**Problem**- [**Frog Jump**](https://www.codingninjas.com/codestudio/problems/frog-jump_3621012?source=youtube&campaign=striver_dp_videos&utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_dp_videos&leftPanelTab=1)

\*\***1. recursive- solution**

**int helper(int ind, vector<int> &heights){**

**if(ind==0)**

**return 0;**

**//1 step**

**int left = helper(ind-1, heights) + abs(heights[ind]- heights[ind-1]);**

**//ind==1, can't take 2 step**

**int right = (ind>1) ? helper(ind-2, heights) + abs(heights[ind]- heights[ind-2]) : 1e4; //2-step**

**//store minimum**

**return min(left, right);**

**}**

**int frogJump(int n, vector<int> &heights)**

**{**

**return helper(n-1, heights);**

**}**

//TC = O(N)

//SC = O(N) for recursion stack space

\*\***2. DP- memoization solution**

**int helper(int ind, vector<int> &heights, vector<int> &dp){**

**if(ind==0)**

**return 0;**

**if(dp[ind] != -1) //return value for overlapping subproblems**

**return dp[ind];**

**//1 step**

**int left = helper(ind-1, heights, dp) + abs(heights[ind]- heights[ind-1]);**

**//ind==1, can't take 2 step**

**int right = (ind>1) ? helper(ind-2, heights, dp) + abs(heights[ind]- heights[ind-2]) : 1e4; //2-step**

**dp[ind] = min(left, right); //store minimum**

**return dp[ind];**

**}**

**int frogJump(int n, vector<int> &heights)**

**{**

**vector<int> dp(n, -1);**

**return helper(n-1, heights, dp);**

}

//TC = O(N)

//SC = O(N) for recursive stack space + O(N) for dp[]

\*\***3. DP- tabulation**

**//DP- tabulation(bottom- up approach)**

**int frogJump(int n, vector<int> &heights)**

**{**

**vector<int> dp(n, -1);**

**dp[0] = 0; //have a base**

**for(int i=1; i<=n-1; i++){**

**int left = dp[i-1] + abs(heights[i]- heights[i-1]); //1-step**

**int right = (i>1) ? dp[i-2] + abs(heights[i]- heights[i-2]) : 1e4; //2step**

**dp[i] = min(left, right);**

**}**

**return dp[n-1];**

**}**

//TC = O(N)

//SC = O(N)

\***\*4. DP- optimized space**

**int frogJump(int n, vector<int> &heights)**

**{**

**int prev1= 0;**

**int prev2= 0;**

**for(int i=1; i<=n-1; i++){**

**int left = prev1 + abs(heights[i]- heights[i-1]);**

**int right = (i>1) ? prev2+ abs(heights[i]- heights[i-2]) : 1e4;**

**int cur = min(left, right);**

**prev2 = prev1;**

**prev1 = cur;**

**}**

**return prev1;**

**}**

//TC = O(N)

//SC = O(1)