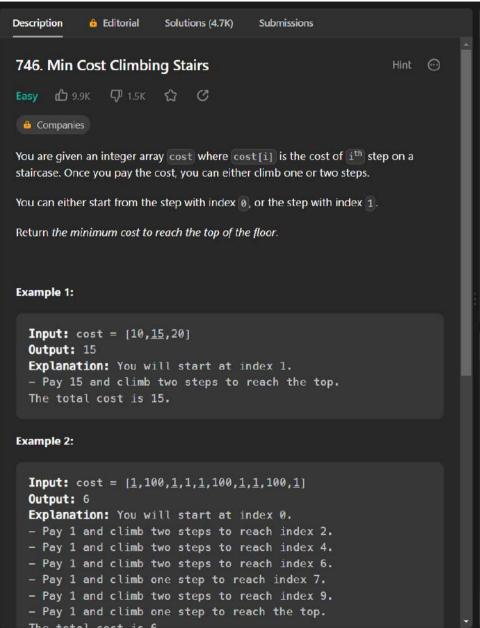


```
i Java · Auto
     class Solution {
         private HashMap<Pair<Integer, Integer>, Double> memo = new HashMap<>();
         public double soupServings(int N) {
              if (N > 4451) {
                  return 1.0;
             N = (N + 24) / 25;
             return dp(N, N);
         private double dp(int a, int b) {
              if (a <= 0 && b <= 0) {
                  return 0.5;
              if (a <= 0) {
                  return 1.0:
Testcase
         Result
Accepted Runtime: 0 ms
  • Case 1

 Case 2

Input
  50
Output
  0.62500
Console v
                                                                                 Run
                                                                                           Submit
```



```
i Java · Auto
         public int minCostClimbingStairs(int[] cost) {
             int[] dp = new int[cost.length];
             return Math.min(helper(cost, 0, dp), helper(cost, 1, dp));
         private int helper(int[] cost, int i, int[] dp) {
             if(i >= cost.length) return 0;
             if(dp[i] != 0) return dp[i];
             int oneStep = cost[i] + helper(cost, i + 1, dp);
             int twoStep = cost[i] + helper(cost, i + 2, dp);
             return dp[i] = Math.min(oneStep, twoStep);
Testcase
         Result
Accepted Runtime: 0 ms
  • Case 1

 Case 2

Input
  [10, 15, 20]
Output
 15
Console v
                                                                                 Run
                                                                                           Submit
```