Problem 2100: Find Good Days to Rob the Bank

Problem Information

Difficulty: Medium
Acceptance Rate: 50.94%
Paid Only: No
Tags: Array, Dynamic Programming, Prefix Sum

Problem Description

You and a gang of thieves are planning on robbing a bank. You are given a **0-indexed** integer array `security`, where `security[i]` is the number of guards on duty on the `ith` day. The days are numbered starting from `0`. You are also given an integer `time`.

The `ith` day is a good day to rob the bank if:

* There are at least `time` days before and after the `ith` day, * The number of guards at the bank for the `time` days **before** `i` are **non-increasing** , and * The number of guards at the bank for the `time` days **after** `i` are **non-decreasing**.

More formally, this means day `i` is a good day to rob the bank if and only if `security[i - time] >= security[i - time + 1] >= ... >= security[i] <= ... <= security[i + time - 1] <= security[i + time]`.

Return _a list of**all** days **(0-indexed)** that are good days to rob the bank_._The order that the days are returned in does******not** matter._

Example 1:

Input: security = [5,3,3,3,5,6,2], time = 2 **Output:** [2,3] **Explanation:** On day 2, we have security[0] >= security[1] >= security[2] <= security[3] <= security[4]. On day 3, we have security[1] >= security[2] >= security[3] <= security[4] <= security[5]. No other days satisfy this condition, so days 2 and 3 are the only good days to rob the bank.

Example 2:

Input: security = [1,1,1,1,1], time = 0 **Output:** [0,1,2,3,4] **Explanation:** Since time equals 0, every day is a good day to rob the bank, so return every day.

Example 3:

Input: security = [1,2,3,4,5,6], time = 2 **Output:** [] **Explanation:** No day has 2 days before it that have a non-increasing number of guards. Thus, no day is a good day to rob the bank, so return an empty list.

Constraints:

* `1 <= security.length <= 105` * `0 <= security[i], time <= 105`

Code Snippets

C++:

```
class Solution {
public:
vector<int> goodDaysToRobBank(vector<int>& security, int time) {

}
};
```

Java:

```
class Solution {
public List<Integer> goodDaysToRobBank(int[] security, int time) {

}
}
```

Python3:

```
class Solution:
def goodDaysToRobBank(self, security: List[int], time: int) -> List[int]:
```