656. Coin Path

Hard 208 94 Add to List Share

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`

Accepted 11,767 | Submissions 38,272

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google

Related Topics

Array Dynamic Programming

Similar Questions

House Robber Medium

House Robber II Medium

Java Autocomplete

```java
public class Solution {
    public List < Integer > cheapestJump(int[] A, int B) {
        int[] next = new int[A.length];
        long[] dp = new long[A.length];
        Arrays.fill(next, -1);
        List < Integer > res = new ArrayList();
        for (int i = A.length - 2; i >= 0; i--) {
            long min_cost = Integer.MAX_VALUE;
            for (int j = i + 1; j <= i + B && j < A.length; j++) {
                if (A[j] >= 0) {
                    long cost = A[i] + dp[j];
                    if (cost < min_cost) {
                        min_cost = cost;
                        next[i] = j;
                    }
                }
            }
            dp[i] = min_cost;
        }
        int i;
        for (i = 0; i < A.length && next[i] > 0; i = next[i])
            res.add(i + 1);
        if (i == A.length - 1 && A[i] >= 0)
            res.add(A.length);
        else
            return new ArrayList < Integer > ();
        return res;
    }
}
```