Dynamic Programming

Notes for Interviews

Dynamic programming: using calculated results to formulate the result

* **Steps to solve a DP problem**:
  1. **Identify if it is a DP problem**
  2. Decide a **state expression** with least parameters
  3. Formulate **state relationship**
  4. Do tabulation (or add memoization)
* Identify if it is a DP problem: typically, optimization & counting problems. Typical properties:
  1. **Overlapping subproblems property**: (always) problem can be solved by solving sub-problems
  2. **Optimal substructure property**: (often) optimal solution of the given problem can be obtained by using optimal solutions of its subproblems
* Decide a **state expression** with least parameters
  1. Careful: the state transition depends on the **choice of state definition** you make (= what uniquely identifies a subproblem). States should use least parameters possible to reduce state space (e.g. in knapsack, describe using index & weight)
* **Tabulation VS memorization**:
  + **Tabulation**: Bottom-Up. Start our transition from our base state i.e dp[0] and follow our state transition relation to reach our destination state dp[n].
    - More efficient if all sub-problems must be solved at least once
  + **Memoization**: Top-Down. Start our journey from the top-most destination state and compute its answer by taking in count the values of states that can reach the destination state (storing each sub-result), till we reach the bottom-most base state.
    - Slower due to recursive calls
    - Unlike tabulation, not all entries filled – entried are filled on demand