Rainforest Jumps

In the tropical rainforest of Sumatra, there are N trees in a row numbered from 0 to $N-1$ from left to right. All trees have **distinct heights**, with tree i having the height $H[i]$.

Pak Dengklek is training an orangutan to jump from tree to tree. In a single jump, the orangutan can jump from the top of a tree to the top of the closest tree, either to the left or to the right, whose height is higher than the tree she is currently at. Formally, if the orangutan is currently at tree x, then she can jump to tree y if and only if either one of these is satisfied:

- y is the largest non-negative integer smaller than x such that $H[y] > H[x]$; or
- y is the smallest non-negative integer larger than x such that $H[y] > H[x]$.

Pak Dengklek has Q jumping plans, each can be represented as four integers A, B, C, and D ($A \le B < C \le D$). For each plan, Pak Dengklek would like to know whether it is possible for the orangutan to start from some tree s ($A \le s \le B$) and end at some tree e ($C \le e \le D$) using a sequence of jumps. If it is possible, Pak Dengklek would like to know the minimum number of jumps the orangutan needs for that plan.

Implementation Details

You should implement the following procedures:

```
void init(int N, int[] H)
```

- N: the number of trees.
- H: an array of length N, where $H[i]$ is the height of tree i.
- This procedure is called exactly once, before any calls to `minimum_jumps`.

```
int minimum_jumps(int A, int B, int C, int D)
```

- A, B: the range of trees that the orangutan must start at.
- C, D: the range of trees that the orangutan must end at.
- This procedure should return the minimum number of jumps to carry out the plan, or -1 if it is impossible to do so.
- This procedure is called exactly Q times.

Example

Consider the following call:

```
init(7, [3, 2, 1, 6, 4, 5, 7])
```

After initialization has been done, consider the following call:

```
minimum_jumps(4, 4, 6, 6)
```

This means the orangutan must start at tree 4 (with height 4) and end at tree 6 (with height 7). One way to achieve the minimum number of jumps is to first jump to tree 3 (with height 6), then jump to tree 6. Another way is to jump to tree 5 (with height 5), then jump to tree 6. Therefore, the procedure `minimum_jumps` should return 2.

Consider another possible call:

```
minimum_jumps(1, 3, 5, 6)
```

This means the orangutan must start at either tree 1 (with height 2), tree 2 (with height 1), or tree 3 (with height 6) and end at either tree 5 (with height 5) or tree 6 (with height 7). The only way to achieve the minimum number of jumps is to start at tree 3, then jump to tree 6 using only one jump. Therefore, the procedure `minimum_jumps` should return 1.

Consider another possible call:

```
minimum_jumps(0, 1, 2, 2)
```

This means the orangutan must start at either tree 0 (with height 3) or tree 1 (with height 2) and end at tree 2 (with height 1). Since tree 2 is the shortest tree, it is impossible to be reached from any tree taller than it. Therefore, the procedure `minimum_jumps` should return -1.

Constraints

- $2 \le N \le 200\,000$
- $1 \le Q \le 100\,000$
- $1 \le H[i] \le N$ (for all $0 \le i \le N-1$)
- $H[i] \ne H[j]$ (for all $0 \le i < j \le N-1$)
- $0 \le A \le B < C \le D \le N-1$

Subtasks

1. (4 points) $H[i] = i+1$ (for all $0 \le i \le N-1$)
2. (8 points) $N \le 200$, $Q \le 200$

3. (13 points) $N \leq 2000$, $Q \leq 2000$
4. (12 points) $Q \leq 5$
5. (23 points) $A = B$, $C = D$
6. (21 points) $C = D$
7. (19 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

- line 1: $N \ Q$
- line 2: $H[0] \ H[1] \ \ldots \ H[N-1]$
- line $3 + i$ ($0 \leq i \leq Q - 1$): $A \ B \ C \ D$ for the i-th call to `minimum_jumps`

The sample grader prints your answers in the following format:

- line $1 + i$ ($0 \leq i \leq Q - 1$): return value of the i-th call to `minimum_jumps`