Description | Editorial | Solutions (536) | Submissions

1575. Count All Possible Routes

Hint

Hard | 1.1K | 42

Companies

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 `10^9 + 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
```

Java | Auto

```java

public class Solution {
    int[][] dp;
    int n;
    int finish;

    public int solve(int[] locations, int currCity, int remainingFuel) {
        if (remainingFuel < 0) {
            return 0;
        }
        if (dp[currCity][remainingFuel] != -1) {
            return dp[currCity][remainingFuel];
        }

        int ans = currCity == finish ? 1 : 0;
        for (int nextCity = 0; nextCity < n; nextCity++) {
            if (nextCity != currCity) {
                ans = (ans + solve(locations, nextCity,
                        remainingFuel - Math.abs(locations[currCity] - locations[nextCity]))) %
1000000007;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

locations =

[2,3,6,8,4]

start =

1

finish =

Console | Run | Submit

Description | Editorial | Solutions (1.5K) | Submissions

693. Binary Number with Alternating Bits

Easy | 1.2K | 109

Companies

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

Example 1:

```
Input: n = 5
Output: true
Explanation: The binary representation of 5 is: 101
```

Example 2:

```
Input: n = 7
Output: false
Explanation: The binary representation of 7 is: 111.
```

Example 3:

```
Input: n = 11
Output: false
Explanation: The binary representation of 11 is: 1011.
```

Constraints:

- $1 <= n <= 2^{31} - 1$

Java | Auto

```java
class Solution {
    public boolean hasAlternatingBits(int n) {
        String s = Integer.toBinaryString(n);
        char prev = s.charAt(0);
        for(int i=1; i<s.length(); i++) {
            int x = Character.compare(prev, s.charAt(i));
            System.out.println(x);
            if(x == 0) {
                return false;
            }
            prev = s.charAt(i);
        }
        return true;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

n =
5

Stdout

```
1
-1
```

Console | Run | Submit