Praktikum Kecerdasan Buatan
Algoritma Pencarian
Breadth First Search dan Depth First Search

Dosen Pengampu
Entin Martiana Kusumaningtyas S.Kom, M.kom

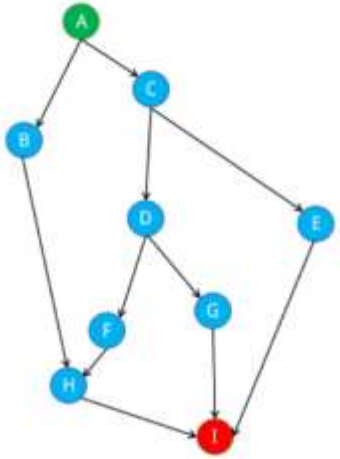
Ratri Maria Manik
3121600039
2 D4 IT - B

O1

Tugas Praktikum

1. Buatlah coding algoritma:
   - Breadth First Search
   - Depth First Search

   Untuk Graph disamping!
2. Tampilkan Traversal atau perjalanan dari Node A ke Node I



Breadth First Search

```
graph = {
  'A' : ['B','C'],
  'B' : ['H'],
  'C' : ['D', 'E'],
  'D' : ['F','G'],
  'E' : ['I'],
  'F' : ['H'],
  'G' : ['I'],
  'H' : ['I'],
  'I' : []
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")
    if(m == 'I'):
      break
    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, 'A')    # function calling
```

```
Following is the Breadth-First Search
A B C H D E I
```

Depth First Search

```
import sys, traceback

# Using a Python dictionary to act as an adjacency list
graph = {
  'A' : ['B','C'],
  'B' : ['H'],
  'C' : ['D', 'E'],
  'D' : ['F','G'],
  'E' : ['I'],
  'F' : ['H'],
  'G' : ['I'],
  'H' : ['I'],
  'I' : []
}

visited = set() # Set to keep track of visited nodes of graph.
```

```
def dfs(visited, graph, node, nodeEnd):  #function for dfs
    if node not in visited:
        print (node)
        visited.add(node)
        if node == nodeEnd:
            sys.exit("Sudah menemukan tujuan")
        for neighbour in graph[node]:
          dfs(visited, graph, neighbour, nodeEnd)


# Driver Code
print("Following is the Depth-First Search")
dfs(visited, graph, 'A', 'I')
```

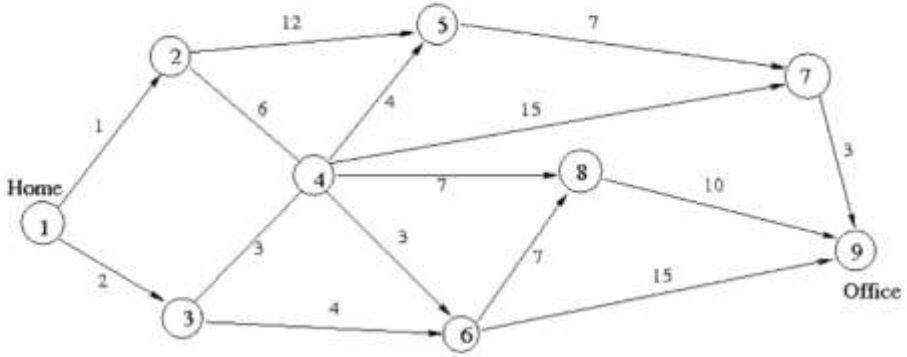
```
Following is the Depth-First Search
A
B
H
I
An exception has occurred, use %tb to see the full traceback.

SystemExit: Sudah menemukan tujuan
```

Tugas Praktikum

1. Buatlah coding algoritma:
   - Breadth First Search
   - Depth First Search

   Untuk Graph disamping!
2. Tampilkan Traversal atau perjalanan dari Home ke Office

Figure 1



Breadth First Search

```
graph = {
  '1' : ['2','3'],
  '2' : ['4','5'],
  '3' : ['4','6'],
  '4' : ['5','6','8'],
  '5' : ['7'],
  '6' : ['8','9'],
  '7' : ['9'],
  '8' : ['9'],
  '9' : [],
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")
```

```
    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '1')    # function calling
```

```
Following is the Breadth-First Search
1 2 3 4 5 6 8 7 9
```

Depth First Search

```
# Using a Python dictionary to act as an adjacency list
graph = {
  '1' : ['2','3'],
  '2' : ['4','5'],
  '3' : ['4','6'],
  '4' : ['5','6','8'],
  '5' : ['7'],
  '6' : ['8','9'],
  '7' : ['9'],
  '8' : ['9'],
  '9' : [],
}

visited = set() # Set to keep track of visited nodes of graph.

def dfs(visited, graph, node):  #function for dfs
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
          dfs(visited, graph, neighbour)


# Driver Code
print("Following is the Depth-First Search")
dfs(visited, graph, '1')
```

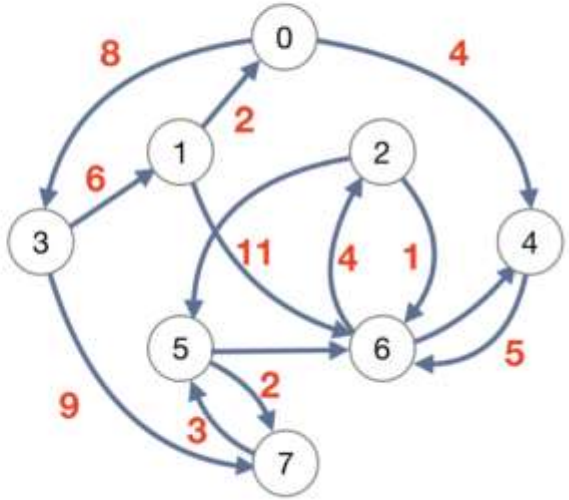
```
Following is the Depth-First Search
1
2
4
5
7
9
6
8
3
```

03

Tugas Praktikum

1. Buatlah coding algoritma:
   - Breadth First Search
   - Depth First Search

   Untuk Graph disamping!
2. Tampilkan Traversal atau perjalanan dari Node-node tersebut



Breadth First Search

```
graph = {
  '0' : ['3','4'],
  '1' : ['0','6'],
  '2' : ['5','6'],
  '3' : ['1','7'],
  '4' : ['6'],
  '5' : ['6','7'],
  '6' : ['2','4'],
  '7' : ['5'],
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)
```

```
# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '0')    # function calling
```

```
Following is the Breadth-First Search
0 3 4 1 7 6 5 2
```

Depth First Search

```
# Using a Python dictionary to act as an adjacency list
graph = {
  '0' : ['3','4'],
  '1' : ['0','6'],
  '2' : ['5','6'],
  '3' : ['1','7'],
  '4' : ['6'],
  '5' : ['6','7'],
  '6' : ['2','4'],
  '7' : ['5'],
}

visited = set() # Set to keep track of visited nodes of graph.

def dfs(visited, graph, node):  #function for dfs
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
          dfs(visited, graph, neighbour)


# Driver Code
print("Following is the Depth-First Search")
dfs(visited, graph, '0')
```

```
Following is the Depth-First Search
0
3
1
6
2
5
7
4
```