Paint House

There are a row of *n* houses, each house can be painted with one of the three colors: red, blue or green. The cost of painting each house with a certain color is different. You have to paint all the houses such that no two adjacent houses have the same color.

The cost of painting each house with a certain color is represented by a `n x 3` cost matrix. For example, `costs[0][0]` is the cost of painting house 0 with color red; `costs[1][2]` is the cost of painting house 1 with color green, and so on... Find the minimum cost to paint all houses.

Note:
All costs are positive integers.

Solution 1

The 1st row is the prices for the 1st house, we can change the matrix to present sum of prices from the 2nd row. i.e, the costs[1][0] represent minimum price to paint the second house red plus the 1st house.

```
public class Solution {
public int minCost(int[][] costs) {
    if(costs==null||costs.length==0){
        return 0;
    }
    for(int i=1; i<costs.length; i++){
        costs[i][0] += Math.min(costs[i-1][1],costs[i-1][2]);
        costs[i][1] += Math.min(costs[i-1][0],costs[i-1][2]);
        costs[i][2] += Math.min(costs[i-1][1],costs[i-1][0]);
    }
    int n = costs.length-1;
    return Math.min(Math.min(costs[n][0], costs[n][1]), costs[n][2]);
}
```

}

written by ya09208 original link here

Solution 2

```
public class Solution {
    public int minCost(int[][] costs) {

        if(costs==null || costs.length==0) return 0;
        int[] prevRow = costs[0];
         for(int i=1;i<costs.length;i++)
        {
            int[] currRow = new int[3];
            for(int j=0;j<3;j++)
                currRow[j]=costs[i][j]+Math.min(prevRow[(j+1)%3],prevRow[(j+2)%3]
);
            prevRow = currRow;
        }
        return Math.min(prevRow[0],Math.min(prevRow[1],prevRow[2]));
    }
}
```

written by prachibhans original link here

Solution 3

the idea is to store current house's minimum cost in different colors.

```
def minCost(self, costs):
    size = len(costs)
    if size == 0:
        return 0

    pre = costs[0][:]
    now = [0]*3

    for i in xrange(size-1):
        now[0] = min(pre[1], pre[2]) + costs[i+1][0]
        now[1] = min(pre[0], pre[2]) + costs[i+1][1]
        now[2] = min(pre[0], pre[1]) + costs[i+1][2]
        pre[:] = now[:]

    return min(pre)
```

written by autekwing original link here