Nama : Wahyu Miftahul aflah

NIM : 1910511060

Kelas : B

Suatu **graph** didefinisikan oleh himpunan verteks dan himpunan sisi (edge). keterhubungan antara verteks. Biasanya untuk suatu graph G digunakan notasi matematis. Verteks menyatakan entitas-entitas data dan sisi menyatakan $G = (V, E)$ Dimana :

G = Graph

V = Simpul atau Vertex, atau Node, atau Titik

E = Busur atau Edge, atau arc

OPERASI DALAM GRAF

Misalkan graf G adalah graf dengan himpunan titik V(G) dan himpunan sisi X(G), serta H adalah graf dengan himpunan titik V(H) dan himpunan sisi X(H) . Maka:

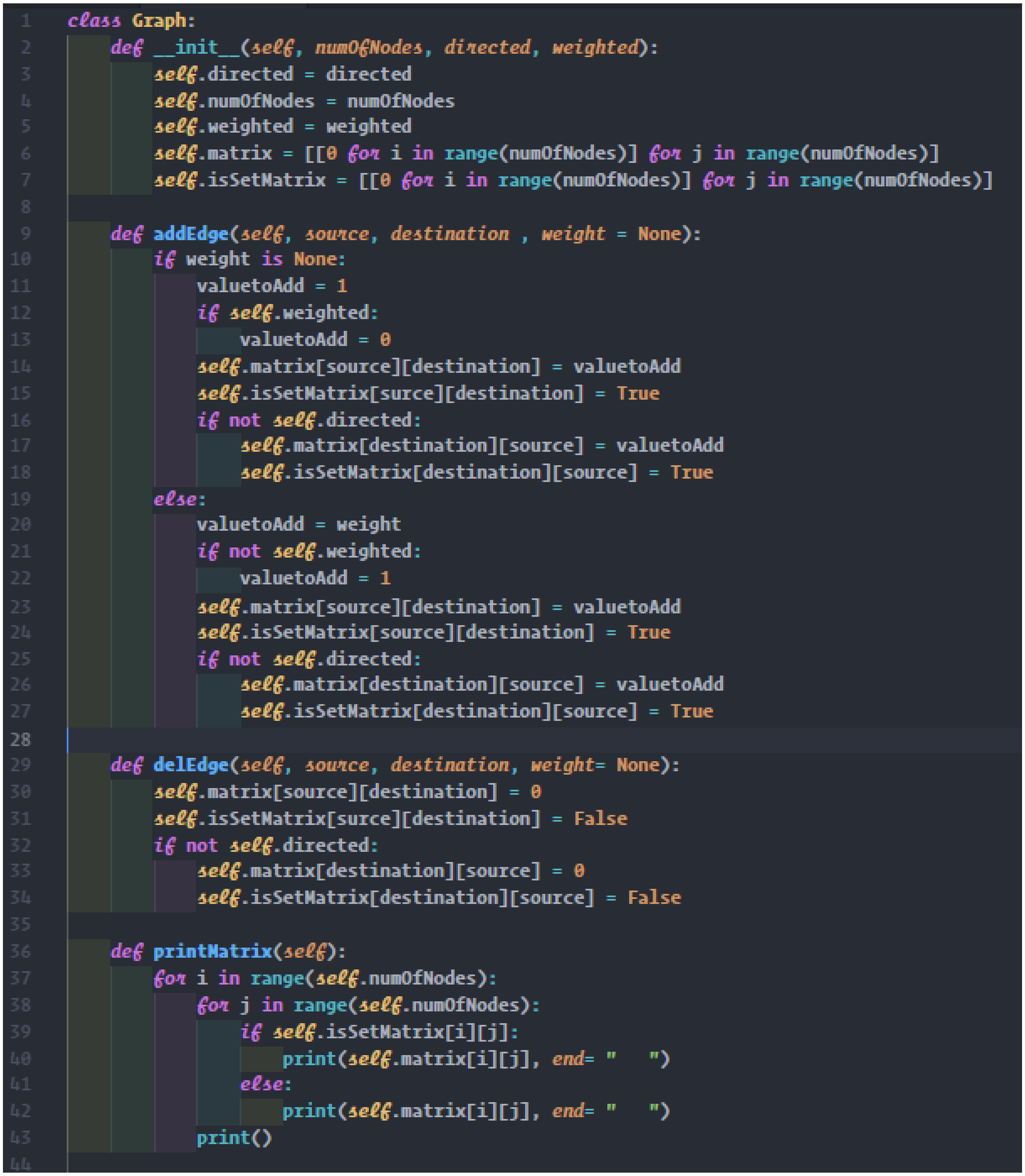
1. Graf gabung (union graph) antara G dan H ditulis $G \cup H$, adalah graf dengan himpunan titik $V(G \cup H) = V(G) \cup V(H)$ dan himpunan sisi $X(G \cup H) = X(G) \cup X(H)$;
2. Graf tambah (join graph) antara G dan H ditulis G+H, adalah graf dengan himpunan titik $V(G+H) = V(G) \cup V(H)$ dan himpunan sisi $X(G+H) = X(G) \cup X(H) \cup \{uv: u\ V(G), v\ V(H) \}$.
3. Graf kali (GxH) adalah graf dengan himpunan titik $V(GxH) = V(G) \cup V(H)$ dan himpunan sisi $E(GxH) = V(G) \cup V(H)$

JENIS JENIS GRAF

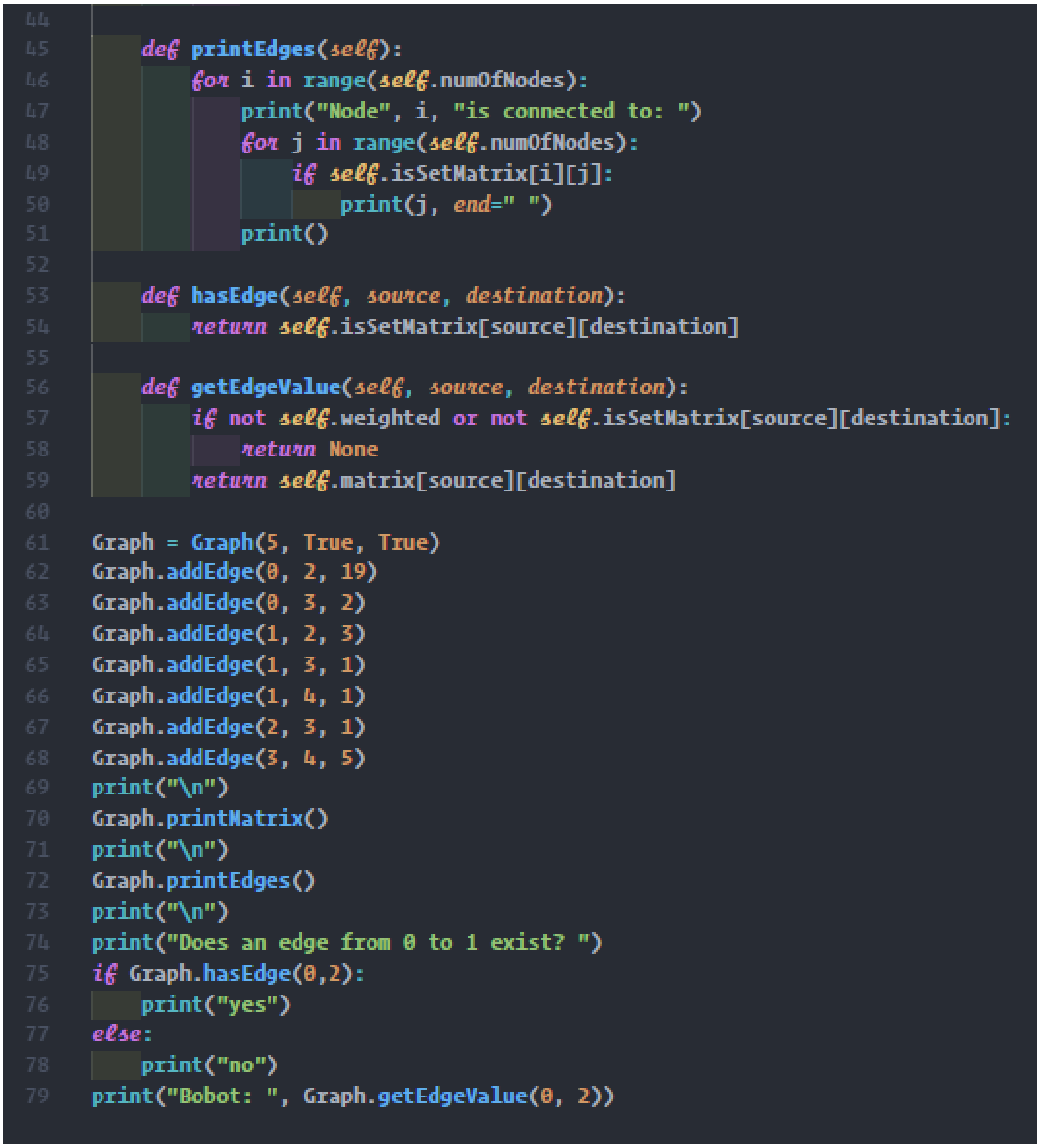
- Directed graph (edge memiliki sisi satu arah)
- Undirected graph (edge memiliki sisi banyak arah)

IMPLEMENTASI GRAF

Source code

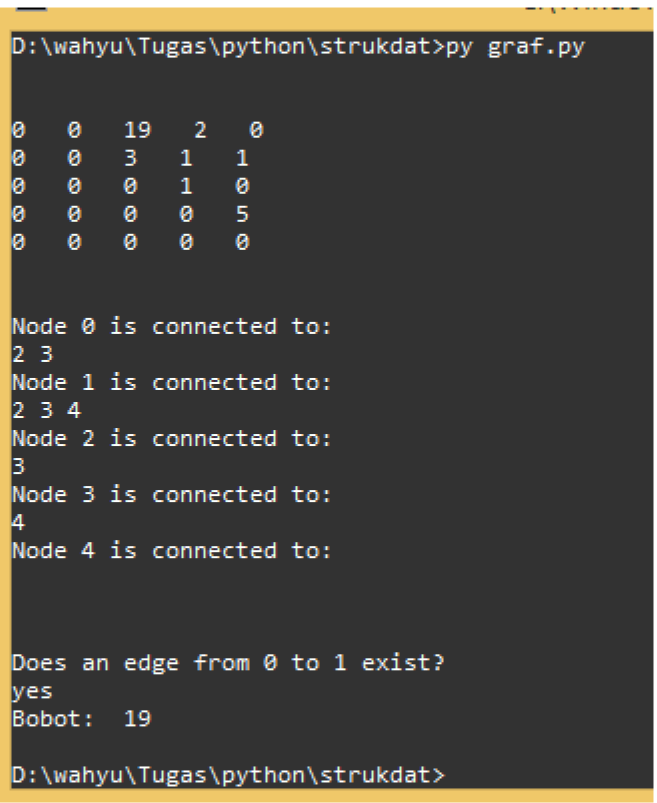


```
class Graph:
    def __init__(self, numOfNodes, directed, weighted):
        self.directed = directed
        self.numOfNodes = numOfNodes
        self.weighted = weighted
        self.matrix = [[0 for i in range(numOfNodes)] for j in range(numOfNodes)]
        self.isSetMatrix = [[0 for i in range(numOfNodes)] for j in range(numOfNodes)]

    def addEdge(self, source, destination , weight = None):
        if weight is None:
            valuetoAdd = 1
            if self.weighted:
                valuetoAdd = 0
            self.matrix[source][destination] = valuetoAdd
            self.isSetMatrix[surce][destination] = True
            if not self.directed:
                self.matrix[destination][source] = valuetoAdd
                self.isSetMatrix[destination][source] = True
        else:
            valuetoAdd = weight
            if not self.weighted:
                valuetoAdd = 1
            self.matrix[source][destination] = valuetoAdd
            self.isSetMatrix[source][destination] = True
            if not self.directed:
                self.matrix[destination][source] = valuetoAdd
                self.isSetMatrix[destination][source] = True

    def delEdge(self, source, destination, weight= None):
        self.matrix[source][destination] = 0
        self.isSetMatrix[surce][destination] = False
        if not self.directed:
            self.matrix[destination][source] = 0
            self.isSetMatrix[destination][source] = False

    def printMatrix(self):
        for i in range(self.numOfNodes):
            for j in range(self.numOfNodes):
                if self.isSetMatrix[i][j]:
                    print(self.matrix[i][j], end= "   ")
                else:
                    print(self.matrix[i][j], end= "   ")
            print()
```



```

    def printEdges(self):
        for i in range(self.numOfNodes):
            print("Node", i, "is connected to: ")
            for j in range(self.numOfNodes):
                if self.isSetMatrix[i][j]:
                    print(j, end=" ")
            print()

    def hasEdge(self, source, destination):
        return self.isSetMatrix[source][destination]

    def getEdgeValue(self, source, destination):
        if not self.weighted or not self.isSetMatrix[source][destination]:
            return None
        return self.matrix[source][destination]

Graph = Graph(5, True, True)
Graph.addEdge(0, 2, 19)
Graph.addEdge(0, 3, 2)
Graph.addEdge(1, 2, 3)
Graph.addEdge(1, 3, 1)
Graph.addEdge(1, 4, 1)
Graph.addEdge(2, 3, 1)
Graph.addEdge(3, 4, 5)
print("\n")
Graph.printMatrix()
print("\n")
Graph.printEdges()
print("\n")
print("Does an edge from 0 to 1 exist? ")
if Graph.hasEdge(0,2):
    print("yes")
else:
    print("no")
print("Bobot: ", Graph.getEdgeValue(0, 2))
```

Output



```
D:\wahyu\Tugas\python\strukdat>py graf.py


0   0   19   2   0
0   0   3   1   1
0   0   0   1   0
0   0   0   0   5
0   0   0   0   0


Node 0 is connected to:
2 3
Node 1 is connected to:
2 3 4
Node 2 is connected to:
3
Node 3 is connected to:
4
Node 4 is connected to:



Does an edge from 0 to 1 exist?
yes
Bobot:  19

D:\wahyu\Tugas\python\strukdat>
```