**JOBSHEET MATERI 3**

**UI-INFORMED SEARCH**

# 1.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Memahami konsep strategi pencarian uninformed.
2. Memahami manfaat strategi pencarian uninformed.
3. Membangun solusi dari permasalahan yang ada melalui penerapan strategi pencarian.

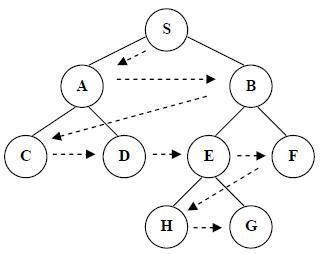
# 1.2 Uninformed Search

Uninformed Search sering disebut juga dengan Blind Search. Istilah tersebut menggambarkan bahwa teknik pencarian ini tidak memiliki informasi tambahan mengenai kondisi diluar dari yang disediakan oleh definisi masalah. Yang dilakukan oleh algoritma ini adalah melakukan generate dari successor dan membedakan goal state dari non-goal state.

Pencarian dilakukan berdasarkan pada urutan mana saja node yang hendak di-expand.

## 1.2.1 Breadth First Search (BFS)

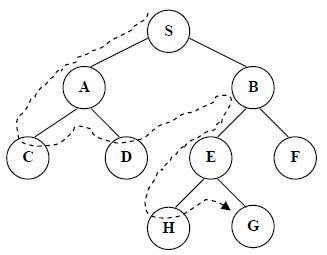
Pencarian dengan Breadth First Search adalah teknik penelusuran data pada semua node dalam suatu level atau satu tingkatan sebelum ke level atau tingkatan di bawahnya. Keuntungan pencarian dengan Teknik ini adalah semua node akan dicek secara menyeluruh pada setiap tingkatan node. Kekurangan teknik ini terletak pada waktu yang dibutuhkan dan sangat lama apabila solusi berada pada posisi tingkatan paling bawah sehingga menjadi tidak efisien. Langkah pertama adalah root node diekspansi, setelah itu dilanjutkan semua successor dari root node juga di-expand. Hal ini terus dilakukan berulang-ulang hingga leaf (node pada level paling bawah yang sudah tidak mempunyai successor lagi).



*Gambar 1 Penelusuran Ekspansi Node pada Breadth First Search*

## 1.2.2 Depth First Search (DFS)

Depth First Search adalah Teknik penelusuran data pada node-node secara vertical dan sudah didefinisikan, misalnya dari kiri ke kanan. Keuntungan pencarian ini adalah penelusuran masalah dapat digali secara mendalam sampai ditemukannya kepastian suatu solusi yag optimal. Kekurangan Teknik ini adalah membutuhkan waktu yang sangat lama untuk ruang lingkup yang besar. Langkah DFS adalah dengan melakukan ekspansi menuju node yang paling dalam pada tree. Node paling dalam dicirikan dengan tidak adanya successor dari node itu. Setelah node itu selesai diekspansi, maka node tersebut akan ditinggalkan, dan dilakukan ke node paling dalam lainnya yang masih memiliki successor yang belum diekspansi.



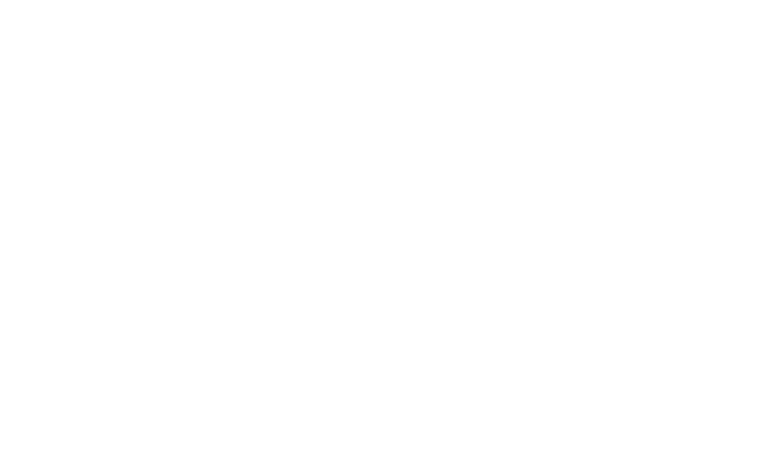
*Gambar 2 Penelusuran Ekspansi Node pada Depth First Search*

# 1.3 Percobaan 1 - Breadth-First Search

Didalam praktikum ini, kita akan mempraktekkan bagaimana membuat graf, membuat fungsi BFS kemudian menggunakan BFS untuk menelusuri graf.

## 1.3.1 Langkah-langkah Percobaan

1. Buat graf sesuai dengan gambar di bawah ini:



A



B



D



C



F



E



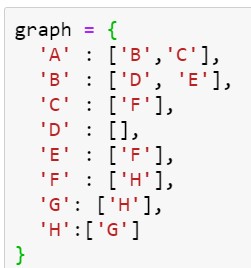
H



G

Syntax pada python untuk pembuatan graf menggunakan adjacency list seperti di bawah

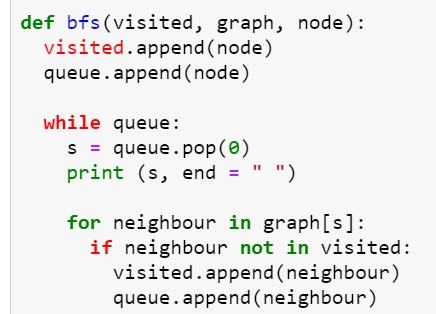
ini



1. Buat Array untuk menyimpan node yang dikunjungi dan menyimpan antrian.



1. Buat fungsi BFS dengan menggunakan queue.

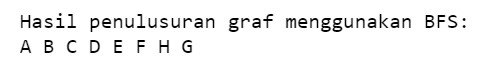


1. Tambahkan kode pemanggilan fugsi BFS. Kemudian klik button Run untuk menjalankan program



## 1.3.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



## 1.3.3 Pertanyaan

1. Jelaskan detail dari Langkah ketiga pada percobaan 1

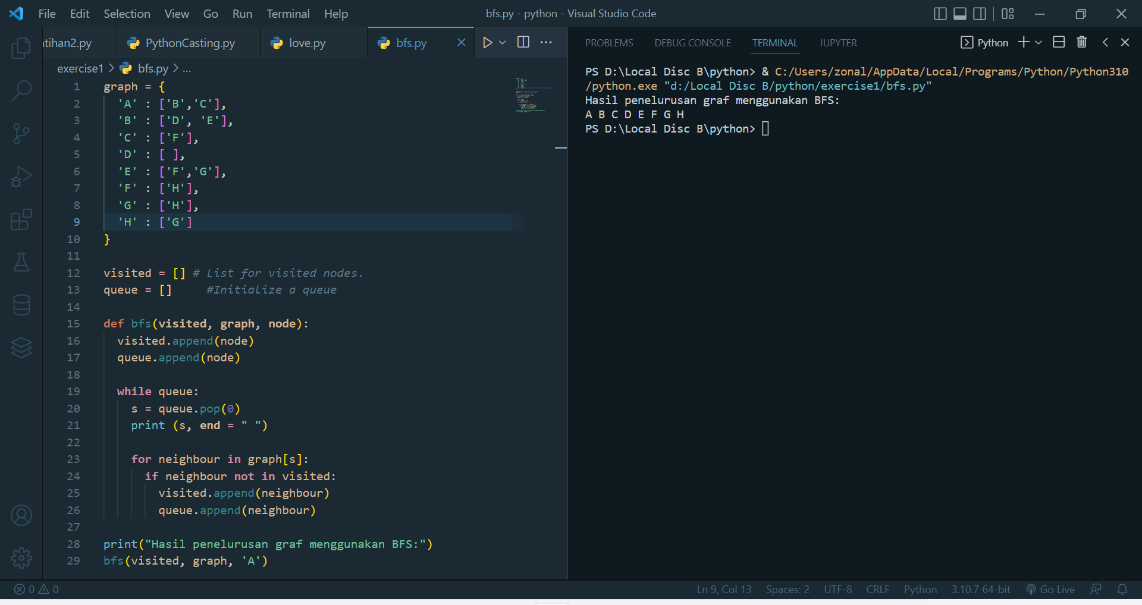
Jawab :

* mendeklarasikan fungsi dengan parameter sebagai node yang dikunjungi grafik itu sendiri dan node masing-masing. Dan di dalam suatu fungsi akan ditambahkan daftar yang telah dikunjungi dan daftar antrian.
* Kemudian menjalankannya dengan while loop untuk antrian pada saat mengunjungi node dan lalu menghapus node yang sama dan mencetaknya saat dikunjungi.
* Terakhir, kita akan menjalankan for loop untuk memeriksa node yang tidak dikunjungi dan kemudian menambahkannya dari daftar yang dikunjungi dan antrian.

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

Jawab : Penelurusan akan menghasilkan sama dengan program sebelumnya.

Karena ketika code progam dijalankan maka akan mengunjungi setiap antrian node dan menghapus node yang sama lalu mencetaknya saat dikunjungi.

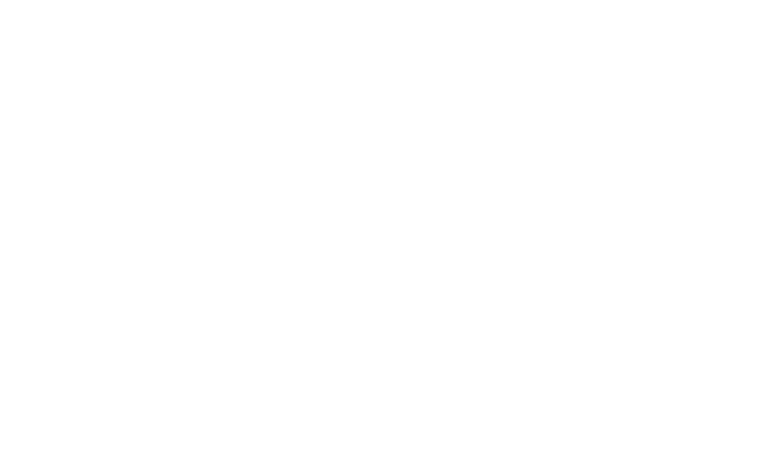
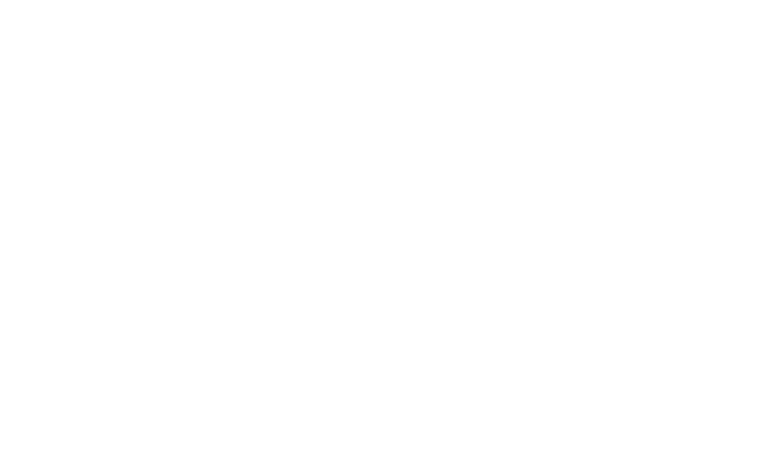


# 1.4 Praktikum 2 - Depth-First Search

Didalam praktikum ini, kita akan mempraktekkan bagaimana membuat graf, membuat fungsi DFS kemudian menggunakan DFS untuk menelusuri graf.

## 1.4.1 Langkah-langkah Percobaan

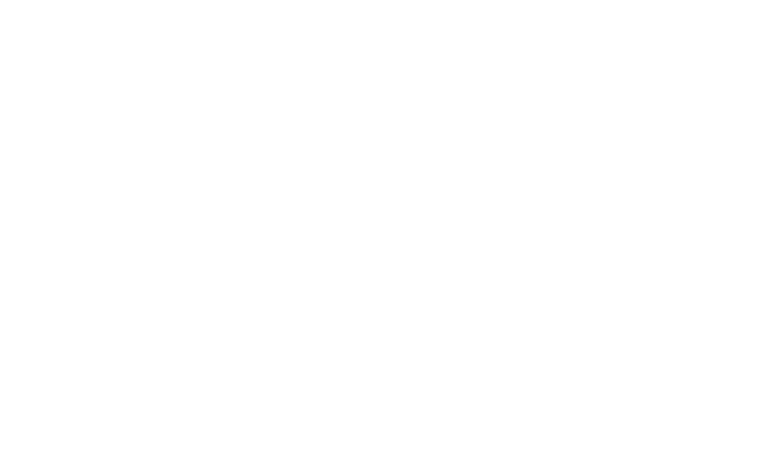
1. Buat graf sesuai dengan gambar di bawah ini:



A



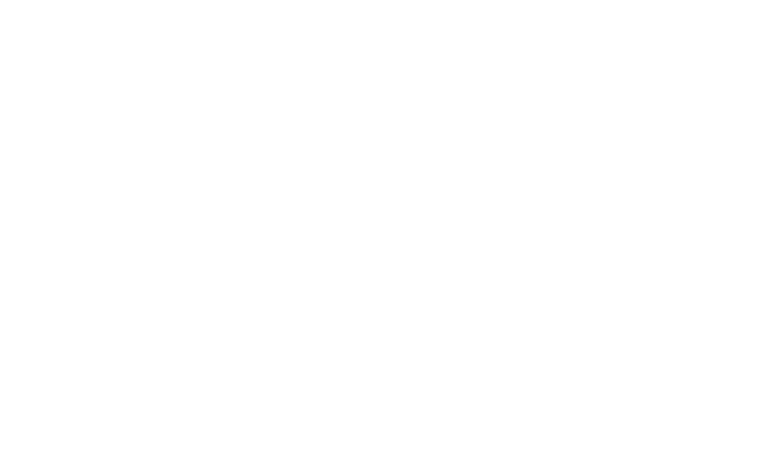
B



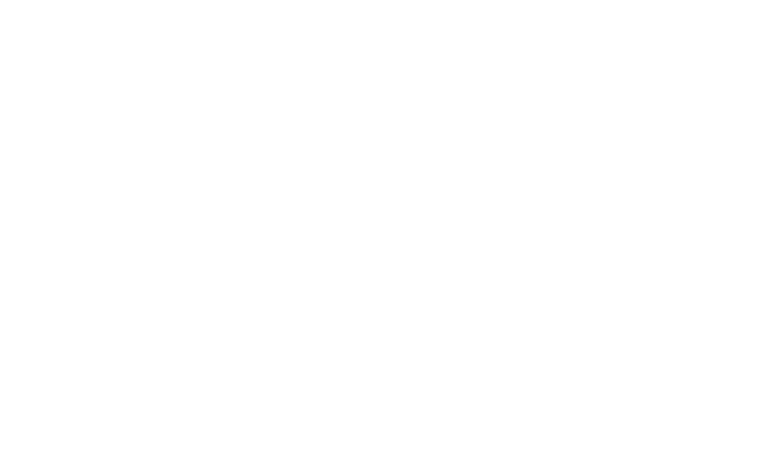
D



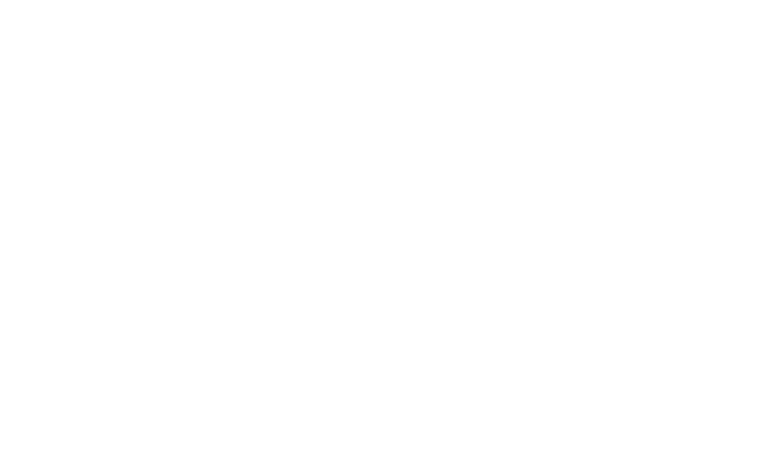
C



F



E



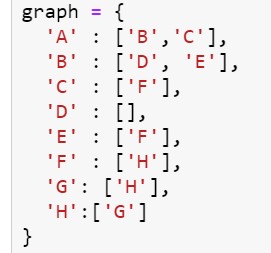
H



G

Syntax pada python untuk pembuatan graf menggunakan adjacency list seperti di bawah

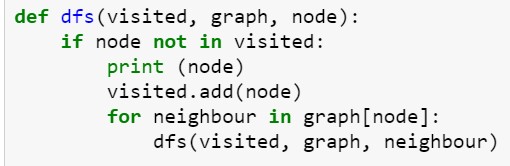
ini



1. Buat Array untk menyimpan node yang dikunjungi.



1. Buat fungsi DFS

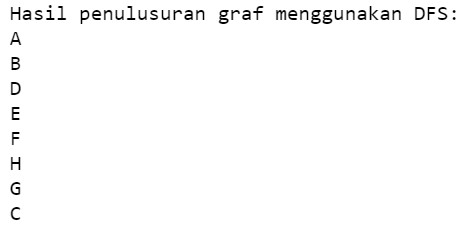


1. Tambahkan kode pemanggilan fugsi DFS. Kemudian klik button Run untuk menjalankan program



## 1.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



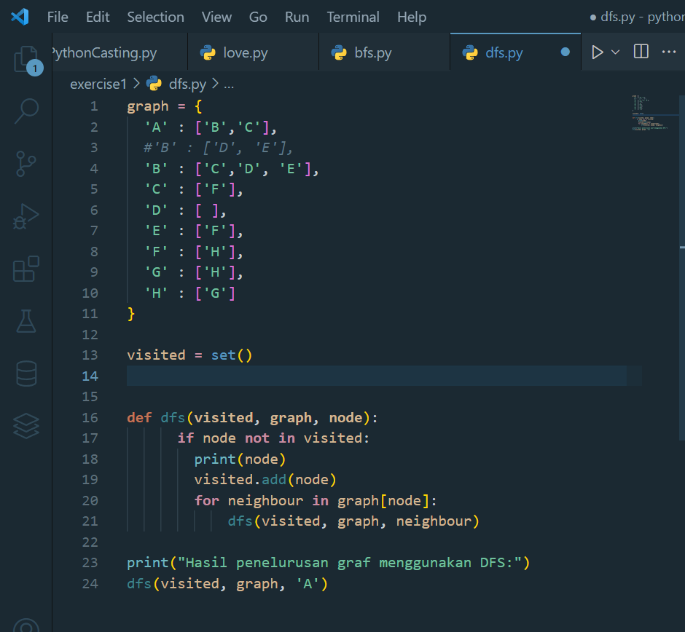
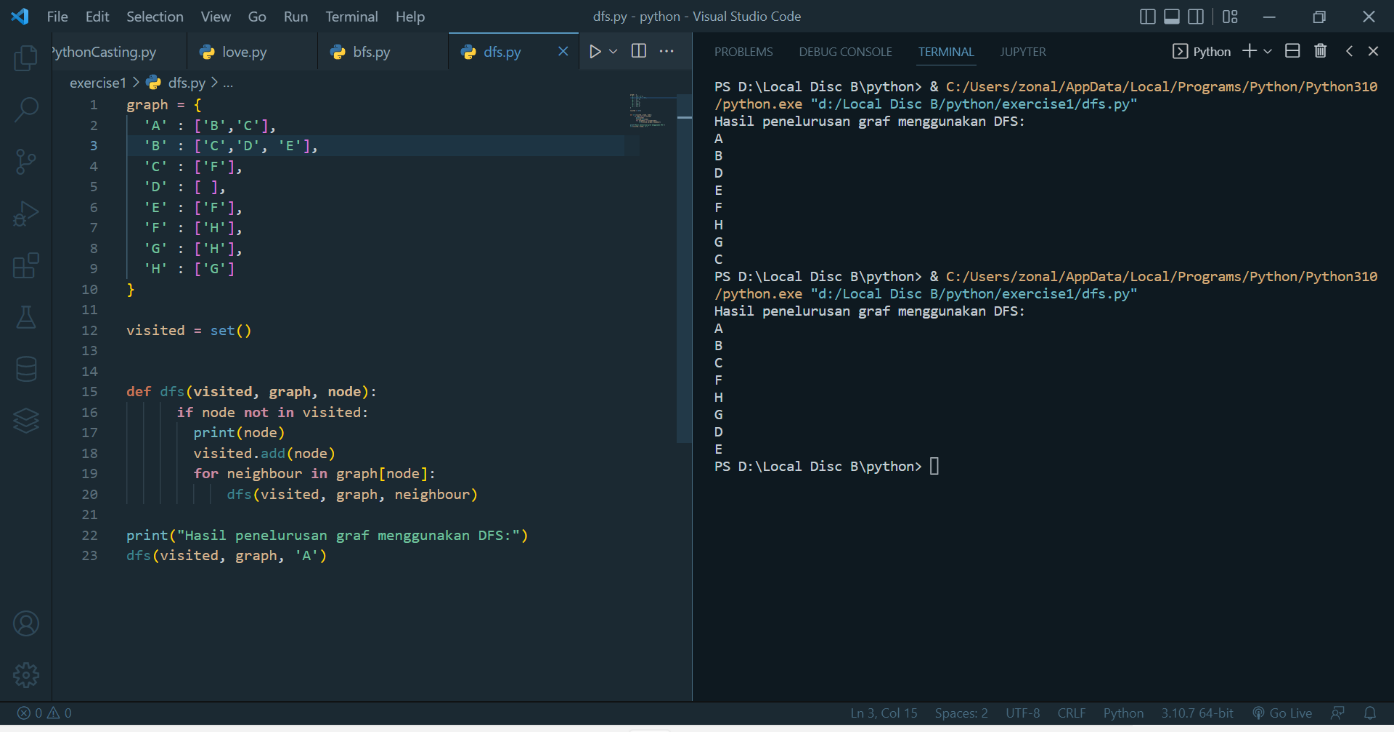
**1.4.3 Pertanyaan**

1. Jelaskan secara rinci Langkah ketiga dari percobaan 2!

Jawab :

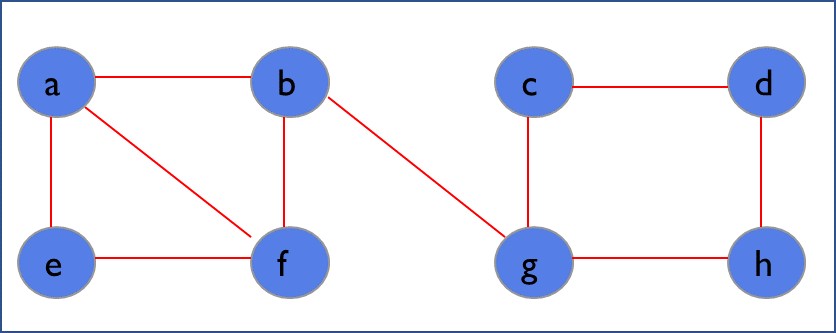
* Mendeklarasikan fungsi paramater sebagai node yang akan dikunjungi masing-masing graph dan node. Setelah itu memeriksa apakah ada simpul dari graph yang telah dikunjungi, yaitu dengan kondisi true or false. Jika tidak, maka akan mencetak node dan menambahkannya ke set node yang telah dikunjungi.
* Kemudian pergi ke node neighbour dari graph dan kembali memanggil fungsi dfs untuk menggunakan parameter neighbour

1. Apakah terdapat perbedaan hasil penelusuran bila node B terhubung dengan node C? Mengapa? Jawab : Iya, ada perbedaan karena pada algoritma DFS akan memeriksa semua simpul dari graph yang telah dikunjungi, yaitu dengan kondisi true or false. Jika tidak, maka akan mencetak node dan menambahkannya ke set node yang telah dikunjungi.



# 1.5 Latihan Praktikum

1. Telusuri graf berikut menggunakan:
   1. BFS
   2. DFS



1. Buatlah program untuk menelusuri mencari huruf ‘d’ pada graf berikut menggunakan: a. BFS
   1. DFS
   2. Mana yang lebih efektif? Mengapa?

