```
1
2  # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  # Path   : uebung12/al/aufgabe02
4  # Version: Fri Dec  1 17:48:30 CET 2023
5
6  import sys
7
8  from uebung12.graphs.graph import Graph
9  from uebung12.graphs.graph_adv import GraphADV
10 from uebung12.al.aufgabe02.directed_dfs import DirectedDFS
11
12
13 if __name__ == '__main__':
14
15   print("Example this exercise 12, task 1c:\n")
16
17   graph = Graph() # without ADV
18   #graph = GraphADV() # with ADV
19
20   v_01 = graph.insert_vertex("01")
21   v_02 = graph.insert_vertex("02")
22   v_03 = graph.insert_vertex("03")
23   v_04 = graph.insert_vertex("04")
24   v_05 = graph.insert_vertex("05")
25   v_06 = graph.insert_vertex("06")
26   v_07 = graph.insert_vertex("07")
27   v_08 = graph.insert_vertex("08")
28   v_09 = graph.insert_vertex("09")
29   v_10 = graph.insert_vertex("10")
30
31   graph.insert_edge(v_01, v_08)
32   graph.insert_edge(v_01, v_10)
33   graph.insert_edge(v_02, v_06)
34   graph.insert_edge(v_03, v_01)
35   graph.insert_edge(v_03, v_07)
36   graph.insert_edge(v_03, v_06)
37   graph.insert_edge(v_04, v_10)
38   graph.insert_edge(v_04, v_05)
39   graph.insert_edge(v_05, v_10)
40   graph.insert_edge(v_06, v_07)
41   graph.insert_edge(v_07, v_08)
42   graph.insert_edge(v_07, v_02)
43   graph.insert_edge(v_08, v_09)
44   graph.insert_edge(v_09, v_08)
45   graph.insert_edge(v_10, v_03)
46
47   directed_dfs = DirectedDFS()
48   directed_dfs.search(graph)
49
50   directed_dfs.print_maps()
51
52   if len(directed_dfs._edge_map) != 15:
53     print("\nERROR: DirectedDFS.edgeMap should have a size of 15 !")
54     sys.exit(11)
55
```



```
56
57 """ Session-Log:
58
59 Example this exercise 12, task 1c:
60
61 DirectedDFS.search() : 01
62 Testing              : 01-08: DISCOVERY
63 DirectedDFS.search() : 08
64 Testing              : 08-09: DISCOVERY
65 DirectedDFS.search() : 09
66 Testing              : 09-08: BACK
67 Testing              : 01-10: DISCOVERY
68 DirectedDFS.search() : 10
69 Testing              : 10-03: DISCOVERY
70 DirectedDFS.search() : 03
71 Testing              : 03-01: BACK
72 Testing              : 03-06: DISCOVERY
73 DirectedDFS.search() : 06
74 Testing              : 06-07: DISCOVERY
75 DirectedDFS.search() : 07
76 Testing              : 07-02: DISCOVERY
77 DirectedDFS.search() : 02
78 Testing              : 02-06: BACK
79 Testing              : 07-08: CROSS
80 Testing              : 03-07: FORWARD
81 DirectedDFS.search() : 04
82 Testing              : 04-05: DISCOVERY
83 DirectedDFS.search() : 05
84 Testing              : 05-10: CROSS
85 Testing              : 04-10: CROSS
86
87 DirectedDFS.print_maps():
88 Vertex-Map : {01=VISITED, 02=VISITED, 03=VISITED, 04=VISITED, 05=VISITED, 06=VISITED,
   07=VISITED, 08=VISITED, 09=VISITED, 10=VISITED}
89 Edge-Map   : {01-08=DISCOVERY, 01-10=DISCOVERY, 02-06=BACK, 03-01=BACK, 03-06=DISCOVER
   Y, 03-07=FORWARD, 04-05=DISCOVERY, 04-10=CROSS, 05-10=CROSS, 06-07=DISCOVERY, 07-02=DI
   SCOVERY, 07-08=CROSS, 08-09=DISCOVERY, 09-08=BACK, 10-03=DISCOVERY}
90
91 """
```

```
1
2  # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  # Path   : uebung12/al/aufgabe02
4  # Version: Fri Dec  1 17:48:30 CET 2023
5
6  import enum
7
8
9  class DirectedDFS:
10
11   class _VertexLabel(enum.Enum):
12     UNEXPLORED = enum.auto()
13     VISITED = enum.auto()
14
15   class _EdgeLabel(enum.Enum):
16     UNEXPLORED = enum.auto()
17     DISCOVERY = enum.auto()
18     BACK = enum.auto()
19     FORWARD = enum.auto()
20     CROSS = enum.auto()
21
22   def __init__(self):
23     self._vertex_map = dict()
24     self._edge_map = dict()
25     self._parent_map = dict()
26     # The parent-map maps a child to its parent
27     self._graph = None
28
29   def search(self, graph):
30     self._graph = graph
31     self._vertex_map = graph.get_vertex_map()
32     self._edge_map = graph.get_edge_map()
33
34     # TODO: Implement here...
35
36
37   def _search(self, graph, v):
38     print("{:21s}: {}".format("DirectedDFS.search()", str(v)))
39
40     # TODO: Implement here...
41
42
43   def print_maps(self):
44     self._graph.printing_maps(True)
45     print("\nDirectedDFS.print_maps():")
46     print("Vertex-Map : {", end = "")
47     mappings = list()
48     for v in self._vertex_map:
49       mappings.append(v.__str__() + "=" + self._get_enum_name(self._vertex_map[v]))
50     print(", ".join(mappings), end = "")
51     print("}")
52     print("Edge-Map   : {", end = "")
53     mappings = list()
54     for e in self._edge_map:
55       mappings.append(e.__str__() + "=" + self._get_enum_name(self._edge_map[e]))
56     mappings.sort()
57     print(", ".join(mappings), end = "")
58     print("}")
59     self._graph.printing_maps(False)
60
61   def _get_enum_name(self, enum_value):
62     return enum_value.__str__().split(".")[1]
```