**DP-2:**

**Coin Change-2**

**BruteForce Approach:**

class Solution {

public int change(int amount, int[] coins) {

if(coins == null || coins.length == 0)

return 0;

return helper(coins, amount, 0, 0);

}

private int helper(int[]coins,int amount, int index, int count){

//base case

if(amount == 0) return count+1;

if(amount< 0 || index == coins.length) return count;

//choose

int case1 = helper(coins, amount-coins[index],index,count);

//dont choose

int case2 = helper(coins, amount, index+1,count);

return case1+ case2;

}

}

**DP Approach:**

class Solution {

public int change(int amount, int[] coins) {

if(coins == null || coins.length == 0)

return 0;

int [][] dp = new int[coins.length+1][amount+1];

//filling first column

for(int i = 0; i<= coins.length;i++)

dp[i][0] = 1;

for(int i = 1; i<= coins.length; i++){

for(int j = 1; j<= amount; j++){

//amount< denomination

if(j<coins[i-1])

dp[i][j]= dp[i-1][j];

else

dp[i][j]= dp[i-1][j]+dp[i][j-coins[i-1]];

}

}

return dp[coins.length][amount];

}

}

**Paint house:**

**BruteForce Approach:**

class Solution {

public int minCost(int [][] costs) {

if(costs == null || costs.length == 0)

return 0;

//costs, row, color, mincost

int case1 = helper(costs, 0, 0,0);

int case2 = helper(costs, 0, 1, 0);

int case 3 = helper(costs, 0 , 2, 0);

return Math.min(case1, Math.min(case2,case3);

}

private int helper(int[][]costs,int row, int color, int minCost){

if(row == costs.length) return minCost;

if(color == 0){

Math.min(costs, row+1,1, minCost+costs[row][color],costs, row+1,2, minCost+costs[row][color]);

}

if(color == 1){

Math.min(costs, row+1,0, minCost+costs[row][color],costs, row+1,2, minCost+costs[row][color]);

}

if(color == 2){

Math.min(costs, row+1,1, minCost+costs[row][color],costs, row+1,0, minCost+costs[row][color]);

}