Subtask 1:

Can just Bitmask and 2^n
So like for each house without a colour either try to let him be red or blue (try both!!)

Subtask 2:
dp(i, j) = where cost to let house i be colour j
If that house has a fixed colour (for example red, then we can jus set
dp[i][blue]=-100000000000)
After each query just recompute dp. Resulting in O(NQ) time complexity.

Subtask 3:
Updates only occur at one point x. So do a dp_prefix(i, j) and a dp suffix (i, j) and the answer after an update is simply

dp_prefix(x-1, j) + (colour of house x) + dp_suffix(x+1, k) (for all values of j and k.)

The above works becos all queries only ask from 1 to n

Subtask 4:
Construct a dynamic segment tree with each node having a dp state dp(i, j) where i stands for the colour of the first guy and j stands for the colour of the last guy (in ur interval!) Then when u merge two nodes in ur segtree u try all possible pairs of first guy last guy and see if left->colour_of_last_guy != right_colour_of_first_guy then then + 1.

Similar to the maxsum segment tree (ur can google abt this now i guess)