**Problem**- [**Frog Jump**](https://atcoder.jp/contests/dp/tasks/dp_b)

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

**int solveUtil(int ind, vector<int>& height, int k){**

**if(ind==0) return 0;**

**int mmSteps = INT\_MAX;**

**for(int j=1;j<=k;j++){**

**if(ind-j>=0){**

**int jump = solveUtil(ind-j, height, k)+ abs(height[ind]- height[ind-j]);**

**mmSteps= min(jump, mmSteps);**

**}**

**}**

**return mmSteps;**

**}**

**int solve(int n, vector<int>& height , int k){**

**return solveUtil(n-1, height, k);**

**}**

//TC = O(N\*K)

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

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

**int solveUtil(int ind, vector<int>& height, vector<int>& dp, int k){**

**if(ind==0) return 0;**

**if(dp[ind]!=-1) return dp[ind];**

**int mmSteps = INT\_MAX;**

**for(int j=1;j<=k;j++){**

**if(ind-j>=0){**

**int jump = solveUtil(ind-j, height, dp, k)+ abs(height[ind]- height[ind-j]);**

**mmSteps= min(jump, mmSteps);**

**}**

**}**

**return dp[ind]= mmSteps;**

**}**

**int solve(int n, vector<int>& height , int k){**

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

**return solveUtil(n-1, height, dp, k);**

**}**

//TC = O(N\*k)

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

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

**int solveUtil(int n, vector<int>& height, vector<int>& dp, int k){**

**dp[0]=0;**

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

**int mmSteps = INT\_MAX;**

**for(int j=1;j<=k;j++){**

**if(i-j>=0){**

**int jump = dp[i-j]+ abs(height[i]- height[i-j]);**

**mmSteps= min(jump, mmSteps);**

**}**

**}**

**dp[i]= mmSteps;**

**}**

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

**}**

**int solve(int n, vector<int>& height , int k){**

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

**return solveUtil(n, height, dp, k);**

**}**

//TC = O(N\*k)

//SC = O(N)

**\*\*4. DP- optimized space:** not possible!