**Kosaraju’s algorithm**

**def** dfs1(self, vertex):  
 self.expl[vertex] = **True  
 for** child **in** self.get\_outbound(vertex):  
 **if** self.expl[child] == **False**:  
 self.dfs1(child)  
  
 self.stack.append(vertex)  
  
**def** dfs2(self, vertex):  
 self.expl[vertex] = **True  
 for** child **in** self.get\_inbound(vertex):  
 **if** self.expl[child] == **False**:  
 self.dfs2(child)  
  
 self.component.append(vertex)  
  
**def** str\_conected(self):  
 *''' Kosaraju's algorithm for strongly connected components'''  
  
 #all the vertices in the graph are marked as not visited/explored* **for** vertex **in** self.graph.get\_vertices():  
 self.expl[vertex] = **False  
  
 for** vertex **in** self.graph.get\_vertices():  
 **if** self.expl[vertex] == **False**:  
 self.dfs1(vertex)  
  
 **for** vertex **in** self.graph.get\_vertices():  
 self.expl[vertex] = **False  
  
 while** self.stack:  
 *#build the strongly connected components* vertex = self.stack[-1]  
 **if** self.expl[vertex] == **False**:  
 self.component.clear()  
 self.dfs2(vertex)  
 self.str\_conn.append(deepcopy(self.component))  
  
 self.stack.pop(-1)  
  
 i = 1  
 *#printing the strongly connected components* **for** comp **in** self.str\_conn:  
 comp.sort()  
 print(**"component: "**, i)  
 print(comp)  
 **for** v1 **in** comp:  
 **for** v2 **in** comp:  
 **if** v1 != v2:  
 **try**:  
 *#printing the edges in a strongly connected component* self.validator.valid\_edge(v1, v2, self.graph.get\_out())  
 print(**"edge: "**, v1, **"->"**, v2)  
 **except** ValueError **as** ve:  
 **continue** i += 1  
  
 self.str\_conn.clear()  
 self.expl.clear()  
 self.stack.clear()