


Algorithmics	Student information	Date	Number of session
	UO: 296503	17/03/2025	Session4
	Surname: Mulet Alonso	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Sergio		



## Activity 1. Map coloring

The complexity of my greedy algorithm is  $O(n)$ , for the heuristic, I decide to order the nodes in descending order according to its degree, so I start with the one with the highest degree and end with the lowest.

```
List<String> sortedNodes = graph.keySet().stream()
    .sorted
    ((a, b) -> Integer.compare(graph.get(b).size(), graph.get(a).size()))
    .toList();
```

After having the nodes ordered, we simply iterate through them and print them with a color that is not used by their neighbors.

```
for (String node : sortedNodes) {
    for (String color : colours) {
        if (isNotUsed(color, graph.get(node))) {
            graphColoured.put(node, color);
            break;
        }
    }
}
return graphColoured;
```

Note that despite having a nested loop, the second for is iterating through the array of colors, which has a fixed size of 8, which is not  $n$ , so the complexity at the end, would be  $O(n)$ .

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/03/2025	Session4
	Surname: Mulet Alonso		
	Name: Sergio		

### Times measured:

n	t(ms)
8	39
16	0
32	2
64	4
128	5
256	10
512	19
1024	40
2048	86
4096	170
8192	354
16384	729
32768	1494
65536	3072

Note that for measuring times, I have changed Greedy class, so it does not waste time writing the solution file, however I have first run it with it so I have the solution files (and times have improved).

The algorithm seems to be good as for graph with 128 nodes we have the same number of colors as in the example, 6 colors.