**LAPORAN KECERDASAN BUATAN**

**Jobsheet 4: ProblemSolving With Searching**

Diajukan Untuk Memenuhi Tugas

Mata Kuliah Kecerdasan Buatan

Semester Ganjil Tahun Akademik 2021/2022

****

**Disusun Oleh:**

**Aditya Raihan Setyoputra**

**(2041720236)**

**PROGRAM STUDI D-IV TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2021**

Percobaan 1

Code Program  
graph={

    'A' : ['B','C'],

    'B' : ['D','E'],

    'C' : ['F'],

    'D' : [],

    'E' : ['F'],

    'F' : ['H'],

    'G' : ['H'],

    'H' : ['G']

}

visited = []#List untuk menyimpan node yang dikunjungi

queue = [] #menambahkan queue

def dfs(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)

#Memanggil Fungsi BFS

print("Hasil penelusuran graf menggunakan BFS: ")

dfs(visited,graph,'A')

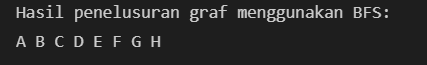
Hasil Percobaan 1



Pertanyaan  
1. Jelaskan detail dari Langkah ketiga pada percobaan 1!

-Pertama membuat variable Graph yang berisikan beberapa index didalamnya,visited untuk menyimpan node yang telah dikunjungi,queue menambahkan list,kemudian membuat fungsi BFS yang berparameter(visited,graph,node).Didalam fungsi tersebut terdapat append untuk visited dan queue menggunakan parameter node yang dimana akan menambahkan nilai nya pada variable visited dan queue.kemudian melakukan while queue pengulanan untuk variable queue dan didalam loop tersebut terdapat variable s untuk menampung nilai dalam queue index 0 dengan menggunakan pop untuk menarik nilai dari queue,kemudian dilakukan print nilai s tadi dan end yang berisikan “(Spasi)”.Setelah itu terdapat loop lagi menggunakan for variable neighbor mengembalikan index/nilai pada graph yang didalamnya menggunakan index berdasarkan variable ‘s’.Jika neighbour tidak sama visited maka dilakukan penambahan pada variable visited dan queue berdasarkan neighbour.

2.Bagaimana hasil penelusuran graf diatas bila node E terhubung dengan node F dan G (neighbour dari E adalah F dan G)? Mengapa?

-Maka Node G akan diprint terlebih dahulu seperti di gambar dibawah: 

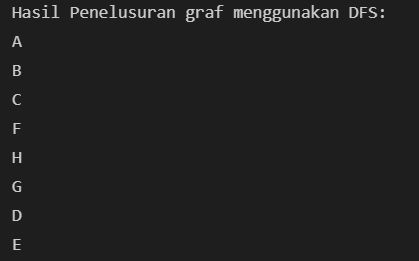
Karena Node yang dieksekusi berdasarkan node dalam node E yang berisikan F dan G

Percobaan 2

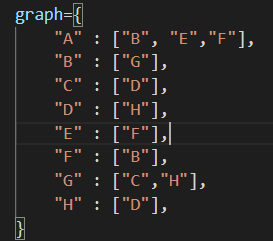
1. Jelaskan secara rinci Langkah ketiga dari percobaan 2!
   * Pertama membuat variable array Graph,kemudian membuat variable visited dengan set

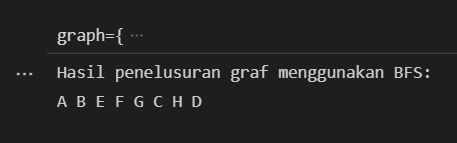
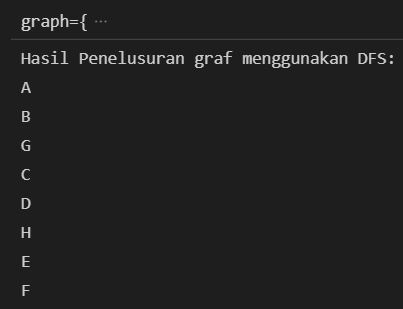
Kemudian membuat fungsi dfs yang berparameter visited graph node,lalu ditemukan code program if untuk menentukan kondisi atau state jika node tidak sama dengan visited lalu dilakukan print(node),kemudian variable visited akan ditambahkan dengan nilai node kemudian dilakukan loop untuk neighbour dan melakukan return in pada graph[node] didalamnya akan memanggil fungsi dfs Kembali untuk menjalankan dari atas akan tetapi mengubah parameter yang digunakan.setelah fungsi dibuat melakukan print dan dibawahnya digunakan untuk memanggil fungsi dfs tadi yang telah dibuat

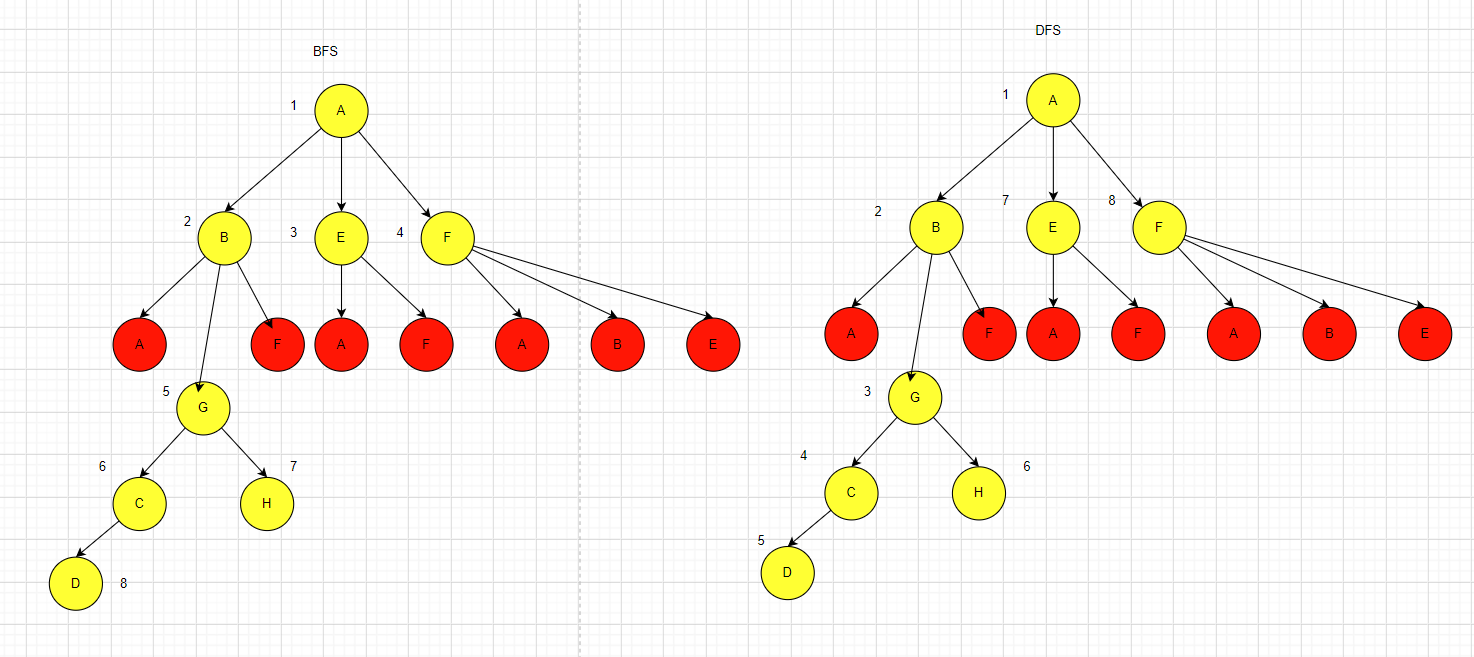
2. Apakah terdapat perbedaan hasil penelusuran bila node B terhubung dengan node C? Mengapa?

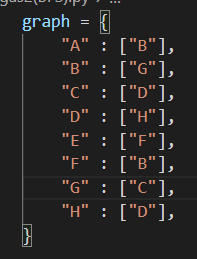
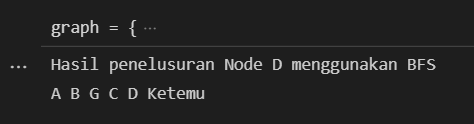
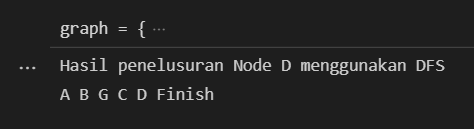
* graph={
* 'A' : ['B','C'],
* 'B' : ['C','D','E'],
* 'C' : ['F'],
* 'D' : [],
* 'E' : ['F'],
* 'F' : ['H'],
* 'G' : ['H'],
* 'H' : ['G']
* }
* Hasil output:
* Berbeda karena node C berada di sisi paling kiri yang bersangkutan dengan node B   
  karena fungsi tersebut setelah memanggil index dari array paling awal kemudian memanggil node array baru dan memanggil index 0 nya

Tugas(Praktikum)

1. 

* 1. 
  2. 



1. 
   1. 
   2. 
   3. DFS karena penerapan yang dilakukan secara rekursif yang dimana tidak perlu menyeleksi secara keseluruhan