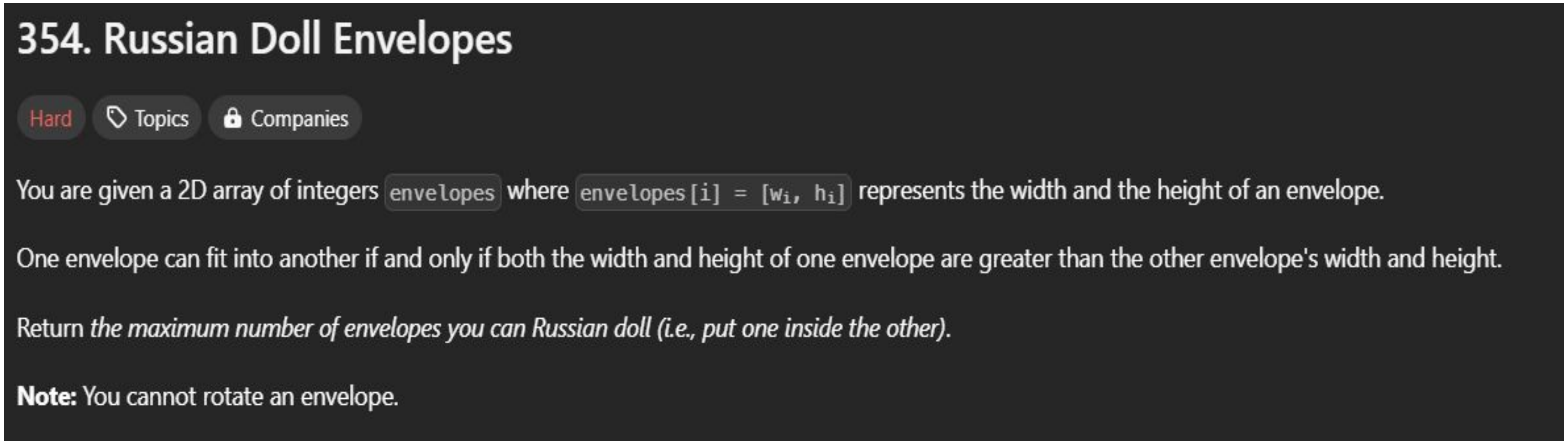
354. Russian Doll Envelopes

BY ROHAN GAJMER

Problem Explanation



354. Russian Doll Envelopes

Hard | Topics | Companies

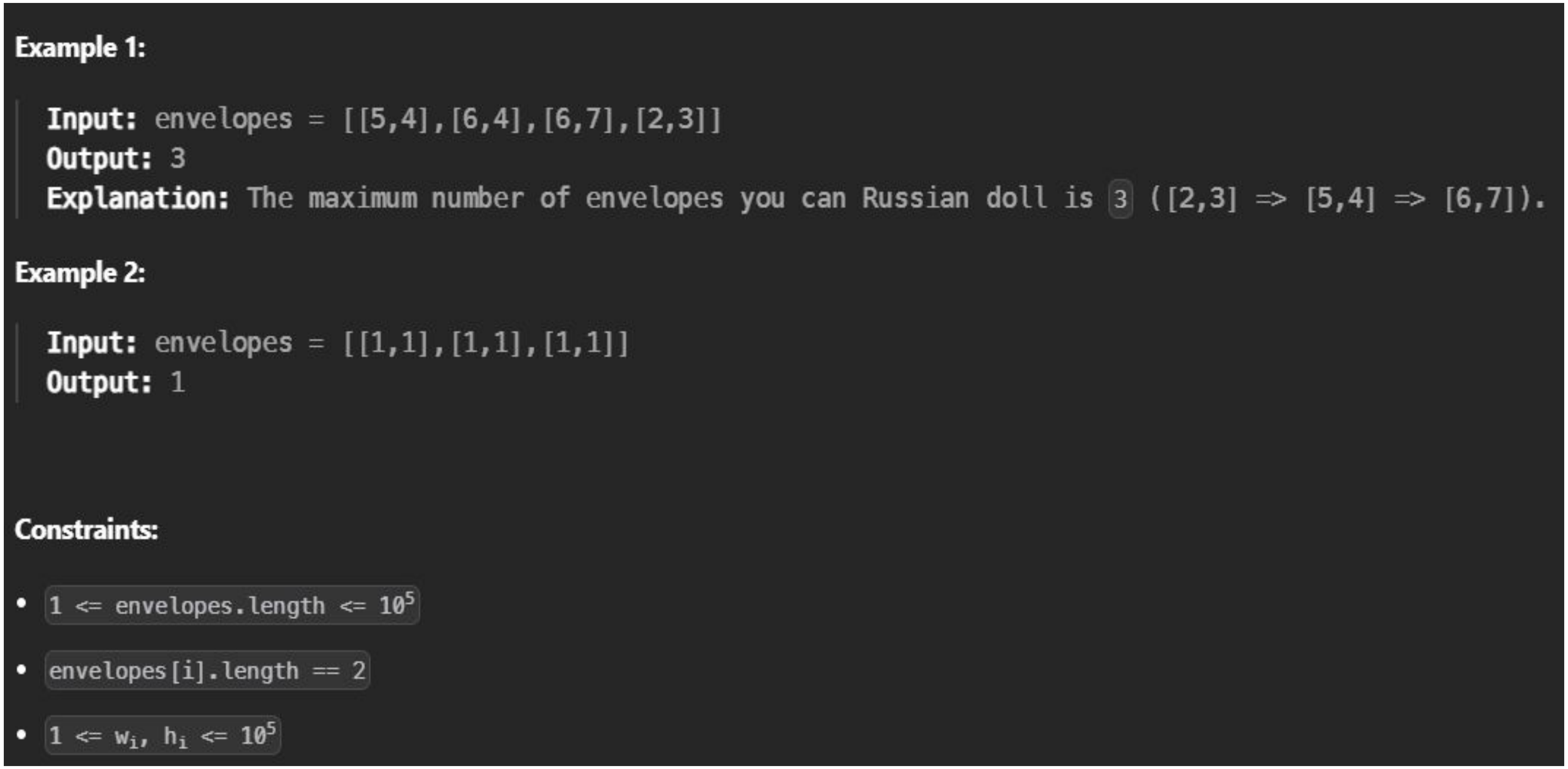
You are given a 2D array of integers `envelopes` where `envelopes[i] = [wi, hi]` represents the width and the height of an envelope.

One envelope can fit into another if and only if both the width and height of one envelope are greater than the other envelope's width and height.

Return *the maximum number of envelopes you can Russian doll (i.e., put one inside the other).*

Note: You cannot rotate an envelope.

Russian Doll Envelope



Example 1:

```
Input: envelopes = [[5,4],[6,4],[6,7],[2,3]]
Output: 3
Explanation: The maximum number of envelopes you can Russian doll is 3 ([2,3] => [5,4] => [6,7]).
```

Example 2:

```
Input: envelopes = [[1,1],[1,1],[1,1]]
Output: 1
```

Constraints:

- `1 <= envelopes.length <=` 10^5
- `envelopes[i].length == 2`
- `1 <=` w_i, h_i `<=` 10^5

Why This Problem?

Topic covered: ArrayBinary,SearchDynamic,
ProgrammingSorting

- - Combines sorting and dynamic programming concepts.
- - Involves multi-dimensional constraints.
- - Highlights a cool optimization using binary search.
- - Encourages discussion on 2D-to-1D problem translation.

Approach to Solve

- 1. Sort envelopes by width (ascending), then by height (descending) for duplicates.
- 2. Extract heights and find the Longest Increasing Subsequence (LIS).
-    - Use binary search for LIS to achieve O(n log n) time.
- 3. Return the length of LIS as the result.

Challenges and Insights

- - Roadblocks:
- - Sorting by descending height for same widths.
- - Translating 2D problem to 1D LIS.

- - Insights:
- - Binary search approach significantly improves efficiency.
- - Sorting decisions are critical for correct LIS.

Time Complexity with DP LIS

- Sorting: O(nlogn)
- DP LIS: O(n^2).

Without binary search, the LIS algorithm has **$O(n^2)$** complexity.

Binary search optimizes the process of finding or replacing elements in the LIS, reducing it to O(nlogn).

Coded Solution in Python

```
• def maxEnvelopes(envelopes):
•     envelopes.sort(key=lambda x: (x[0], -x[1]))
•     heights = [h[1] for h in envelopes]
•     def LIS(arr):
•         dp = []
•         for x in arr:
•             i = bisect_left(dp, x)
•             if i == len(dp): dp.append(x)
•             else: dp[i] = x
•         return len(dp)
•     return LIS(heights)
```