**class** Graph(object):  
  
 **def** \_\_init\_\_(self, n):  
 *"""  
 Constructor for class Graph.* **:param** *n: an integer positive number  
 """* self.\_\_in = {}  
 self.\_\_out = {}  
 self.\_\_edges = {}  
 **for** i **in** range(0, n):  
 self.\_\_in[i] = []  
 self.\_\_out[i] = []  
  
 **def** get\_in(self):  
 *'''  
 Method to return the inbound  
 dictionary.* **:return***: inbound dictionary  
 '''* **return** self.\_\_in  
  
 **def** get\_out(self):  
 *'''  
 Method to return the outbound  
 dictionary.* **:return***: outbound dictionary  
 '''* **return** self.\_\_out  
  
 **def** get\_edges(self):  
 *'''  
 Method to return the edges'  
 dictionary.* **:return***: edges' dictionary  
 '''* **return** self.\_\_edges  
  
 **def** add\_vertex(self, vertex):  
 *'''  
 Method to add a vertex.* **:param** *vertex: positive integer* **:return***:  
 '''* self.\_\_in[vertex] = []  
 self.\_\_out[vertex] = []  
  
 **def** add\_edge(self, x, y, e):  
 *'''  
 method to add an edge.* **:param** *x: positive integer* **:param** *y: positive integer* **:param** *e: integer* **:return***:  
 '''* self.\_\_out[x].append(y)  
 self.\_\_in[y].append(x)  
 self.\_\_edges[(x, y)] = e  
  
 **def** is\_edge(self, x, y):  
 *'''  
 This function checks if an edge  
 exists or not. It returns true if it  
 exists and false otherwise* **:param** *x: positive integer* **:param** *y: positive integer* **:return***: true/false  
 '''* **if** x **in** self.\_\_in[y]:  
 **return True  
 else**:  
 **return False  
  
 def** set\_edge(self, v1, v2, cost):  
 *'''  
 Method to update an edge.* **:param** *v1: positive integer* **:param** *v2: positive integer* **:param** *cost: integer* **:return***:  
 '''* self.\_\_edges[(v1, v2)] = cost  
  
 **def** remove\_vertex(self, vertex):  
 *'''  
 Method to remove a vertex.* **:param** *vertex: positive integer* **:return***:  
 '''* **for** i **in** self.\_\_out:  
 **if** (vertex, i) **in** self.\_\_edges:  
 self.\_\_edges.pop((vertex, i))  
 **if** (i, vertex) **in** self.\_\_edges:  
 self.\_\_edges.pop((i, vertex))  
 **if** vertex **in** self.\_\_in[i]:  
 (self.\_\_in[i]).remove(vertex)  
 **if** vertex **in** self.\_\_out[i]:  
 (self.\_\_out[i]).remove(vertex)  
  
 self.\_\_in.pop(vertex)  
 self.\_\_out.pop(vertex)  
  
 **def** remove\_edge(self, v1, v2):  
 *"""  
 Method to remove an edge.* **:param** *x: positive integer* **:param** *y: positive integer* **:return***: -  
 """* self.\_\_edges.pop((v1, v2))  
 self.\_\_out[v1].remove(v2)  
 self.\_\_in[v2].remove(v1)