How to Identify This Pattern?

The key observation here is that the problem asks for a **minimum number of time units** to complete all t

1. The answer lies in a range -> 1 to n (where n is the number of processes).
2. If it's possible to do it in m time units, it should also be possible in m+1 (monotonic property).
3. Binary search efficiently finds the minimum m where it's possible.

General Steps to Recognize This Pattern

1. Does the problem ask for a "minimum" or "maximum" value that satisfies a condition?

   - Not "find exact match" -> Binary search doesn't help.
   - "Find the smallest X such that condition holds" -> Binary search on answer is useful.
   - "Find the largest X such that condition holds" -> Binary search on answer is useful.

2. Can we check if a given X is valid?

   - If there is a clear function isPossible(X) to check whether X is a valid answer, binary search works.
   - This function should be monotonic (if X works, then X+1 must also work OR vice versa).

3. Does brute force take too long?

   - If X can be anywhere between 1 and N (large N), brute force is too slow.
   - Binary search reduces the checks from O(N) to O(log N).

How to Find Patterns in Future Problems?

Here are some common problems where this pattern applies:

1. "Minimum" or "Maximum" Feasibility Search

- Example: What is the minimum time required to complete all tasks?
  Solution: Binary search on time -> Check feasibility.

- Example: What is the maximum weight a bridge can hold before collapsing?
  Solution: Binary search on weight.

---

2. "Minimax" Optimization Problems

- Example: Split an array into k subarrays such that the maximum sum of any subarray is minimized.
  Solution: Binary search on subarray sum -> Check feasibility.

- Example: Find the minimum capacity of a container to store N items in K days.
  Solution: Binary search on capacity.

---

3. "Fair Distribution" or "Resource Allocation"

- Example: Assign jobs to workers so that the maximum workload is minimized.
  Solution: Binary search on maximum workload.

- Example: Find the least number of days to complete all tasks given worker constraints.
  Solution: Binary search on days.

How to Practice This Pattern?

Since you're already good at dynamic programming and sorting, this type of binary search on answer can b

1. Recognize the structure in problems -> Look for "minimum," "maximum," "smallest possible X," or "large
2. Try these problems on Leetcode:
   - Split Array Largest Sum (Leetcode 410)
   - Capacity To Ship Packages Within D Days (Leetcode 1011)
   - Aggressive Cows (SPOJ - AGGRCOW)
   - Allocate Minimum Number of Pages (GFG)
3. Keep solving problems that require resource allocation, task scheduling, or feasibility checks.

Final Thoughts

- You successfully implemented a binary search on the answer.
- You optimized it by sorting processSize and capacity beforehand.
- You understand the greedy allocation strategy that ensures correctness.

Now, the next step is to recognize similar patterns in future problems and solve them faster! Keep up the g