| Algorithmics | Student information | Date | Number of session |
|---|---|---|---|
| | UO: 300047 | 17/03/2025 | 4 |
| | Surname: Rodríguez Torres | | |
| | Name: Luisa Natalia | | |

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

# Activity 1. Map Coloring

To solve this exercise I have implemented a method where we have a list of used colors (usedColors) and the new Map with the colors added (coloredGraph).

We must first iterate through the entire original graph, which means iterating over all the nodes, making the complexity of this first loop O(n).

Then we have another loop, which depends on the number of neighbors each node has to add the already used colors in usedColors. Since this is a constant value k, we have that the complexity of the two loops together is O(n*k), or once simplified, O(n).

Finally we have another loop (not inside the second one) which iterates through the usedColors to check which color to assign to the node introduced in coloredGraph. This is also a constant k, we have that the complexity of the two loops together (the first and the third) is O(n*k), or once simplified, O(n).

We have that the complexity of this method is linear. O(n).

| **n** | **t Coloring (ms)** |
|---|---|
| 8 | 0 |
| 16 | 0 |
| 32 | 0 |
| 64 | 0 |
| 128 | 0 |
| 256 | 0 |
| 512 | 2 |

| Algorithmics | Student information | Date | Number of session |
|---|---|---|---|
| | UO: 300047 | 17/03/2025 | 4 |
| | Surname: Rodríguez Torres | | |
| | Name: Luisa Natalia | | |

| | |
|---|---|
| 1024 | 4 |
| 2048 | 7 |
| 4096 | 12 |
| 8192 | 11 |
| 16284 | 30 |
| 32768 | 60 |
| 65536 | 100 |

**We can see that this algorithm works very well for smaller numbers, and does indeed grow with a n log(n) complexity. However, for bigger numbers, the amount of function calls becomes too much for Java, since it must allocate memory in each of them, so we get and OutOfMemory exception.**