Distinguishing Between Dynamic Programming and Binary Search on Answer
----

It's great that you recognize "minimum" or "maximum" time as key hints for optimization problems. The cha

Use **Dynamic Programming (DP)** When:
1. **Overlapping subproblems & optimal substructure exist**
   - Example: Shortest path in a grid ? Solve smaller subproblems and combine results.
   - Example: DP on strings (edit distance, LCS) ? Break into smaller string problems.

2. **You need to explore all possible ways**
   - Example: Knapsack problem ? Try including and excluding each item.
   - Example: DP on subsets (partitioning, sums, etc.).

3. **The problem has a clear recurrence relation**
   - Example: "To find the best result for dp[i], use previous dp[j] values."
   - Example: "State dp[i] depends on previous states like dp[i-1]."

Use **Binary Search on Answer** When:
1. **The answer lies in a sorted range (monotonic property exists)**
   - Example: "Find the minimum time X where all tasks can be done."
   - If X=5 works, then X=6 also works ? Monotonic property.

2. **You have a function isPossible(X) to check feasibility**
   - Example: "Can we process all tasks in X time?"
   - Example: "Can we ship packages with X weight per day?"
   - If isPossible(X) is easy to implement, binary search is a good fit.

3. **You?re asked for the minimum or maximum valid value**
   - Example: "Minimum largest sum in subarrays."
   - Example: "Maximum number of tasks that can be completed."

Quick Test to Decide:
Ask yourself:

? "Can I write a function isPossible(X) to check if X is a valid answer?" ? **Binary Search on Answer**

? "Do I need to break the problem into smaller states?" ? **Dynamic Programming**

Since you naturally lean toward DP, try practicing problems where binary search is more suitable (e.g., Lee