Dynamic Programming:

* It is bottom-up approach.
* Used when given problem can be broken down to iterative sub problems.
  + Similar to divide and conquer however where as in divide and conquer if a problem is divided they are independent of each other. In dynamic programming problems depend on answers of its previous problem -> needs ordering of problems, which one gets executed first.
* 3 steps:

1. Split into subproblems
2. Solve from smallest to largest subproblems and store into a table to avoid computing same problem again.
   1. What is considered small, based on what measurement?
3. Going through table created move bottom-up

* Not all questions can be approached using dynamic programming approach, the problem needs to be in optimal substructure
  + Optimal substructure: where largest subproblem is answered by using answers from smaller subproblems.
  + If it does not satisfy optimal substructure condition it means that previous answers from previous problems are not being used then there Is no need to store answers in a table since it would be waste of space therefore in that case dynamic programming wouldn’t be most efficient.

MIT lecture:

* Pre cursor of bellman ford algorithm
* “Dynamic programming” termed by Bellman Ford.

Tabulation:

* Bottom up approach

Memoization:

* Whenever we compute a subproblem store it into a dictionary, if it already is in the dictionary just return it.

Q:

* Applications of Fibonacci number?
* When to use memorization over tabulation?

Tabulation avoid multiple lookups therefore save function call overhead time however it needs to compute all subproblems.

Memoization: only solve subproblems that are needed.

Eg: Longest Common Subsequence

More intuitive to write e.g: matrix chain multiplication