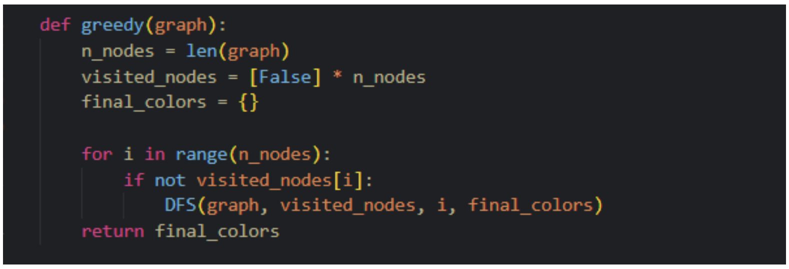
Algorithmics	Student information	Date	Number of session
	UO: 300829	19/03/2025	4
	Surname: Cid Lazcano		
	Name: Izan		

Activity 1. Divide and conquer by substraction

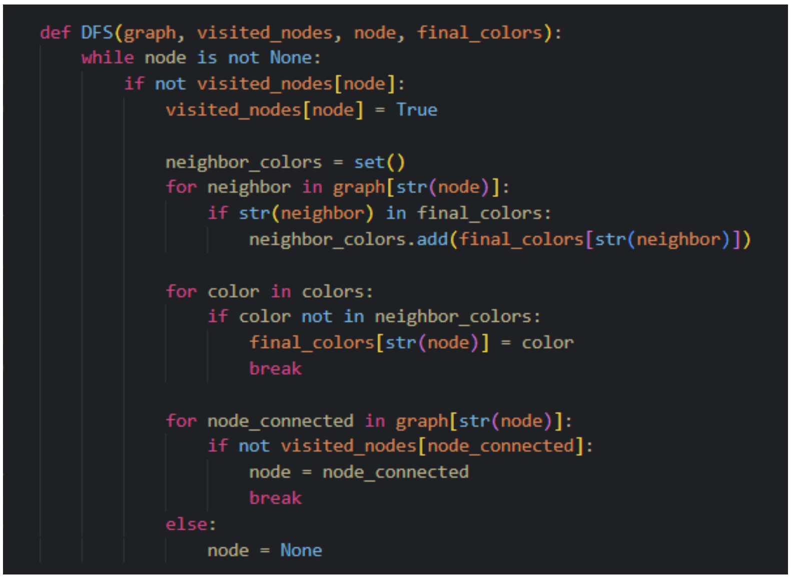
n	t Colouring (ms)
8	0,015
16	0,027
32	0,054
64	0,126
128	0,325
256	0,585
512	1,144
1024	2,839
2048	5,016
4096	10,432
8192	20,707
16384	45,574
32768	95,583
65536	209,302

The given algorithm is a greedy approach for graph coloring using DFS traversal. The greedy() function iterates over all nodes and calls DFS() for unvisited ones, leading to an overall DFS traversal time of O(n+m), where n is the number of nodes and m is the number of edges. Within DFS, each node processes its neighbors to determine the first available color, which takes O(d), where d is the node's degree. In the worst case (graphs strongly connected where m=O(n2), a node may check up to O(n) neighbours, making the coloring process O(n2). Thus, the worst-case complexity is O(n^2). However, in average graphs the complexity would be O(n).



```python
def greedy(graph):
    n_nodes = len(graph)
    visited_nodes = [False] * n_nodes
    final_colors = {}

    for i in range(n_nodes):
        if not visited_nodes[i]:
            DFS(graph, visited_nodes, i, final_colors)
    return final_colors
```



```python
def DFS(graph, visited_nodes, node, final_colors):
    while node is not None:
        if not visited_nodes[node]:
            visited_nodes[node] = True

            neighbor_colors = set()
            for neighbor in graph[str(node)]:
                if str(neighbor) in final_colors:
                    neighbor_colors.add(final_colors[str(neighbor)])

            for color in colors:
                if color not in neighbor_colors:
                    final_colors[str(node)] = color
                    break

            for node_connected in graph[str(node)]:
                if not visited_nodes[node_connected]:
                    node = node_connected
                    break
            else:
                node = None
```