Problem 656: Coin Path

Problem Information

Difficulty: Hard
Acceptance Rate: 34.15%
Paid Only: Yes
Tags: Array, Dynamic Programming

Problem Description

You are given an integer array `coins` (**1-indexed**) of length `n` and an integer `maxJump`. You can jump to any index `i` of the array `coins` if `coins[i] != -1` and you have to pay `coins[i]` when you visit index `i`. In addition to that, if you are currently at index `i`, you can only jump to any index `i + k` where `i + k <= n` and `k` is a value in the range `[1, maxJump]`.

You are initially positioned at index `1` (`coins[1]` is not `-1`). You want to find the path that reaches index n with the minimum cost.

Return an integer array of the indices that you will visit in order so that you can reach index n with the minimum cost. If there are multiple paths with the same cost, return the **lexicographically smallest** such path. If it is not possible to reach index n, return an empty array.

A path `p1 = [Pa1, Pa2, ..., Pax]` of length `x` is **lexicographically smaller** than `p2 = [Pb1, Pb2, ..., Pbx]` of length `y`, if and only if at the first `j` where `Paj` and `Pbj` differ, `Paj < Pbj`; when no such `j` exists, then `x < y`.

Example 1:

Input: coins = [1,2,4,-1,2], maxJump = 2 **Output:** [1,3,5]

Example 2:

Input: coins = [1,2,4,-1,2], maxJump = 1 **Output:** []

Constraints:

* `1 <= coins.length <= 1000` * `-1 <= coins[i] <= 100` * `coins[1] != -1` * `1 <= maxJump <= 100`

Code Snippets

C++:

```
class Solution {
public:
vector<int> cheapestJump(vector<int>& coins, int maxJump) {

}
};
```

Java:

```
class Solution {
public List<Integer> cheapestJump(int[] coins, int maxJump) {

}
}
```

Python3:

```
class Solution:
def cheapestJump(self, coins: List[int], maxJump: int) -> List[int]:
```