class Solution:

"""

@param costs: n x k cost matrix

@return: an integer, the minimum cost to paint all houses

"""

def minCostII(self, costs):

# write your code here

if not costs:

return 0

n,k = len(costs),len(costs[0])

dp = [[0 for i in range(k)] for i in range(n+1)]

for i in range(1,n+1):

for j in range(k):

a = float("inf")

for l in range(k):

if j!=l:

a = min(a,dp[i-1][l])

a+=costs[i-1][j]

dp[i][j] = a

return min(dp[-1])

问题的关键在于怎么把O(N\*K^2) reduece 到 O(NK), 否则当K很大的时候必然超时：

在找不同颜色的最小值不是遍历所有不同颜色，而是用min1和min2来记录之前房子的最小和第二小的花费的颜色，如果当前房子颜色和min1相同，那么我们用min2对应的值计算，反之我们用min1对应的值

def minCostII(self, costs):

""" 优化后, AC过"""

if len(costs) < 1:

return 0

if len(costs[0]) < 2:

return sum([i[0] for i in costs])

for idx, cost in enumerate(costs[1:]):

sort = list(sorted([(i, j) for j, i in enumerate(costs[idx])]))

for idy in range(len(cost)):

cost[idy] += sort[0][0] if idy != sort[0][1] else sort[1][0]

# end for

return min(costs[-1])