Marathon Match Problem Statement

Contest: Poland Lightning Round Normal view

Problem: MapRecoloring

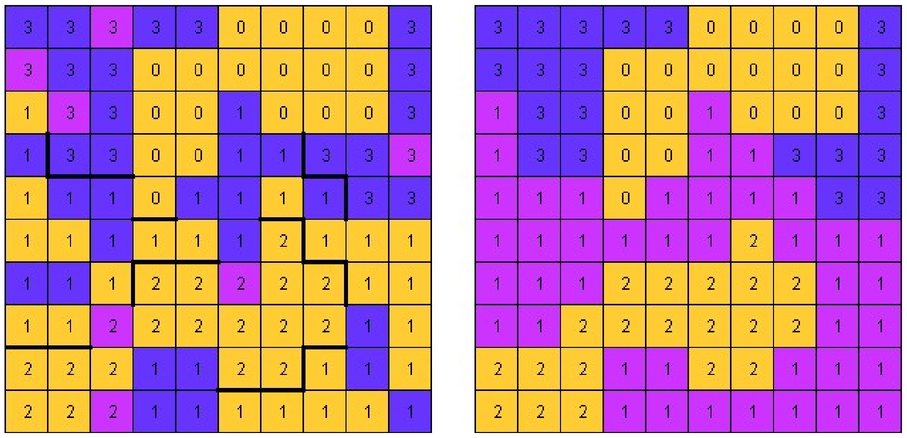
Problem Statement

You are given a rectangular map of **H** x **W** pixels. The map is divided into **R** regions. Each region consists of one or several contiguous groups of pixels. Each pixel of the map starts painted with some color. Pixels of the same region can be painted with different colors.

Your task is to assign each region a color so that

1. no two adjacent regions (i.e. regions which have pixels adjacent vertically or horizontally) are assigned the same color,
2. as few different colors are used as possible, and
3. the number of pixels that have to be repainted when painting each region its assigned color is minimized.

For example, the map shown on the left side of the image has four regions, all of them contiguous except region 3 which consists of two contiguous groups of pixels. Pixels of region 0 are all painted the same color, the rest of regions have pixels of different colors. One valid color assignment is {2, 1, 2, 0}, which requires recoloring 49 pixels (shown on the right side of the image).



Implementation

Your code should implement a single method **recolor**(integer **H**, tuple (integer) **regions**, tuple (integer) **oldColors**).

- **H** gives you the height of the map.
- **regions** gives you the assignment of map pixels to the regions. Pixel with coordinates (row, col) belongs to a region number **regions**[row * **W** + col]. (**regions** contains exactly **H** * **W** elements, so you can figure out **W** as **regions**.length / **H**.) Regions are numbered 0 through **R**-1.
- **oldColors** gives you the starting colors of map pixels. Pixel with coordinates (row, col) has color **oldColors**[row * **W** + col]. **oldColors** contains **H** * **W** elements. Colors are numbered starting with 0.

The method should return an array of **R** integers, i-th integer denoting the color assigned to i-th region. You can use at most **R** colors, and each element of your return must be between 0 and **R**-1, inclusive. The indices of the colors in your return match the indices of the colors in the original coloring.

Scoring

Your raw score for an individual test case will be calculated as (100000 * total number of colors used + number of pixels recolored). If your return was invalid (was formatted incorrectly, returned an invalid map coloring, etc.), the score for this test case will be -1.

Your normalized score for an individual test case will be calculated in the following way. For each test case where your score is not -1, the minimal number of colors **C_min** and the minimal number of pixels recolored **P_min** currently obtained on this test case will be calculated (considering only the last submission from each competitor). Your normalized score for that test case will be (**C_min** / **C_your**) * (**P_min** / **P_your**), where **C_your** and **P_your** are the number of colors you used and the number of pixels you recolored, respectively.

Finally, your total score is equal to the arithmetic average of normalized scores on all test cases, multiplied by 1,000,000.

Tools

An offline tester is available here. You can use it to test/debug your solution locally. You can also check its source code for exact implementation of test case generation and score calculation. That page also contains links to useful information and sample solutions in several languages.

Definition

Class:	MapRecoloring
Method:	recolor
Parameters:	integer, tuple (integer), tuple (integer)
Returns:	tuple (integer)
Method signature:	def recolor(self, H, regions, oldColors):

(be sure your method is public)

Notes

- The time limit is 10 seconds per test case (this includes only the time spent in your code). The memory limit is 1024 megabytes.
- There is no explicit code size limit. The implicit source code size limit is around 1 MB (it is not advisable to submit codes of size close to that or larger). Once your code is compiled, the binary size should not exceed 1 MB.
- The compilation time limit is 30 seconds. You can find information about compilers that we use and compilation options here.
- There are 10 example test cases and 100 full submission (provisional) test cases. There will be 2000 test cases in the final testing.
- The match is rated.
- There are prizes for top 3 finishers among attendees of Warsaw regional event and top 3 finishers among virtual participants. See the rules for details.

Constraints

- Both dimensions of the map **H** and **W** will be between 20 and 200, inclusive.
- The number of the colors used for coloring the map initially will be between 2 and 5, inclusive.
- The number of the regions on the map will be between (**H** * **W**) / 10 and (**H** * **W**) / 50.

Examples

0)

```
seed = 1
H = 20
W = 20
R = 16
Regions:
15 15 15 15 15 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
15 15 15 15 15 1 1 1 1 1 1 9 5 5 2 2 2 2 2 2
15 15 15 15 11 11 11 1 1 1 1 9 5 5 15 2 2 2 2 2
15 15 15 15 11 11 11 11 11 9 9 9 4 15 15 2 2 2 2 2
15 15 15 15 0 0 11 11 11 11 9 4 4 9 9 8 2 2 2 2
15 15 0 0 0 0 11 11 11 12 4 4 4 9 9 9 2 2 2 14
0 0 0 0 0 7 7 12 12 12 6 6 4 9 7 9 9 14 14 14
3 0 0 0 0 7 7 6 6 6 6 10 10 10 7 7 14 14 14 14
3 3 0 0 0 0 15 1 6 6 6 10 10 7 7 7 14 14 14 14
3 13 13 13 0 15 15 3 6 10 10 10 10 10 7 7 14 14 14 7
3 13 13 13 15 15 15 15 6 6 10 10 10 15 7 7 7 14 7 7
3 13 3 3 15 2 2 15 12 6 14 14 14 14 14 7 7 7 7 7
3 3 3 3 1 2 2 9 12 6 6 14 14 6 6 7 7 7 7 7
3 3 3 3 1 2 9 9 9 11 11 14 14 6 6 7 7 7 7 7
5 5 5 7 1 2 6 9 6 11 11 14 6 6 6 6 6 7 7 7
7 7 7 7 7 6 6 6 6 6 13 6 6 6 6 7 7 7 7 7
7 7 7 1 1 6 6 6 6 6 13 14 14 12 12 12 12 7 7 7
7 7 1 1 1 1 6 6 12 12 12 12 12 12 12 12 12 12 7 7
1 1 1 1 1 1 2 6 12 12 12 12 12 12 12 12 12 12 7 7
1 1 1 1 1 2 2 12 12 12 12 12 12 12 12 12 12 12 7 7
Old map colors:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0
0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
1 1 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0
1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 1
1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0
1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

1)

```
seed = 2
H = 200
W = 200
R = 1290
```

2)

```
seed = 3
H = 183
W = 55
```

```
R = 214
```

3)

```
seed = 4
H = 141
W = 198
R = 1642
```

4)

```
seed = 5
H = 156
W = 23
R = 83
```

5)

```
seed = 6
H = 82
W = 76
R = 207
```

6)

```
seed = 7
H = 112
W = 145
R = 854
```

7)

```
seed = 8
H = 105
W = 146
R = 806
```

8)

```
seed = 9
H = 41
W = 144
R = 227
```

9)

```
seed = 10
H = 46
W = 164
R = 359
```