Problem 3540: Minimum Time to Visit All Houses

Problem Information

Difficulty: Medium
Acceptance Rate: 68.77%
Paid Only: Yes
Tags: Array, Prefix Sum

Problem Description

You are given two integer arrays `forward` and `backward`, both of size `n`. You are also given another integer array `queries`.

There are `n` houses _arranged in a circle_. The houses are connected via roads in a special arrangement:

* For all `0 <= i <= n - 2`, house `i` is connected to house `i + 1` via a road with length `forward[i]` meters. Additionally, house `n - 1` is connected back to house 0 via a road with length `forward[n - 1]` meters, completing the circle. * For all `1 <= i <= n - 1`, house `i` is connected to house `i - 1` via a road with length `backward[i]` meters. Additionally, house 0 is connected back to house `n - 1` via a road with length `backward[0]` meters, completing the circle.

You can walk at a pace of **one** meter per second. Starting from house 0, find the **minimum** time taken to visit each house in the order specified by `queries`.

Return the **minimum** total time taken to visit the houses.

Example 1:

Input: forward = [1,4,4], backward = [4,1,2], queries = [1,2,0,2]

Output: 12

Explanation:

The path followed is `_0_(0) -> _1_(1) ->■■■■■■■ _2_(5) _->_ 1(7) _->_ _0_(8) _->_ _2_(12)`.

Note: The notation used is `node(total time)`, `->` represents forward road, and `_->_` represents backward road.

Example 2:

Input: forward = [1,1,1,1], backward = [2,2,2,2], queries = [1,2,3,0]

Output: 4

Explanation:

The path travelled is `_0_ ->■■■■■■■ _1_ ->■■■■■■■ _2_ ->■■■■■■■ _3_ -> _0_`. Each step is in the forward direction and requires 1 second.

Constraints:

* `2 <= n <= 105` * `n == forward.length == backward.length` * `1 <= forward[i], backward[i] <= 105` * `1 <= queries.length <= 105` * `0 <= queries[i] < n` * `queries[i] != queries[i + 1]` * `queries[0]` is not 0.

Code Snippets

C++:

```
class Solution {
public:
long long minTotalTime(vector<int>& forward, vector<int>& backward,
vector<int>& queries) {

}
};
```

Java:

```
class Solution {
public long minTotalTime(int[] forward, int[] backward, int[] queries) {
}
}
```

Python3:

```
class Solution:
def minTotalTime(self, forward: List[int], backward: List[int], queries:
List[int]) -> int:
```