LAPORAN PRAKTIKUM KECERDASAN BUATAN

JOB SHEET 4
PROBLEM SOLVING PART 1

Oleh:
FAREL PUTRA HIDAYAT
NIM. 2041723016

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
SEPTEMBER 2020

1.3.3 Pertanyaan Percobaan 1

1. Langkah ketiga menjelaskan pembuatan function bfs yang memiliki tiga parameter. Selanjutnya variable visited dan queue ditambahkan objek baru bernama node dengan menggunakan fungsi pop dan append. Membuat looping menggunakan while dengan kondisi menggunakan queue, jika queue null maka looping akan dihentikan. Membuat sebuah variable s yang berisi data teratas dari queue. Setelah itu melakukan cetak value dari variable s yang merupakan node yang telah di kunjungi. Membuat looping menggunakan for dengan batasan loopingnya jumlah neighbour didalam graph. Pada looping for tedapat pengecekan kondisi menggunakan logika percabangan if dimana jika kondisi neighbour belum dikunjungi maka neighbour akan dimasukkan ke dalam visited dan queue. Jika semua kondisi terpenuhi maka function bfs selesai.
2. Output yang dihasilkan akan berubah, Karena pada node E dihubungkan dengan node G yang bukan merupakan urutan dari node F yang dimana seharusnya node F terhubung dengan node H.

1.4.3 Pertanyaan Percobaan 2

1. Langkah ketiga menjelaskan pembuatan function dfs yang memiliki tiga parameter. Selanjutnya dilakukan logika percabangan if yang berfungsi untuk melakukan pengecekan kondisi jika node belum dikunjungi maka lakukan cetak node yang merupakan root. Selanjutnya variable visited ditambahkan objek baru bernama node dengan menggunakan fungsi add dan append. Didalam logika if terdapat looping menggunakan for yang memiliki kondisi lakukan looping neighbour sampai jumlah graph. Dalam looping tersebut terdapat pemanggilan function itu sendiri.
2. Output yang dihasilkan akan berbeda, karena saat mengunjungi node B maka neighbour yang akan terpilih untuk dikunjungi yaitu node C. Pada neighbour node C yaitu node F dan seterusnya.

1.5 Latihan Praktikum

1. Telusuri graf berikut menggunakan:
   a. BFS

In [1]:
```
graph = {
    'A' : ['E', 'F','B'],
    'B' : ['F', 'G'],
    'C' : ['D', 'G'],
    'D' : ['H'],
    'E' : ['F'],
    'F' : [],
    'G' : ['C', 'H'],
    'H' : ['G', 'D']
}

visited = []
queue = []

def bfs(visited, graph, node):
    visited.append(node)
    queue.append(node)

    while queue:
        s = queue.pop(0)
        print (s, end = " ")

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

print("Hasil penelusuran graf menggunakan BFS:")
bfs(visited, graph, 'A')
```

```
Hasil penelusuran graf menggunakan BFS:
A E F B G C H D
```

   b. DFS

In [2]:
```
graph = {
    'A' : ['E', 'F','B'],
    'B' : ['F', 'G'],
    'C' : ['D', 'G'],
    'D' : ['H'],
    'E' : ['F'],
    'F' : [],
    'G' : ['C', 'H'],
    'H' : ['G', 'D']
}

visited = set()

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

print("Hasil penelusuran graf menggunakan DFS:")
dfs(visited, graph, 'A')
```

```
Hasil penelusuran graf menggunakan DFS:
A
E
F
B
G
C
D
H
```

2. Buatlah program untuk menelusuri mencari huruf ‘d’ pada graf berikut menggunakan:
   a. BFS

In [3]:
```
graph = {
    'A' : ['B'],
    'B' : ['F', 'G'],
    'C' : ['D'],
    'D' : ['H'],
    'E' : [],
    'F' : ['E'],
    'G' : ['C', 'H'],
    'H' : []
}

visited = []
queue = []

def bfs(visited, graph, node, search):
    visited.append(node)
    queue.append(node)

    while queue:
        s = queue.pop(0)
        print (s, end = " ")

        for neighbour in graph[s]:
            if search not in visited:
                visited.append(neighbour)
                queue.append(neighbour)

print("Hasil penelusuran graf menggunakan BFS:")
bfs(visited, graph, 'A', 'D')
```

```
Hasil penelusuran graf menggunakan BFS:
A B F G E C H D
```

   b. DFS

In [7]:
```
graph = {
    'A' : ['B'],
    'B' : ['F', 'G'],
    'C' : ['D'],
    'D' : ['H'],
    'E' : [],
    'F' : ['E'],
    'G' : ['C', 'H'],
    'H' : []
}

visited = set()

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

print("Hasil penelusuran graf menggunakan DFS:")
dfs(visited, graph, 'A', 'D')
```

```
Hasil penelusuran graf menggunakan DFS:
A
B
F
E
G
C
D
```

c. Mana yang lebih efektif? Mengapa?
   Menurut pendapat saya yang lebih pencarian yang lebih efektif yaitu menggunakan DFS, karena pada kasus tersebut huruf yang ingin dicari yaitu D yang posisinya berada dibawah. Sehingga saat pencarian huruf D dapat lebih cepat ditemukan.