LAPORAN TUGAS STRUKTUR DATA GRAPH

DISUSUN OLEH :

1. Dimas Wijil Pamungkas (23091397201)
2. M. Raka Phaedra Agus Putra (23091397210)
3. Vani Fransiska (23091397193)

DOSEN PENGAMPU :

I Gde Agung Sri Sidhimantra,S.Kom., M.Kom

PRODI D4 MANAJEMEN INFORMATIKA

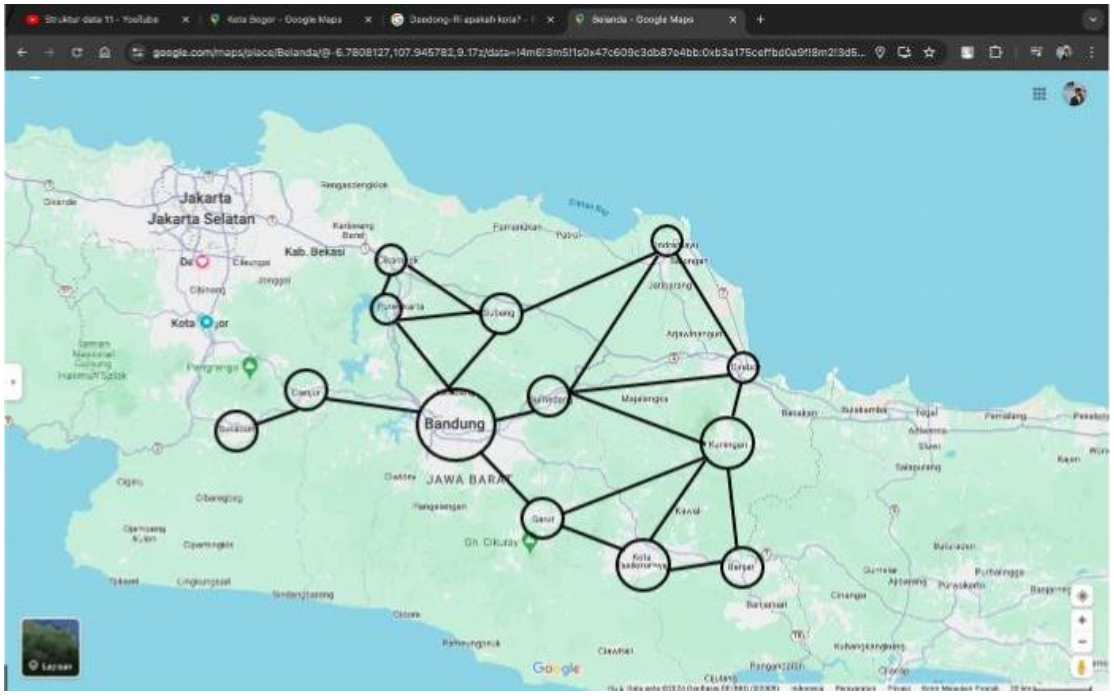
FAKULTAS VOKASI

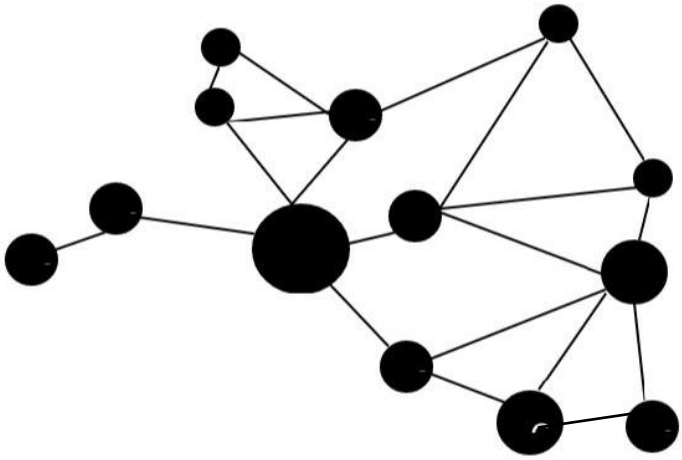
UNIVERSITAS NEGERI SURABAYA

2024

GRAPH PROVINSI JAWA BARAT

Berikut adalah peta Jawa Barat:





Gambar tersebut adalah peta dari Jawa Barat, Indonesia. Dan kota kota yang dilinkari Adalah kota kota yang akan dihubungkan sehingga membentuk Graph. Adapun kota yang akan dihubungkan dalam graph ini adalah:

1) Cikampek
2) Purwakarta
3) Subang
4) Bandung
5) Sumedang
6) Garut
7) Cianjur
8) Sukabumi
9) Tasikmalaya
10) Kuningan
11) Cirebon
12) Banjar
13) indramayu

Source Code :

```
class Peta:
    def __init__(self):
        self.cityList = {}

    def printPeta(self):
        for kota in self.cityList:
            print(kota, ":",self.cityList[kota])

    def tambahkanKota(self,kota):
        if kota not in self.cityList:
            self.cityList[kota] = []
            return True
        return False

    def hapusKota(self,kotaDihapus):
        #cek apakah kota yang ingin dihapus ada di list
        if kotaDihapus in self.cityList:
        #iterasi setiap kotalain untuk hapus kotadihapus
            for kotalain in self.cityList:
                #cek apakah kota yang ingin dihapus ada jalannya ke kotalain
                if kotaDihapus in self.cityList[kotalain]:
                    self.cityList[kotalain].remove(kotaDihapus)
            del self.cityList[kotaDihapus]
            return True
        return False

    def tambahkanJalan(self,kota1,kota2):
        if kota1 in self.cityList and kota2 in self.cityList:
            #masukkan kota 1 di list kota2
            self.cityList[kota2].append(kota1)
            #masukkan kota 2 di list kota1
            self.cityList[kota1].append(kota2)
            return True
        return False

    def hapusJalan(self,kota1,kota2):
        if kota1 in self.cityList and kota2 in self.cityList:
            #hapus kota 1 di list kota2
            self.cityList[kota2].remove(kota1)
```

```
            #hapus kota 2 di list kota1
            self.cityList[kota1].remove(kota2)
            return True
        return False


petajawabarat = Peta()
petajawabarat.tambahkanKota("Cikampek")
petajawabarat.tambahkanKota("Purwakarta")
petajawabarat.tambahkanKota("Subang")
petajawabarat.tambahkanKota("Bandung")
petajawabarat.tambahkanKota("Sumedang")
petajawabarat.tambahkanKota("Garut")
petajawabarat.tambahkanKota("Cianjur")
petajawabarat.tambahkanKota("Sukabumi")
petajawabarat.tambahkanKota("Tasikmalaya")
petajawabarat.tambahkanKota("Kuningan")
petajawabarat.tambahkanKota("Cirebon")
petajawabarat.tambahkanKota("Banjar")
petajawabarat.tambahkanKota("Indramayu")

petajawabarat.tambahkanJalan("Cikampek","Purwakarta")
petajawabarat.tambahkanJalan("Subang","Cikampek")
petajawabarat.tambahkanJalan("Subang","Purwakarta")
petajawabarat.tambahkanJalan("Bandung","Subang")
petajawabarat.tambahkanJalan("Cianjur","Sukabumi")
petajawabarat.tambahkanJalan("Cianjur","Bandung")
petajawabarat.tambahkanJalan("Bandung","Sumedang")
petajawabarat.tambahkanJalan("Bandung","Purwakarta")
petajawabarat.tambahkanJalan("Bandung","Garut")
petajawabarat.tambahkanJalan("Garut","Tasikmalaya")
petajawabarat.tambahkanJalan("Garut","Kuningan")
petajawabarat.tambahkanJalan("Cirebon","Kuningan")
petajawabarat.tambahkanJalan("Sumedang","Kuningan")
petajawabarat.tambahkanJalan("Tasikmalaya","Kuningan")
petajawabarat.tambahkanJalan("Banjar","Kuningan")
petajawabarat.tambahkanJalan("Banjar","Tasikmalaya")
petajawabarat.tambahkanJalan("Cirebon","Sumedang")
petajawabarat.tambahkanJalan("Cirebon","Indramayu")
petajawabarat.tambahkanJalan("Subang","Indramayu")
petajawabarat.tambahkanJalan("Sumedang","Indramayu")
petajawabarat.tambahkanJalan("Subang","Indramayu")
```

```
petajawabarat.tambahkanJalan("Sumedang","Indramayu")
petajawabarat.printPeta()
print('------------------------------')
petajawabarat.hapusKota("Banjar")
petajawabarat.printPeta()
```

Penjelasan Step By Step:

- Membuat sebuah kelas dengan nama **Peta.**
    - Kelas adalah sebuah blueprint untuk membuat objek-objek yang memiliki properti dan metode tertentu.
    - **def __init__(self)**adalah metode khusus yang disebut **__init__**. Metode ini adalah konstruktor kelas yang dipanggil secara otomatis setiap kali sebuah objek dari kelas dibuat. **self** merujuk pada objek yang sedang dibuat. Dalam konstruktor ini, kita memulai pembuatan objek dengan menginisialisasi properti-properti awalnya.
    - **self.cityList = {}**: Di dalam metode konstruktor, kita membuat atribut **cityList** untuk setiap objek yang dibuat dari kelas **Peta**. **cityList** adalah sebuah kamus (dictionary) kosong. Dengan menginisialisasi **cityList** menjadi kamus kosong, kita siap untuk menambahkan kota-kota ke dalamnya nanti.

```
class Peta:
    def __init__(self):
        self.cityList = {}
```

- Membuat fungsi **printPeta.**
    - **def printPeta(self)** adalah definisi metode baru dalam kelas **Peta** yang disebut **printPeta**. Metode ini tidak memiliki argumen selain **self**, yang merujuk pada objek kelas itu sendiri.
    - **for kota in self.cityList** adalah pernyataan loop **for** yang digunakan untuk mengulangi setiap elemen dalam kamus **cityList** dari objek **Peta**. Dalam setiap iterasi, variabel **kota** akan mewakili kunci (nama kota) dalam kamus.
    - **print(kota, ":", self.cityList[kota])** digunakan untuk mencetak nama kota (kunci) diikuti oleh nilai yang terkait (mungkin merupakan informasi tambahan tentang kota tersebut) dari kamus **cityList**. Dengan menggunakan format **print**, kita

mencetak kunci dan nilai dengan menggunakan **:** sebagai pemisah di antara keduanya.

```
    def printPeta(self):
        for kota in self.cityList:
            print(kota, ":",self.cityList[kota])
```

- Membuat Fungsi **tambahkanKota**
    - **def tambahkanKota(self, kota):** adalah definisi metode **tambahkanKota** dalam kelas **Peta**. Metode ini memiliki dua parameter: **self**, yang merujuk pada objek kelas itu sendiri, dan **kota**, yang merupakan nama kota yang ingin ditambahkan ke dalam peta.
    - **if kota not in self.cityList:** adalah pernyataan **if** yang digunakan untuk memeriksa apakah **kota** sudah ada dalam kamus **cityList** dari objek **Peta**. Jika **kota** belum ada dalam **cityList**, maka blok kode di dalam pernyataan **if** akan dieksekusi.
    - **self.cityList[kota] = []**: Di dalam blok **if**, kita menambahkan **kota** ke dalam kamus **cityList** dengan menginisialisasi nilai awalnya sebagai daftar kosong (**[]**). Ini berarti setiap kota dalam peta memiliki daftar kosong untuk menyimpan informasi tambahan yang terkait dengan kota tersebut.
    - **return True**: Setelah menambahkan kota ke dalam **cityList**, metode mengembalikan **True** untuk menunjukkan bahwa penambahan kota berhasil dilakukan.
    - **return False**: Jika **kota** sudah ada dalam **cityList**, maka metode akan langsung mengembalikan **False** tanpa menambahkan kota baru. Ini menunjukkan bahwa penambahan kota gagal dilakukan karena kota tersebut sudah ada sebelumnya dalam peta.

```
    def tambahkanKota(self,kota):
        if kota not in self.cityList:
            self.cityList[kota] = []
            return True
        return False
```

- Membuat fungsi **hapusKota**
    - **def hapusKota(self, kotaDihapus):**adalah definisi metode **hapusKota** dalam kelas **Peta**. Metode ini memiliki dua parameter: **self**, yang merujuk pada objek kelas itu sendiri, dan **kotaDihapus**, yang merupakan nama kota yang ingin dihapus dari peta.

- **if kotaDihapus in self.cityList:** adalah pernyataan **if** yang digunakan untuk memeriksa apakah **kotaDihapus** ada dalam kamus **cityList** dari objek **Peta**. Jika **kotaDihapus** ada dalam **cityList**, maka blok kode di dalam pernyataan **if** akan dieksekusi.
- **for kotalain in self.cityList:** adalah pernyataan **for** yang digunakan untuk mengiterasi setiap kota (**kotalain**) dalam kamus **cityList**.
- **if kotaDihapus in self.cityList[kotalain]:**pernyataan **if** digunakan untuk memeriksa apakah **kotaDihapus** terhubung dengan **kotalain**. Jika ya, maka jalur dari **kotalain** ke **kotaDihapus** akan dihapus.
- **self.cityList[kotalain].remove(kotaDihapus)**: Jika koneksi antara **kotalain** dan **kotaDihapus** ditemukan, kota tersebut akan dihapus dari daftar koneksi (**cityList**) kotalain.
- **del self.cityList[kotaDihapus]**: Setelah menghapus jalur ke dan dari kota yang akan dihapus, kota tersebut dihapus secara keseluruhan dari **cityList**.
- **return True**: Setelah menghapus kota dari **cityList**, metode mengembalikan **True** untuk menunjukkan bahwa penghapusan berhasil dilakukan.
- **return False**: Jika **kotaDihapus** tidak ditemukan dalam **cityList**, maka metode akan langsung mengembalikan **False** tanpa melakukan penghapusan. Ini menunjukkan bahwa penghapusan kota gagal dilakukan karena kota tersebut tidak ditemukan dalam peta.

```
    def hapusKota(self,kotaDihapus):
        #cek apakah kota yang ingin dihapus ada di list
        if kotaDihapus in self.cityList:
        #iterasi setiap kotalain untuk hapus kotadihapus
            for kotalain in self.cityList:
                #cek apakah kota yang ingin dihapus ada jalannya ke kotalain
                if kotaDihapus in self.cityList[kotalain]:
                    self.cityList[kotalain].remove(kotaDihapus)
            del self.cityList[kotaDihapus]
            return True
        return False
```

- Membuat fungsi **tambahkanJalan** untuk menambahkan hubungan di antara dua kota.
    - **def tambahkanJalan(self, kota1, kota2):** adalah definisi metode **tambahkanJalan** dalam kelas **Peta**. Metode ini memiliki tiga parameter: **self**, yang merujuk pada objek kelas itu sendiri, **kota1**, yang merupakan nama kota pertama, dan **kota2**, yang merupakan nama kota kedua.
    - **if kota1 in self.cityList and kota2 in self.cityList:** pernyataan **if** digunakan untuk memeriksa apakah **kota1** dan **kota2** ada dalam kamus **cityList** dari objek **Peta**. Jika

keduanya ada dalam **cityList**, maka blok kode di dalam pernyataan **if** akan dieksekusi.

- **self.cityList[kota2].append(kota1)**: Di dalam blok **if**, kita menambahkan **kota1** ke dalam daftar koneksi (**cityList**) dari **kota2**. Ini berarti menambahkan jalur dari **kota2** ke **kota1**.
- **self.cityList[kota1].append(kota2)**: Selanjutnya, kita juga menambahkan **kota2** ke dalam daftar koneksi (**cityList**) dari **kota1**. Ini berarti menambahkan jalur dari **kota1** ke **kota2**.
- **return True**: Setelah menambahkan jalur antara **kota1** dan **kota2**, metode mengembalikan **True** untuk menunjukkan bahwa penambahan jalur berhasil dilakukan.
- **return False**: Jika salah satu atau kedua kota tidak ditemukan dalam **cityList**, maka metode akan langsung mengembalikan **False** tanpa melakukan penambahan jalur. Ini menunjukkan bahwa penambahan jalur gagal dilakukan karena salah satu atau kedua kota tidak ditemukan dalam peta.

```
def tambahkanJalan(self,kota1,kota2):
    if kota1 in self.cityList and kota2 in self.cityList:
        #masukkan kota 1 di list kota2
        self.cityList[kota2].append(kota1)
        #masukkan kota 2 di list kota1
        self.cityList[kota1].append(kota2)
        return True
    return False
```

- Membuat fungsi **hapusJalan** yang digunakan untuk menghapus jalur atau koneksi antara dua kota dalam peta yang direpresentasikan oleh objek kelas **Peta**.
    - **def hapusJalan(self, kota1, kota2):**adalah definisi metode **hapusJalan** dalam kelas **Peta**.
    - **if kota1 in self.cityList and kota2 in self.cityList:**: Ini adalah pernyataan **if** yang digunakan untuk memeriksa apakah **kota1** dan **kota2** ada dalam kamus **cityList** dari objek **Peta**. Jika keduanya ada dalam **cityList**, maka blok kode di dalam pernyataan **if** akan dieksekusi.
    - **self.cityList[kota2].remove(kota1)**: berarti Di dalam blok **if**, kita menghapus **kota1** dari daftar koneksi (**cityList**) **kota2**. Ini berarti menghapus jalur dari **kota2** ke **kota1**.
    - **self.cityList[kota1].remove(kota2)**: Selanjutnya, kita juga menghapus **kota2** dari daftar koneksi (**cityList**) **kota1**. Ini berarti menghapus jalur dari **kota1** ke **kota2**.

- **return True**: agar setelah menghapus jalur antara **kota1** dan **kota2**, metode mengembalikan **True** untuk menunjukkan bahwa penghapusan jalur berhasil dilakukan.
- **return False**: Jika salah satu atau kedua kota tidak ditemukan dalam **cityList**, maka metode akan langsung mengembalikan **False** tanpa melakukan penghapusan jalur. Ini menunjukkan bahwa penghapusan jalur gagal dilakukan karena salah satu atau kedua kota tidak ditemukan dalam peta.

```
    def hapusJalan(self,kota1,kota2):
        if kota1 in self.cityList and kota2 in self.cityList:
            #hapus kota 1 di list kota2
            self.cityList[kota2].remove(kota1)
            #hapus kota 2 di list kota1
            self.cityList[kota1].remove(kota2)
            return True
        return False
```

- Membuat fungsi untuk menambahkan kota dalam peta

  petajawabarat = Peta(): digunakan untuk membuat objek baru dari kelas **Peta** dan menyimpannya dalam variabel **petajawabarat**.

  tambahkanKota digunakan untuk menambahkan kota dari objek **petajawabarat** sehingga dalam setiap baris perlu dimasukkan kota-kota yang diinginkan.

```
petajawabarat = Peta()
petajawabarat.tambahkanKota("Cikampek")
petajawabarat.tambahkanKota("Purwakarta")
petajawabarat.tambahkanKota("Subang")
petajawabarat.tambahkanKota("Bandung")
petajawabarat.tambahkanKota("Sumedang")
petajawabarat.tambahkanKota("Garut")
petajawabarat.tambahkanKota("Cianjur")
petajawabarat.tambahkanKota("Sukabumi")
petajawabarat.tambahkanKota("Tasikmalaya")
petajawabarat.tambahkanKota("Kuningan")
petajawabarat.tambahkanKota("Cirebon")
petajawabarat.tambahkanKota("Banjar")
petajawabarat.tambahkanKota("Indramayu")
```

- Membuat fungsi **tambahkanJalan** untuk menambahkan jalan yang menghubungkan kota-kota.

```
petajawabarat.tambahkanJalan("Cikampek","Purwakarta")
petajawabarat.tambahkanJalan("Subang","Cikampek")
petajawabarat.tambahkanJalan("Subang","Purwakarta")
petajawabarat.tambahkanJalan("Bandung","Subang")
petajawabarat.tambahkanJalan("Cianjur","Sukabumi")
petajawabarat.tambahkanJalan("Cianjur","Bandung")
petajawabarat.tambahkanJalan("Bandung","Sumedang")
petajawabarat.tambahkanJalan("Bandung","Purwakarta")
petajawabarat.tambahkanJalan("Bandung","Garut")
petajawabarat.tambahkanJalan("Garut","Tasikmalaya")
petajawabarat.tambahkanJalan("Garut","Kuningan")
petajawabarat.tambahkanJalan("Cirebon","Kuningan")
petajawabarat.tambahkanJalan("Sumedang","Kuningan")
petajawabarat.tambahkanJalan("Tasikmalaya","Kuningan")
petajawabarat.tambahkanJalan("Banjar","Kuningan")
petajawabarat.tambahkanJalan("Banjar","Tasikmalaya")
petajawabarat.tambahkanJalan("Cirebon","Sumedang")
petajawabarat.tambahkanJalan("Cirebon","Indramayu")
petajawabarat.tambahkanJalan("Subang","Indramayu")
petajawabarat.tambahkanJalan("Sumedang","Indramayu")
petajawabarat.tambahkanJalan("Subang","Indramayu")
petajawabarat.tambahkanJalan("Sumedang","Indramayu")
```

- Membuat fungsi **printPeta** dari objek **petajawabarat** untuk mengeprint output dalam pemograman ini. Serta membuat fungsi **hapusKota** untuk menghapus kota yang tidak diinginkan.

  print('------------------------------'): Baris ini mencetak garis pemisah horizontal untuk memisahkan output sebelumnya dengan output yang akan datang.
  petajawabarat.hapusKota("Banjar"): ini adalah fungsi **hapusKota** dari objek **petajawabarat.** Yang kali ini kota banjar adalah kota yang ingin dikeluarkan/dihapus dari peta.
  petajawabarat.printPeta(): digunakan untuk mencetak isi peta setelah kota "Banjar" dihapus.

```
petajawabarat.printPeta()
print('------------------------------')
petajawabarat.hapusKota("Banjar")
petajawabarat.printPeta()
```

Output sebelum menghapus salah satu kota :

```
Cikampek : ['Purwakarta', 'Subang']
Purwakarta : ['Cikampek', 'Subang', 'Bandung']
Subang : ['Cikampek', 'Purwakarta', 'Bandung', 'Indramayu', 'Indramayu']
Bandung : ['Subang', 'Cianjur', 'Sumedang', 'Purwakarta', 'Garut']
Sumedang : ['Bandung', 'Kuningan', 'Cirebon', 'Indramayu', 'Indramayu']
Garut : ['Bandung', 'Tasikmalaya', 'Kuningan']
Cianjur : ['Sukabumi', 'Bandung']
Sukabumi : ['Cianjur']
Tasikmalaya : ['Garut', 'Kuningan', 'Banjar']
Kuningan : ['Garut', 'Cirebon', 'Sumedang', 'Tasikmalaya', 'Banjar']
Cirebon : ['Kuningan', 'Sumedang', 'Indramayu']
Banjar : ['Kuningan', 'Tasikmalaya']
Indramayu : ['Cirebon', 'Subang', 'Sumedang', 'Subang', 'Sumedang']
```

Output setelah menghapus salah satu kota (kota Banjar):

```
------------------------------
Cikampek : ['Purwakarta', 'Subang']
Purwakarta : ['Cikampek', 'Subang', 'Bandung']
Subang : ['Cikampek', 'Purwakarta', 'Bandung', 'Indramayu', 'Indramayu']
Bandung : ['Subang', 'Cianjur', 'Sumedang', 'Purwakarta', 'Garut']
Sumedang : ['Bandung', 'Kuningan', 'Cirebon', 'Indramayu', 'Indramayu']
Garut : ['Bandung', 'Tasikmalaya', 'Kuningan']
Cianjur : ['Sukabumi', 'Bandung']
Sukabumi : ['Cianjur']
Tasikmalaya : ['Garut', 'Kuningan']
Kuningan : ['Garut', 'Cirebon', 'Sumedang', 'Tasikmalaya']
Cirebon : ['Kuningan', 'Sumedang', 'Indramayu']
Indramayu : ['Cirebon', 'Subang', 'Sumedang', 'Subang', 'Sumedang']
PS C:\Users\thinkpad>
```