**Question :**

* You re given an number N reduce it to 1 in minimum number of steps.
* In one step , you can either do one of the 3 operation
  + Reduce it by 1
  + Reduce by /2 if it is divisible by 2
  + Reduce by /3 if it is divisible by 3

**Observation :**

* 1 to reach 1 in 0 steps
* 2 to reach 1 in 1 steps
* If a number is divisible by 2 or 3 we find the min of those

**Recurrence Relation :**

dp[i] = min ( dp[i - 1] , dp[i/ 2] , dp[i / 3) + 1

Step 1 : Create an DP array of size “n + 1”

Step 2 : put dp[1] = 1 and dp[2] = 1

Step 3 : Run a for loop from 3 to n and apply the recurrence formula.

**Code :**

class Solution {

private int min(int a , int b , int c){

if(a < b && a < c){

return a;

}

if(b < c){

return b;

}

return c;

}

public int minOperation(int n){

int [] dp = new int[n + 1];

dp[1] = 0;

dp[2] = 1;

for(int i = 3 ; i <= n; i++){

int v1 = dp[i - 1] + 1;

int v2 = Integer.MAX\_VALUE;

int v3 = Integer.MAX\_VALUE;

if(i % 2 == 0){

v2 = dp[i / 2] + 1;

}

if(i % 3 == 0){

v3 = dp[i / 3] + 1;

}

dp[i] = min(v1 , v2 , v3);

}

return dp[n];

}

}

Follow Up :

Reduce n to 1

If the number is even you can divide it by 2.

If the number is odd you can do +1 or -1

Recurrence Relation :

dp[i] = min(dp[i - 1] + 1 , 2 + dp[(i+ 1) / 2])