FROG JUMP WITH K DISTANCES

A Follow-Up to Frog Jump, Here the frog is allowed to jump upto 'K' steps of a time. If K=4, the frog can jump 1, 2, 3 or 4 steps at every index

Sol If K possible jumps are being given we cannot always write those many no. of jumps again & again like jump1, jump2, jump3 ... jumpk. Instead we'll use a for loop

```
f(ind){
if(ind==0) return 0;
min_steps = INT_MAX;
for(j=1;j<=K;j++){ if (ind-j>=0){
jump = f(ind-j) + abs(a[ind]-a[ind-j]);
minSteps = min(minSteps, jump);
}}
return minSteps;
}
```

Space Optimization

There is no need of space optimization, since even if you try to space optimize the ~~time complexity will~~ i.e... if K=3, the space complexity will be O(n) i.e... if you are given n jumps you have to carry on, there is no other way

Memoization ⇒ store values in dp to access them back again.

Tabulation:-

```
int dp[n]; dp[0]=0;
for (i=1; i<n; i++) {
minSteps=INT_MAX;
for(j=1; j<=K;j++){
if (i-j>=0){
jump=dp[i-j] + abs(a[ind] - a[ind-j]);
minSteps = min(minSteps, jump);
}}
dp[i] = minSteps}
print(dp[n-1]);
```