**Task:**

To find the largest link nodes of the same color in a row or column, when the number of columns, rows, and colors in a color grid is not constant.

**Assumption:**

The solution has been implemented such that to return the colour with the highest number of linked nodes.

* Column count : 5
* row count : 5
* color count : 3 (red, green, blue)
* Generated color grid:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

**Considering the color grid above, the expected answer would be:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

•Node Count: 4

•Start X, Y : 4,0

•End X, Y : 4,3

**Approach:**

1. A color grid was created initially (for testing purposes).



The above code segment returns a ColorGrid object having a random row, column, and color count.

1. The grid was iterated across its rows and columns checking for the availability of same color nodes consecutively (using the ‘checkColors’ function). The number of similar colored node count, indices(X, Y) of the last node of the iterated color pattern is then stored in an object (‘ResultDto’).



This method iterates through rows and columns of the grid to find the largest linked nodes of the same color in a single row, and returns a ResultDto object. The ResultDto object contains the number of nodes and the last node row index and column index with the same color.

checkColors method🡪 Iterating through the columns in a row using a nested for loop, the first node (0, 0) is taken and its color is compared with the color of the right-side node (0, 1). If it is the same, the right-side node’s row index, column index is saved to ResultDto. Also, while traversing, the count of similar color nodes is saved in the ‘count’ variable.

If new same color nodes are found, checked if the number of new nodes is greater than the number of saved nodes and the existing results in the ResultDto is updated with the new values.

1. Upon traversing through all the rows in the generated grid using the above function, the same was intended to be performed across the columns as well. In order to use the same checkColor function above, the rows and columns of the existing grid were swapped using the ‘swapRowCols’ method.



Iterated through rows and columns, and set row nodes as column nodes and column nodes as row nodes in temporary ColorGrid and returned the temporary ColorGrid as a new ColorGrid.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

1. From the steps 2, 3 above, two ResultDto objects (rowResultDto, columnResultDto) have been returned. The two node counts retuned are then compared for the maximum count and the final X, Y values are calculated and printed.

**Calculate final X and Y:**

1. If rowResultDto node count is greater than columnResultDto node count, the start X index was calculated by subtracting the node count from X index value of the rowResultDto.



Calculate start X:

1. If rowResultDto node count is less than columnResultDto node count, the start Y index was calculated by subtracting the node count from Y index value of the columnResultDto.



Calculate start Y: