Problem 3

I approached the task as a DP problem and began by pinning down **state**:

- `i` – how many trees are fixed so far
- `g` – groups (beauty) currently formed
- `c` – colour of the last painted tree

So the value I need is

```
dp[i][g][c] = minimum paint to colour trees 0…i-1
            with g groups, ending in colour c
```

First attempt:

I naïvely tried a greedy "paint the cheapest colour each time" pass.

It raced through the sample but blew up on hidden tests: the greedy step often locked me into too many / too few groups and could never back-track.

Second attempt:

Converted the idea to a 1-D DP (`best[i] = min paint up to i`).

Failed again because the beauty constraint depends on **both** the previous colour and the current group count, a single dimension cannot remember both.

Final (accepted)

1. **Initialise**

   `cur[0][0] = 0`, everything else = INF.

2. **Transition** for each tree `pos`

   If the tree is pre-coloured → only that colour is legal, cost 0.

   Else try every colour `col` and add `cost[pos][col]`.

   New group count `ng = g + (col != last)`; skip if `ng > k`.

Update `nxt[ng][col] = min(nxt[ng][col], cur[g][last] + Δcost)`.

3. **Roll arrays** (`cur ↔ nxt`) to keep memory `O(k·m)`.
4. **Answer** = `min(cur[k][col])`, or −1 if all are INF.