Problem 1575: Count All Possible Routes

Problem Information

Difficulty: Hard
Acceptance Rate: 64.90%
Paid Only: No
Tags: Array, Dynamic Programming, Memoization

Problem Description

You are given an array of **distinct** positive integers locations where `locations[i]` represents the position of city `i`. You are also given integers `start`, `finish` and `fuel` representing the starting city, ending city, and the initial amount of fuel you have, respectively.

At each step, if you are at city `i`, you can pick any city `j` such that `j != i` and `0 <= j < locations.length` and move to city `j`. Moving from city `i` to city `j` reduces the amount of fuel you have by `|locations[i] - locations[j]|`. Please notice that `|x|` denotes the absolute value of `x`.

Notice that `fuel` **cannot** become negative at any point in time, and that you are **allowed** to visit any city more than once (including `start` and `finish`).

Return _the count of all possible routes from_`start` _to_ `finish`. Since the answer may be too large, return it modulo `109 + 7`.

Example 1:

Input: locations = [2,3,6,8,4], start = 1, finish = 3, fuel = 5 **Output:** 4 **Explanation:** The following are all possible routes, each uses 5 units of fuel: 1 -> 3 1 -> 2 -> 3 1 -> 4 -> 3 1 -> 4 -> 2 -> 3

Example 2:

Input: locations = [4,3,1], start = 1, finish = 0, fuel = 6 **Output:** 5 **Explanation:** The following are all possible routes: 1 -> 0, used fuel = 1 1 -> 2 -> 0, used fuel = 5 1 -> 2 -> 1 -> 0, used fuel = 5 1 -> 0 -> 1 -> 0, used fuel = 3 1 -> 0 -> 1 -> 0 -> 1 -> 0, used fuel = 5

Example 3:

Input: locations = [5,2,1], start = 0, finish = 2, fuel = 3 **Output:** 0 **Explanation:** It is impossible to get from 0 to 2 using only 3 units of fuel since the shortest route needs 4 units of fuel.

Constraints:

* `2 <= locations.length <= 100` * `1 <= locations[i] <= 109` * All integers in `locations` are **distinct**. * `0 <= start, finish < locations.length` * `1 <= fuel <= 200`

Code Snippets

C++:

```
class Solution {
public:
int countRoutes(vector<int>& locations, int start, int finish, int fuel) {

}
};
```

Java:

```
class Solution {
public int countRoutes(int[] locations, int start, int finish, int fuel) {

}
}
```

Python3:

```
class Solution:
def countRoutes(self, locations: List[int], start: int, finish: int, fuel:
int) -> int:
```