Qn Link : <https://www.desiqna.in/15983/amazon-summer-school-sep-2023-minimum-cost-reach-destination>

**Question Summary :**

You re given array and you can move to either I + 1 or I + 3 .

Each index has a value and you need to reach the index “N” with minimum cost.

Cost to travel an array is calculated as abs(num[i] – nums[I – 1])

**Observation :**

* Cost to reach index 0 , dp[0] = 0
* Cost to reach index 1 , dp[1] = abs( nums[0] – nums[1])
* Cost to reach index 2 , dp[2] = abs( nums[1] – nums[2] )
* Now for index 3 , we have 2 options
  + Either moving from index 2 to index 3
  + Or moving from index 0 to index 3

**Sample TC :**

[4 , 12 , 13 , 18 , 10 , 12]

Dp[0] = 0

Dp[1] = abs(12 – 4) = 8

Dp[2] = abs(13 – 12) + dp[1] = 1 + 8 = 9

Dp[3] = ( from index 2 to index 3 ) or ( from index 0 to index 3)

= ( abs( nums[2] – nums[3]) + dp[2]) or ( abs( nums[0] – nums[3]) + dp[0])

= min ( (5 + 9 ) , (14 + 0))

* 14

So , the recurrence formula is

* Min ( ( abs(nums[i] – nums[I – 1) ) + dp[I – 1] ) , ( ( abs(nums[i] – nums[I –3) + dp[I – 3] ) )

Step 1 : create an DP array of size N

Step 2 : Fill the initial value in the DP array , and run the loop from 3 to n

Step 3 : For each iteration , calculate the value by using the recurrence formula.

Step 4 : return the value of dp[n] .

class Solution {

    public int minCost(int [] nums) {

        int n = nums.length;

         int [] dp = new int[n];

         dp[0]  = 0;

         dp[1]  = Math.abs(nums[0] - nums[1]);

         dp[2] = Math.abs(nums[2] - nums[1]) + dp[1];

         for(int i = 3 ; i < n ; i++){

            dp[i] = Math.min(   Math.abs(nums[i] - nums[i - 1]) + dp[i - 1] ,

                                 Math.abs(nums[i] - nums[i - 3]) + dp[i - 3]);

         }

         return dp[n];

    }

}