Qn Link : <https://atcoder.jp/contests/dp/tasks/dp_a>

Question Summary :

* Given an array, in each iteration you can move to either i + 1 or i + 2
* Find the minimum cost to reach the index N , where N is the size of the array.
* The cost to travel to an index is , abs ( nums[i] - nums[i - 1]).

Step 1 : create an Dp array of size N , fill the initial value for the index ) and !.

Step 2 : Dp[0] = 0 and DP [1] = abs ( nums[i] - nums[i - 1]).

Step 3 :

* For index 2 , we ve two options either from index 2 to index 3 or index 1 to index 3.
* We need the minimum value , so find the min of abs ( nums[i] - nums[i - 1]) , abs ( nums[i] - nums[i - 2]).
* In other words , min ( dp[i -1 ] + abs(nums[i] - nums[i - 1] ,

Dp[i - 2] + abs(nums[i] - nums[i - 2]) )

Recurrence formula :

dp[i] = Min ( dp[i -1 ] + abs(nums[i] - nums[i - 1] , dp[i - 2] + abs(nums[i] - nums[i - 2]) )

private int minJumps (int [] nums , int n){

int [] dp = new int [n];

dp[0] = 0;

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

for(int i = 2 ; i < n; i++){

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

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

}

return dp[n];

}