) Can be divided into subproblems
2) Recursive solution
3) Are there repetitive subproblems?
4) Memoize subproblems



original

fib w/ DP.

- original case is $O(2^n)$

```
public int fib(int n) {

if (n < 2){
   return n;
}

 return fib(n-1) + fib(n-2);
}
```

- DP is $O(n)$

Top-Down

```
class Solution {
   public int fib(int n) {
      return fibHelper(n, new int[n+1]);
   }

   private int fibHelper(int n, int[] memo) {
      if (n < 2 ) {
         return n;
      }

      // Check to see if value is in cache, since array indexes default value to 0
      if (memo[n] != 0) {
         return memo[n];
      }
      else {
         memo[n] = fibHelper(n-1, memo) + fibHelper(n-2, memo);
         return memo[n];
      }
   }
}
}
```

w/ outside memo declaration

not mine ->

```
class Solution {
   //dp memoization
   Map<Integer, Integer> cache = new HashMap<>();
   public int fib(int n) {
      if(n <= 0) return 0;
      if(n == 1) return 1;
      if(cache.containsKey(n)){ return cache.get(n);}
      else{
        int fibn =  fib(n - 1) + fib(n-2);
        cache.put(n, fibn);
        return fibn;
      }
   }
}
```

Bottom - up (iterative)

```
class Solution {
```

Basically:

Start from simpler solution and work your way up higher and higher towards the complex solutions

```
class Solution {
  public int fib(int n) {
    if (n < 2) {
      return n;
    }

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

    return memo[n];

  }
}
```