

LAPORAN TUGAS STRUKTUR DATA

“DATA GRAPH”

2023F



Anggota Kelompok 2:

- | | |
|---------------------------------|-------------|
| 1. Abi Khoir Hidayat | 23091397205 |
| 2. Fina Fadhilah Maulana | 23091397207 |
| 3. Mahestu Bagus Senaru Pratama | 23091397211 |

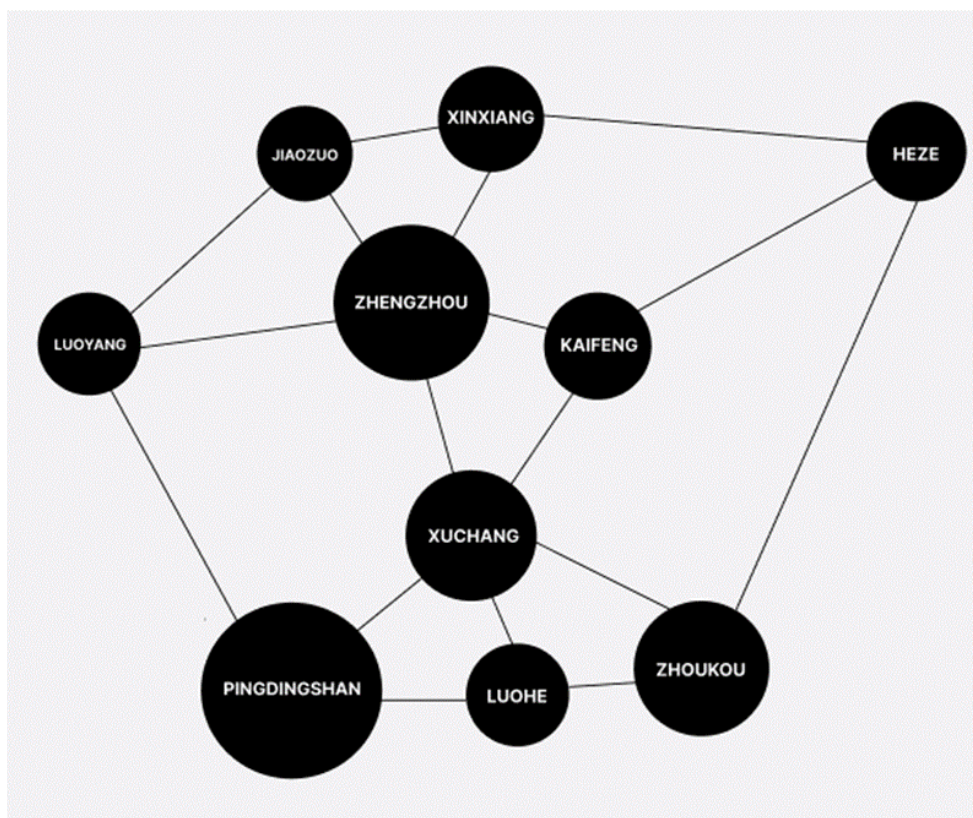
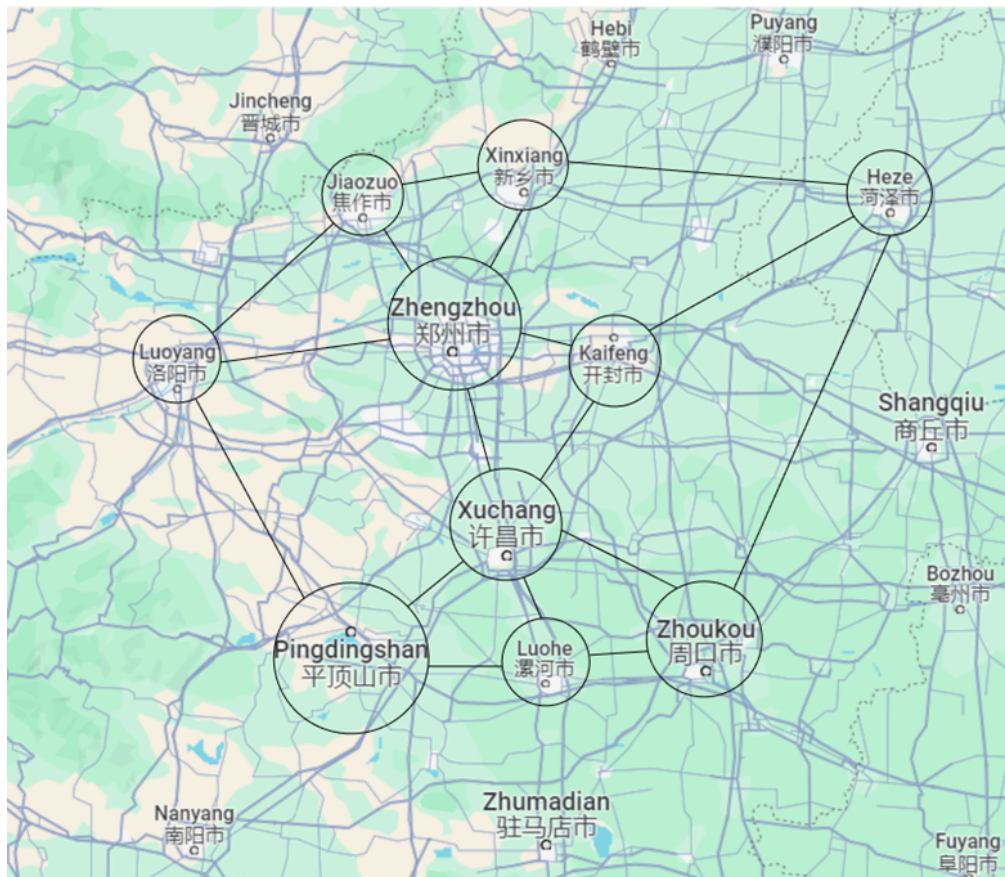
PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2024

GRAPH PADA NEGARA TIONGKOK



Kedua gambar tersebut merupakan peta dari Negara Tiongkok yang saling dihubungkan sehingga menghasilkan gambar graph. Pada gambar atau peta diatas terdiri dari kota antara lain :

1. Xinxiang
2. Jiaozuo
3. Luoyang
4. Pingdingshan
5. Xuchang
6. Luohe
7. Zhoukou
8. Heze
9. Kaifeng
10. Zhengzhou

Source Code :

```
1 class peta:
2     def __init__(self):
3         self.cityList = {}
4
5     def printPeta(self):
6         for kota in self.cityList:
7             print(kota, ":", self.cityList[kota])
8
9     def tambahkanKota(self, kota):
10        if kota not in self.cityList:
11            self.cityList[kota] = []
12            return True
13        return False
14
15    def hapusKota(self, kotaDihapus):
16        #cek apakah kota yang ingin dihapus ada di list
17        if kotaDihapus in self.cityList:
18            #iterasi setiap kotalain untuk hapus kotadihapus
19            for kotalain in self.cityList:
20                #cek apakah kota yang ingin dihapus ada jalannya ke kotalain
21                if kotaDihapus in self.cityList[kotalain]:
22                    self.cityList[kotalain].remove(kotaDihapus)
23            del self.cityList[kotaDihapus]
24            return True
25        return False
26
27    def tambahkanJalan(self, kota1, kota2):
28        if kota1 in self.cityList and kota2 in self.cityList:
29            #masukkan kota 1 di list kota2
30            self.cityList[kota2].append(kota1)
31            #masukkan kota 2 di list kota1
32            self.cityList[kota1].append(kota2)
33            return True
34        return False
35
36    def hapusJalan(self, kota1, kota2):
37        if kota1 in self.cityList and kota2 in self.cityList:
38            #hapus kota 1 di list kota2
39            self.cityList[kota2].remove(kota1)
40            #hapus kota 2 di list kota1
41            self.cityList[kota1].remove(kota2)
42            return True
43        return False
44
45
46    petationgkok = peta()
47    petationgkok.tambahkanKota("xinxiang")
48    petationgkok.tambahkanKota("jiaozuo")
49    petationgkok.tambahkanKota("luoyang")
50    petationgkok.tambahkanKota("pingdingshan")
51    petationgkok.tambahkanKota("xuchang")
52    petationgkok.tambahkanKota("luohe")
53    petationgkok.tambahkanKota("zhoukou")
54    petationgkok.tambahkanKota("heze")
55    petationgkok.tambahkanKota("kaifeng")
56    petationgkok.tambahkanKota("zhengzhou")
57    print("-----")
58    petationgkok.tambahkanJalan("xinxiang", "jiaozuo")
59    petationgkok.tambahkanJalan("xinxiang", "luoyang")
60    petationgkok.tambahkanJalan("xinxiang", "pingdingshan")
61    petationgkok.tambahkanJalan("xinxiang", "xuchang")
62    petationgkok.tambahkanJalan("xinxiang", "luohe")
63    petationgkok.tambahkanJalan("xinxiang", "zhoukou")
64    petationgkok.tambahkanJalan("xinxiang", "heze")
65    petationgkok.tambahkanJalan("xinxiang", "kaifeng")
66    petationgkok.tambahkanJalan("xinxiang", "zhengzhou")
67    petationgkok.tambahkanJalan("jiaozuo", "luoyang")
68    petationgkok.tambahkanJalan("jiaozuo", "pingdingshan")
69    petationgkok.tambahkanJalan("jiaozuo", "xuchang")
70    petationgkok.tambahkanJalan("jiaozuo", "zhoukou")
71    petationgkok.tambahkanJalan("jiaozuo", "heze")
72    petationgkok.tambahkanJalan("jiaozuo", "kaifeng")
73    petationgkok.tambahkanJalan("jiaozuo", "zhengzhou")
74    petationgkok.tambahkanJalan("luoyang", "pingdingshan")
75    petationgkok.tambahkanJalan("luoyang", "xuchang")
76    petationgkok.tambahkanJalan("luoyang", "luohe")
77    petationgkok.tambahkanJalan("luoyang", "zhoukou")
78    petationgkok.tambahkanJalan("luoyang", "heze")
79    petationgkok.tambahkanJalan("luoyang", "kaifeng")
80    petationgkok.tambahkanJalan("luoyang", "zhengzhou")
81    petationgkok.tambahkanJalan("pingdingshan", "xuchang")
82    petationgkok.tambahkanJalan("pingdingshan", "luohe")
83    petationgkok.tambahkanJalan("pingdingshan", "zhoukou")
84    petationgkok.tambahkanJalan("pingdingshan", "heze")
85    petationgkok.tambahkanJalan("pingdingshan", "kaifeng")
86    petationgkok.tambahkanJalan("pingdingshan", "zhengzhou")
87    petationgkok.tambahkanJalan("xuchang", "luohe")
88
89    print("----- OUTPUT AWAL -----")
90    petationgkok.printPeta()
91    print("----- OUTPUT SETELAH MENGHAPUS KOTA LUOYANG -----")
92    petationgkok.hapusKota("luoyang")
93    petationgkok.printPeta()
94    print("----- OUTPUT SETELAH MENGHAPUS JALAN XUCHANG & LUOHE -----")
95    petationgkok.hapusJalan("xuchang", "luohe")
96    petationgkok.printPeta()
```

Penjelasan Step by Step :

➤ Membuat kelas dengan nama '**peta**', Kelas adalah sebuah blueprint untuk membuat objek-objek yang memiliki properti dan metode tertentu. Serta kelas ini digunakan untuk mewakili peta kota dan jalan yang menghubungkannya. Selanjutnya inisialisasi objek pada peta :

- '`__init__`' adalah metode inisialisasi yang akan dipanggil setiap kali objek

Peta dibuat.

- '`self.cityList`' adalah dictionary kosong yang akan menyimpan daftar kota dan daftar jalan yang menghubungkan kota-kota tersebut.



```
1 class peta:
2     def __init__(self):
3         self.cityList = {}
```

➤ Membuat fungsi/ metode print peta, berikut penjelasannya:

- '`printPeta`' sendiri adalah metode untuk mencetak daftar kota dan kota-kota lain yang terhubung dengan masing-masing kota tersebut.
- '`for kota in self.cityList`' mengiterasi semua kota dalam daftar.
- '`print(kota, ":", self.cityList[kota])`' mencetak nama kota dan daftar kota yang terhubung dengannya.



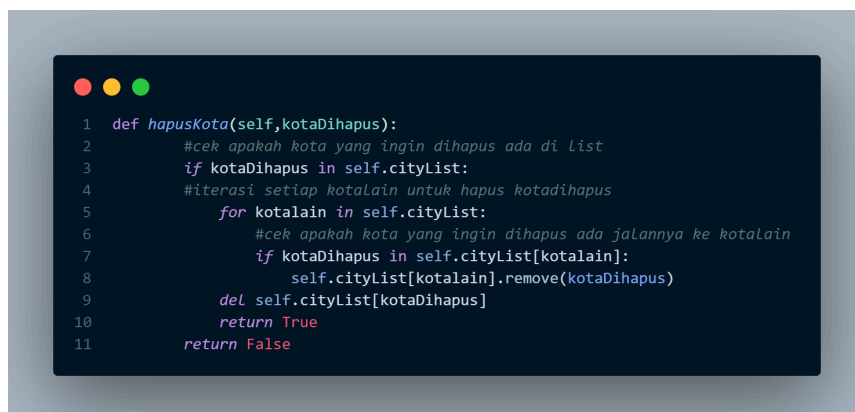
```
1 def printPeta(self):
2     for kota in self.cityList:
3         print(kota, ":", self.cityList[kota])
```

➤ Buat fungsi bernama `tambahkanKota` yang berfungsi untuk menambahkan kota baru ke `cityList`. Jika sebuah kota belum ada dalam `cityList`, maka kota tersebut akan ditambahkan ke daftar kosong (yang nantinya akan diisi dengan kota-kota yang terhubung). Jika kota berhasil ditambahkan, metode akan mengembalikan `True`, jika tidak maka akan mengembalikan `False`.



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

➤ Buat fungsi `hapusKota` untuk menghapus kota dari `cityList`. Jika kota yang ingin hapus ada di `cityList` maka semua referensi ke kota tersebut akan dihapus terlebih dahulu dari daftar kota lainnya. Kota tersebut kemudian akan dihapus dari `cityList` dengan `del`. Jika berhasil, metode akan mengembalikan `True`, jika tidak maka akan mengembalikan `False`.



```
1 def hapusKota(self,kotaDihapus):
2     #cek apakah kota yang ingin dihapus ada di List
3     if kotaDihapus in self.cityList:
4         #iterasi setiap kotalain untuk hapus kotadihapus
5         for kotalain in self.cityList:
6             #cek apakah kota yang ingin dihapus ada jalannya ke kotalain
7             if kotaDihapus in self.cityList[kotalain]:
8                 self.cityList[kotalain].remove(kotaDihapus)
9             del self.cityList[kotaDihapus]
10        return True
11    return False
```

➤ Selanjutnya membuat fungsi/metode `TambahkanJalan` untuk menambahkan hubungan antara dua kota. Jika kedua kota ada di `cityList`, maka `kota1` dan `kota2` saling ditambahkan ke daftar masing-masing, menunjukkan bahwa mereka terhubung. Jika berhasil, metode mengembalikan `True`, jika tidak, mengembalikan `False`.

```

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

```

➤ Berikutnya adalah membuat fungsi/metode HapusJalan untuk menghapus hubungan antara dua kota. Jika kedua kota ada di cityList, maka kota1 dan kota2 saling dihapus dari daftar masing-masing. Jika berhasil, metode mengembalikan True, jika tidak, mengembalikan False.

```

1 def hapusJalan(self,kota1,kota2):
2     if kota1 in self.cityList and kota2 in self.cityList:
3         #hapus kota 1 di list kota2
4         self.cityList[kota2].remove(kota1)
5         #hapus kota 2 di list kota1
6         self.cityList[kota1].remove(kota2)
7         return True
8     return False

```

➤ Membuat variabel baru yaitu PetaTiongkok dan menambahkan kota-kota ke cityList menggunakan metode TambahkanKota.


```

1  petationgkok = peta()
2  petationgkok.tambahkanKota("xinxiang")
3  petationgkok.tambahkanKota("jiaozuo")
4  petationgkok.tambahkanKota("luoyang")
5  petationgkok.tambahkanKota("pingdingshan")
6  petationgkok.tambahkanKota("xuchang")
7  petationgkok.tambahkanKota("luohe")
8  petationgkok.tambahkanKota("zhoukou")
9  petationgkok.tambahkanKota("heze")
10 petationgkok.tambahkanKota("kaifeng")
11 petationgkok.tambahkanKota("zhengzhou")

```

➤ Selanjutnya menambahkan jalan yang menghubungkan antar kota menggunakan metode TambahkanJalan

```

1  petationgkok.tambahkanJalan("xinxiang","jiaozuo")
2  petationgkok.tambahkanJalan("xinxiang","luoyang")
3  petationgkok.tambahkanJalan("xinxiang","pingdingshan")
4  petationgkok.tambahkanJalan("xinxiang","xuchang")
5  petationgkok.tambahkanJalan("xinxiang","luohe")
6  petationgkok.tambahkanJalan("xinxiang","zhoukou")
7  petationgkok.tambahkanJalan("xinxiang","heze")
8  petationgkok.tambahkanJalan("xinxiang","kaifeng")
9  petationgkok.tambahkanJalan("xinxiang","zhengzhou")
10 petationgkok.tambahkanJalan("jiaozuo","luoyang")
11 petationgkok.tambahkanJalan("jiaozuo","pingdingshan")
12 petationgkok.tambahkanJalan("jiaozuo","xuchang")
13 petationgkok.tambahkanJalan("jiaozuo","zhoukou")
14 petationgkok.tambahkanJalan("jiaozuo","heze")
15 petationgkok.tambahkanJalan("jiaozuo","kaifeng")
16 petationgkok.tambahkanJalan("jiaozuo","zhengzhou")
17 petationgkok.tambahkanJalan("luoyang","pingdingshan")
18 petationgkok.tambahkanJalan("luoyang","xuchang")
19 petationgkok.tambahkanJalan("luoyang","luohe")
20 petationgkok.tambahkanJalan("luoyang","zhoukou")
21 petationgkok.tambahkanJalan("luoyang","heze")
22 petationgkok.tambahkanJalan("luoyang","kaifeng")
23 petationgkok.tambahkanJalan("luoyang","zhengzhou")
24 petationgkok.tambahkanJalan("pingdingshan","xuchang")
25 petationgkok.tambahkanJalan("pingdingshan","luohe")
26 petationgkok.tambahkanJalan("pingdingshan","zhoukou")
27 petationgkok.tambahkanJalan("pingdingshan","heze")
28 petationgkok.tambahkanJalan("pingdingshan","kaifeng")
29 petationgkok.tambahkanJalan("pingdingshan","zhengzhou")
30 petationgkok.tambahkanJalan("xuchang","luohe")
31

```


OUTPUT TERMINAL

- Output awal pada saat diprint

```
----- OUTPUT AWAL -----
xinxiang : ['jiaozuo', 'luoyang', 'pingdingshan', 'xuchang', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
jiaozuo : ['xinxiang', 'luoyang', 'pingdingshan', 'xuchang', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
luoyang : ['xinxiang', 'jiaozuo', 'pingdingshan', 'xuchang', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
pingdingshan : ['xinxiang', 'jiaozuo', 'luoyang', 'xuchang', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
xuchang : ['xinxiang', 'jiaozuo', 'luoyang', 'pingdingshan', 'luohe']
luohe : ['xinxiang', 'luoyang', 'pingdingshan', 'xuchang']
zhoukou : ['xinxiang', 'jiaozuo', 'luoyang', 'pingdingshan']
heze : ['xinxiang', 'jiaozuo', 'luoyang', 'pingdingshan']
kaifeng : ['xinxiang', 'jiaozuo', 'luoyang', 'pingdingshan']
zhengzhou : ['xinxiang', 'jiaozuo', 'luoyang', 'pingdingshan']
```

- Output setelah satu kota dihapus

```
----- OUTPUT SETELAH MENGHAPUS KOTA LUOYANG -----
xinxiang : ['jiaozuo', 'pingdingshan', 'xuchang', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
jiaozuo : ['xinxiang', 'pingdingshan', 'xuchang', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
pingdingshan : ['xinxiang', 'jiaozuo', 'xuchang', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
xuchang : ['xinxiang', 'jiaozuo', 'pingdingshan', 'luohe']
luohe : ['xinxiang', 'pingdingshan', 'xuchang']
zhoukou : ['xinxiang', 'jiaozuo', 'pingdingshan']
heze : ['xinxiang', 'jiaozuo', 'pingdingshan']
kaifeng : ['xinxiang', 'jiaozuo', 'pingdingshan']
zhengzhou : ['xinxiang', 'jiaozuo', 'pingdingshan']
```

- Output setelah satu jalan dihapus

```
----- OUTPUT SETELAH MENGHAPUS JALAN XUCHANG & LUOHE -----
xinxiang : ['jiaozuo', 'pingdingshan', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
jiaozuo : ['xinxiang', 'pingdingshan', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
pingdingshan : ['xinxiang', 'jiaozuo', 'luohe', 'zhoukou', 'heze', 'kaifeng', 'zhengzhou']
xuchang : ['xinxiang', 'jiaozuo', 'pingdingshan']
luohe : ['xinxiang', 'pingdingshan']
zhoukou : ['xinxiang', 'jiaozuo', 'pingdingshan']
heze : ['xinxiang', 'jiaozuo', 'pingdingshan']
kaifeng : ['xinxiang', 'jiaozuo', 'pingdingshan']
zhengzhou : ['xinxiang', 'jiaozuo', 'pingdingshan']
```

LINK GITHUB:

<https://github.com/Mahestuu/AlproKelompok2>