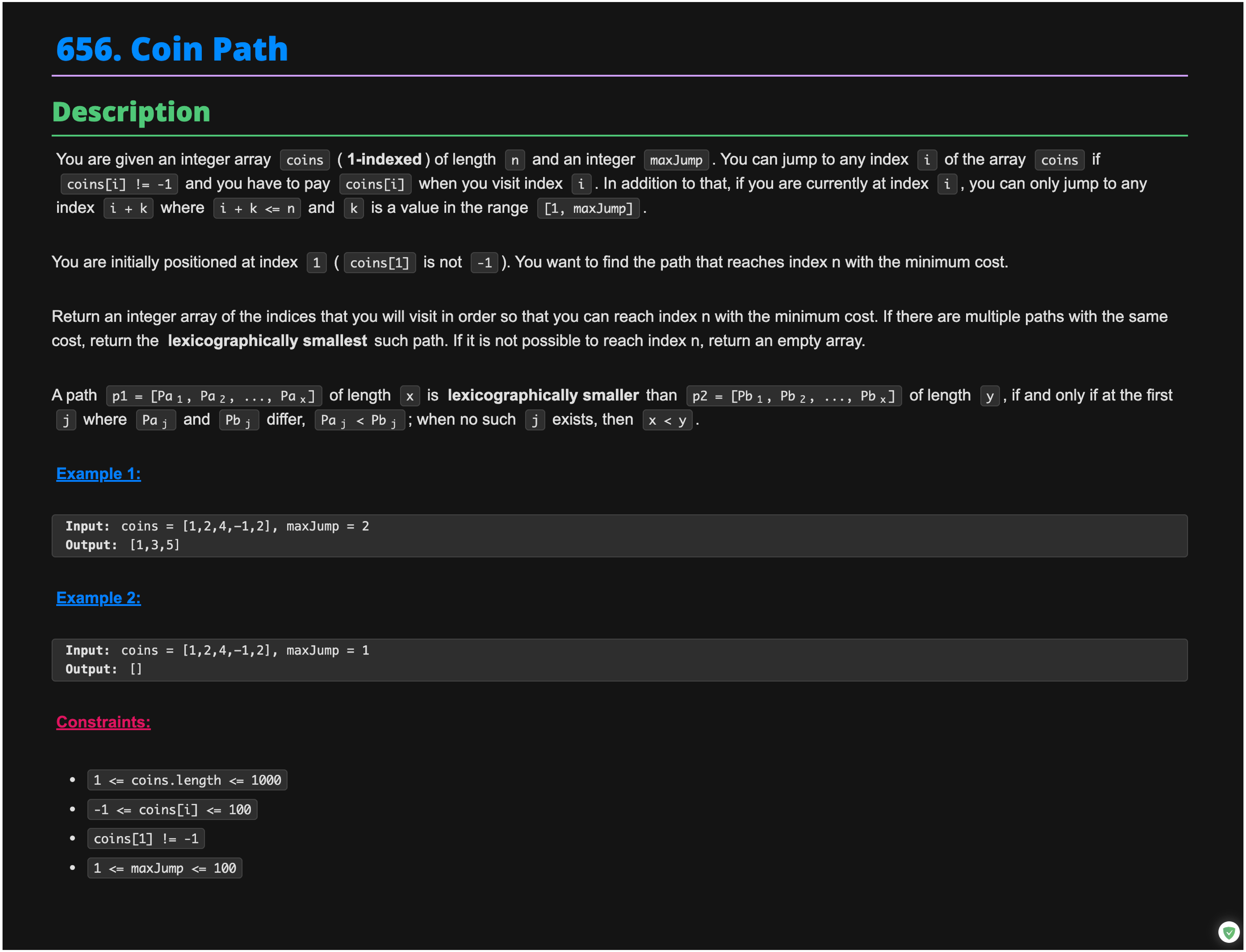

656. Coin Path

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 $p1 = [Pa_1, Pa_2, ..., Pa_x]$ of length `x` is **lexicographically smaller** than $p2 = [Pb_1, Pb_2, ..., Pb_x]$ of length `y`, if and only if at the first `j` where Pa_j and Pb_j differ, $Pa_j < Pb_j$; 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`