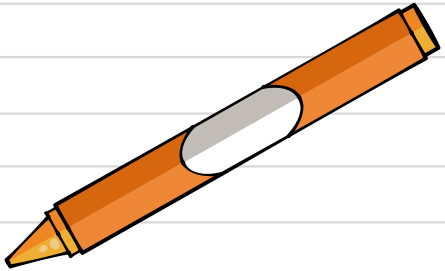
Struktur Data

Meet 09

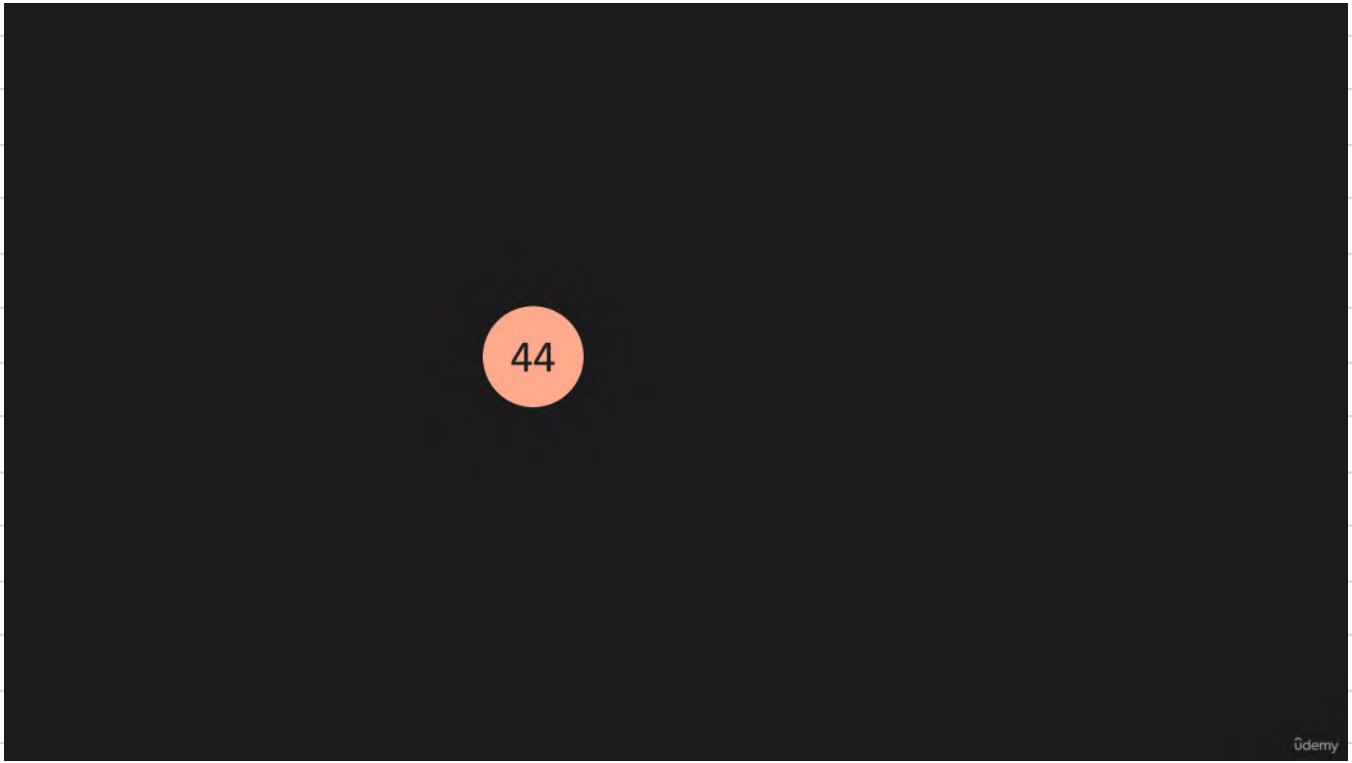
Graph

01

Graph Intro

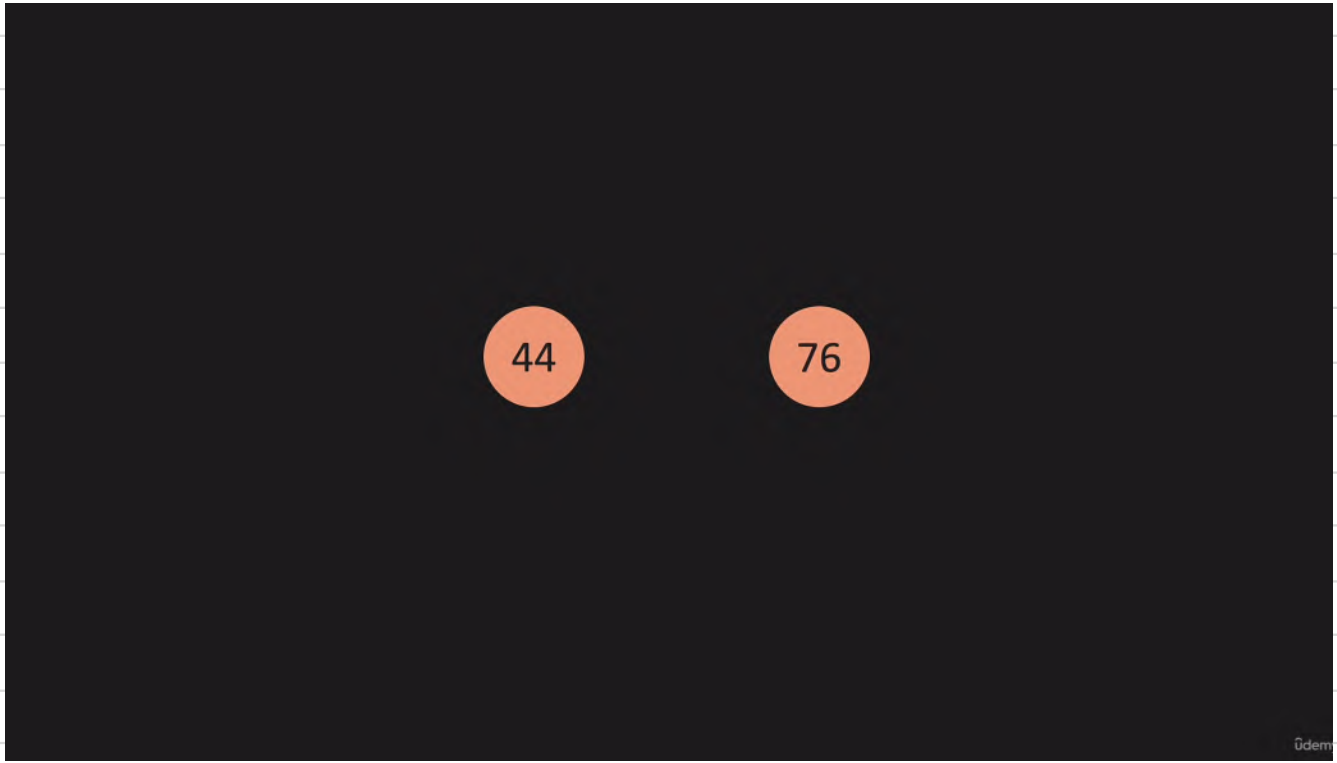


Node Graph



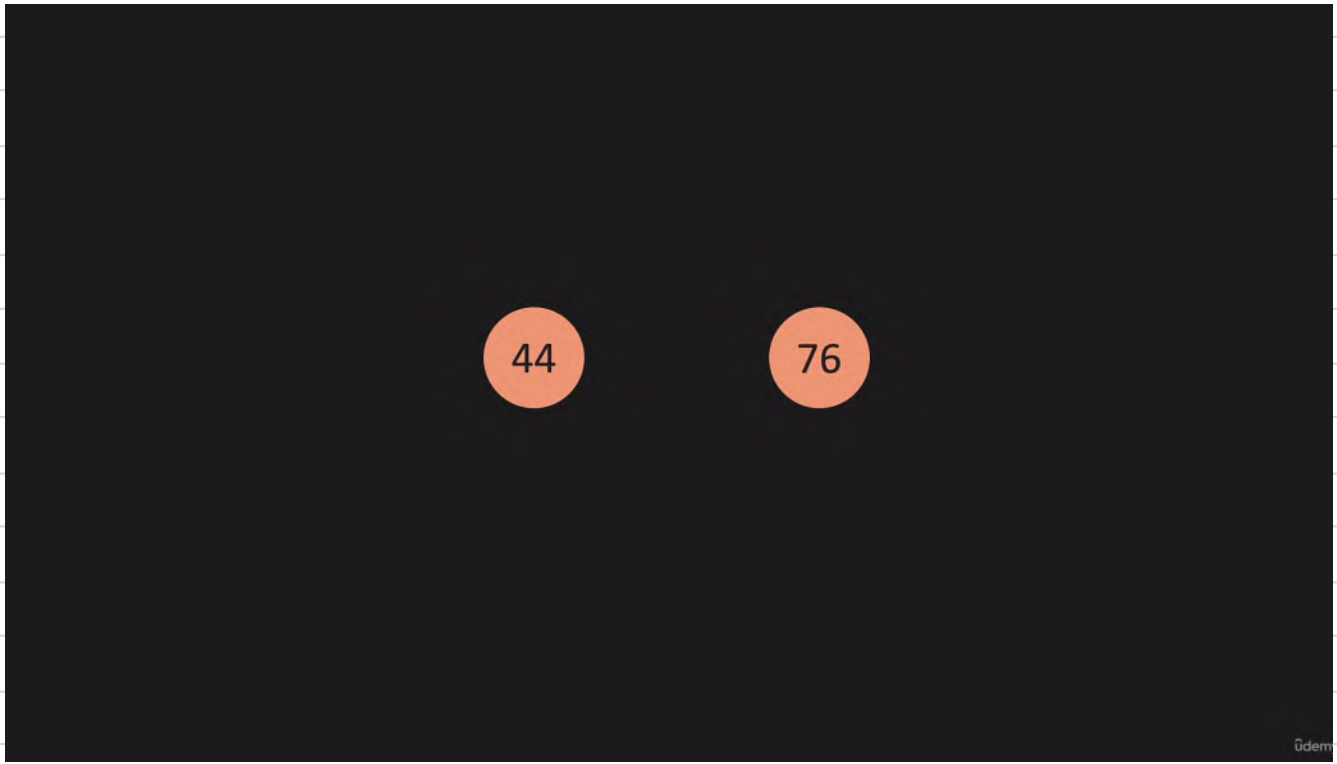
Node Graph

Node Graph

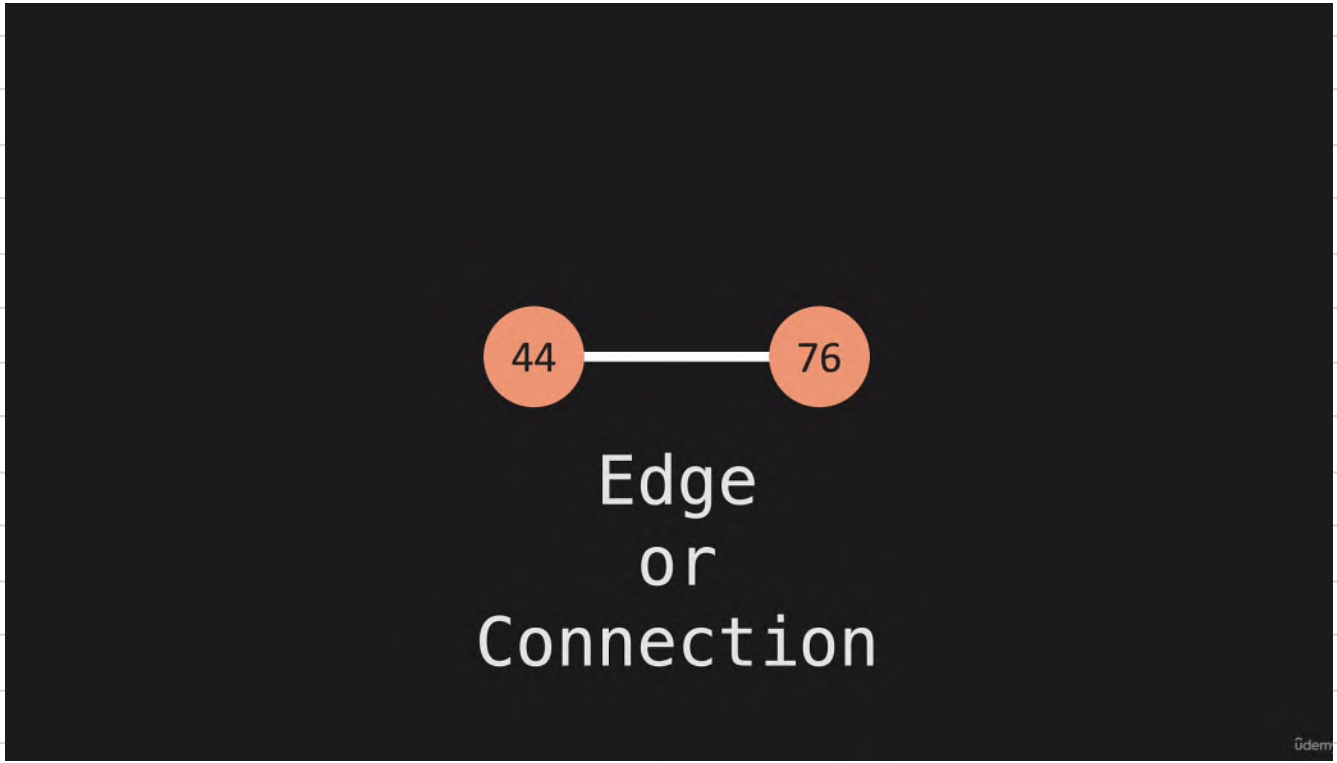


Node Graph

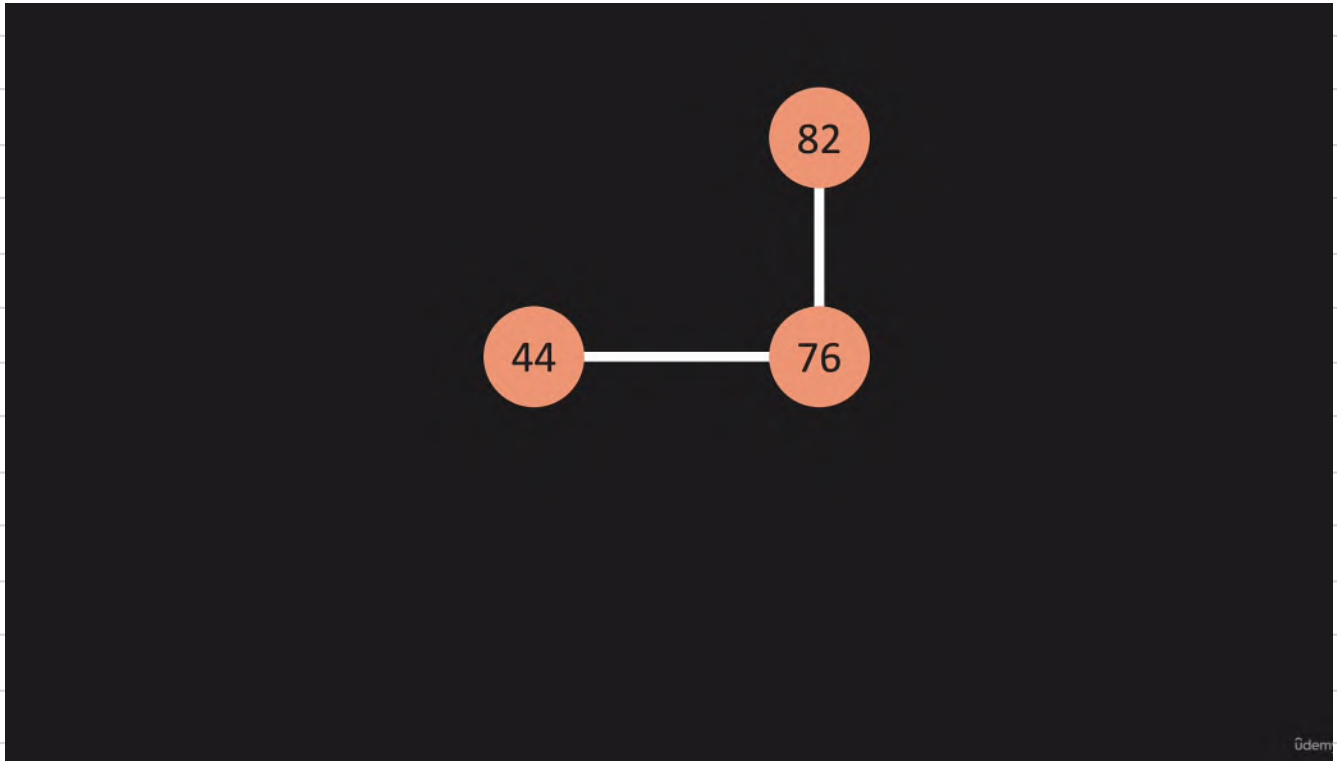
Jika lebih dari 1 vertex
istilah jamaknya menjadi Vertices



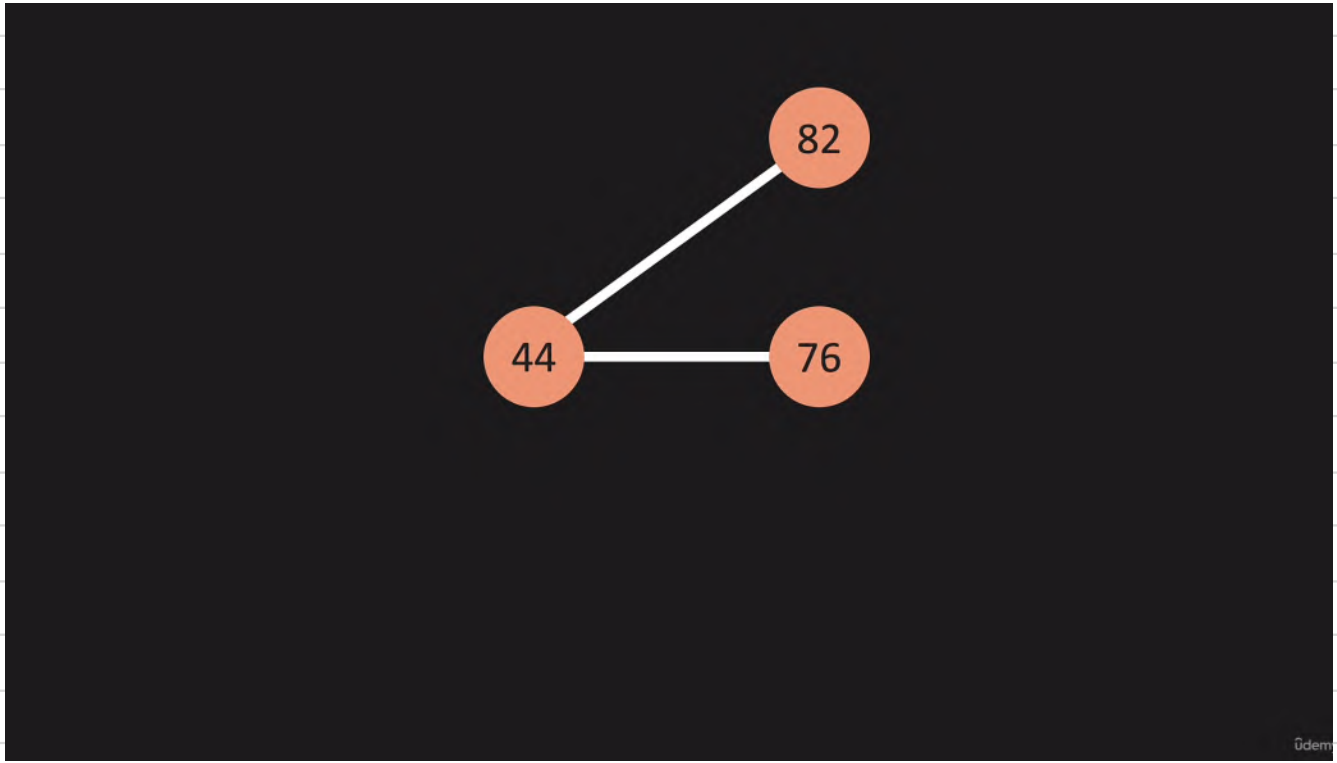
Node Graph



Node Graph

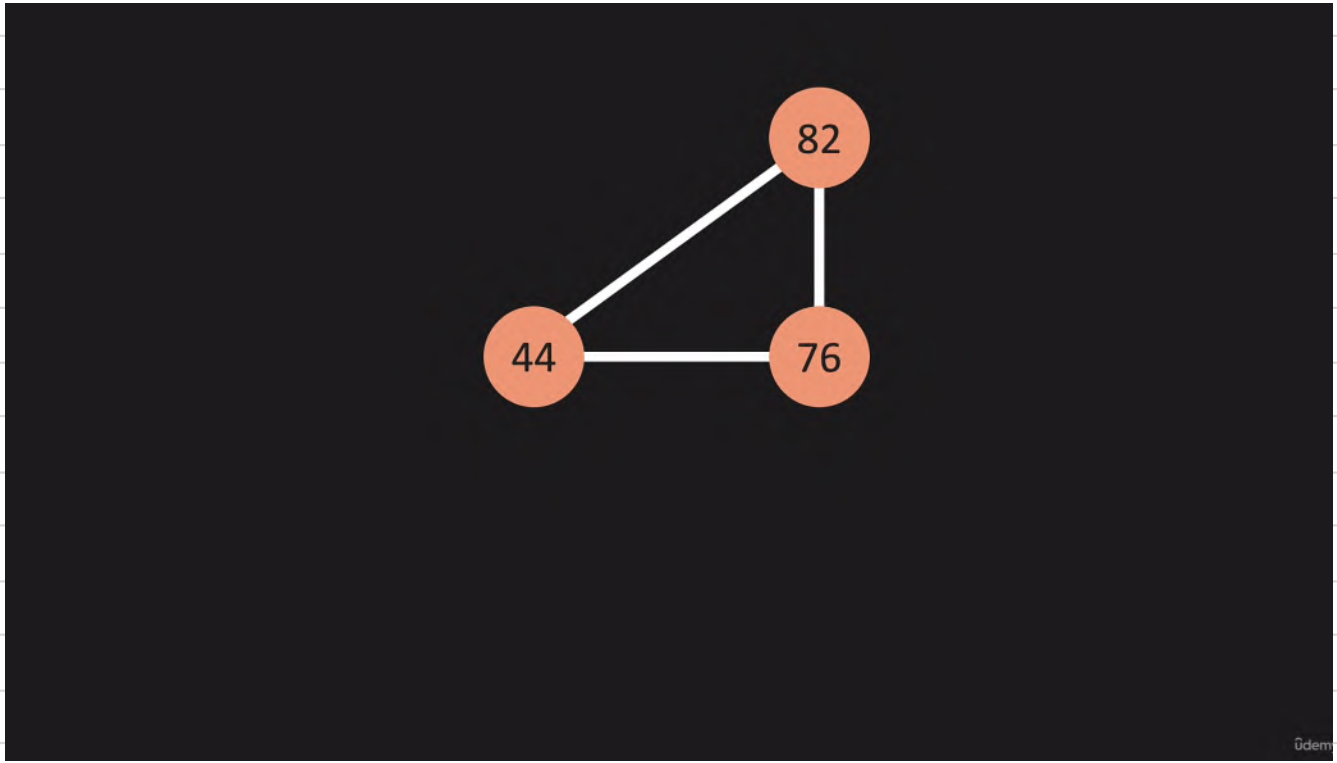


Node Graph



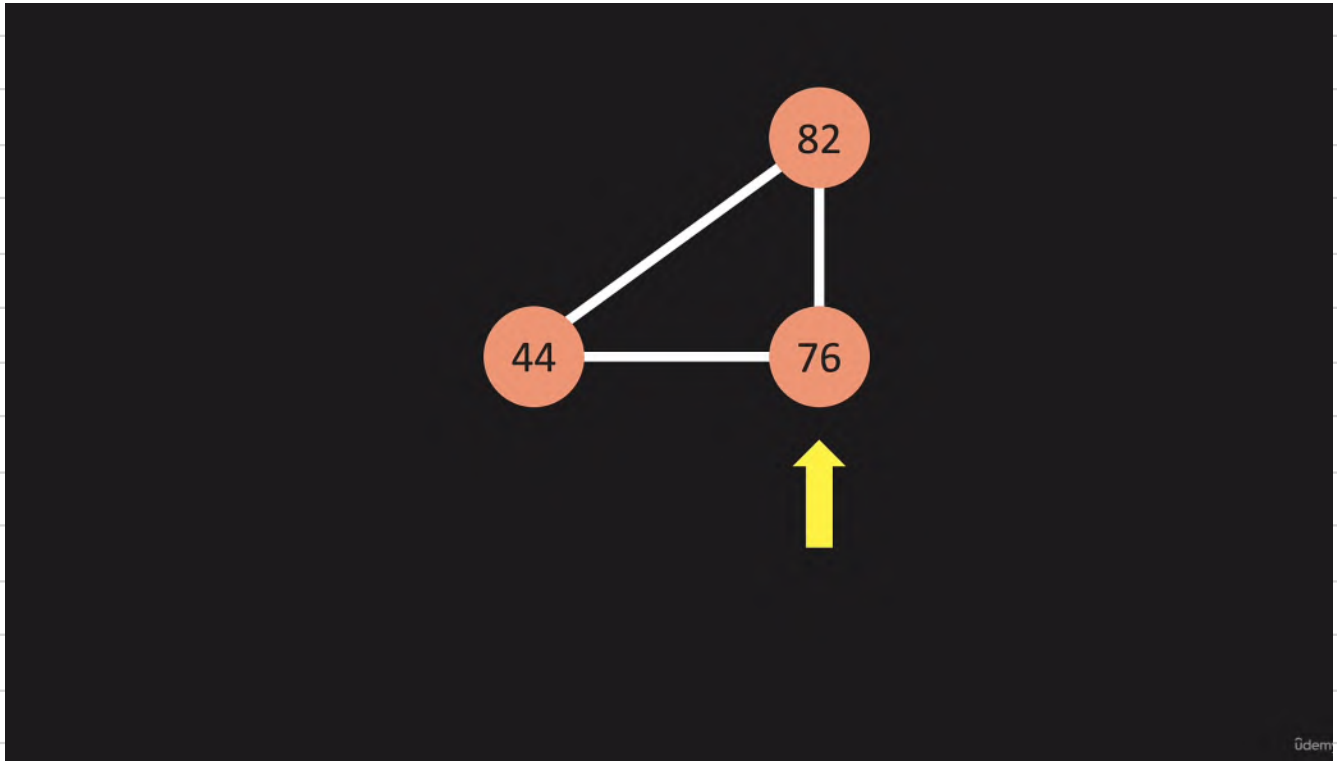
Node Graph

Dalam graph tidak ada batasan berapa node yang bisa terhubung dgn node lain



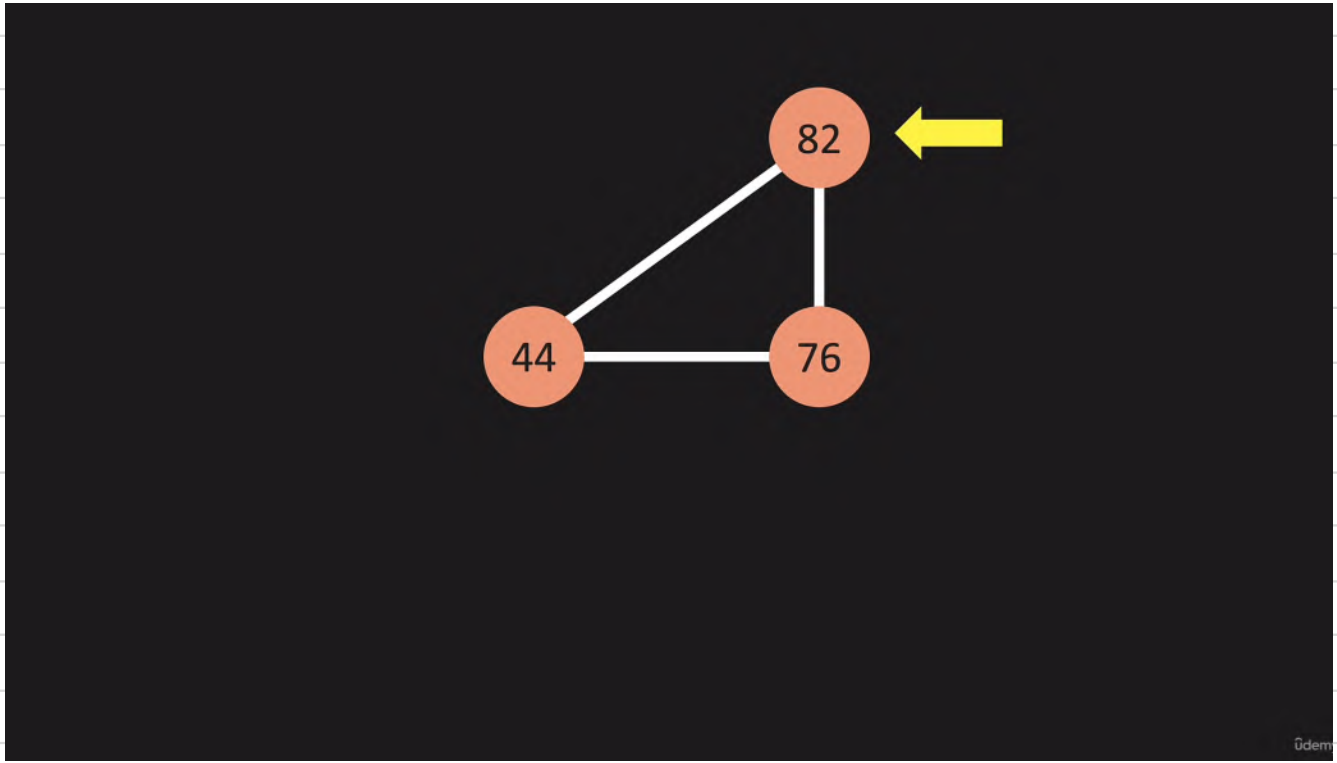
Node Graph

Jika kita berada di vertex 76



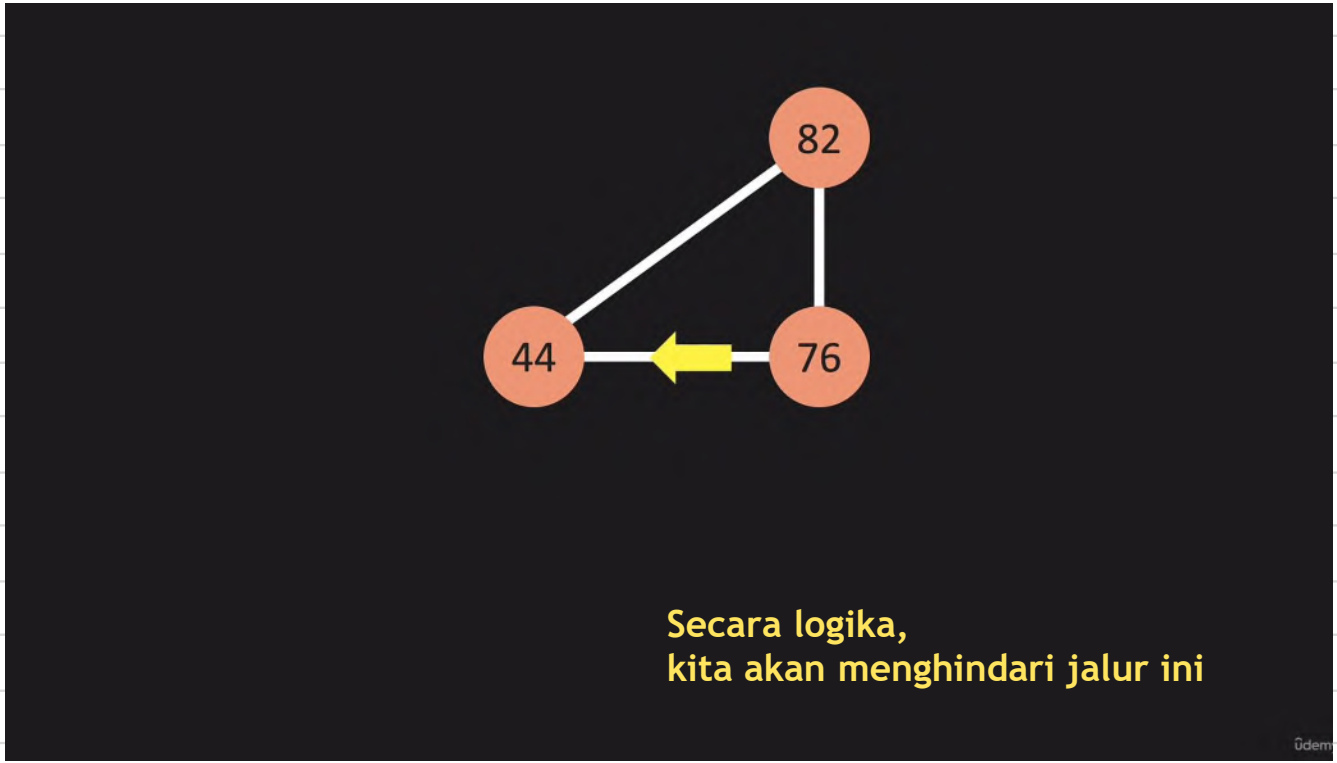
Node Graph

Jika kita berada di vertex 76 dan ingin menuju vertex 82



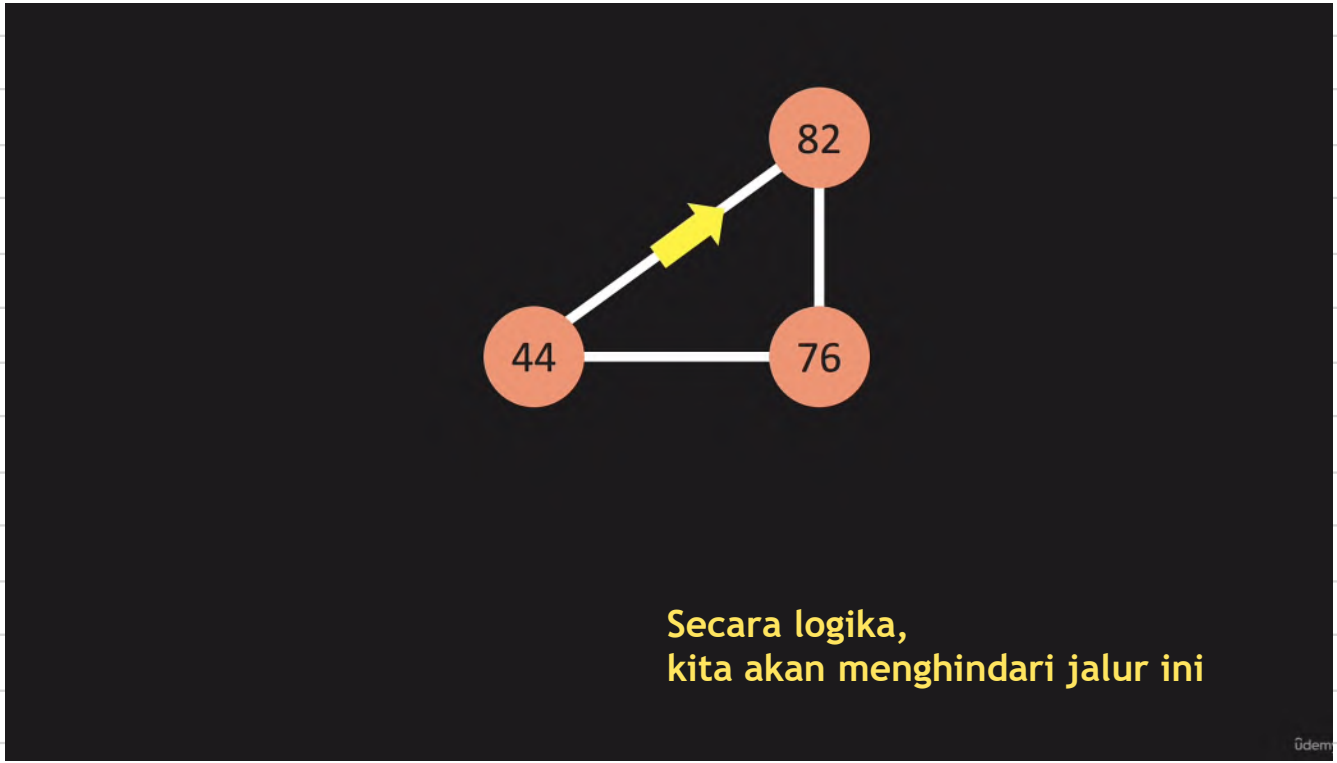
Node Graph

Jika kita berada di vertex 76 dan
ingin menuju vertex 82



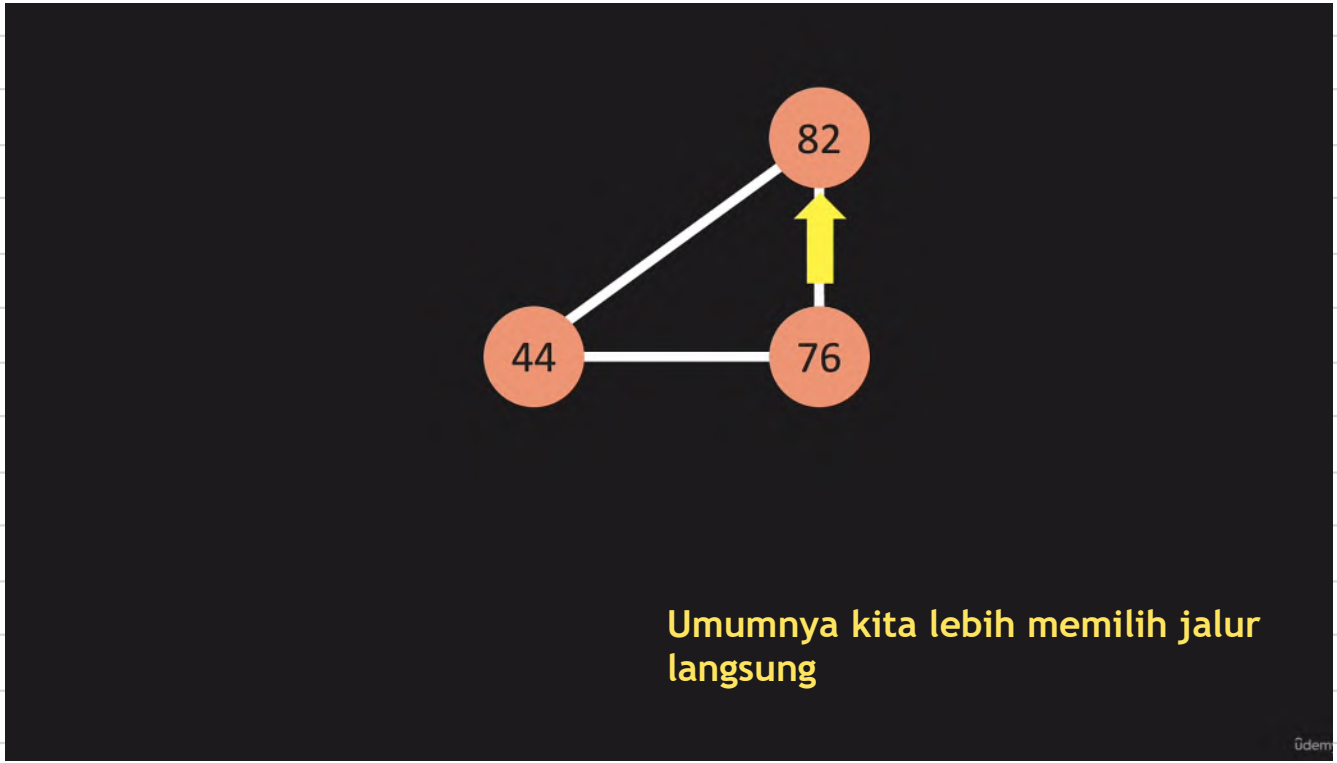
Node Graph

Jika kita berada di vertex 76 dan
ingin menuju vertex 82



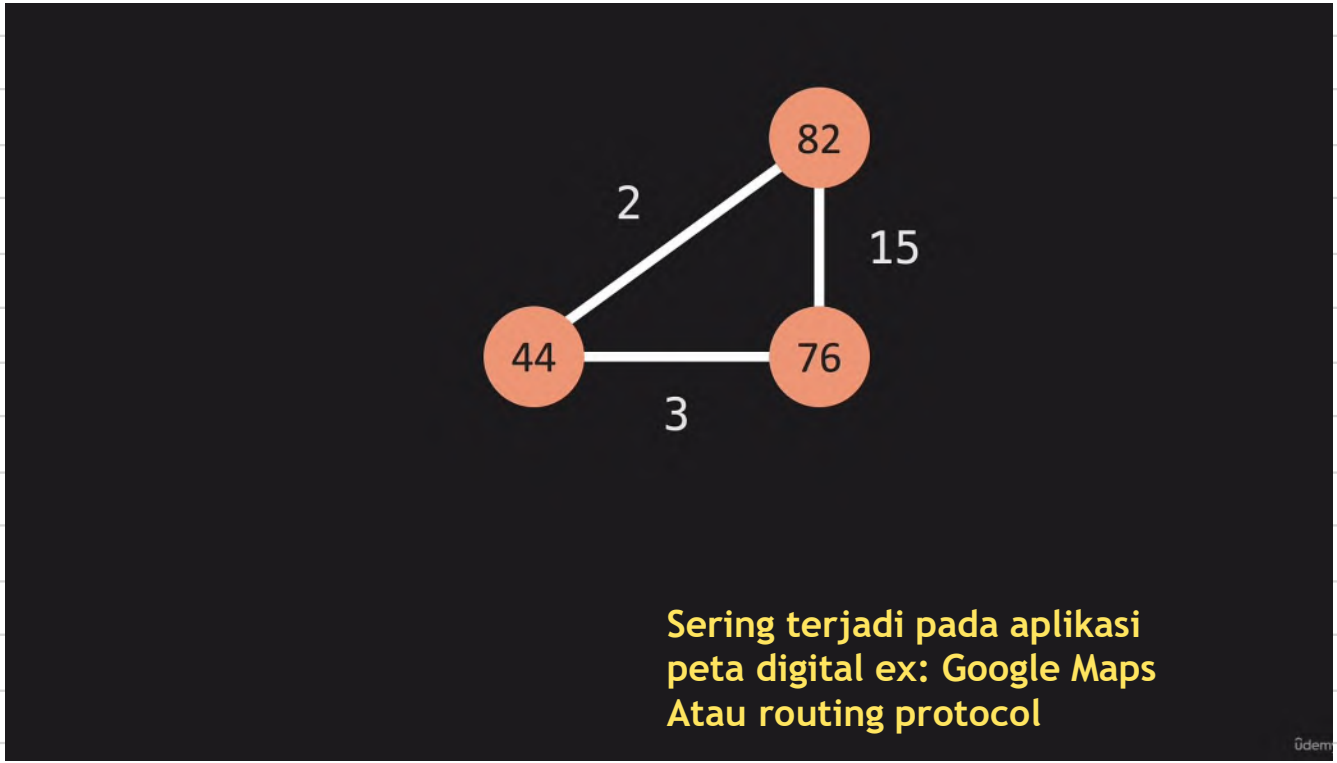
Node Graph

Jika kita berada di vertex 76 dan ingin menuju vertex 82

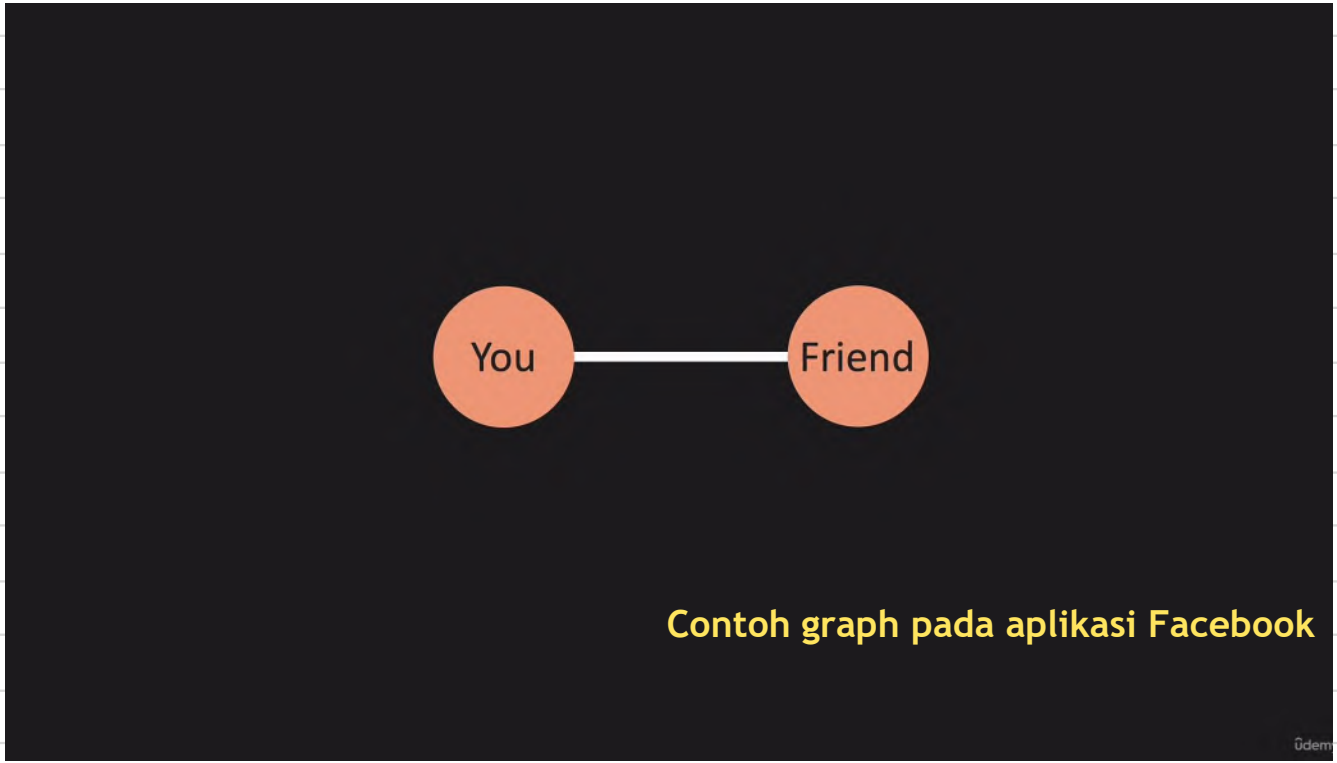


Node Graph

Namun dalam teori Graph, suatu edge bisa bersifat weighted atau non-weighted

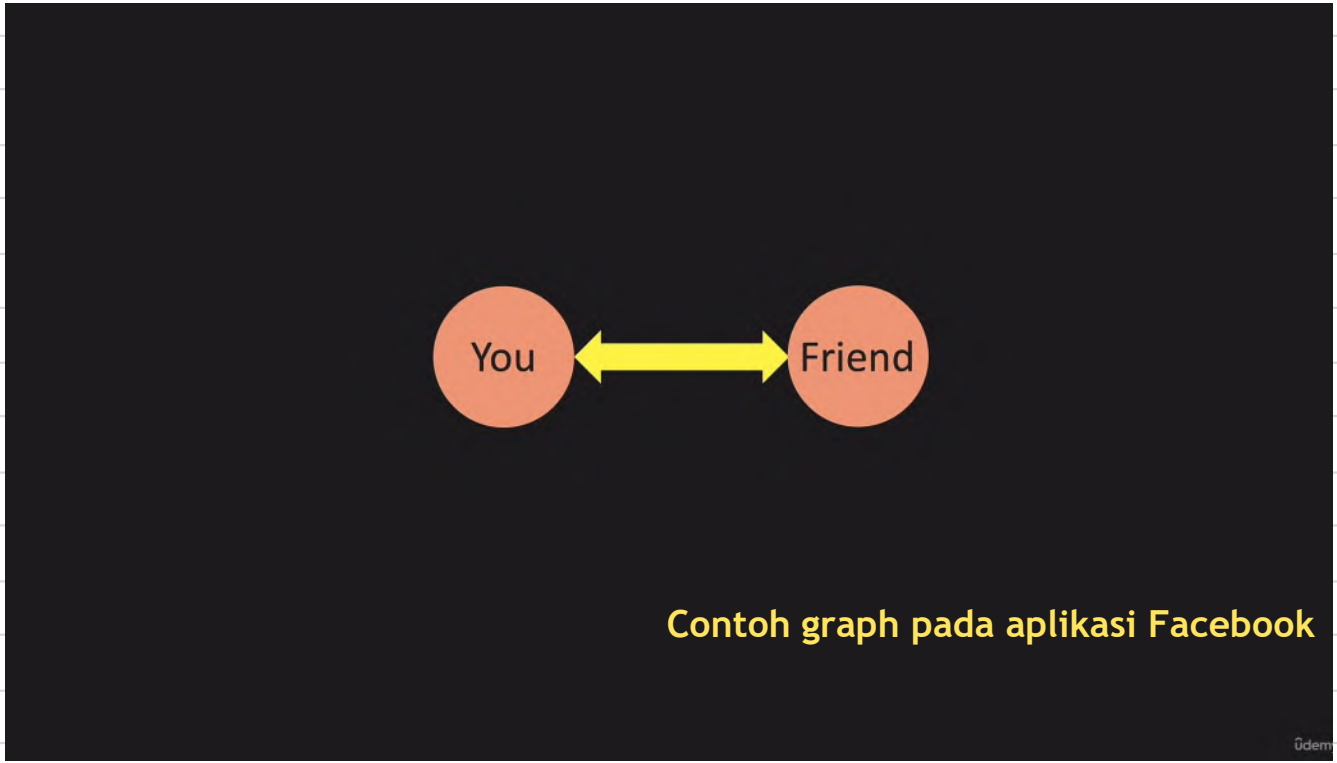


Node Graph



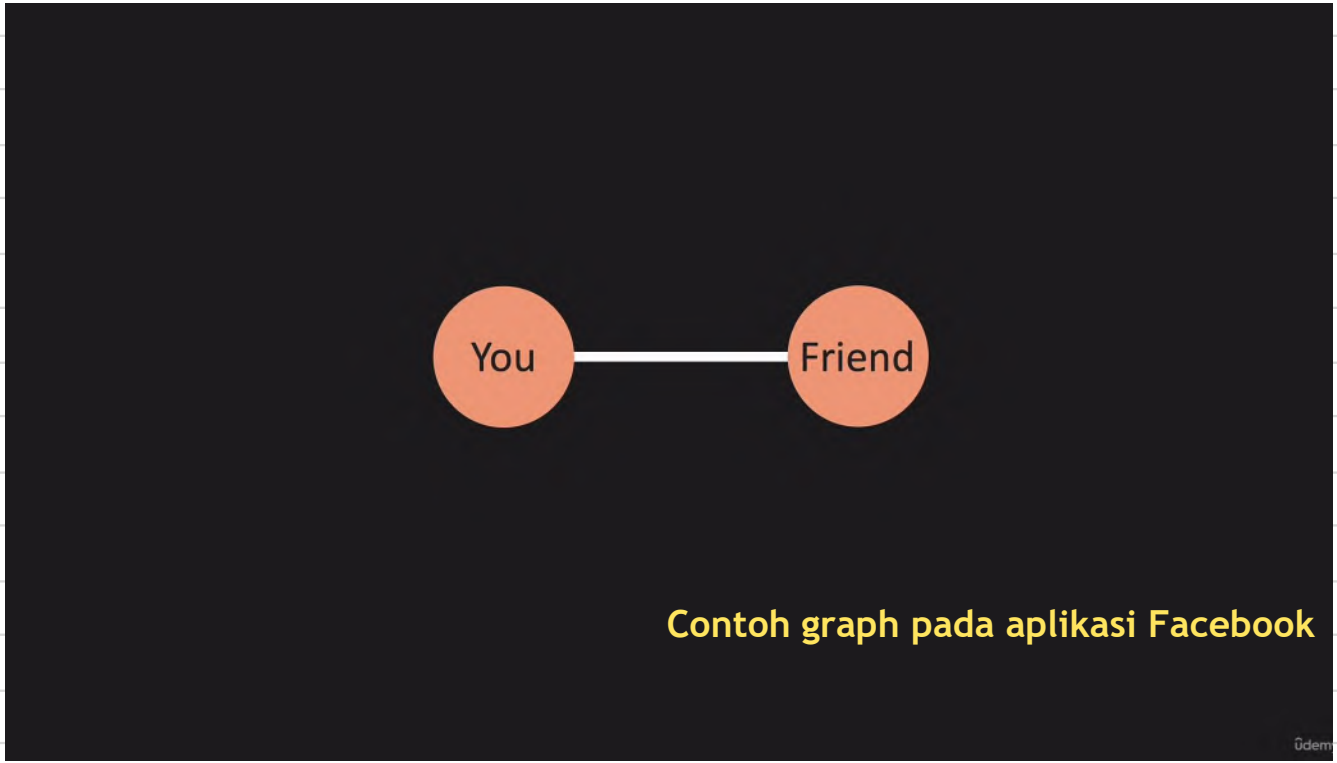
Node Graph

Dalam teori Graph, suatu edge juga bisa bersifat bidirectional



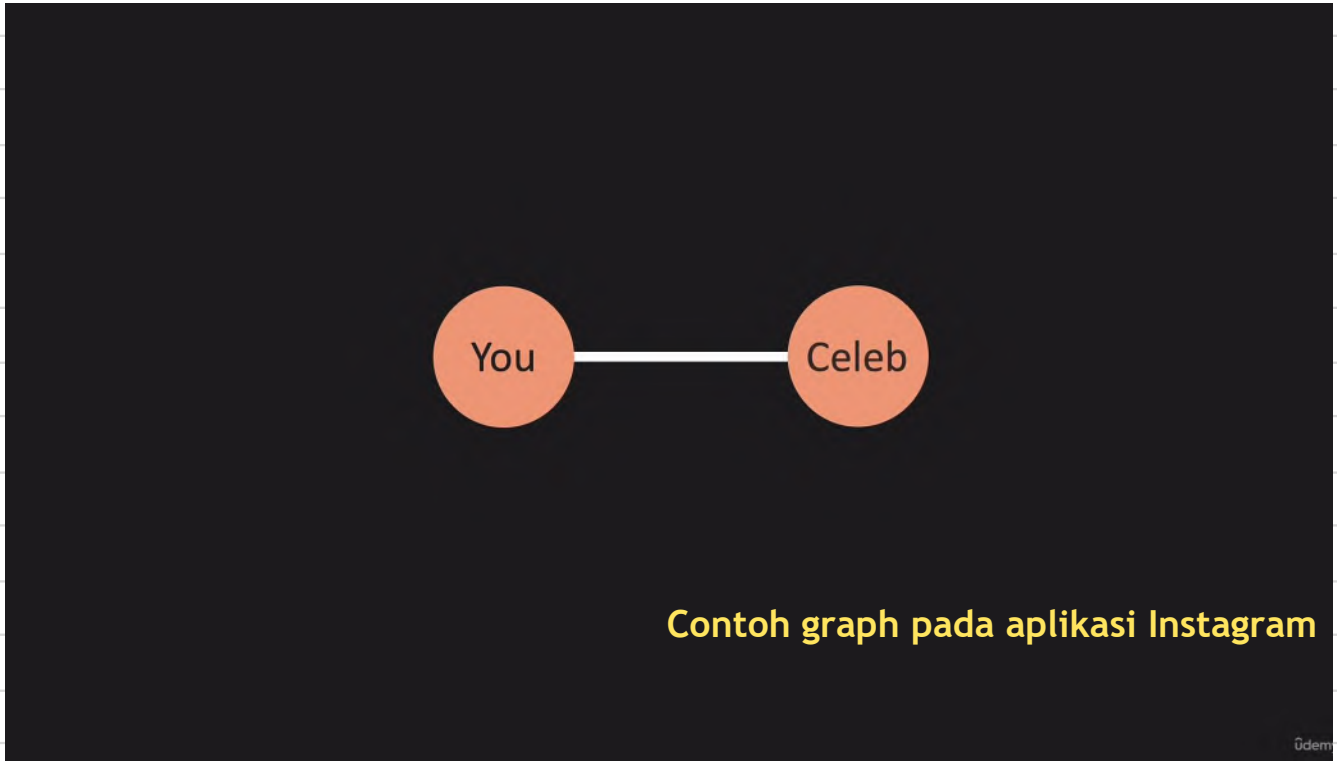
Node Graph

Simbol edge yang bersifat bidirectional
Berupa garis tanpa anak panah



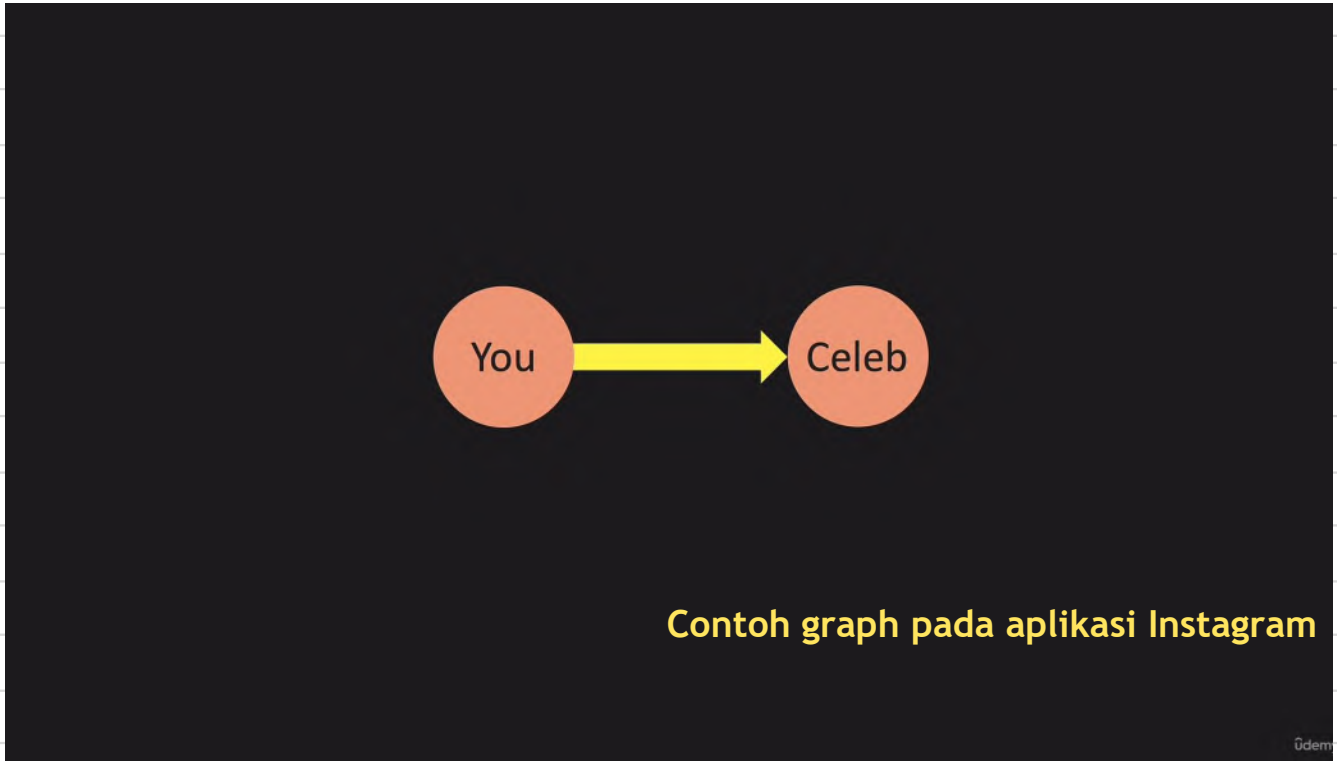
Node Graph

Apakah edge masih bersifat bidirectional?

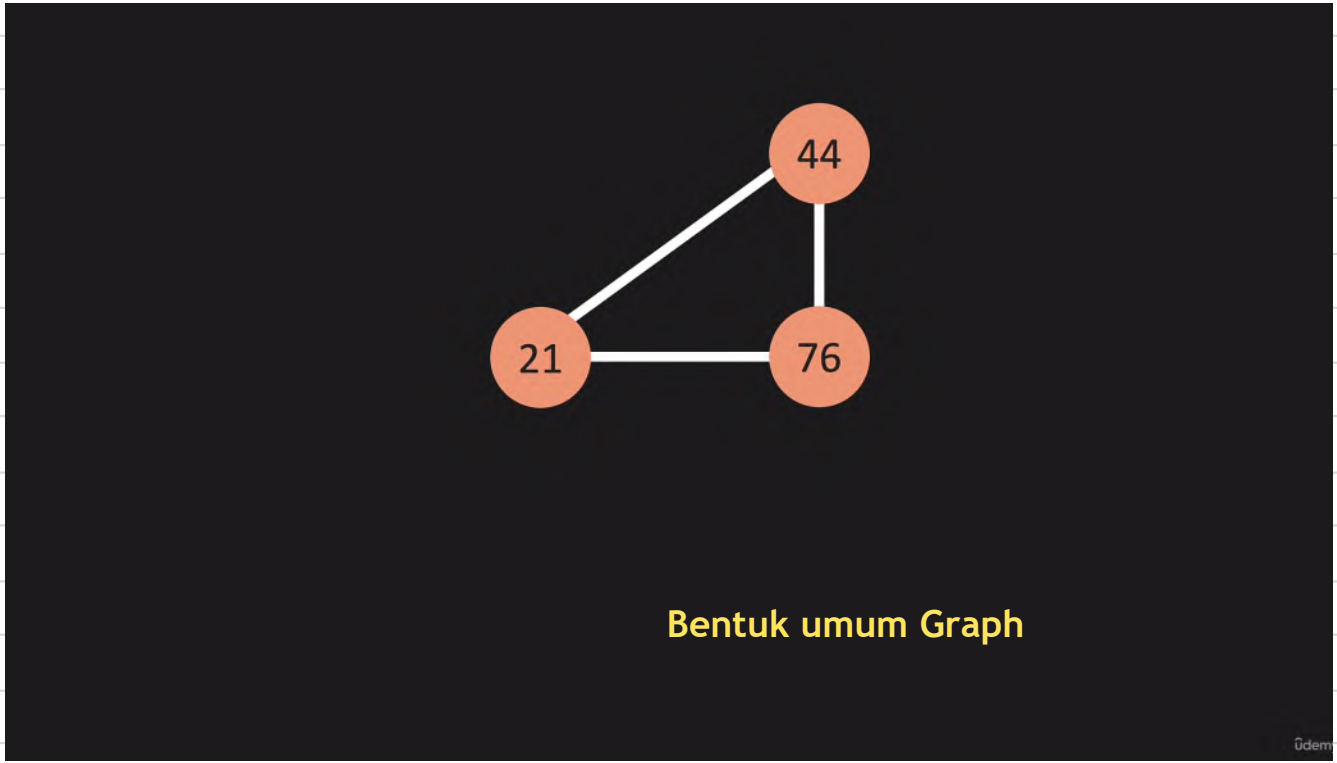


Node Graph

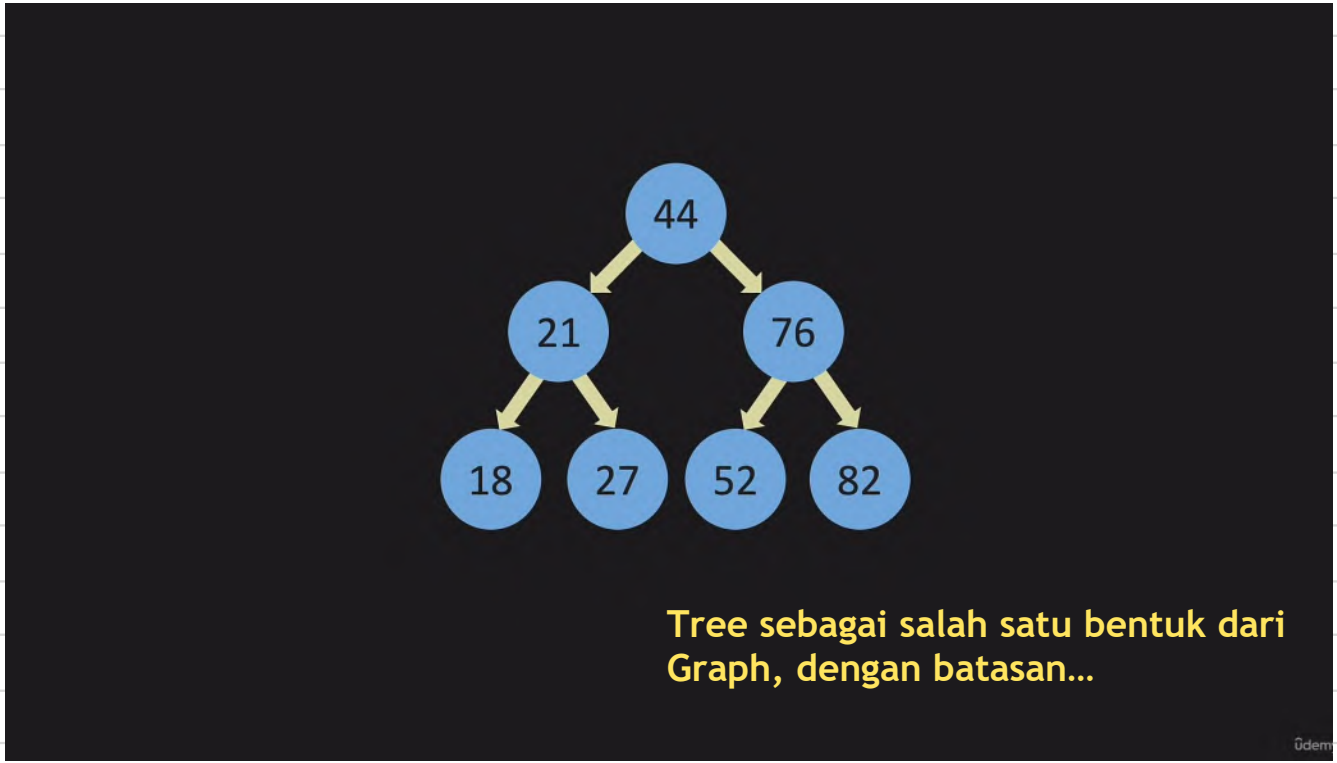
Edge bersifat directional (1 arah)



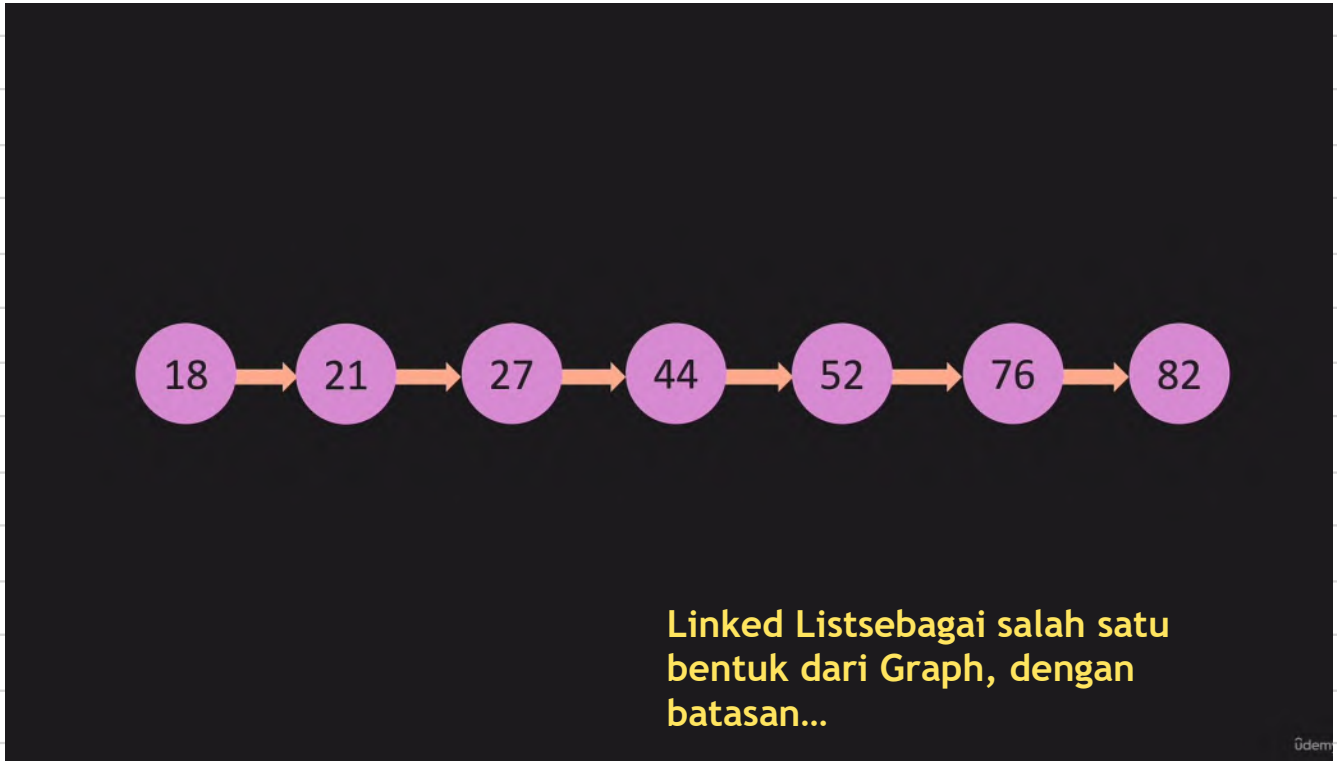
Graph



Tree

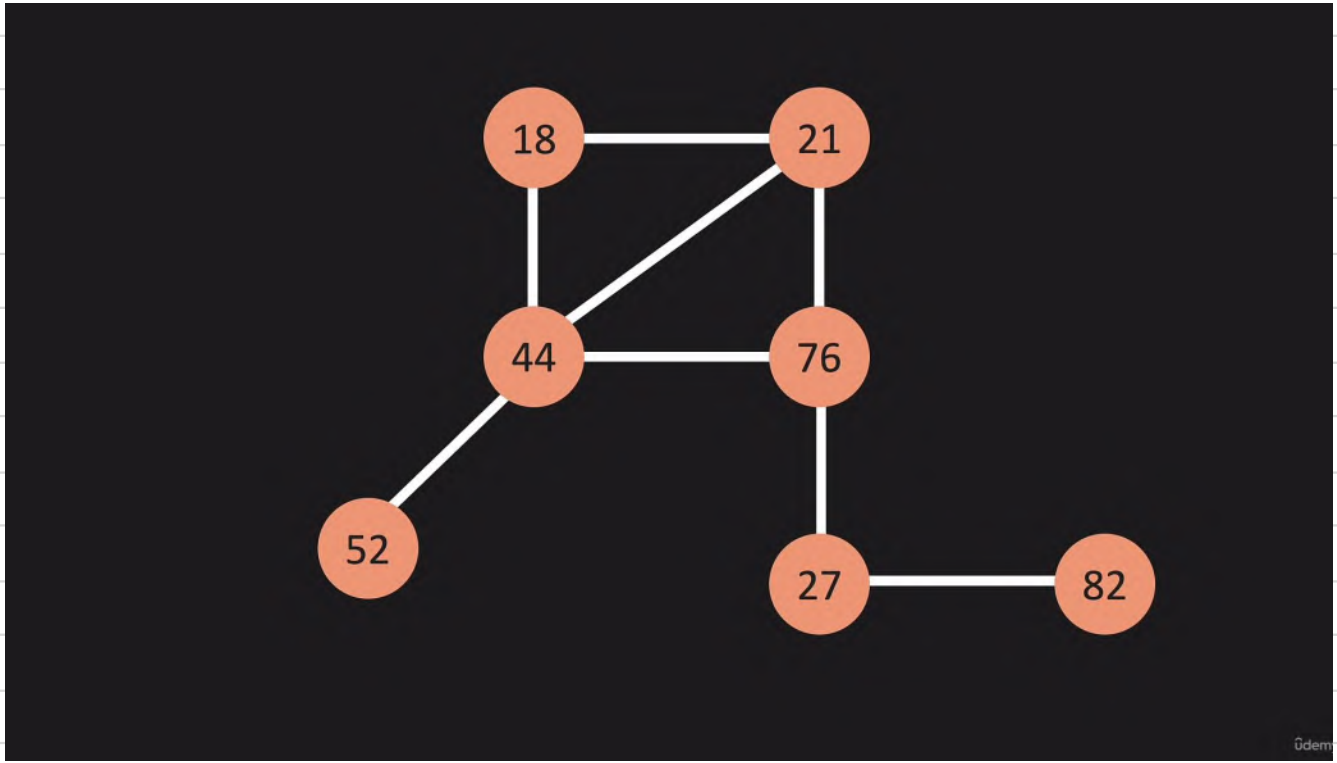

44
21
76
18
27
52
82
Tree sebagai salah satu bentuk dari Graph, dengan batasan...

Linked List


18
21
27
44
52
76
82
Linked Listsebagai salah satu bentuk dari Graph, dengan batasan...

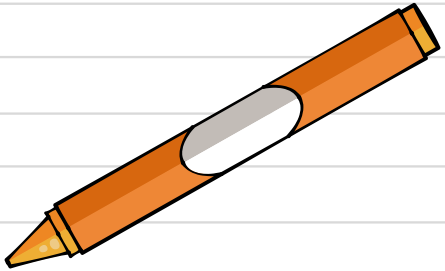
Graph

Graph yang umumnya ada di mindset kita



02

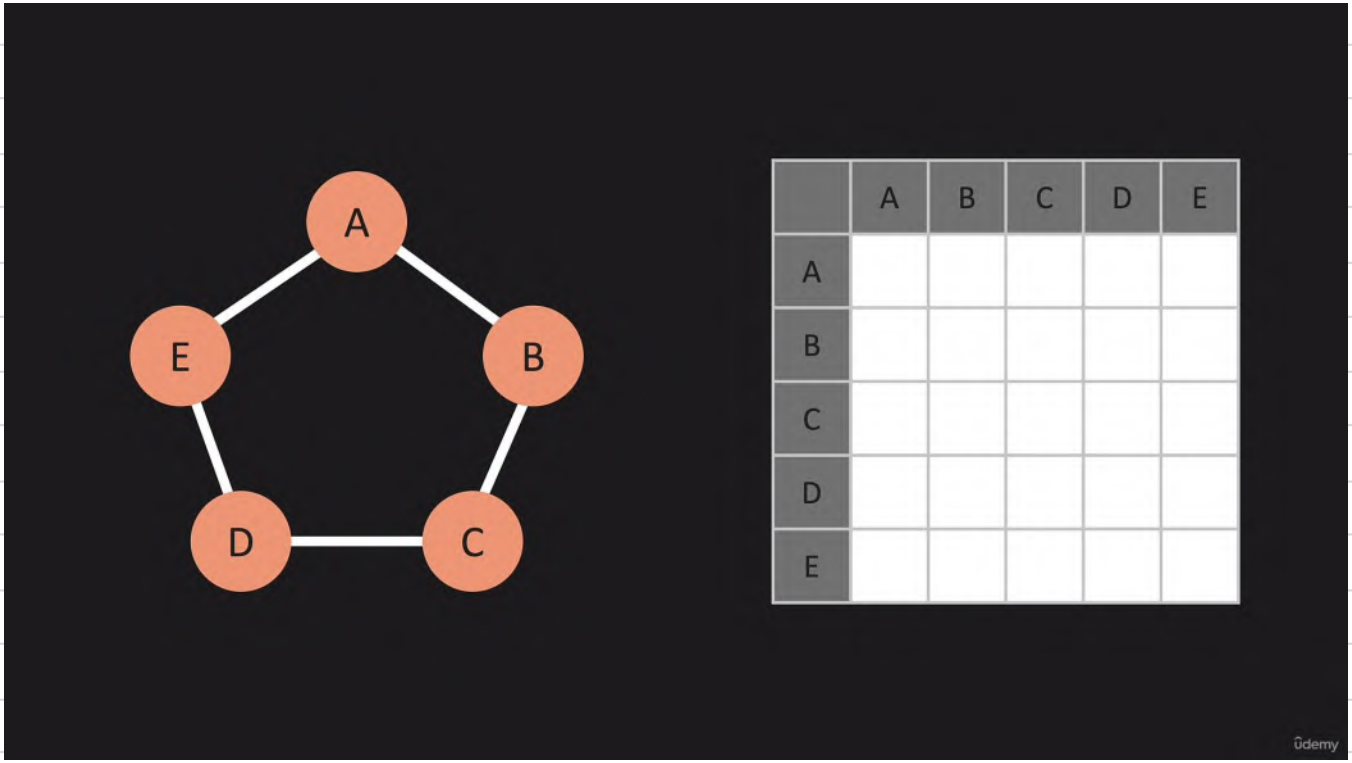
Adjacency Matrix & Adjacency List



Adjacency Matrix

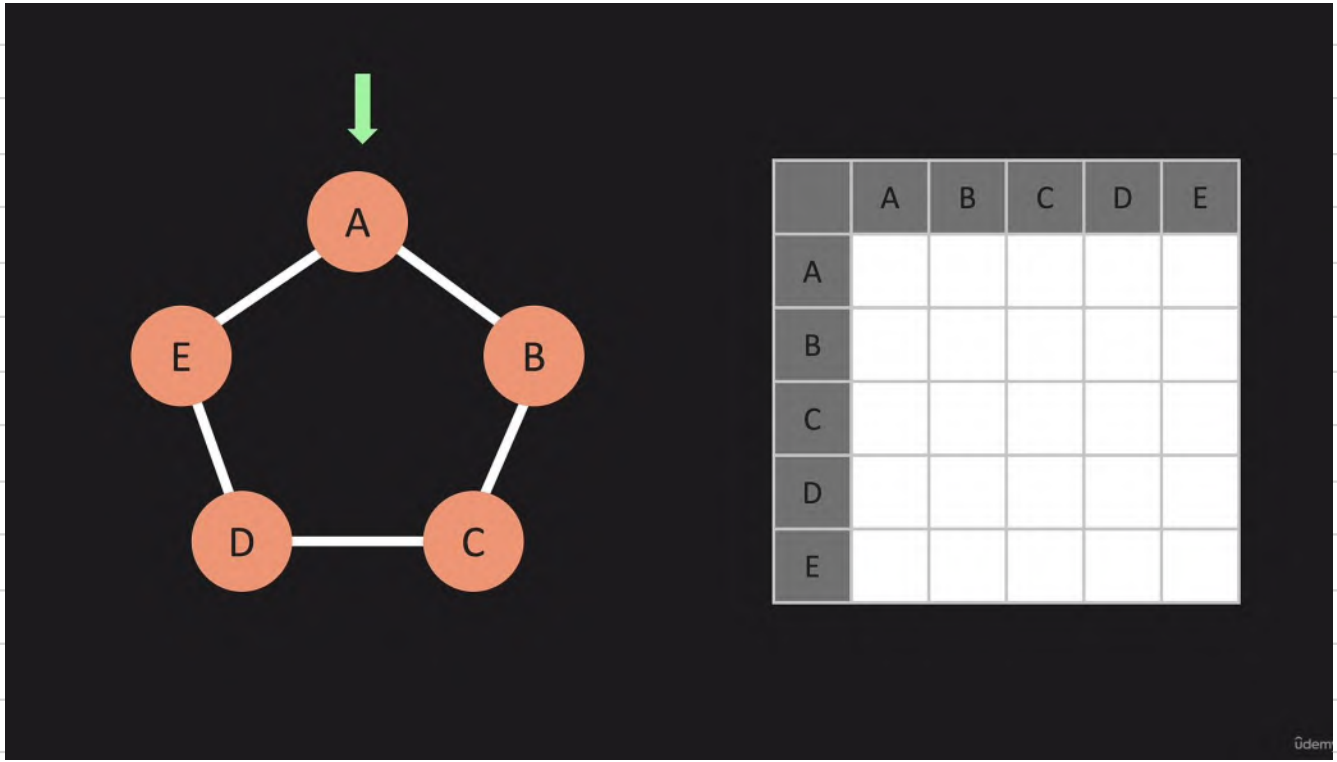
- Terdapat 2 cara untuk mewujudkan suatu Graph di dalam bahasa pemrograman
- Salah satu caranya adalah menggunakan Adjacency Matrix

Adjacency Matrix



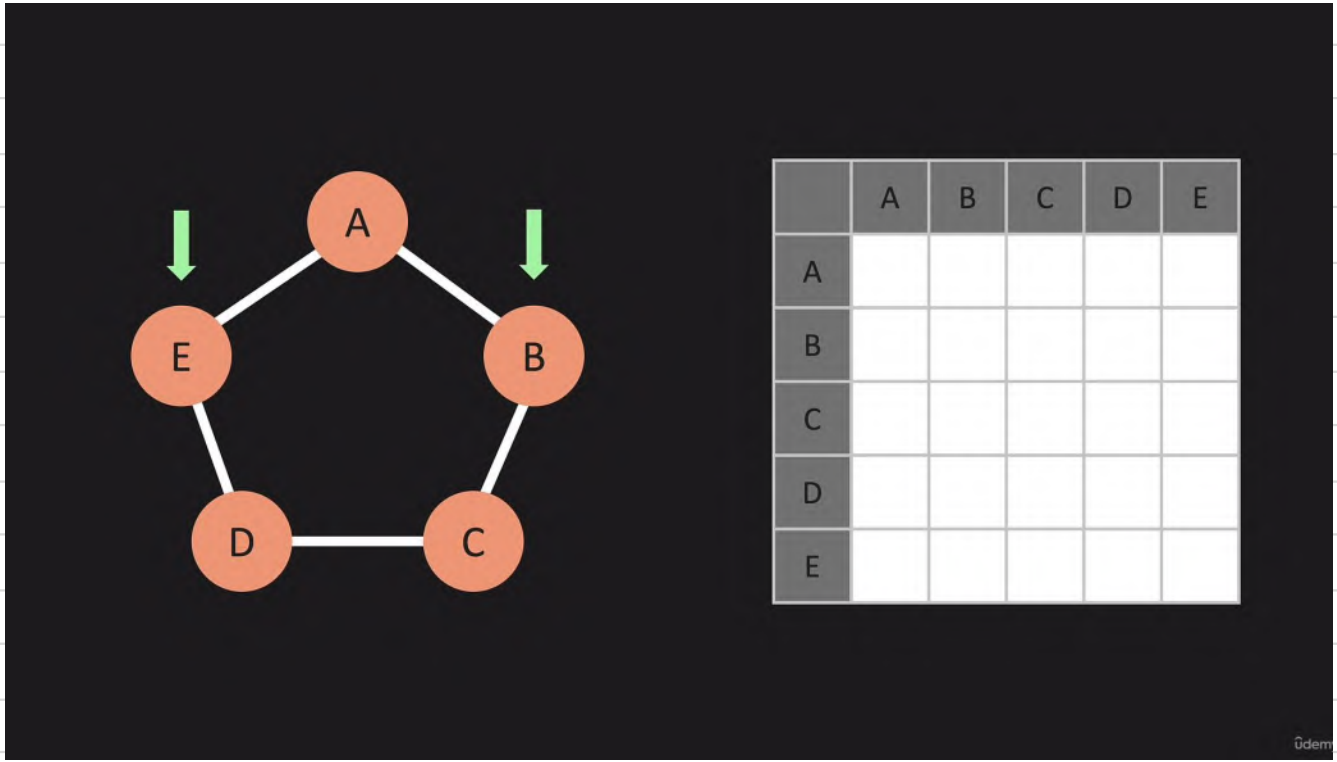
Adjacency Matrix

Jika kita melihat vertex A



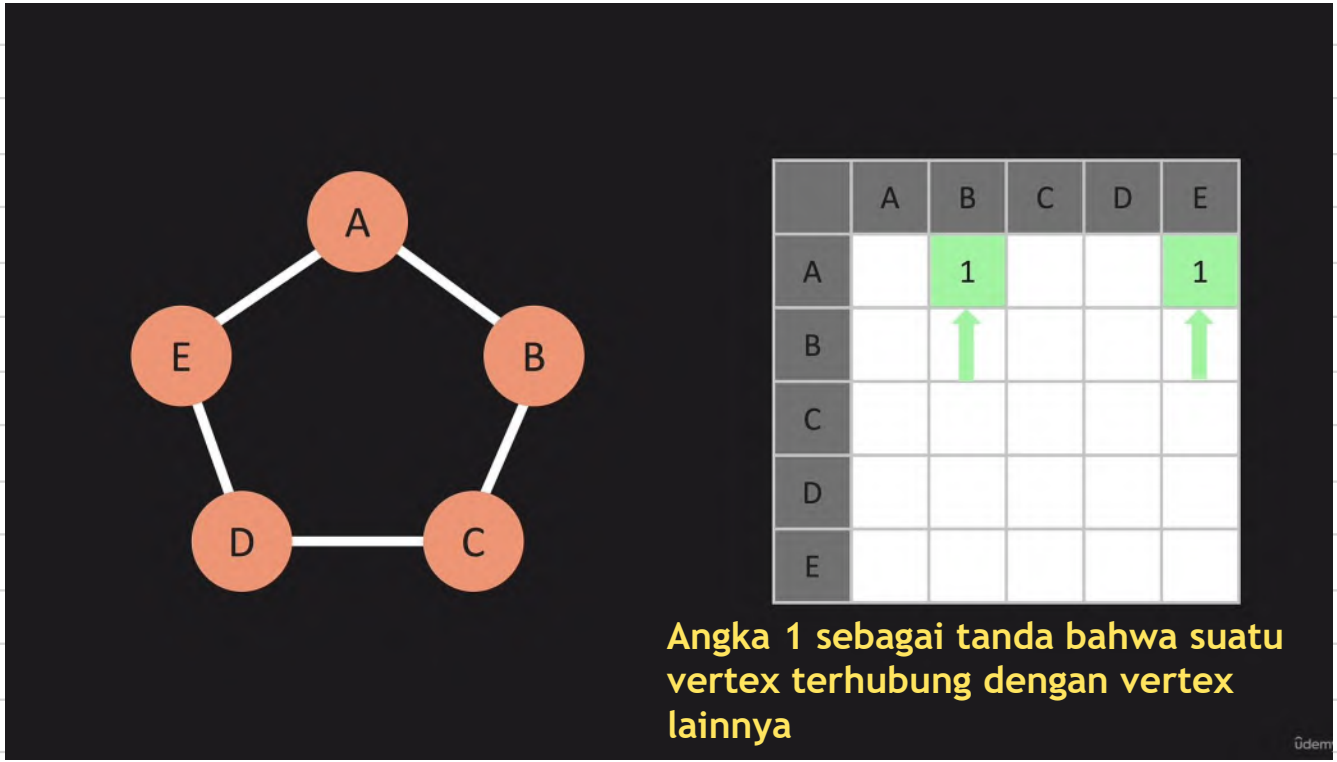
Adjacency Matrix

Vertex A terhubung ke B dan E



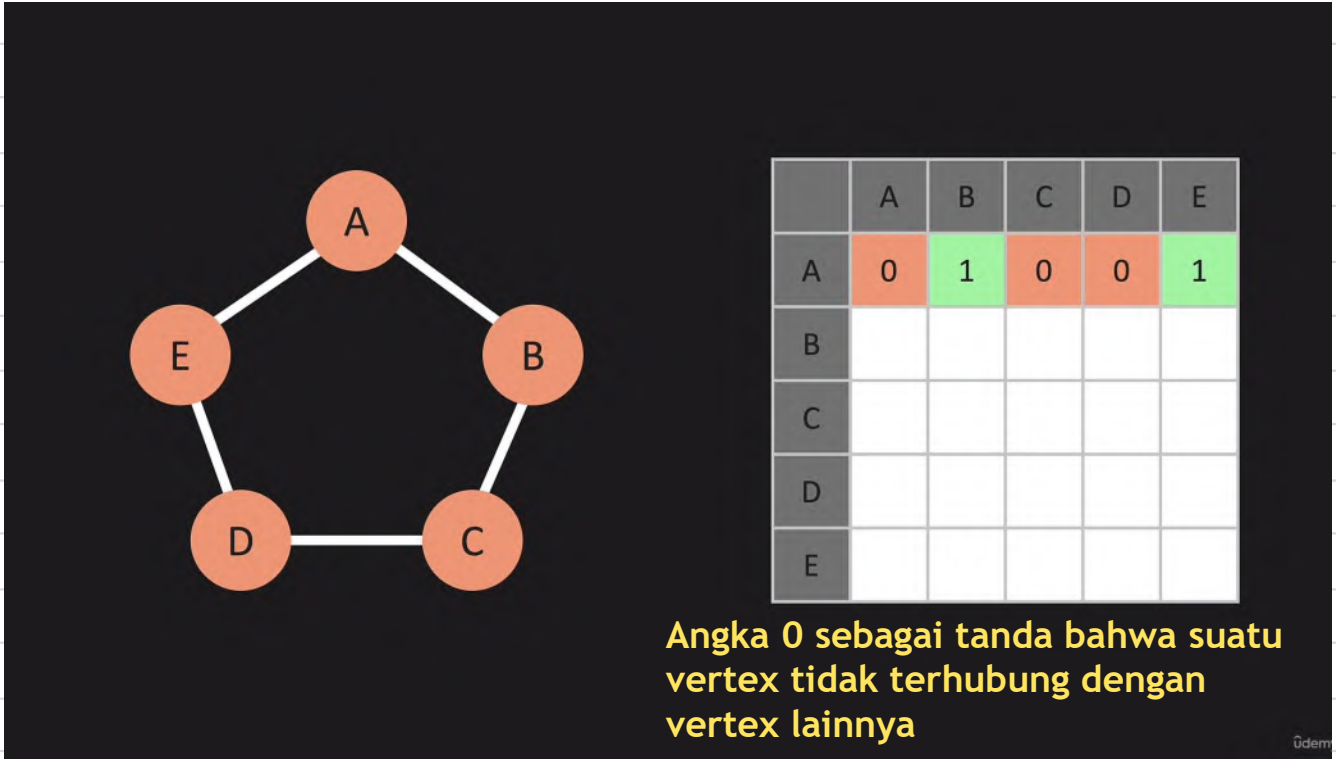
Adjacency Matrix

Maka pada kolom B dan E kita berikan angka 1



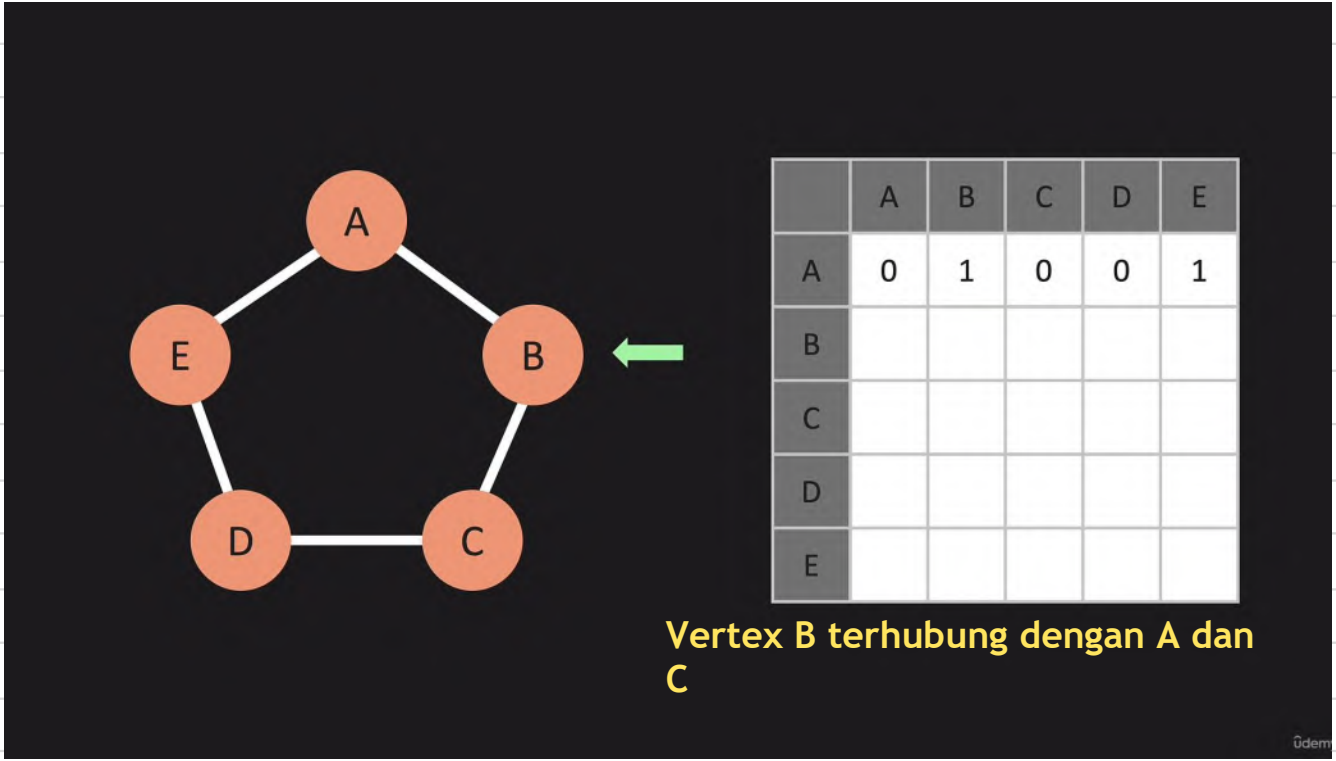
Adjacency Matrix

Maka pada kolom B dan E kita berikan angka 1



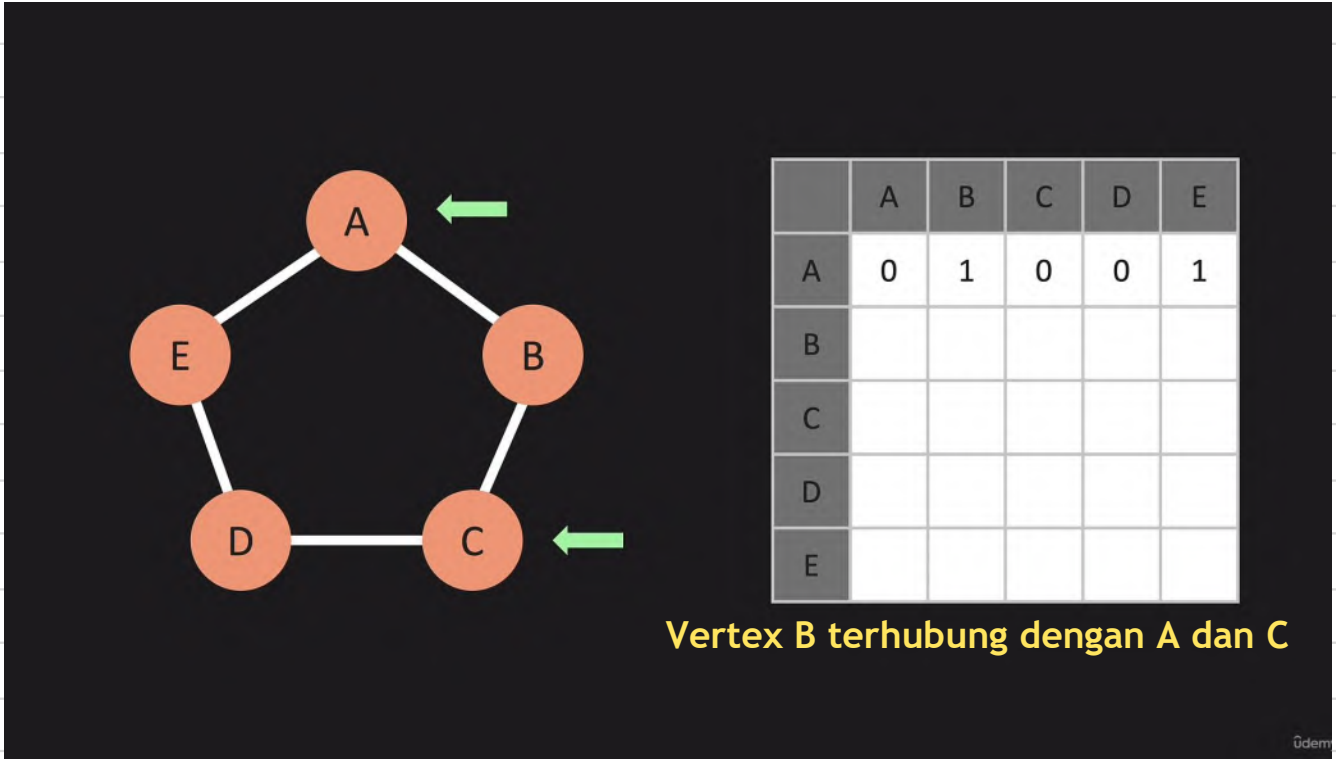
Adjacency Matrix

Jika kita melihat vertex B



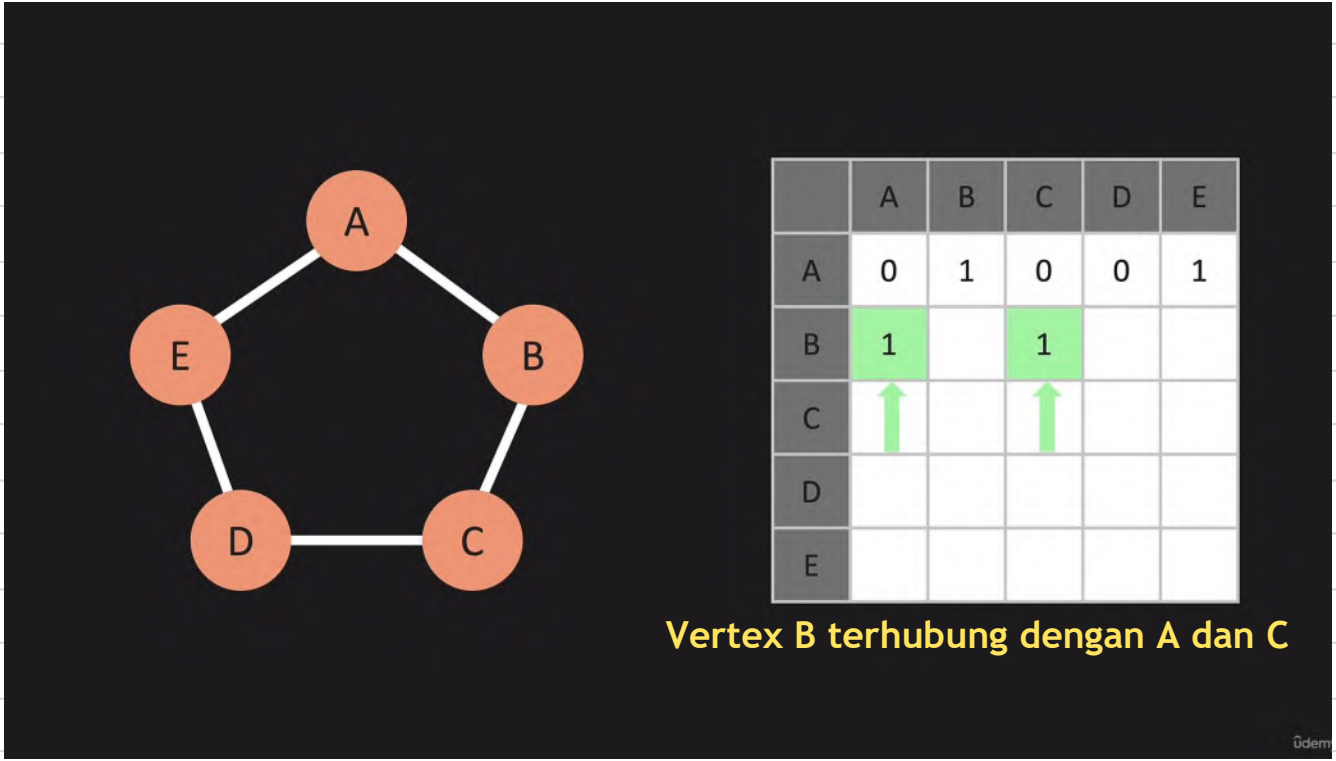
Adjacency Matrix

Jika kita melihat vertex B



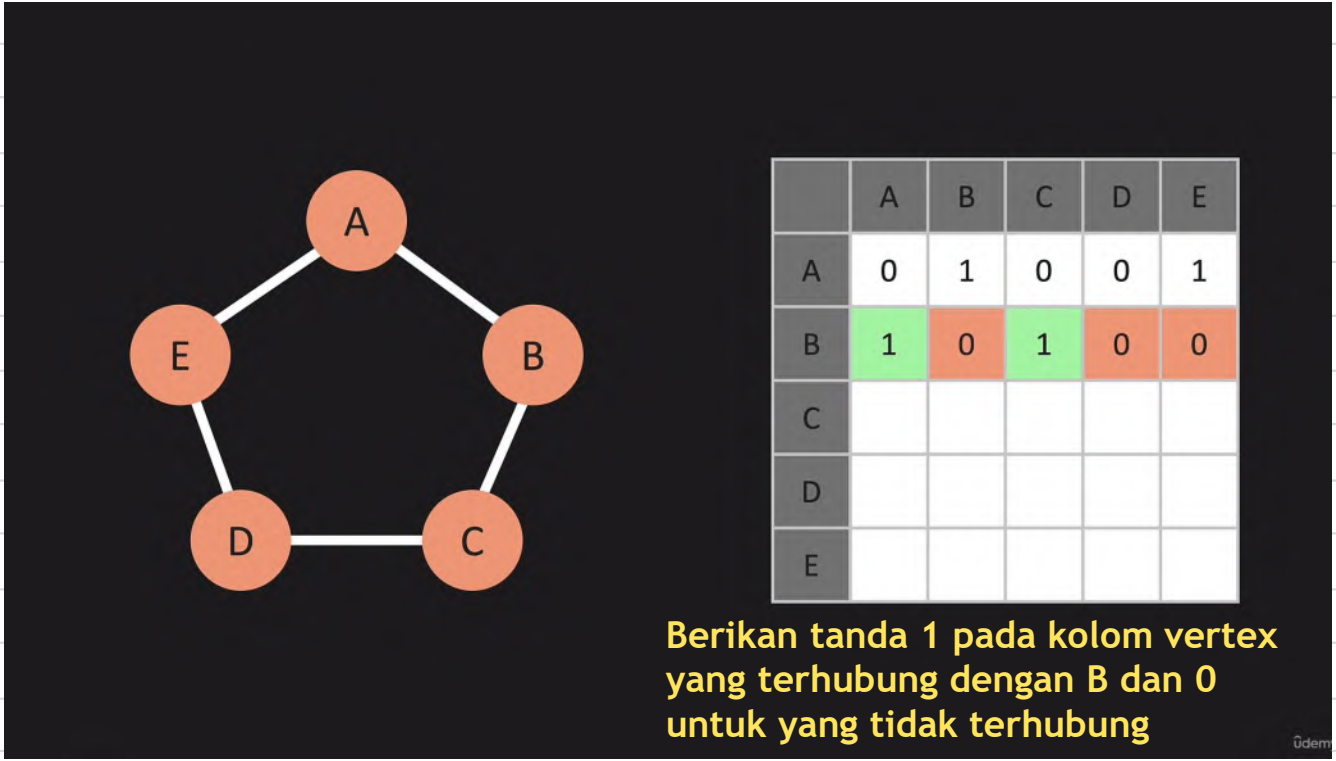
Adjacency Matrix

Jika kita melihat vertex B



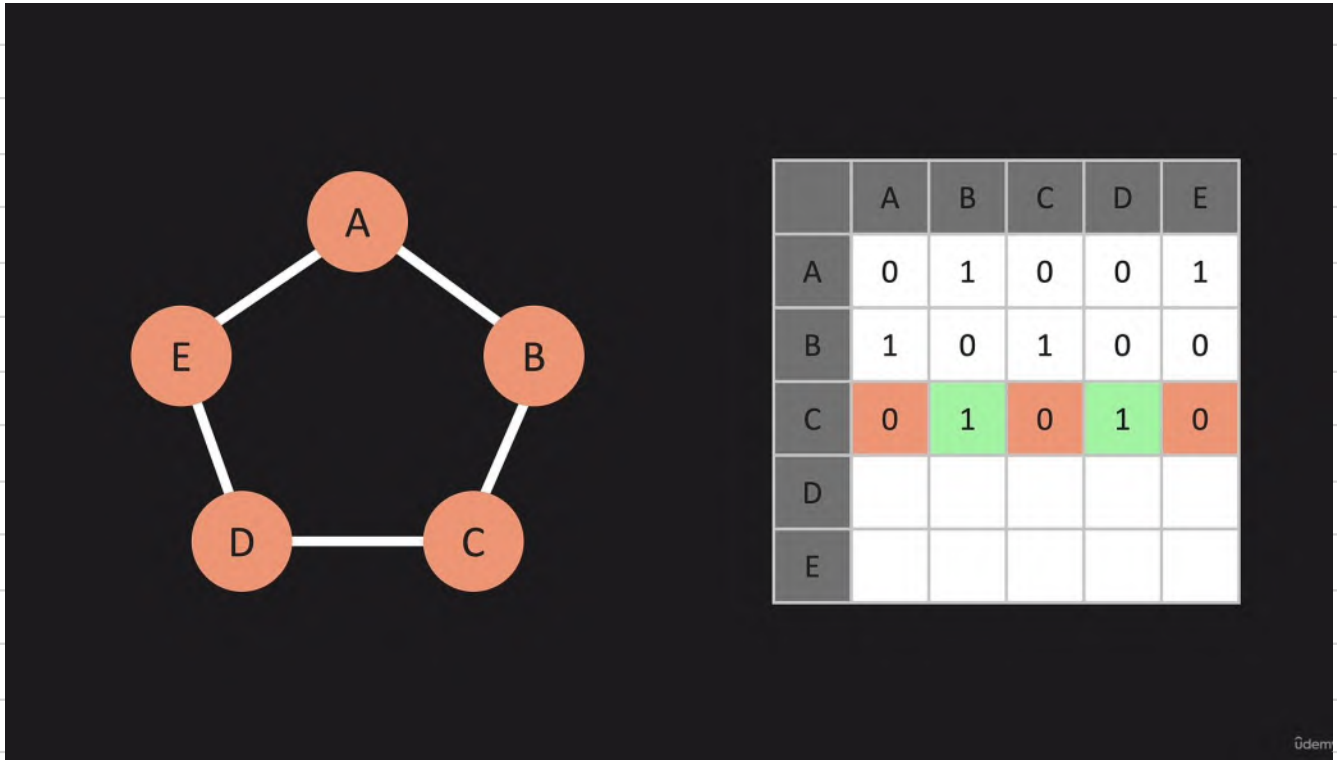
Adjacency Matrix

Jika kita melihat vertex B



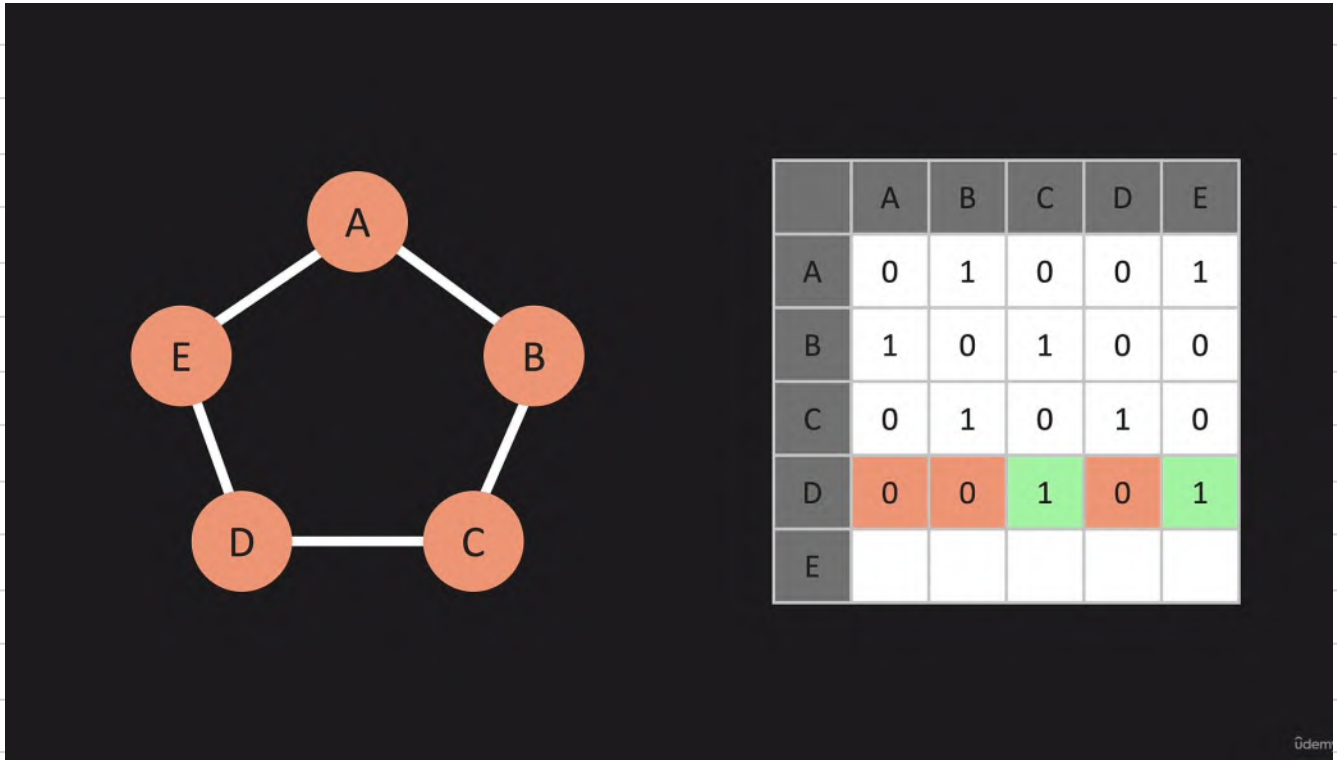
Adjacency Matrix

Isi Adjacency Matrix untuk Vertex C



Adjacency Matrix

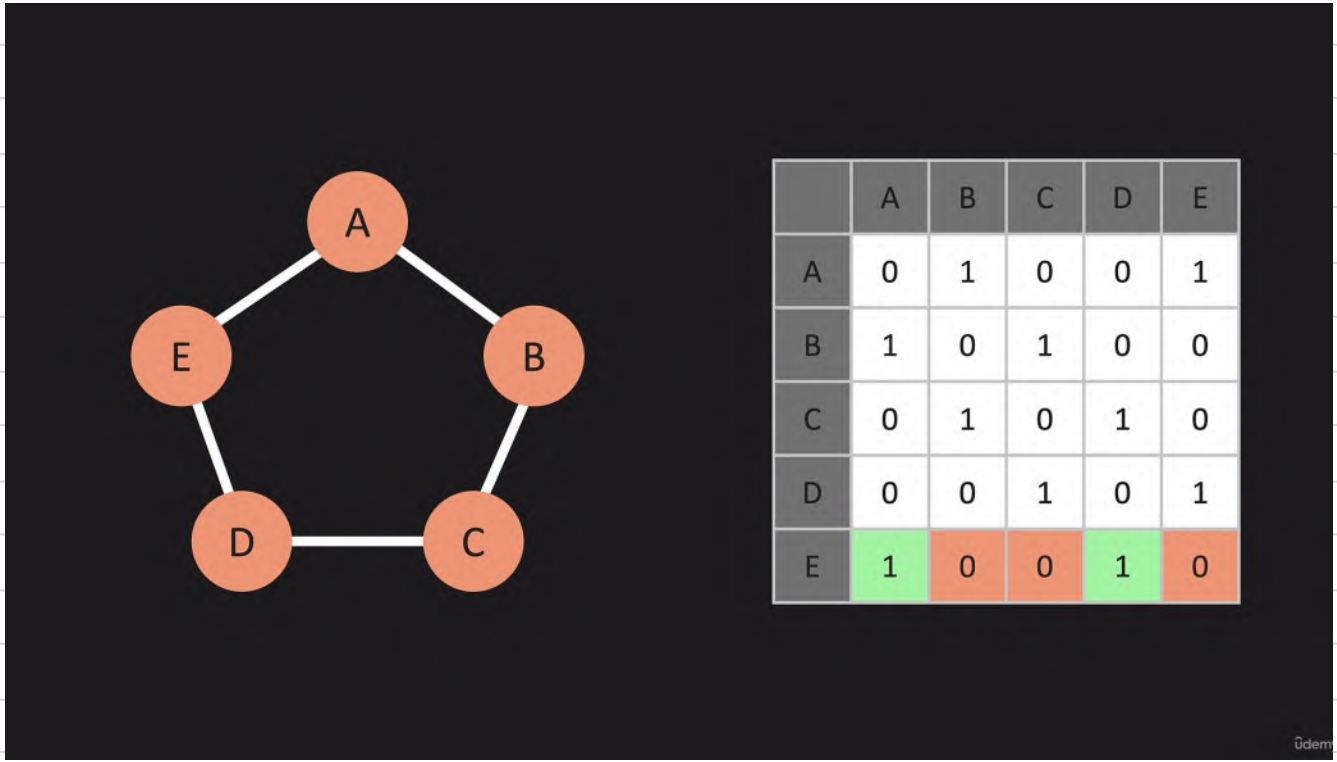
Isi Adjacency Matrix untuk Vertex D



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E					

Adjacency Matrix

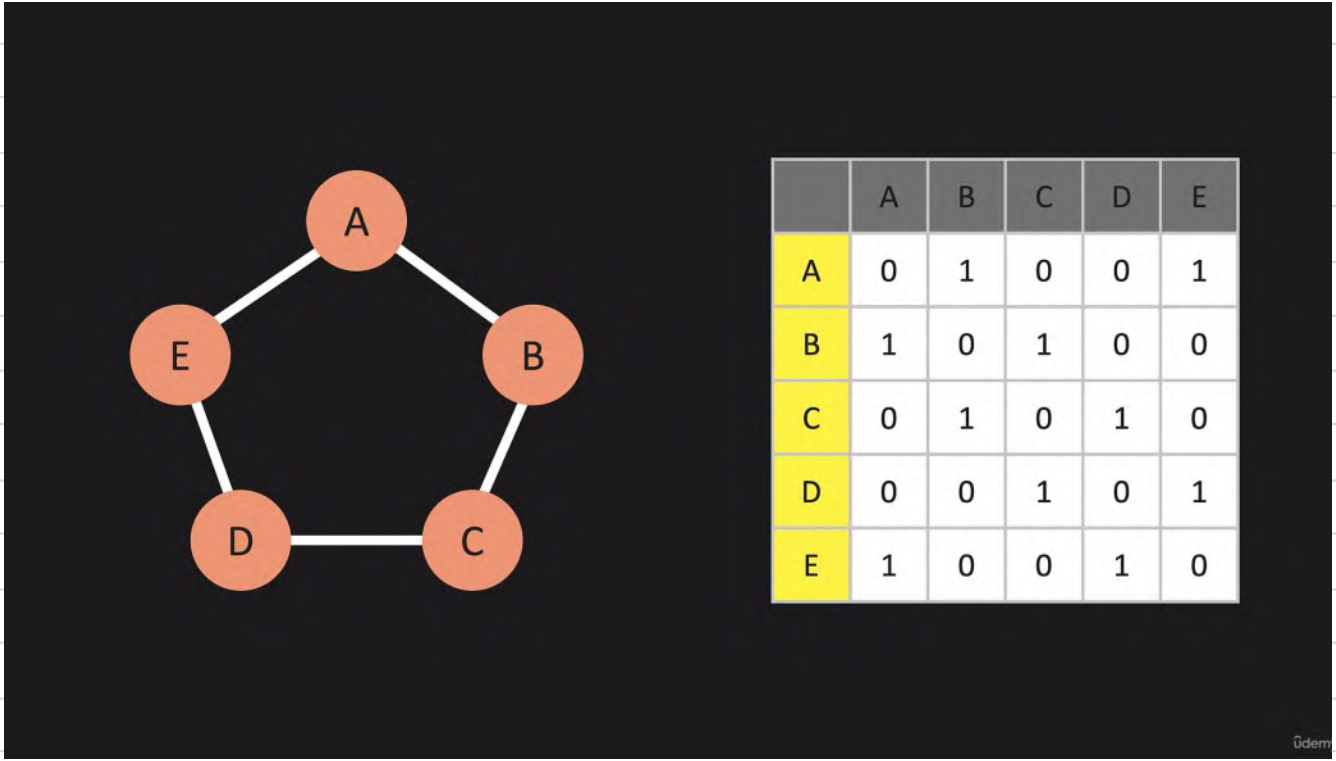
Isi Adjacency Matrix untuk Vertex E



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Adjacency Matrix

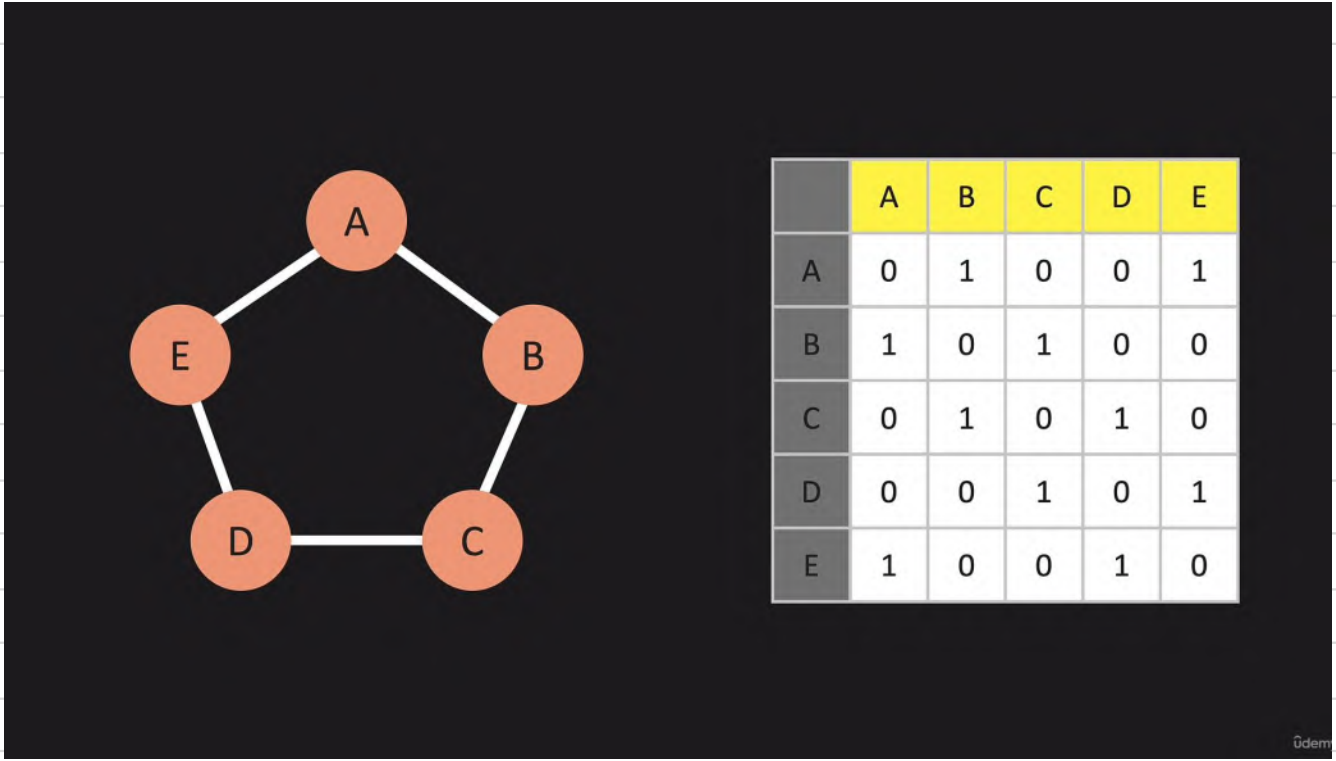
Baris dengan warna kuning adalah vertex sebagai titik asal



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Adjacency Matrix

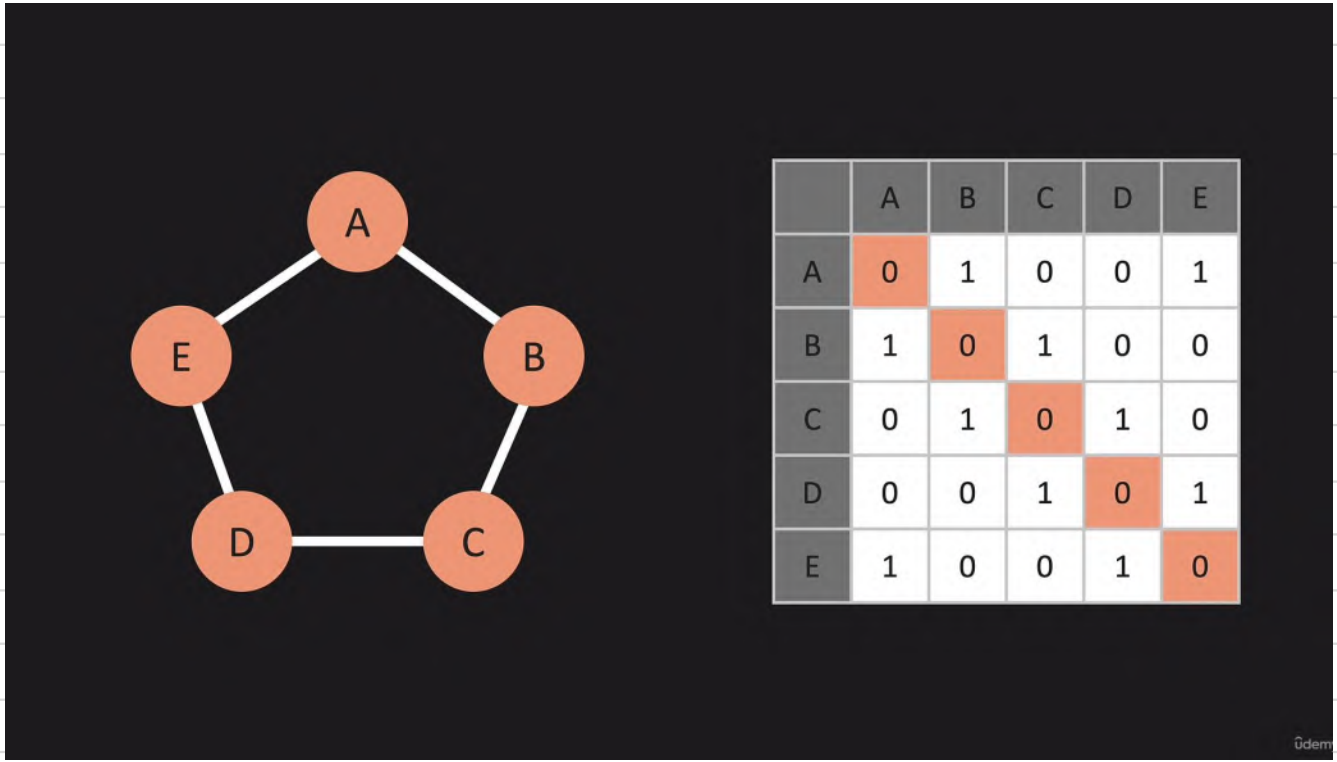
Kolom dengan warna kuning adalah vertex yang terhubung dengan titik asal



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Adjacency Matrix

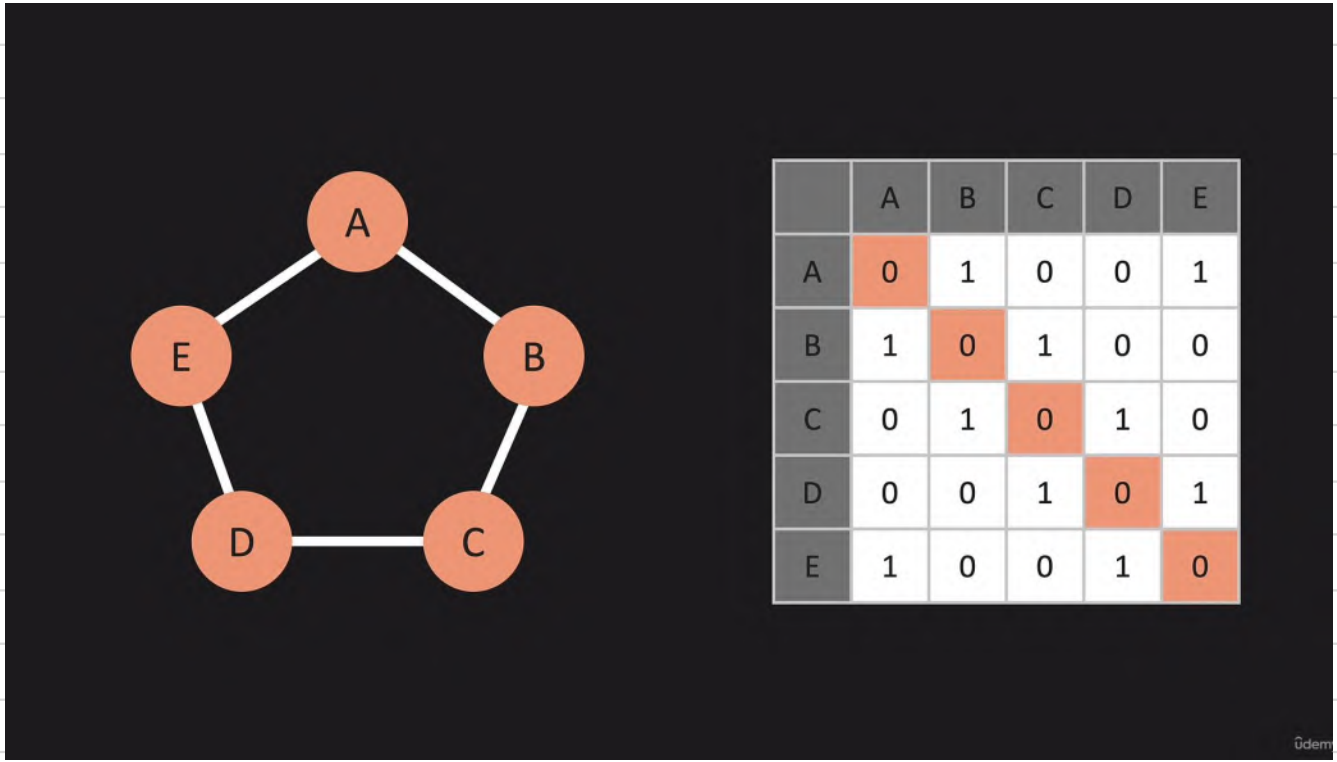
Setiap Adjacency Matrix selalu memiliki diagonal yang bernilai 0 semua



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Adjacency Matrix

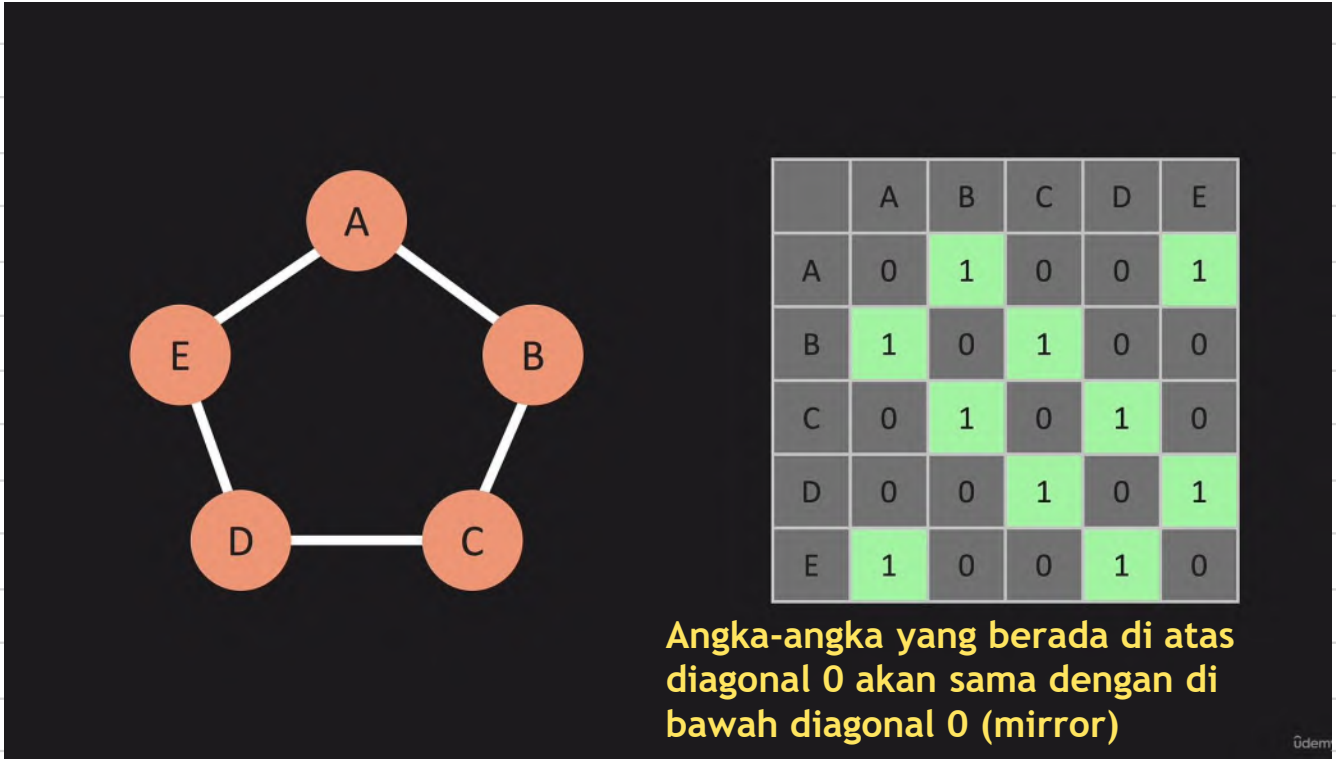
Setiap Adjacency Matrix selalu memiliki diagonal yang bernilai 0 semua



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Adjacency Matrix

Jika Graph bersifat bidirectional

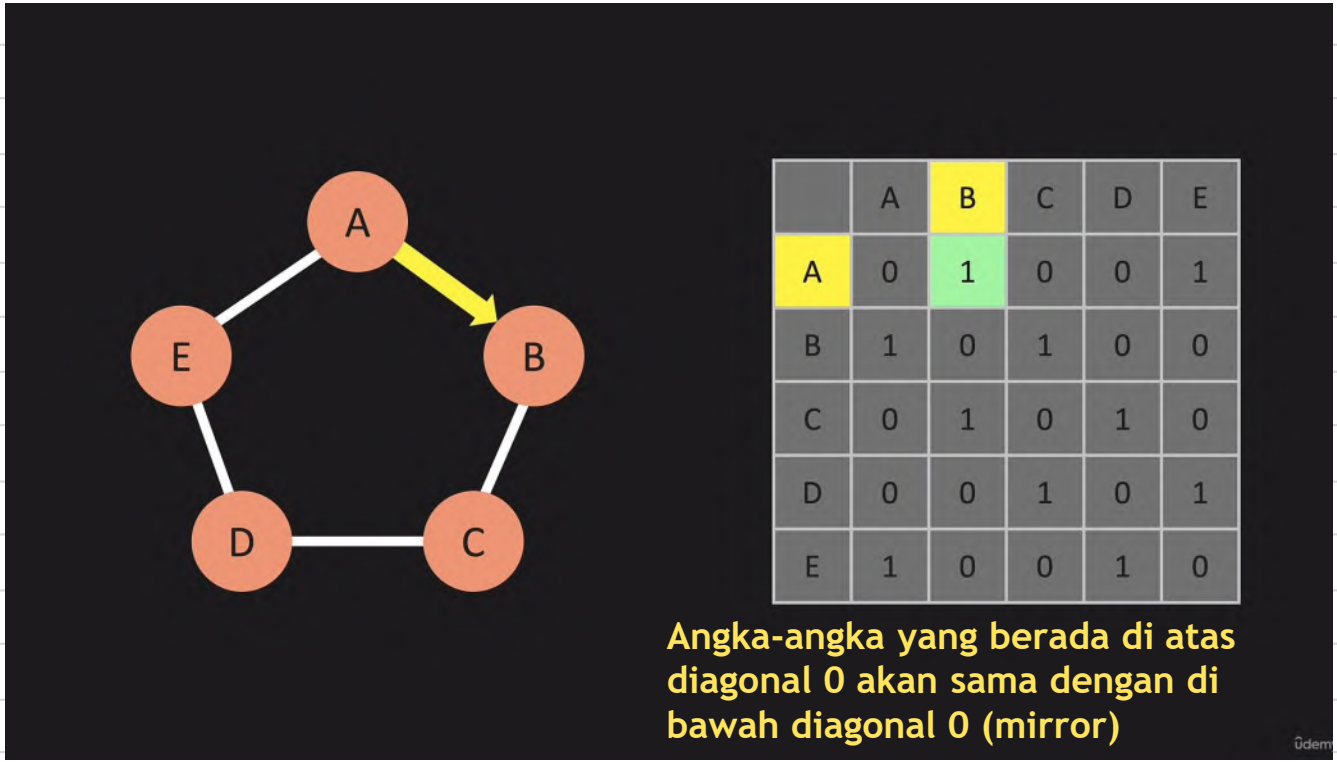


	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Angka-angka yang berada di atas diagonal 0 akan sama dengan di bawah diagonal 0 (mirror)

Adjacency Matrix

Jika Graph bersifat bidirectional

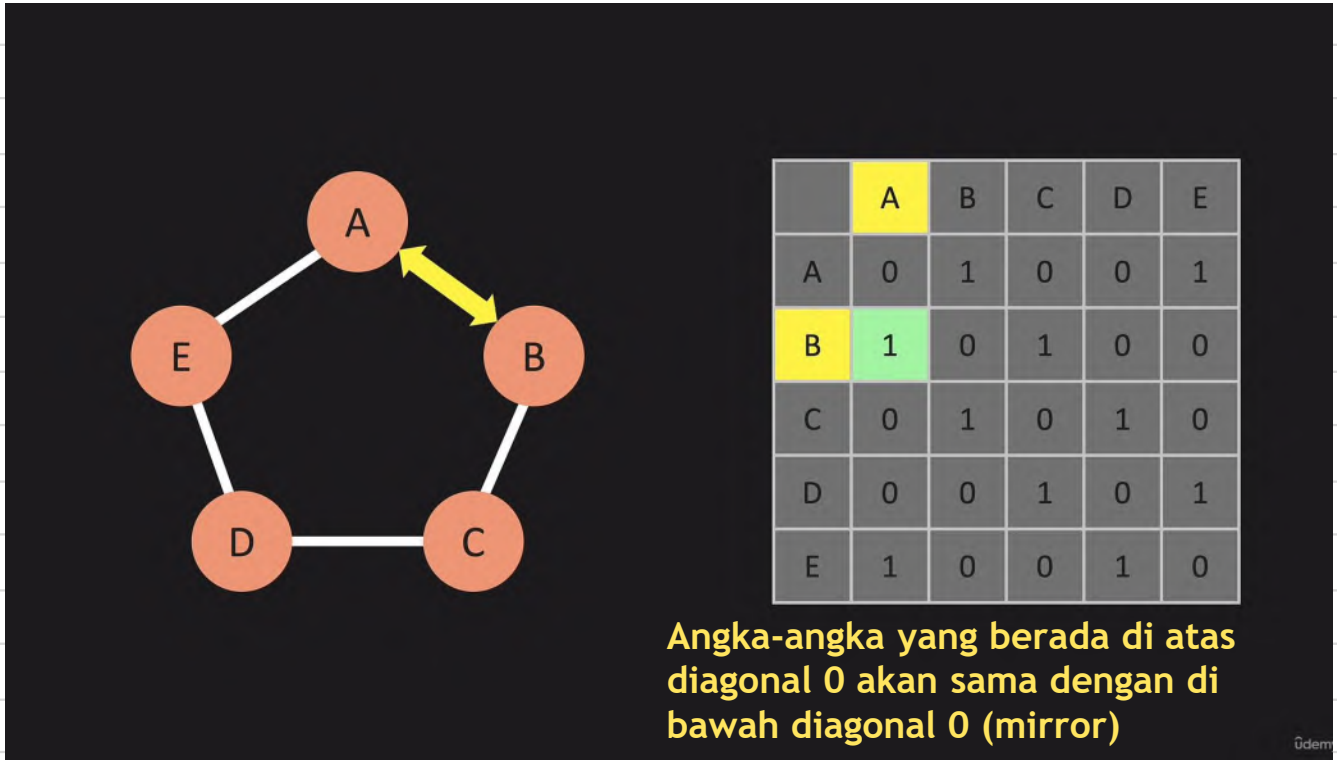


	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Angka-angka yang berada di atas diagonal 0 akan sama dengan di bawah diagonal 0 (mirror)

Adjacency Matrix

Jika Graph bersifat bidirectional

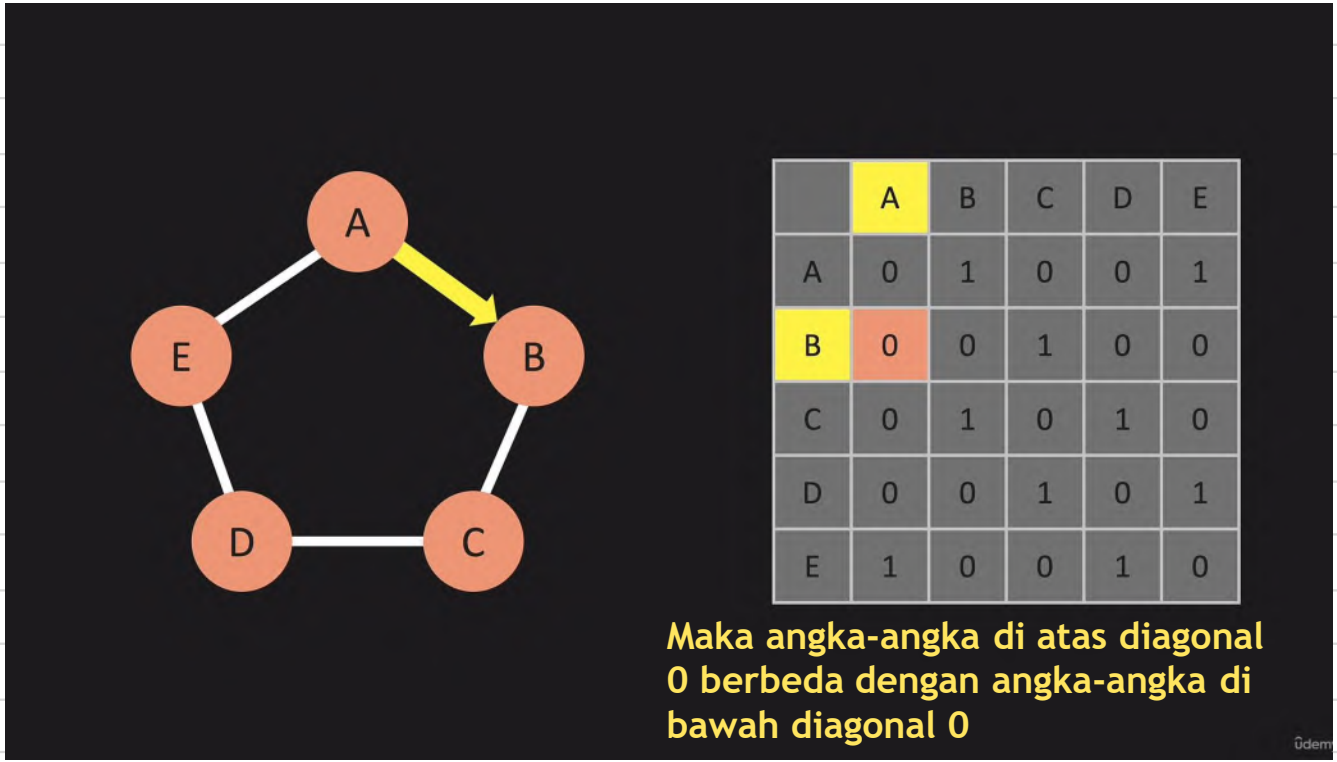


	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Angka-angka yang berada di atas diagonal 0 akan sama dengan di bawah diagonal 0 (mirror)

Adjacency Matrix

Namun, jika Graph bersifat 1 directional

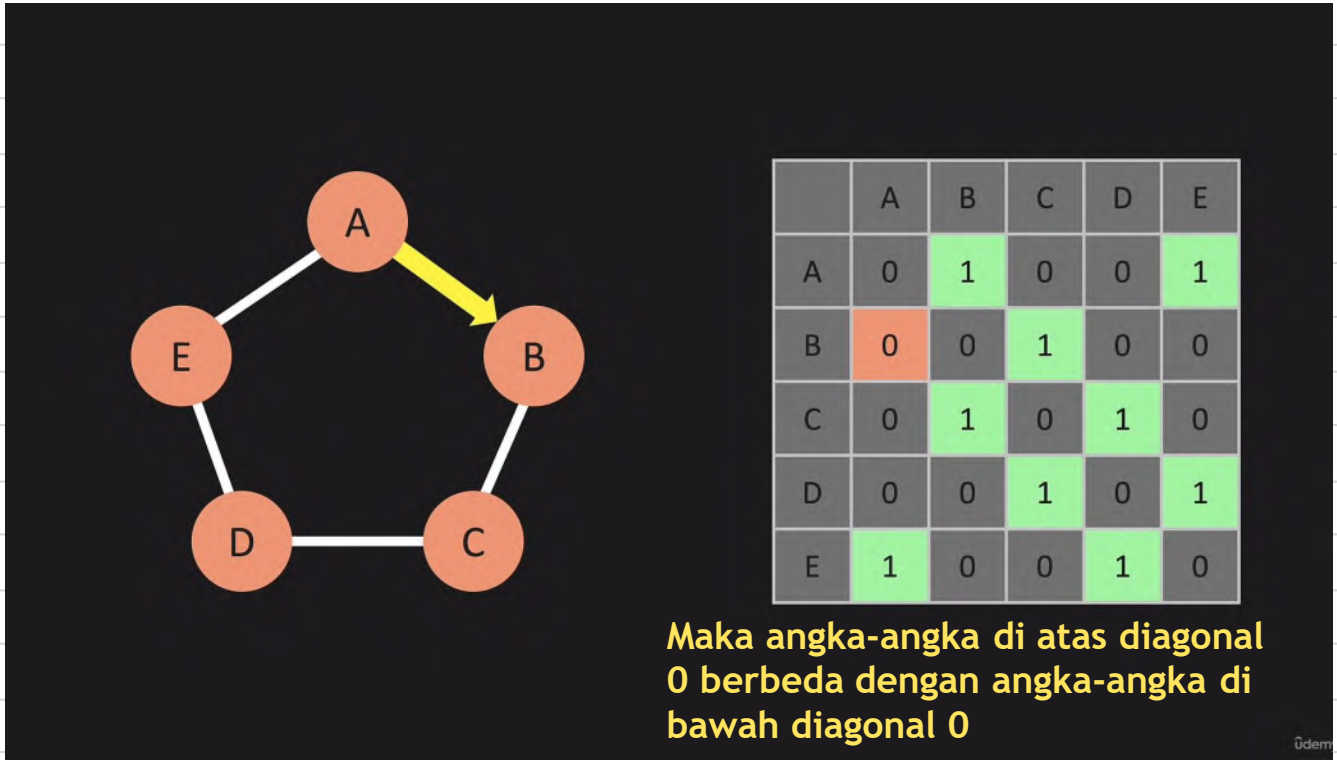


	A	B	C	D	E
A	0	1	0	0	1
B	0	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Maka angka-angka di atas diagonal 0 berbeda dengan angka-angka di bawah diagonal 0

Adjacency Matrix

Namun, jika Graph bersifat 1 directional

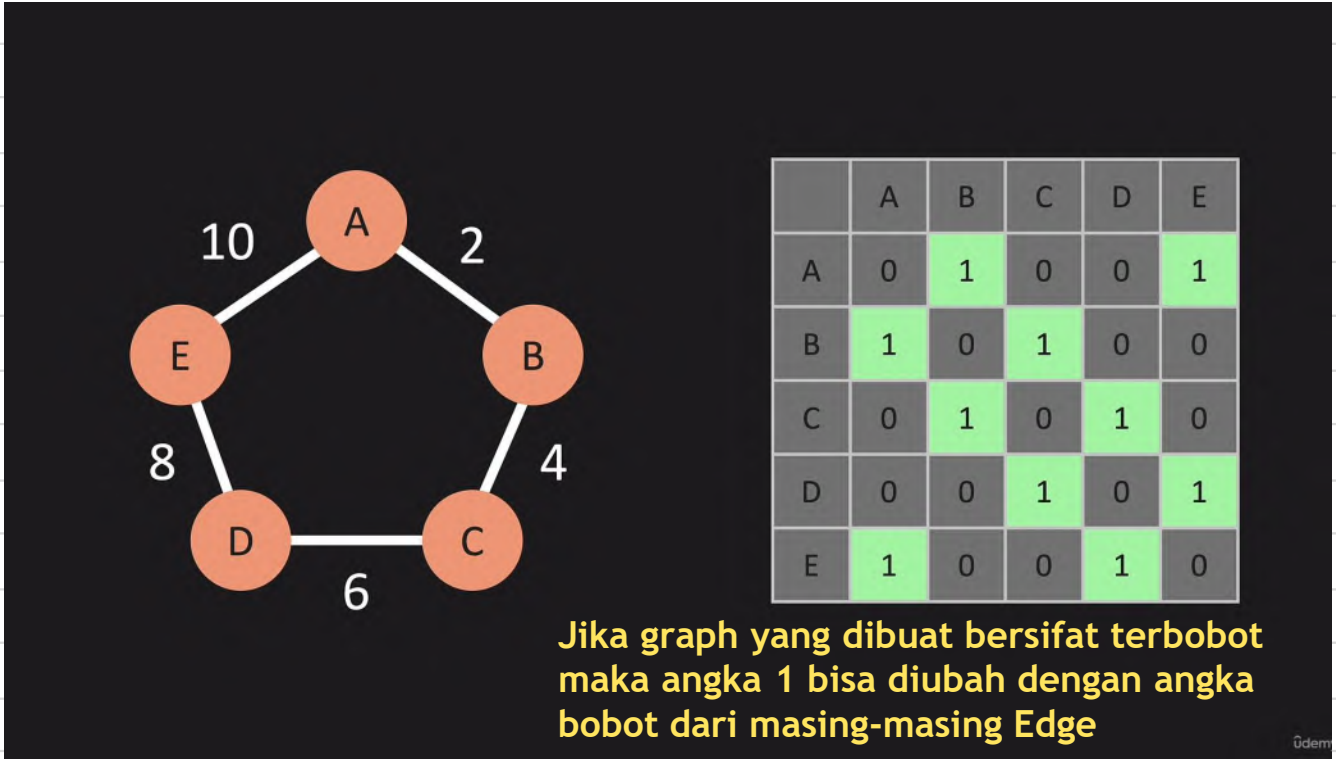


	A	B	C	D	E
A	0	1	0	0	1
B	0	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Maka angka-angka di atas diagonal 0 berbeda dengan angka-angka di bawah diagonal 0

Adjacency Matrix

Graph juga bisa bersifat weighted (terbobot)

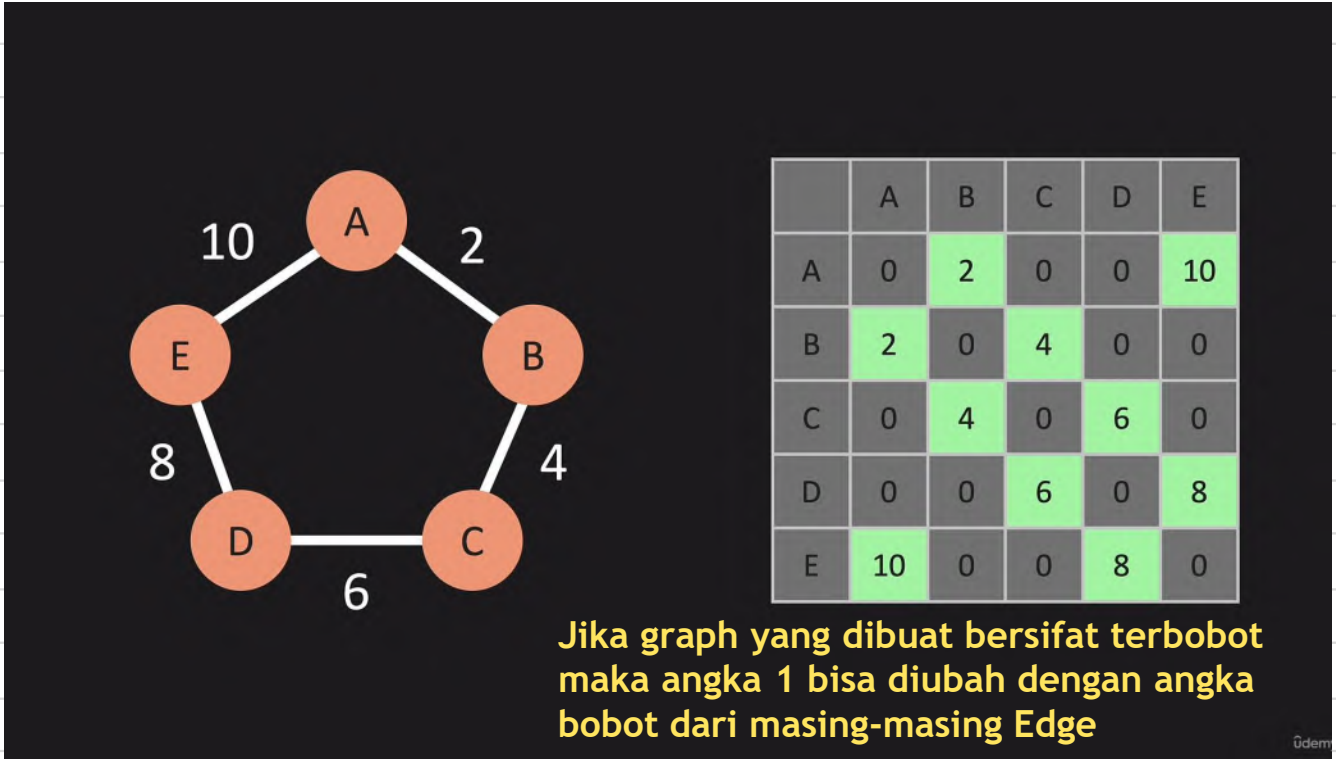


	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Jika graph yang dibuat bersifat terbobot maka angka 1 bisa diubah dengan angka bobot dari masing-masing Edge

Adjacency Matrix

Graph juga bisa bersifat weighted (terbobot)



	A	B	C	D	E
A	0	2	0	0	10
B	2	0	4	0	0
C	0	4	0	6	0
D	0	0	6	0	8
E	10	0	0	8	0

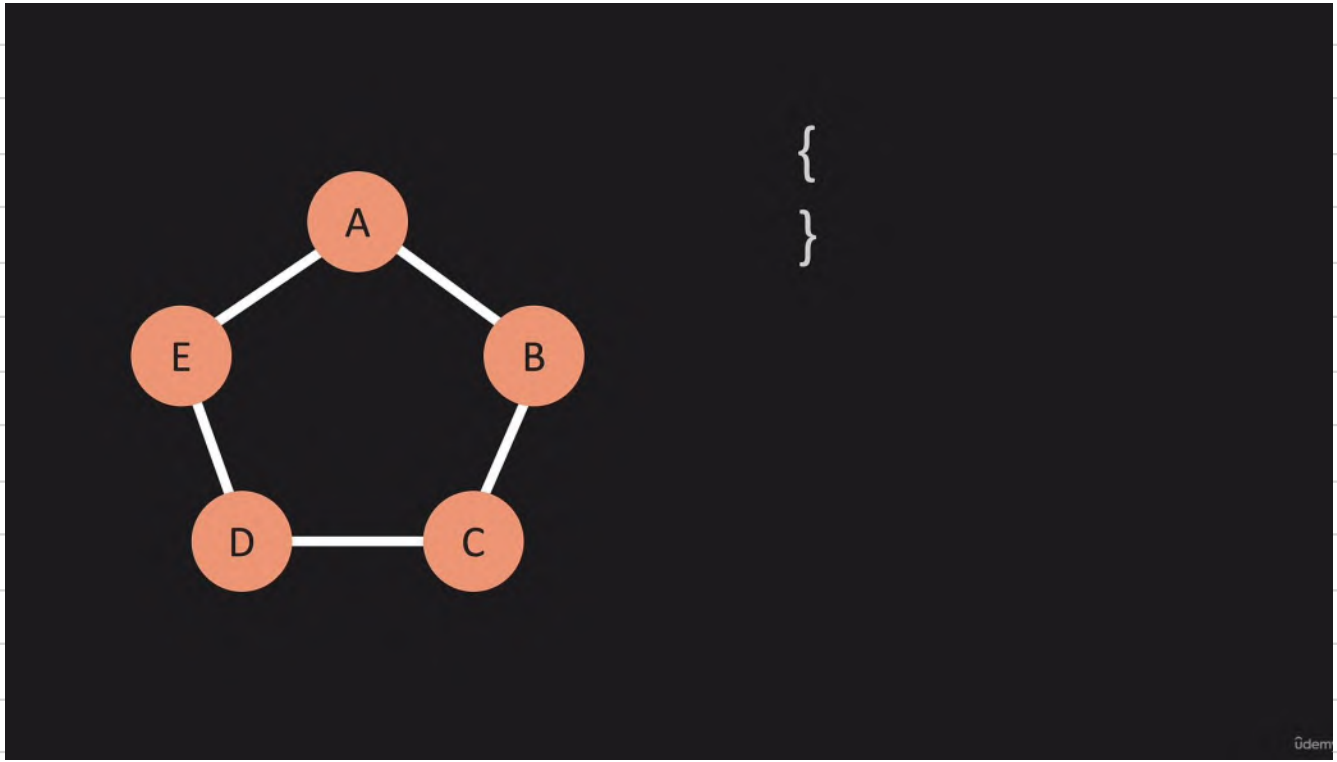
Jika graph yang dibuat bersifat terbobot maka angka 1 bisa diubah dengan angka bobot dari masing-masing Edge

Adjacency Matrix

- Selain menggunakan Adjacency Matrix, Graph juga dapat diwujudkan menggunakan Adjacency List.
- Adjacency List diimplementasikan menggunakan struktur Dictionary.
- Pada mata kuliah ini, kita akan menggunakan pendekatan Adjacency List dalam membangun suatu struktur Graph.

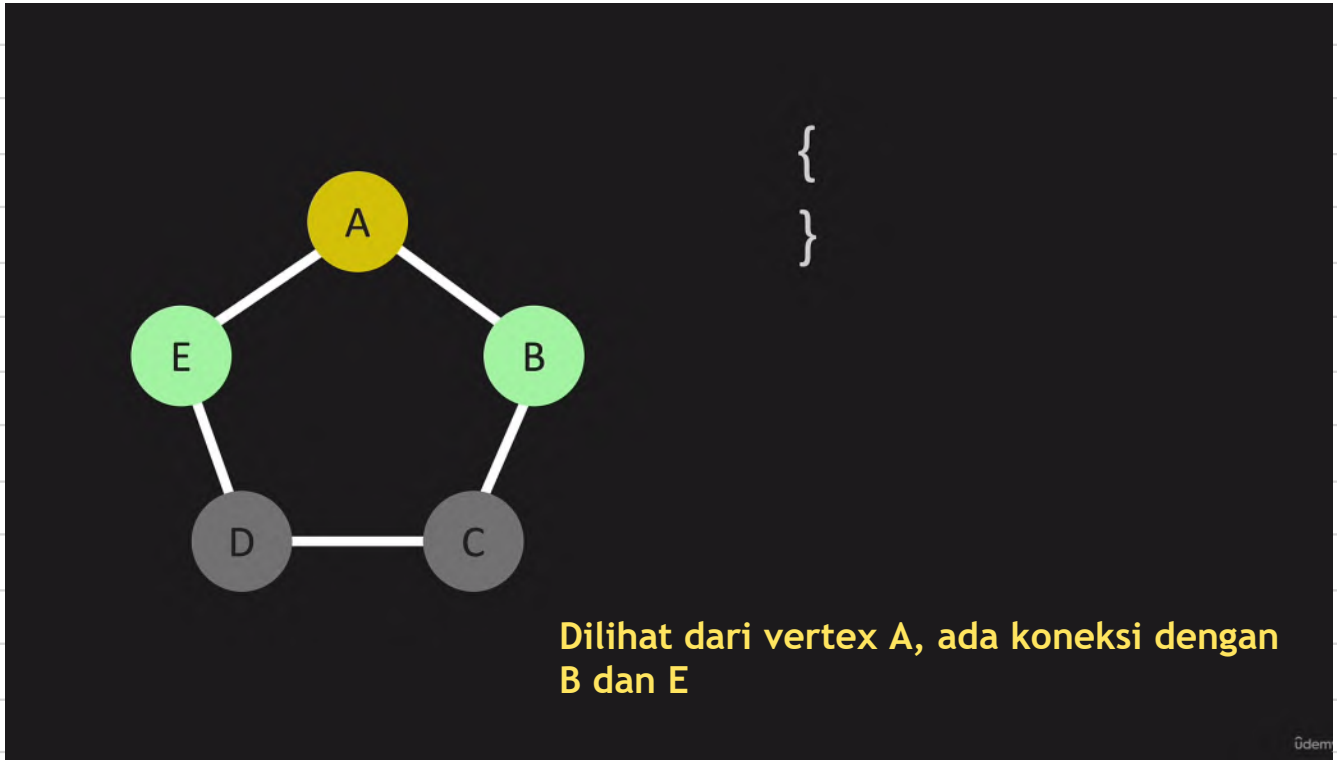
Adjacency List

Jika diketahui Graph sebagai berikut



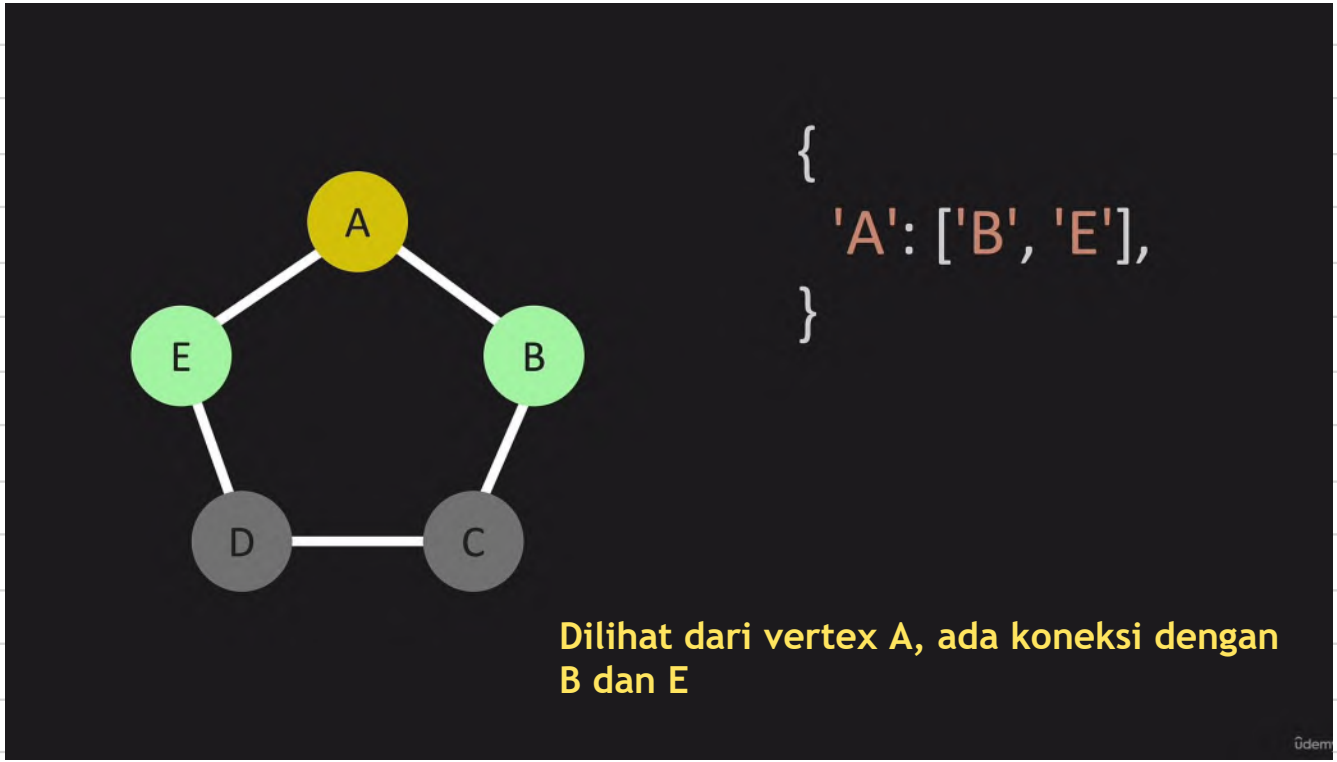
Adjacency List

Jika diketahui Graph sebagai berikut



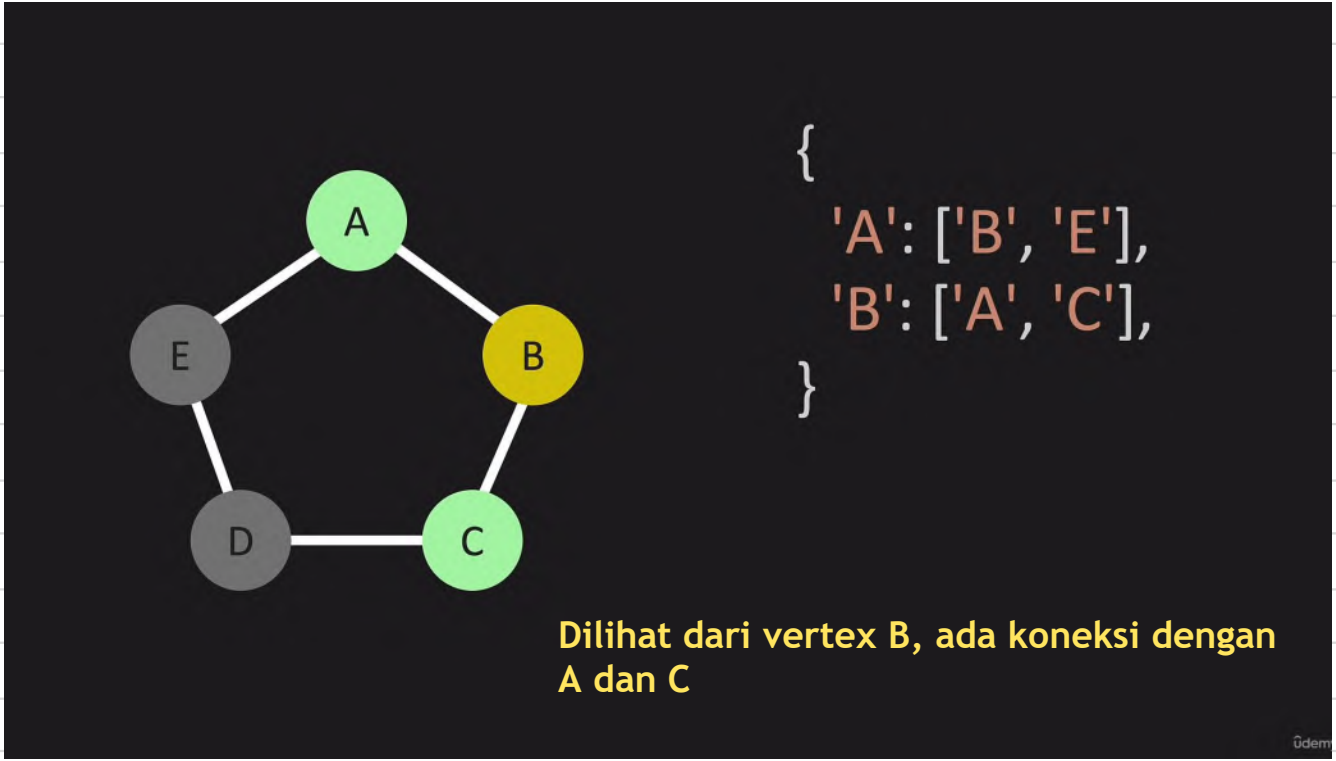
Adjacency List

Jika diketahui Graph sebagai berikut



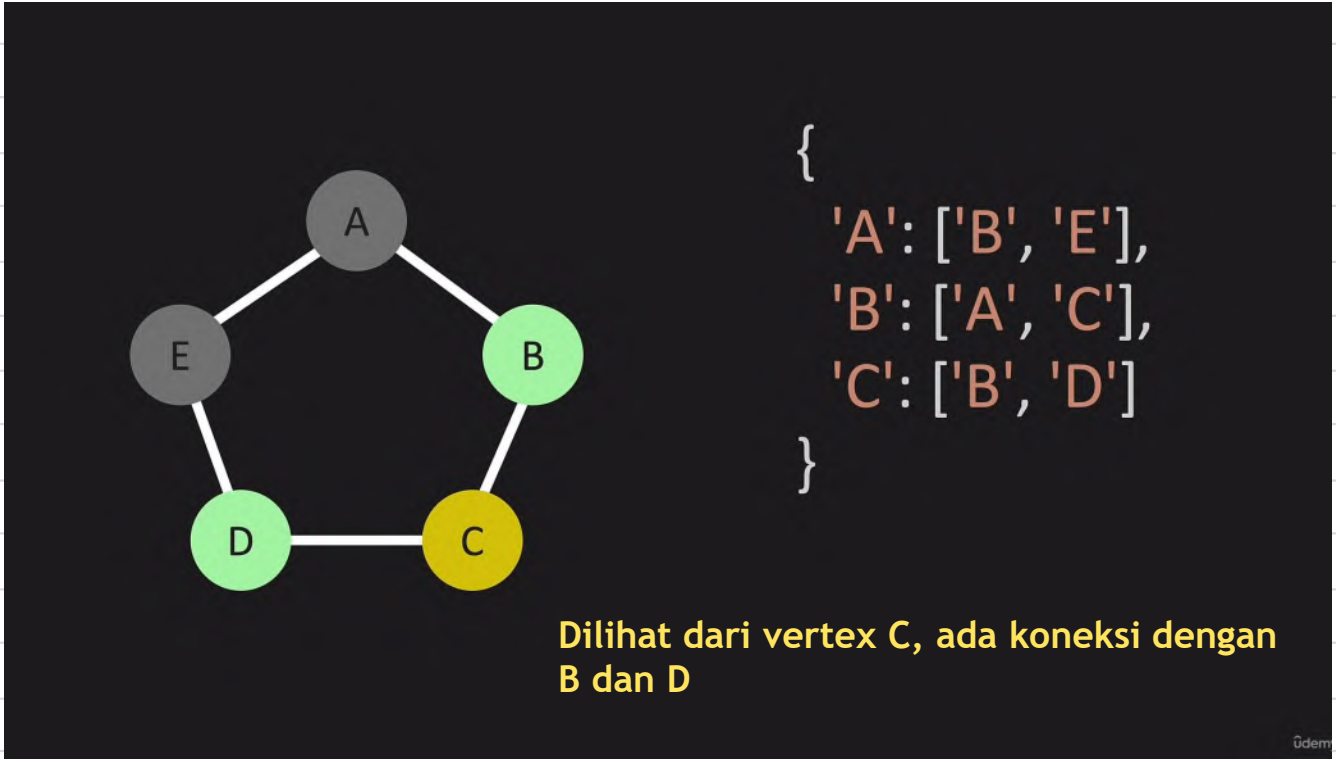
Adjacency List

Jika diketahui Graph sebagai berikut



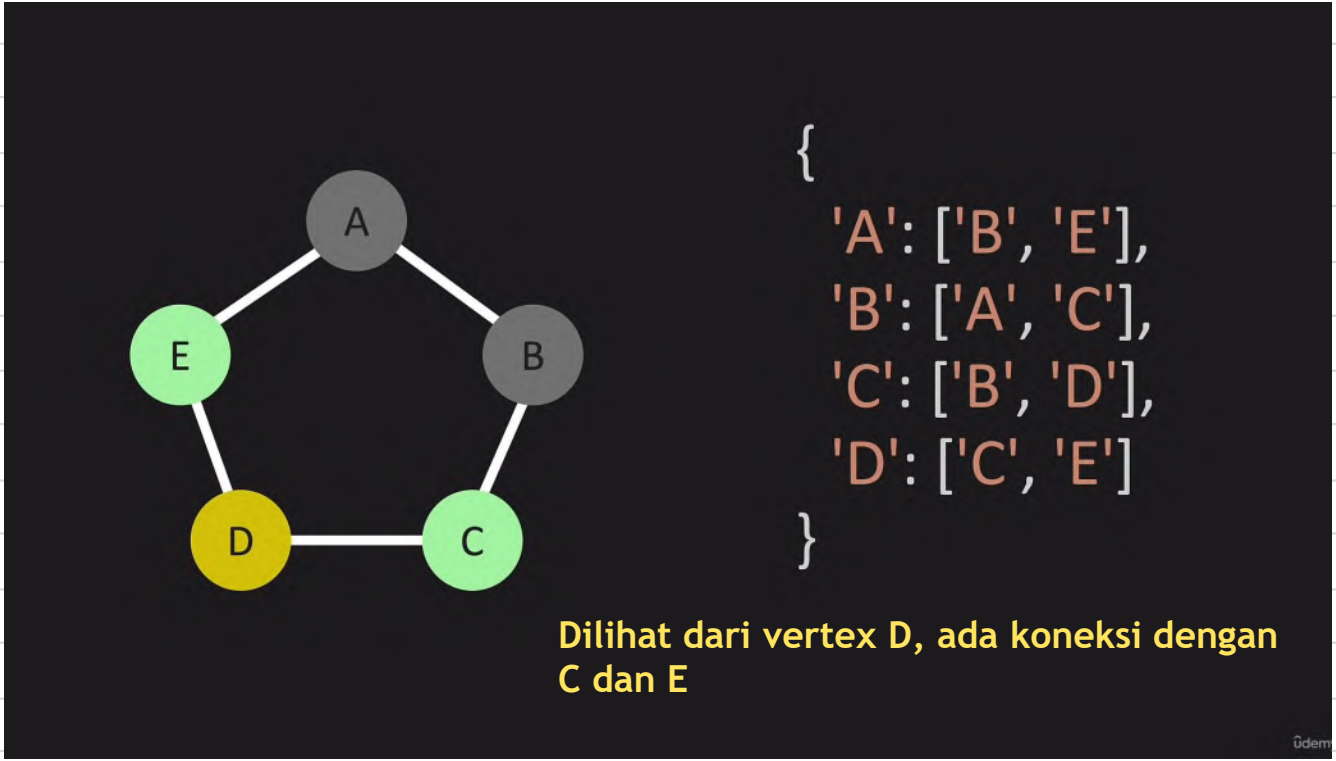
Adjacency List

Jika diketahui Graph sebagai berikut



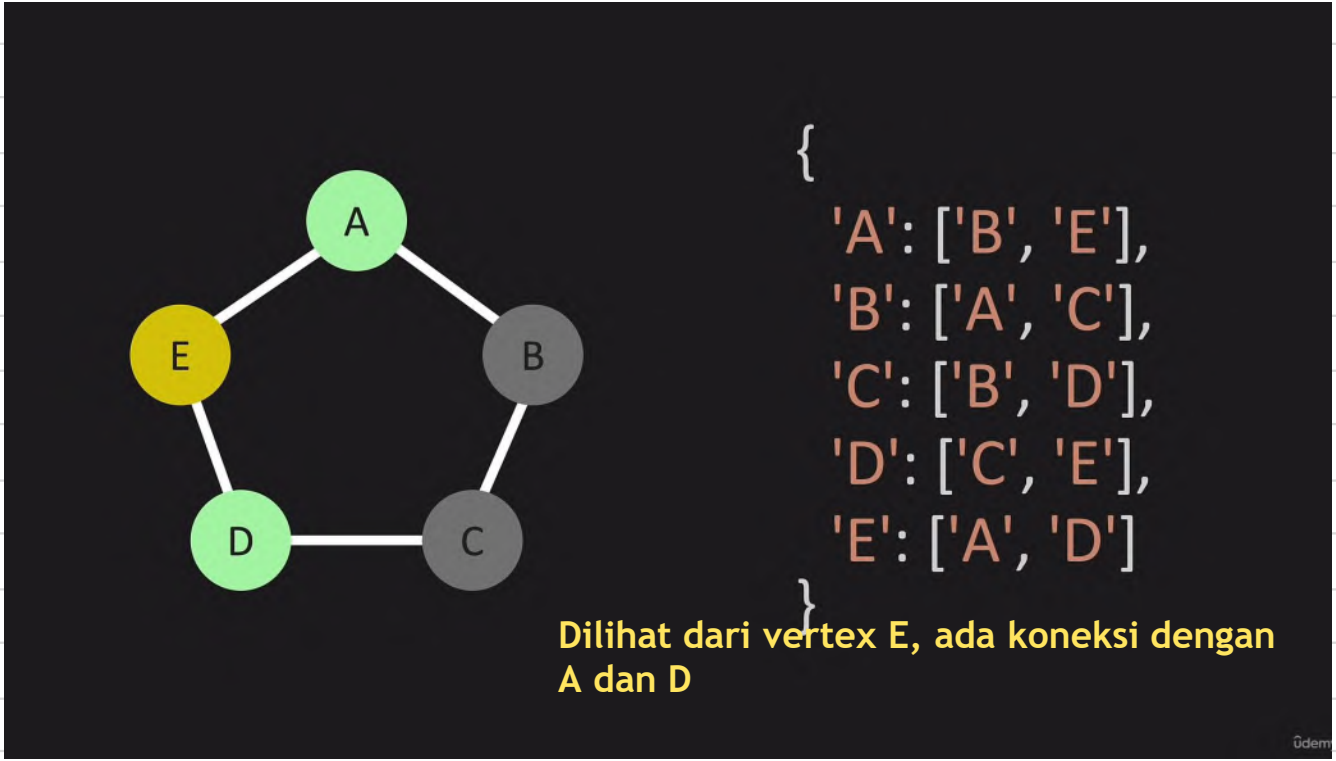
Adjacency List

Jika diketahui Graph sebagai berikut



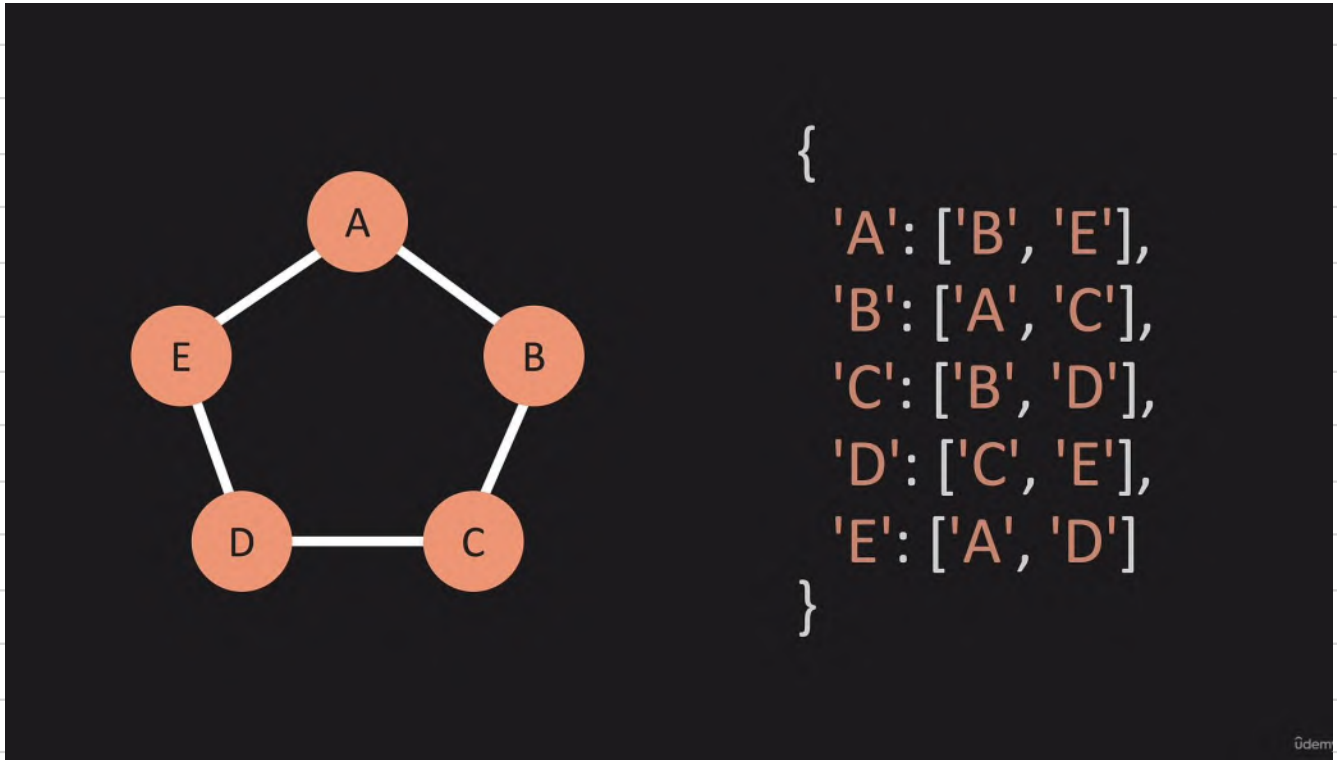
Adjacency List

Jika diketahui Graph sebagai berikut



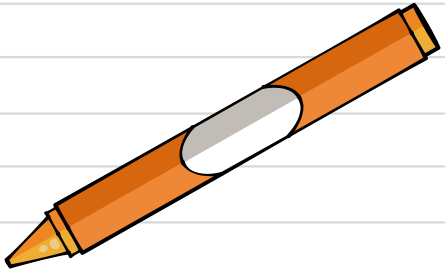
Adjacency List

Hasil akhir Adjacency List



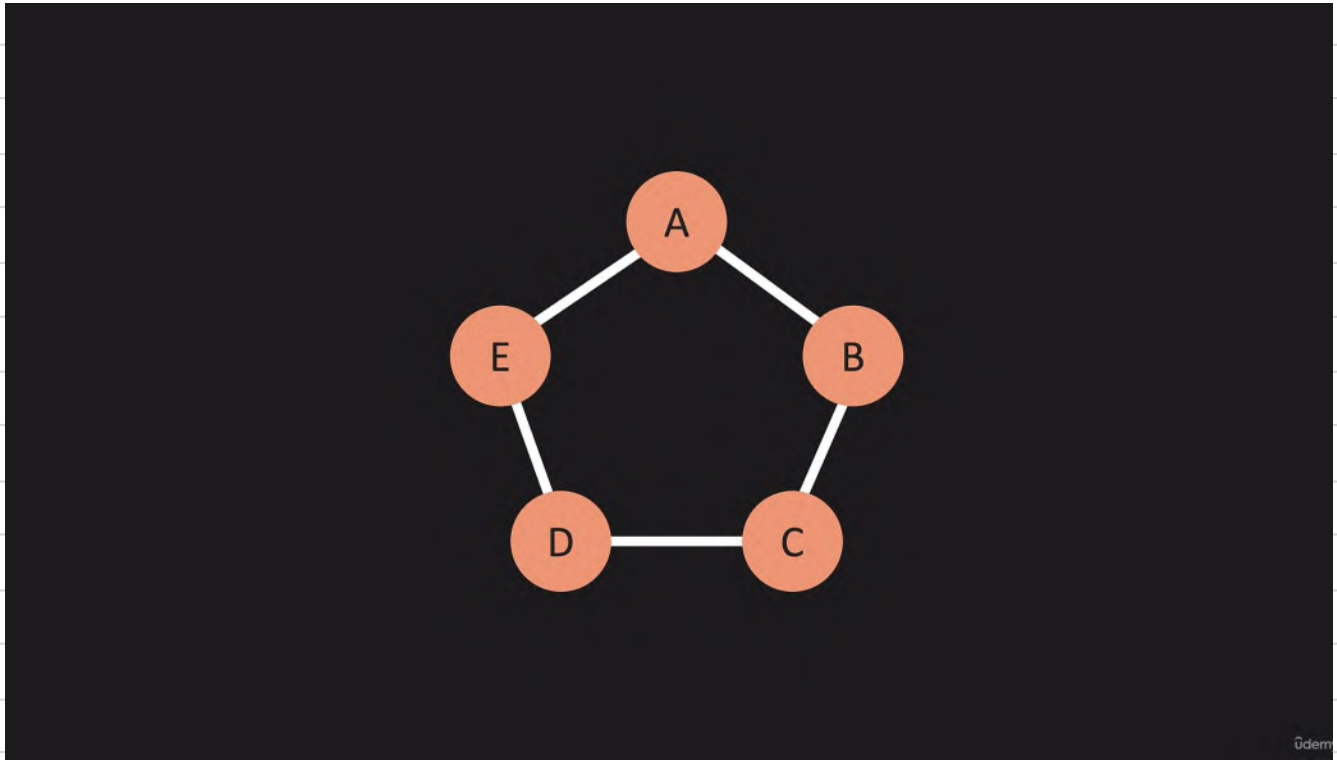
03

Graph
Big O



Graph Big O

Diketahui suatu graph sebagi berikut



Graph Big O

Adj matrix dan Adj list nya sebagai berikut

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

```
{
  'A': ['B', 'E'],
  'B': ['A', 'C'],
  'C': ['B', 'D'],
  'D': ['C', 'E'],
  'E': ['A', 'D']
}
```

Graph Big O

Adj matrix dan Adj list nya sebagai berikut

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

```
{
 'A': ['B', 'E'],
 'B': ['A', 'C'],
 'C': ['B', 'D'],
 'D': ['C', 'E'],
 'E': ['A', 'D']
}
```

Terlihat bahwa adj list hanya menyimpan info vertex yang terhubung

Graph Big O

Adj matrix dan Adj list nya sebagai berikut

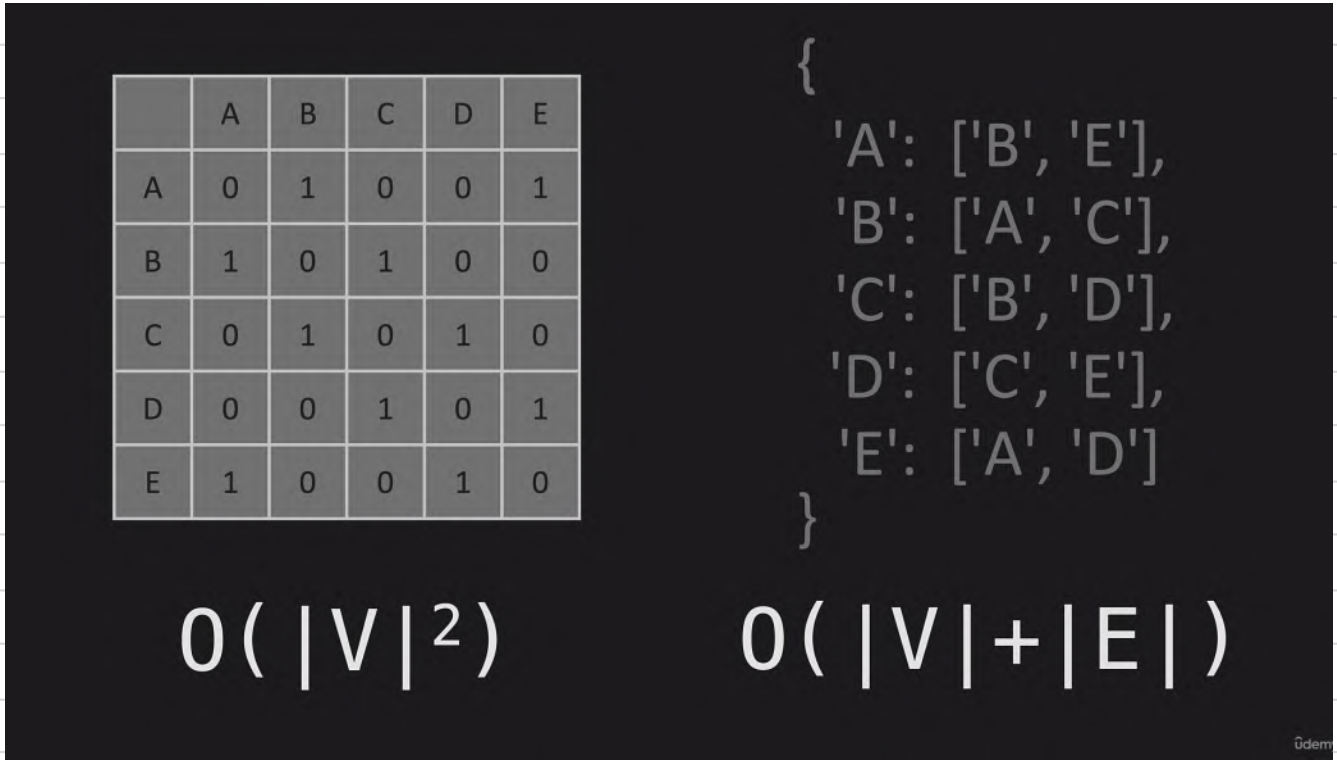
	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

```
{
 'A': ['B', 'E'],
 'B': ['A', 'C'],
 'C': ['B', 'D'],
 'D': ['C', 'E'],
 'E': ['A', 'D']
}
```

Sedangkah adj matrix meyimpan info vertex terhubung dan tidak terhubung

Graph Big O

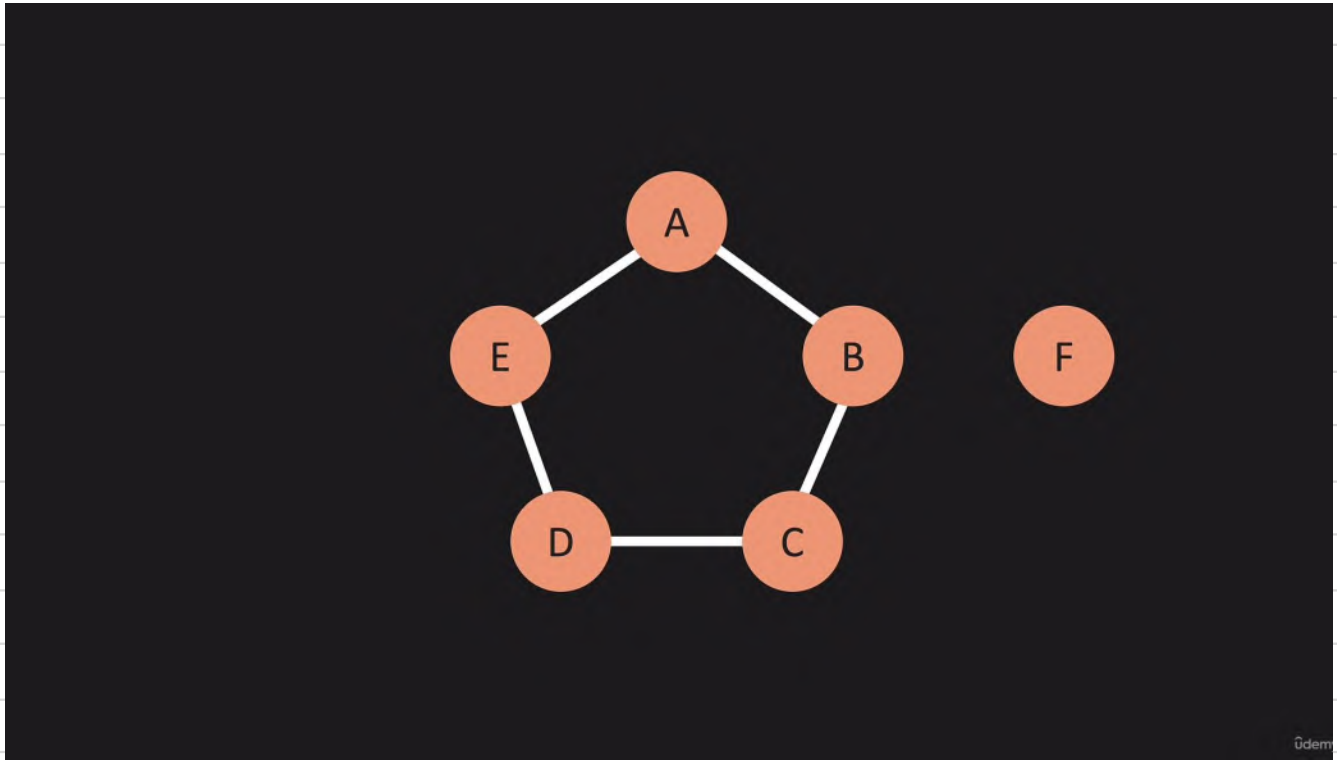
Adj matrix dan Adj list nya sebagai berikut



Dari perspektif ruang, adj list lebih boros karena memerlukan ruang sebanyak vertex kuadrat

Graph Big O:
Operasi Add Vertex

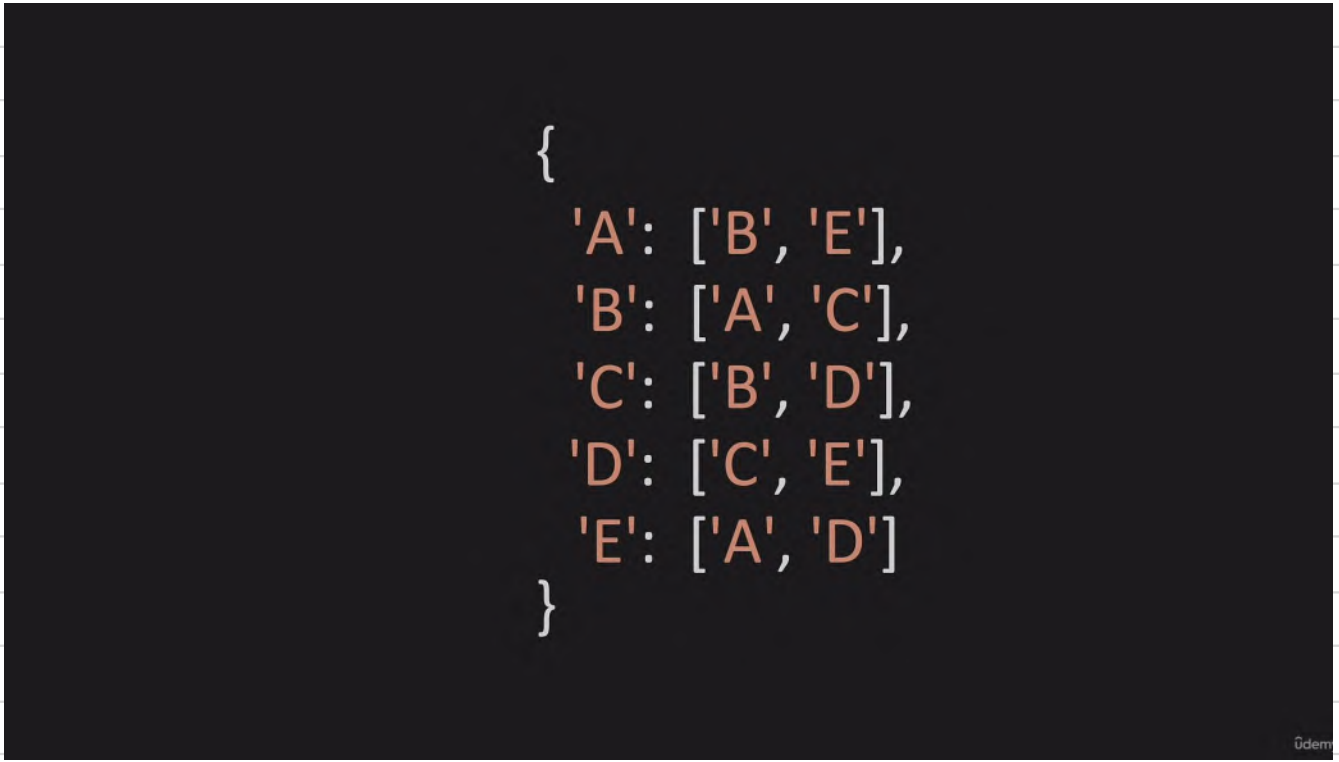
Diketahui suatu graph dan kita ingin
Membuat vertex baru F



Vertex F hanya ditambahkan dan tidak dihubungkan ke vertex lainnya

Graph Big O:
Operasi Add Vertex

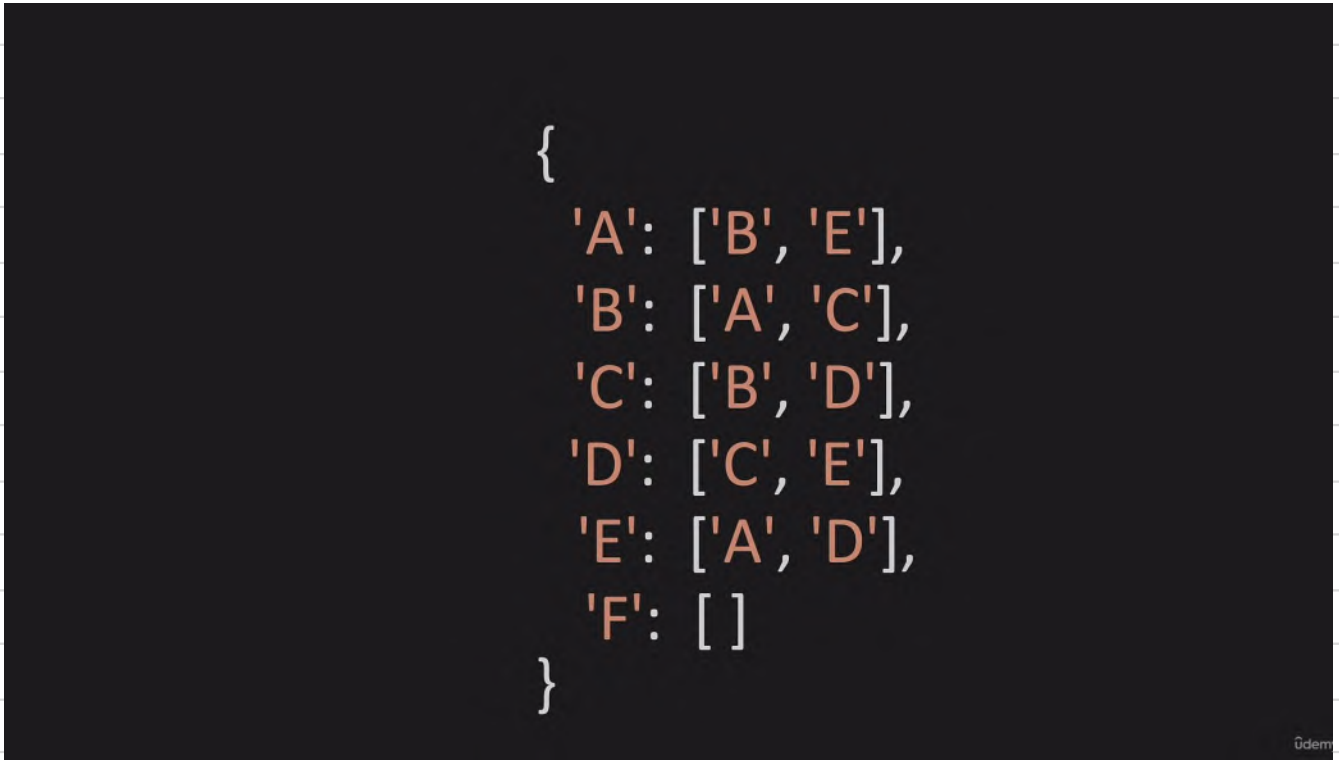
Diketahui suatu graph dan kita ingin
Membuat vertex baru F



Jika menggunakan adj list kita hanya perlu menambahkan F di paling bawah

Graph Big O:
Operasi Add Vertex

Diketahui suatu graph dan kita ingin
Membuat vertex baru F



Jika menggunakan adj list kita hanya perlu menambahkan F di paling bawah

Graph Big O:
Operasi Add Vertex

Diketahui suatu graph dan kita ingin
Membuat vertex baru F

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Jika menggunakan adj matrix kita perlu menulis ulang seluruh matrix karena mengubah ukurannya dari 5x5 menjadi 6x6

Graph Big O:
Operasi Add Vertex

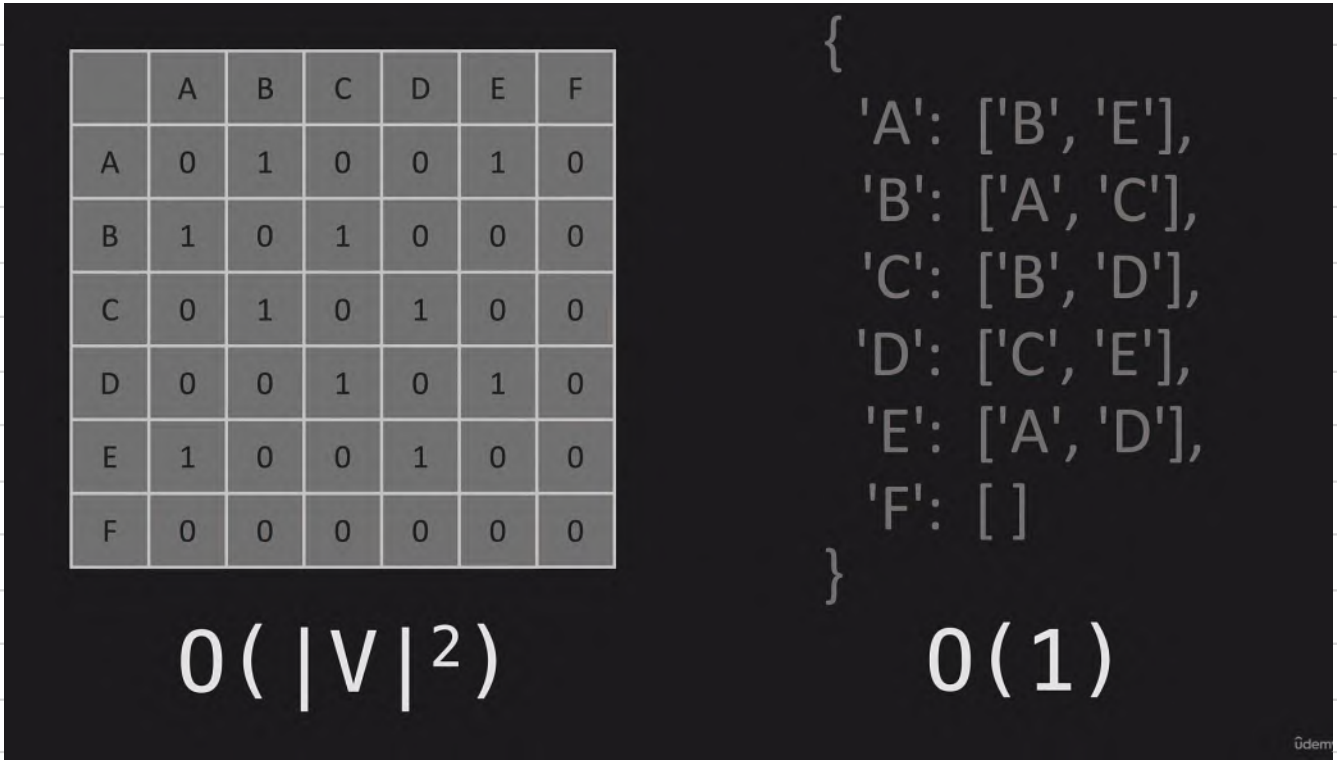
Diketahui suatu graph dan kita ingin
Membuat vertex baru F

	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	1	0	0	0
C	0	1	0	1	0	0
D	0	0	1	0	1	0
E	1	0	0	1	0	0
F	0	0	0	0	0	0

Jika menggunakan adj matrix kita perlu menulis ulang seluruh matrix karena mengubah ukurannya dari 5x5 menjadi 6x6

Graph Big O:
Operasi Add Vertex

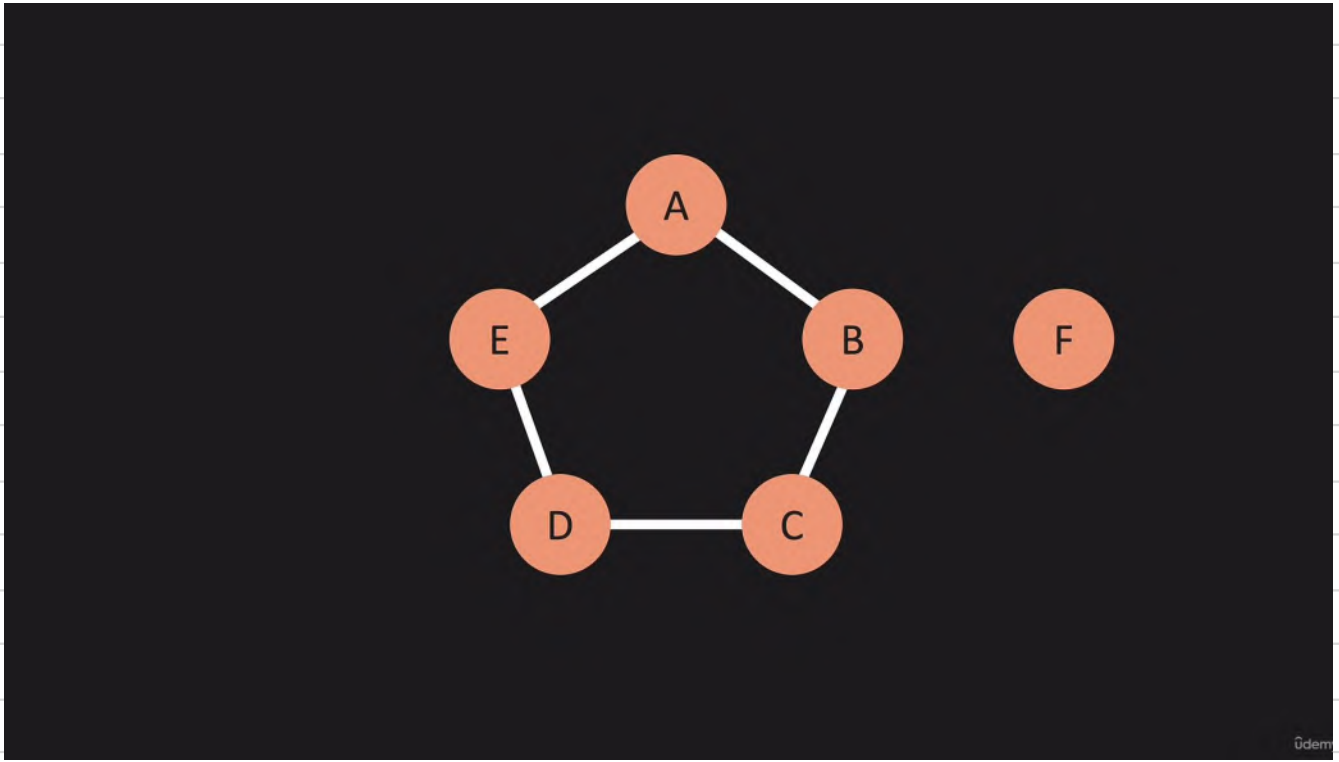
Diketahui suatu graph dan kita ingin
Membuat vertex baru F



Untuk operasi Add Vertex penggunaan Adj List lebih efisien daripada adj Matrix

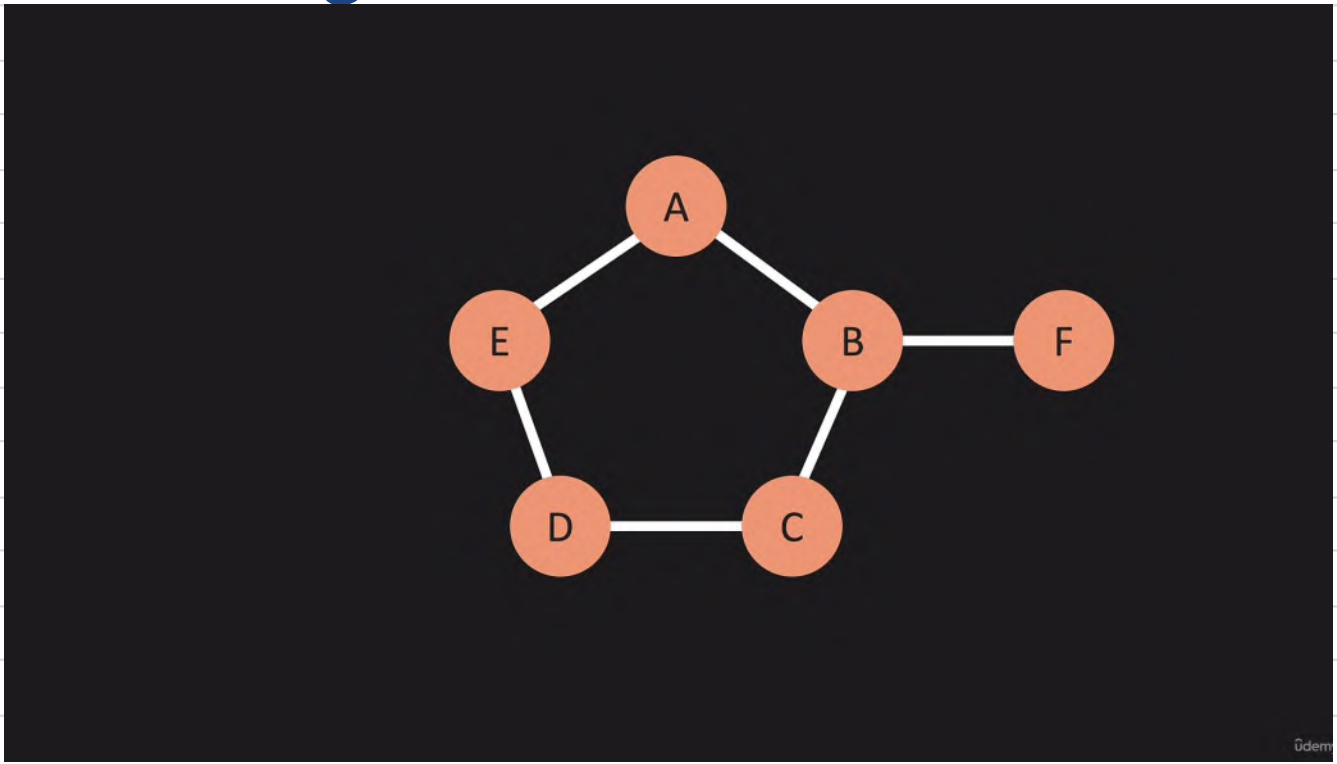
Graph Big O:
Operasi Add Edge

Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



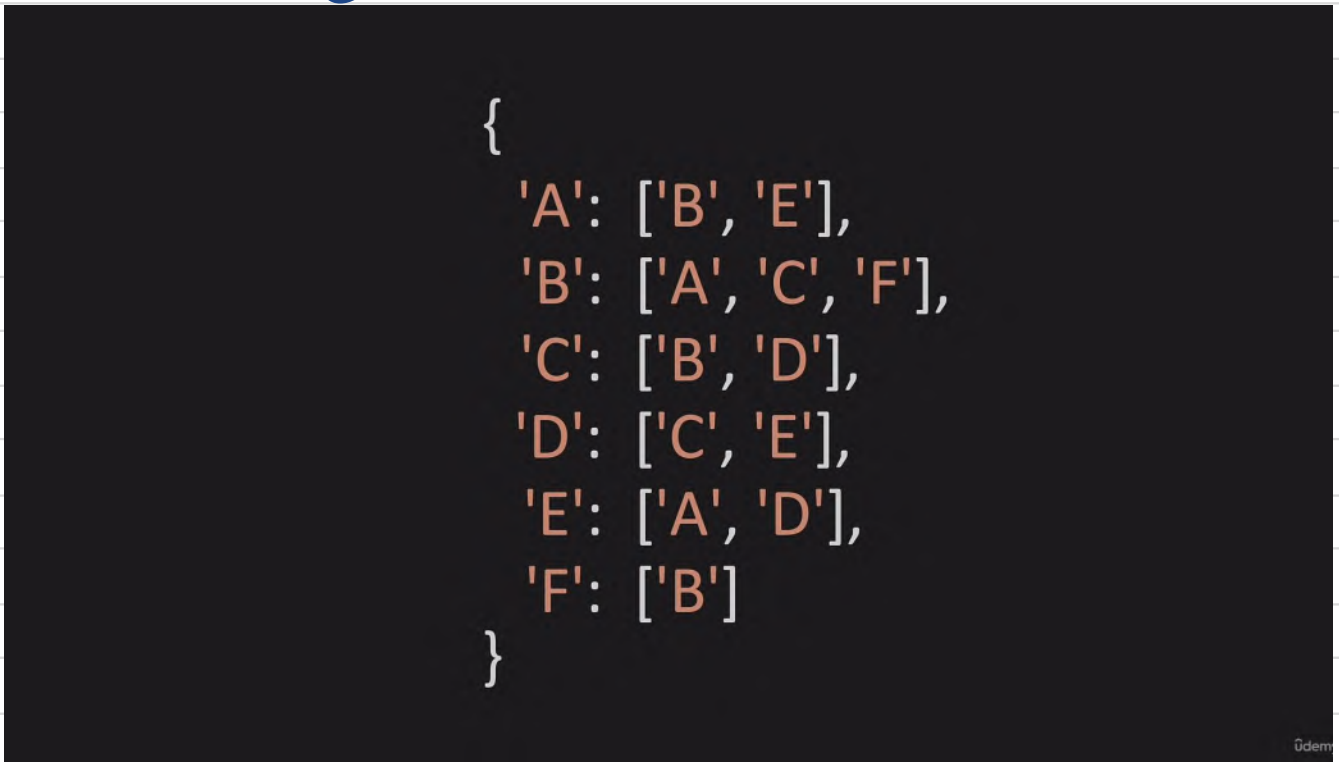
Graph Big O:
Operasi Add Edge

Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Graph Big O:
Operasi Add Edge

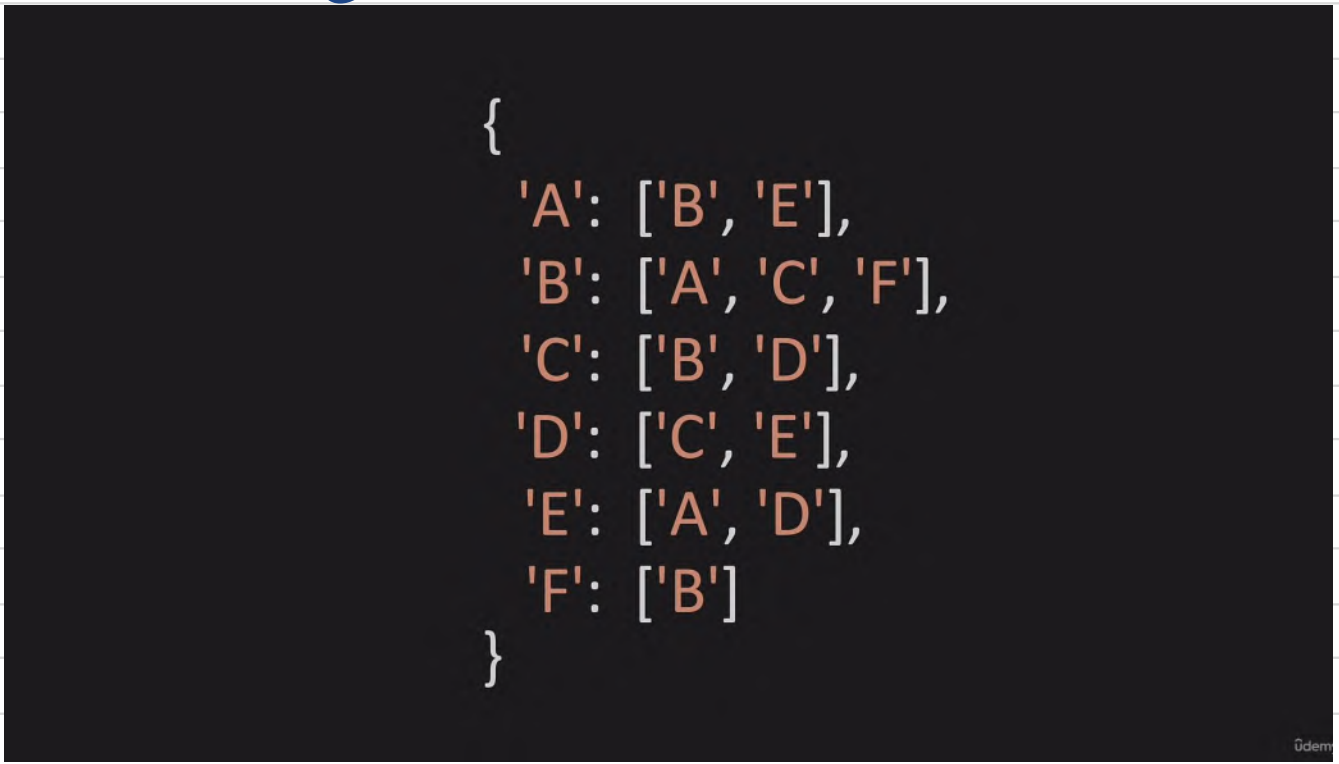
Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Jika menggunakan adj list kita hanya perlu menambahkan B pada List milik vertex F di paling bawah

Graph Big O:
Operasi Add Edge

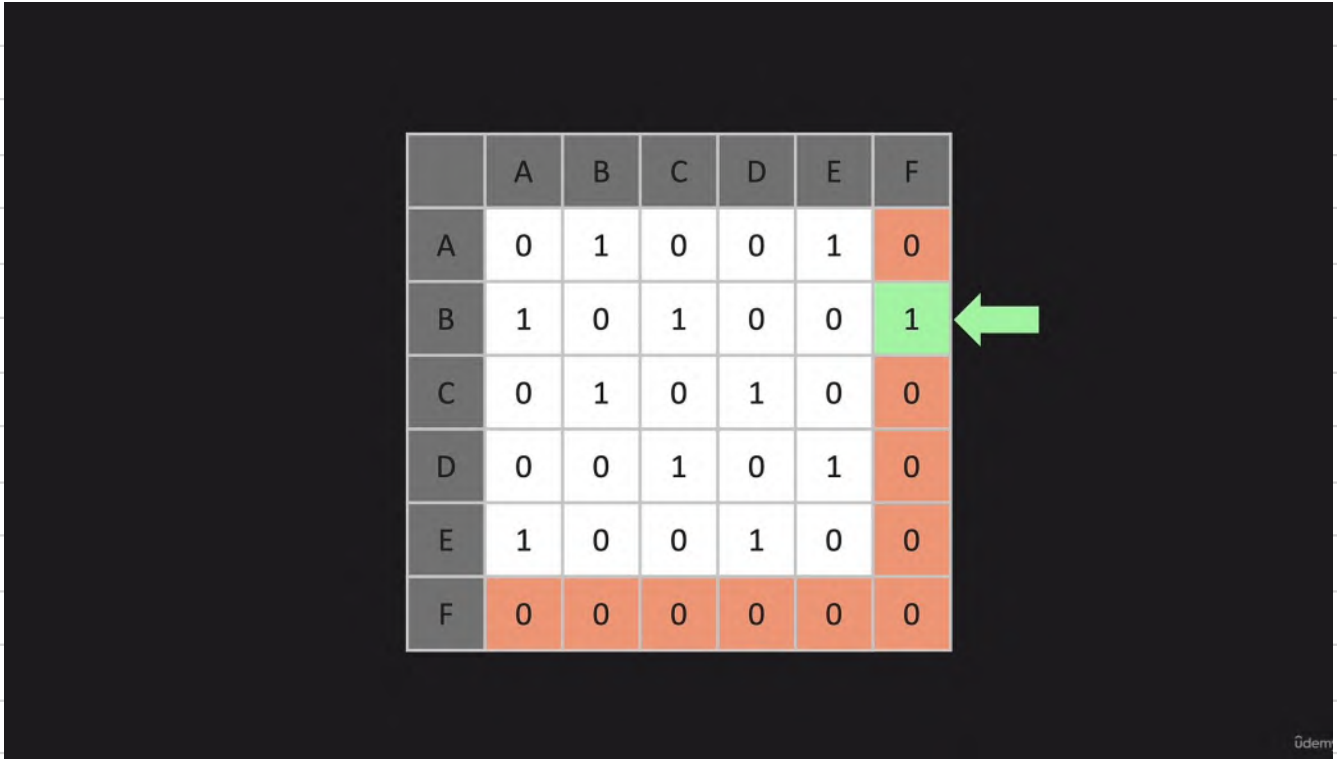
Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Jika menggunakan adj list kita hanya perlu menambahkan B pada List milik vertex F di paling bawah

Graph Big O:
Operasi Add Edge

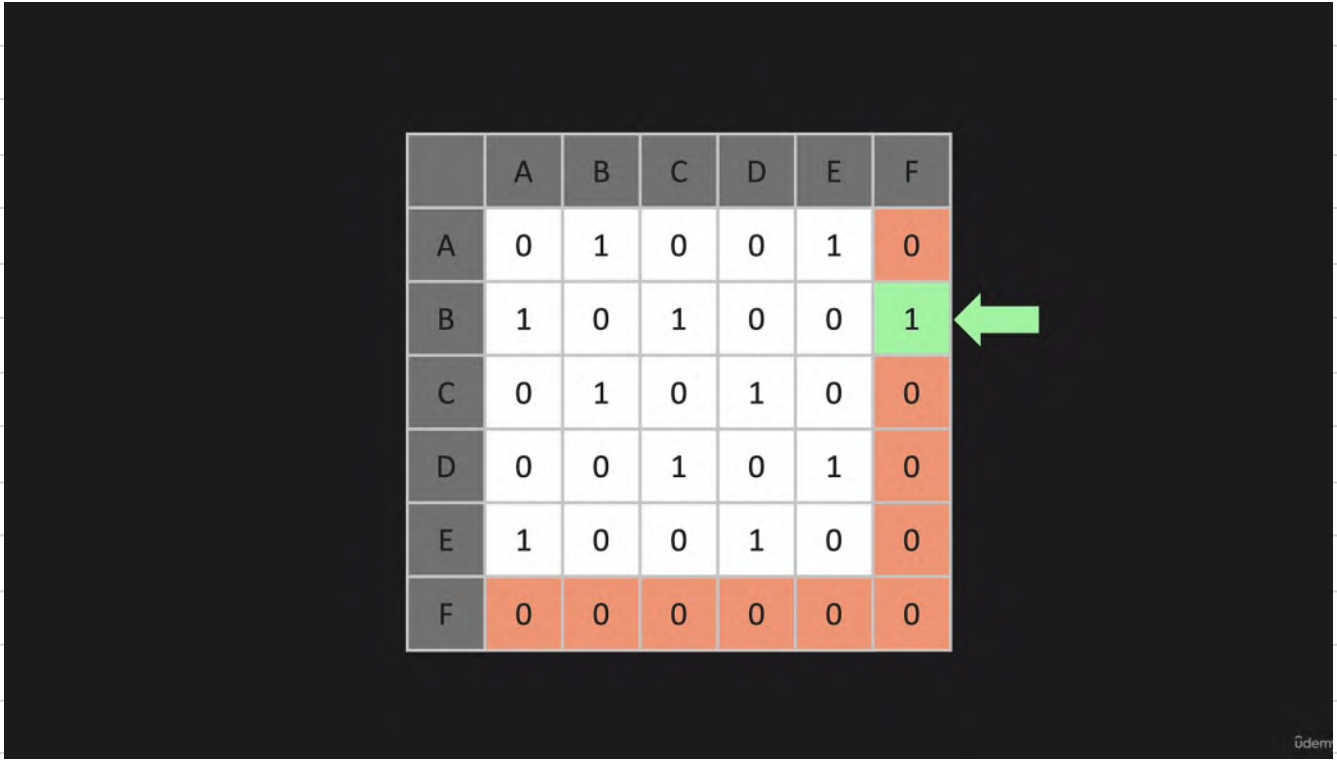
Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Jika menggunakan adj matrix kita hanya perlu mengubah nilai 0 menjadi 1 pada kolom dan baris B & F

Graph Big O:
Operasi Add Edge

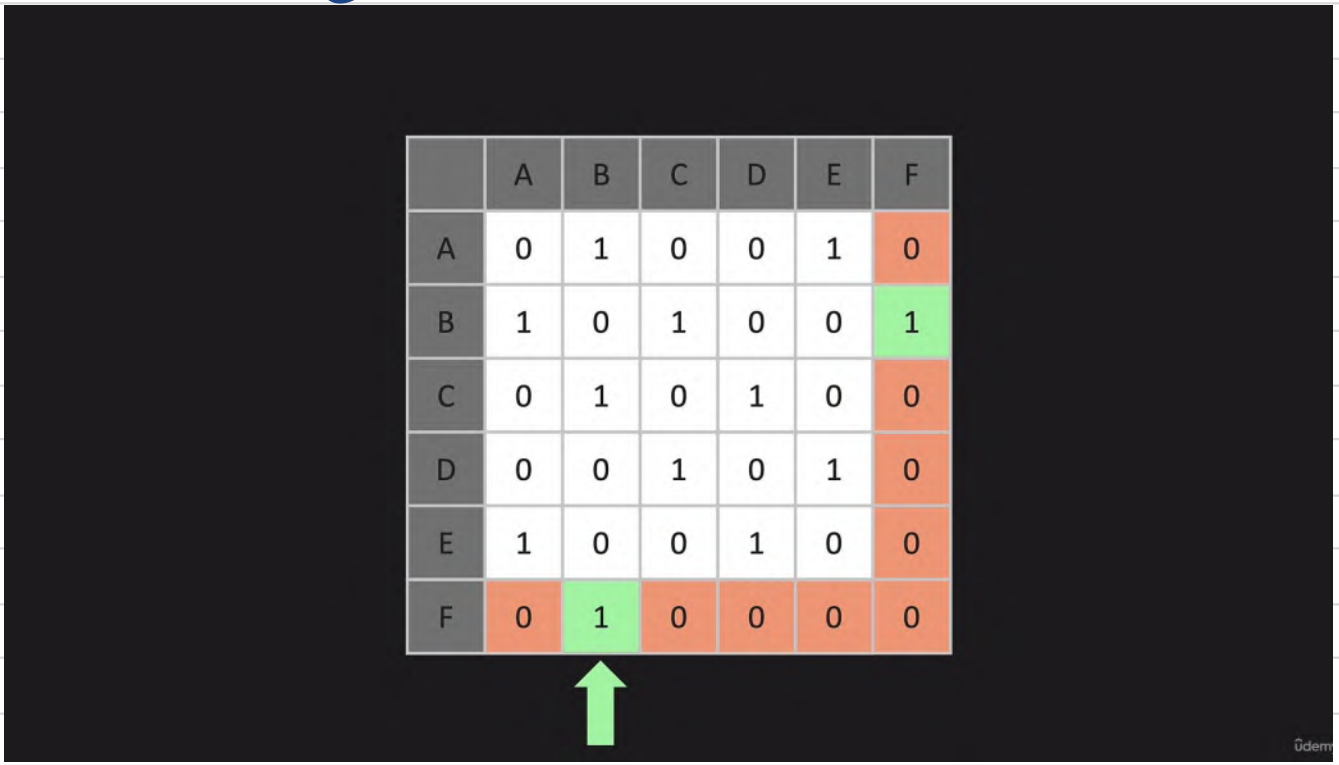
Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Jika menggunakan adj matrix kita hanya perlu mengubah nilai 0 menjadi 1 pada kolom dan baris B & F

Graph Big O:
Operasi Add Edge

Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B

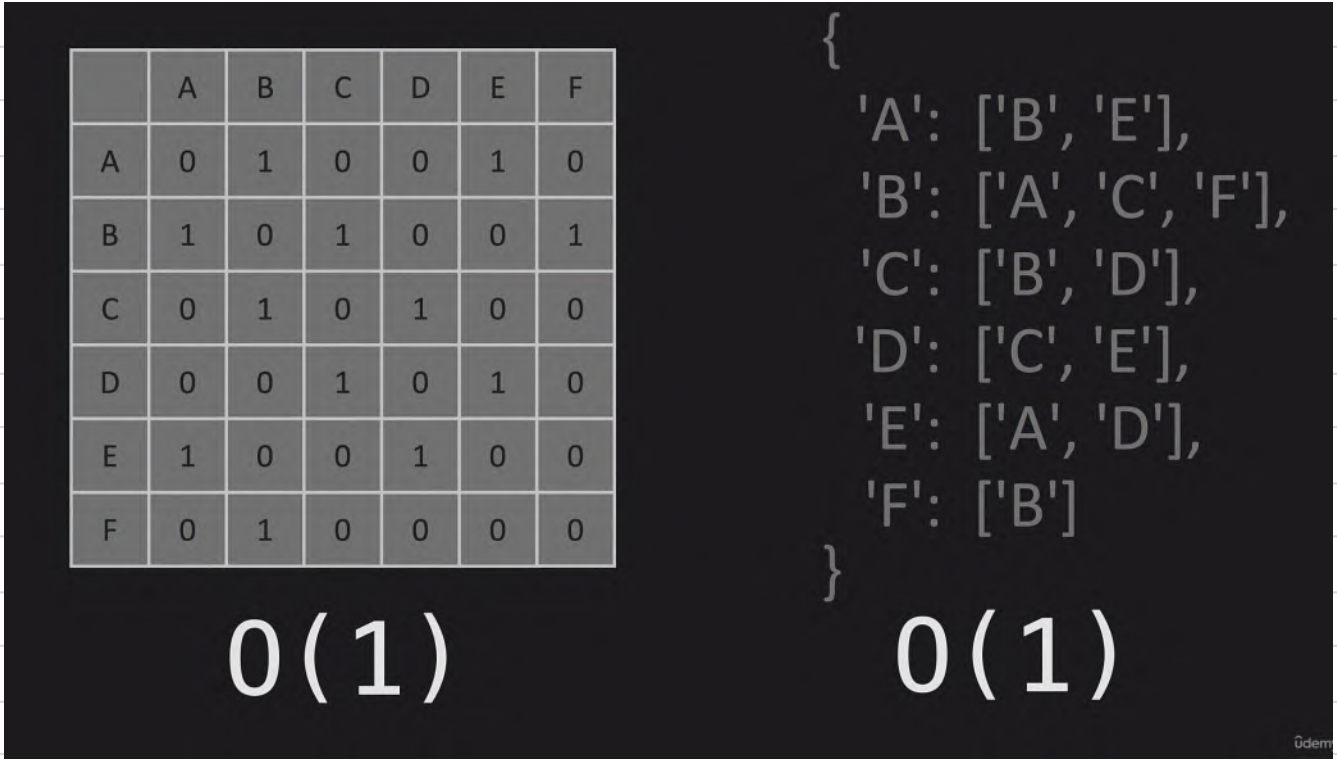


	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	1	0	0	1
C	0	1	0	1	0	0
D	0	0	1	0	1	0
E	1	0	0	1	0	0
F	0	1	0	0	0	0

Jika menggunakan adj matrix kita hanya perlu mengubah nilai 0 menjadi 1 pada kolom dan baris B & F

Graph Big O:
Operasi Add Edge

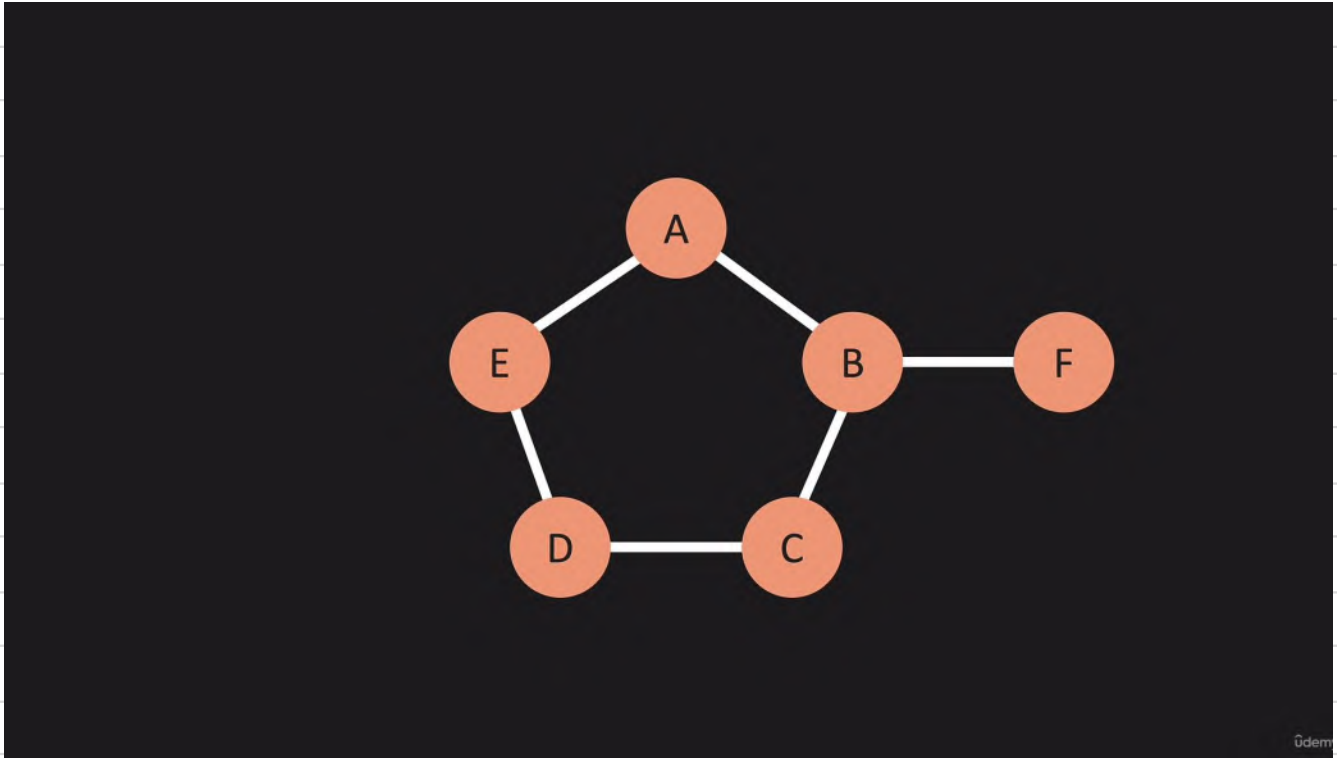
Diketahui suatu graph dan kita ingin
Menyambung vertex baru F ke B



Untuk operasi Add Edge keduanya memiliki efisiensi yang sama

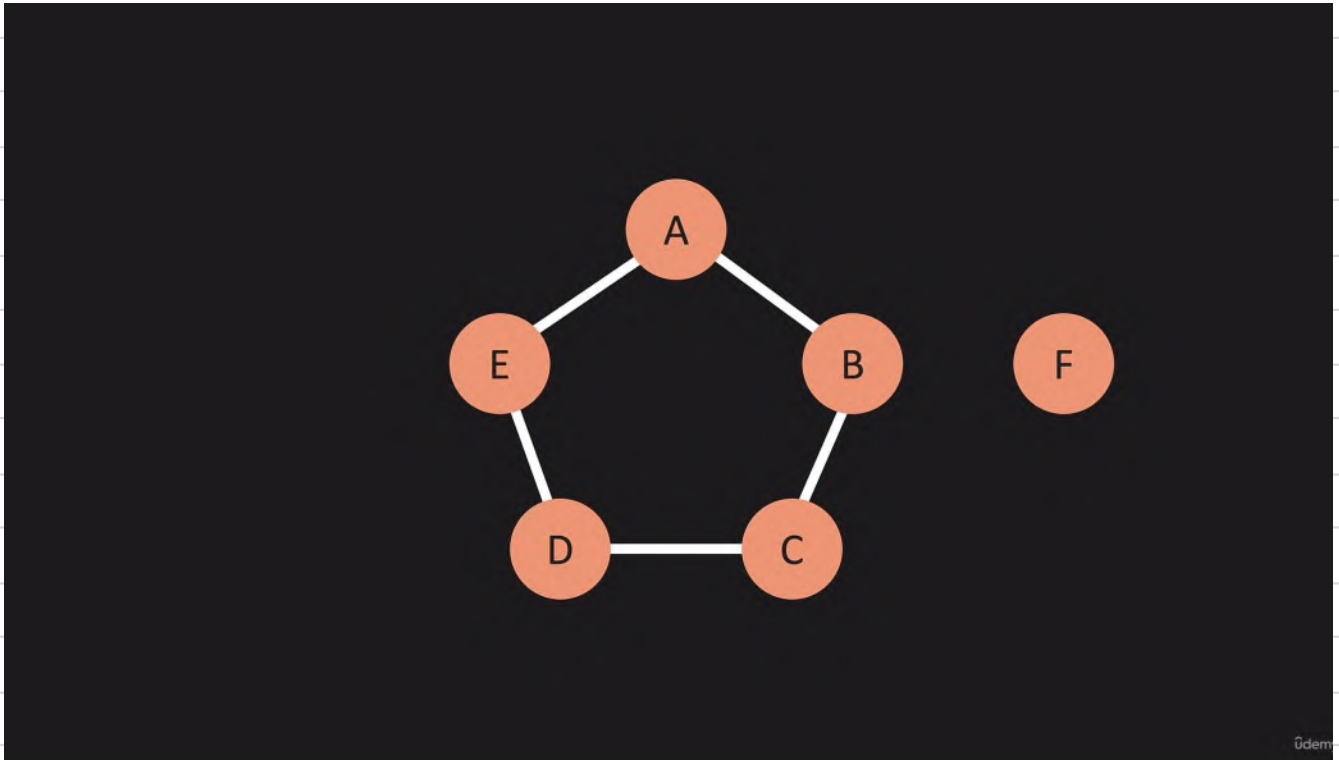
Graph Big O:
Operasi Remove Edge

Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



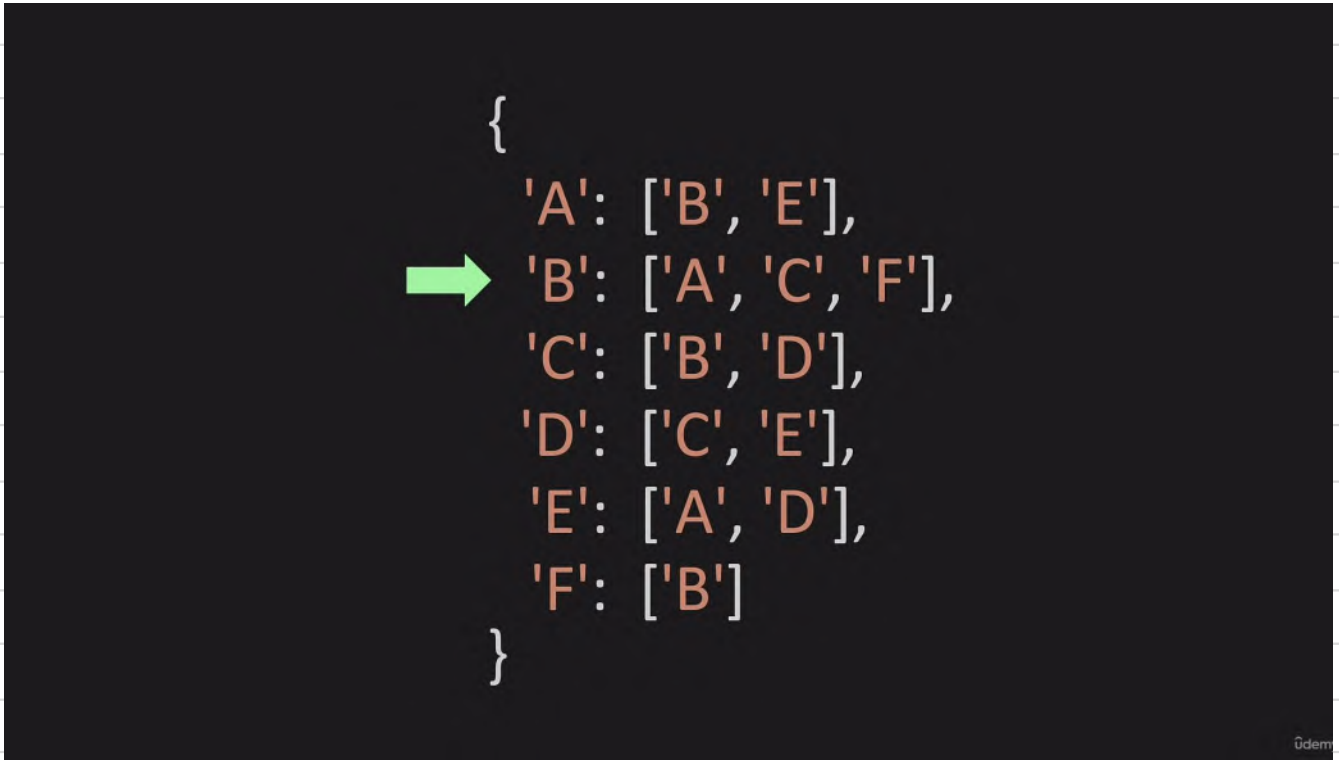
Graph Big O:
Operasi Remove Edge

Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Graph Big O:
Operasi Remove Edge

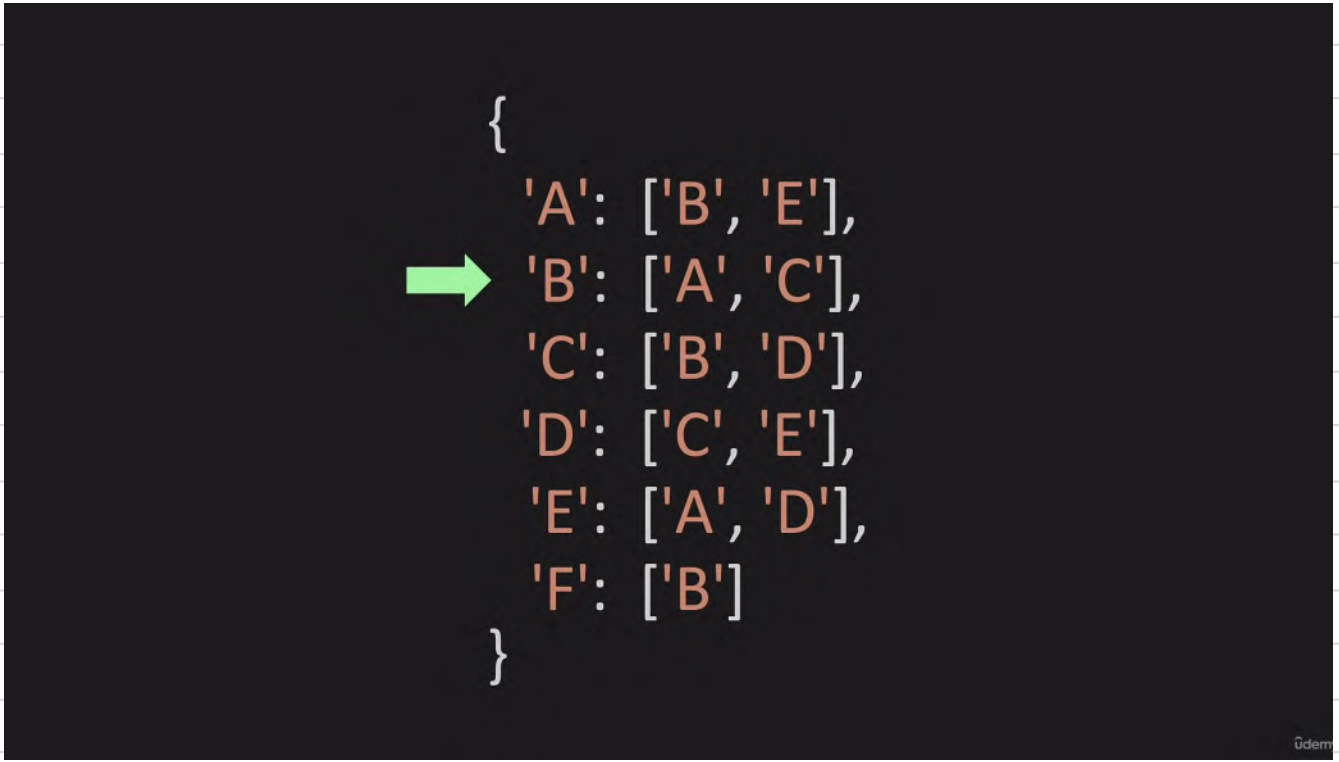
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Jika menggunakan adj list maka kita akan mengunjungi list vertex B dan melakukan penelusuran untuk menemukan vertex F dan menghapusnya

Graph Big O:
Operasi Remove Edge

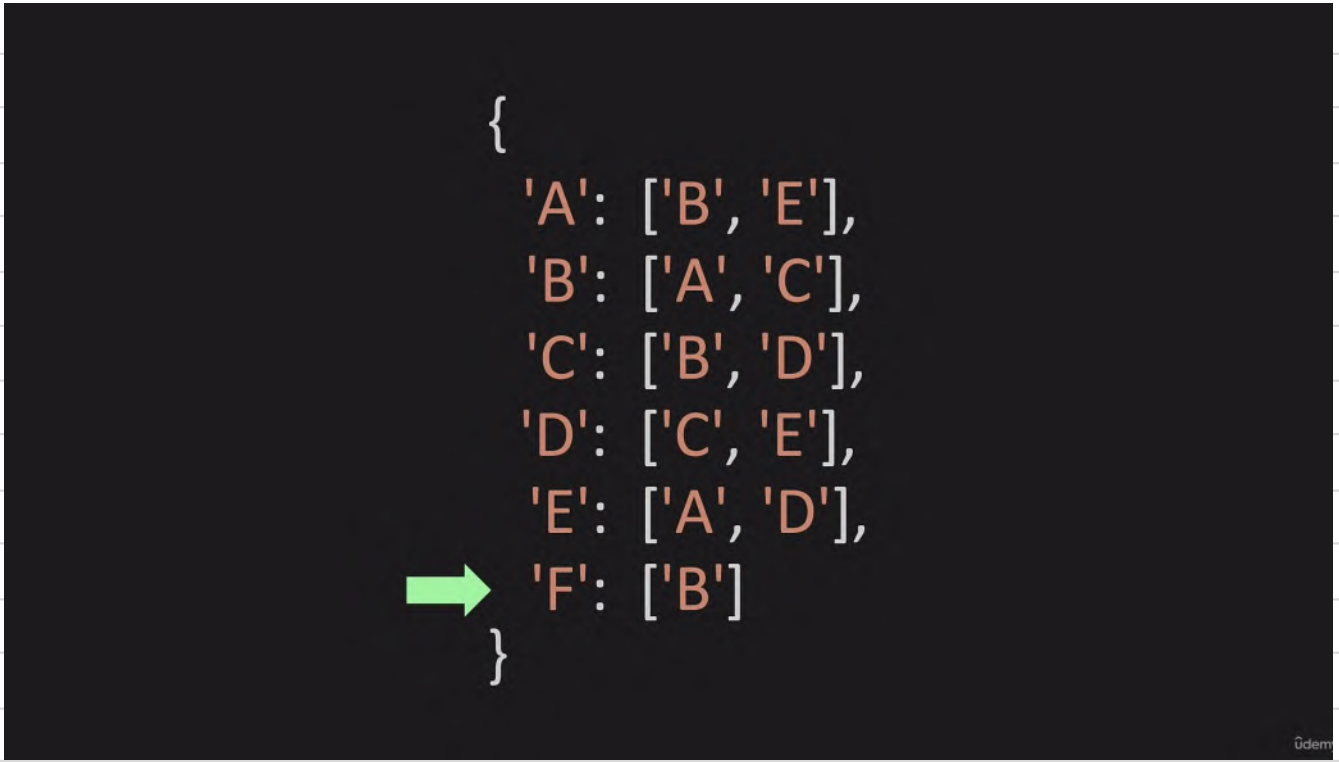
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Jika menggunakan adj list maka kita akan mengunjungi list vertex B dan melakukan penelusuran untuk menemukan vertex F dan menghapusnya

Graph Big O:
Operasi Remove Edge

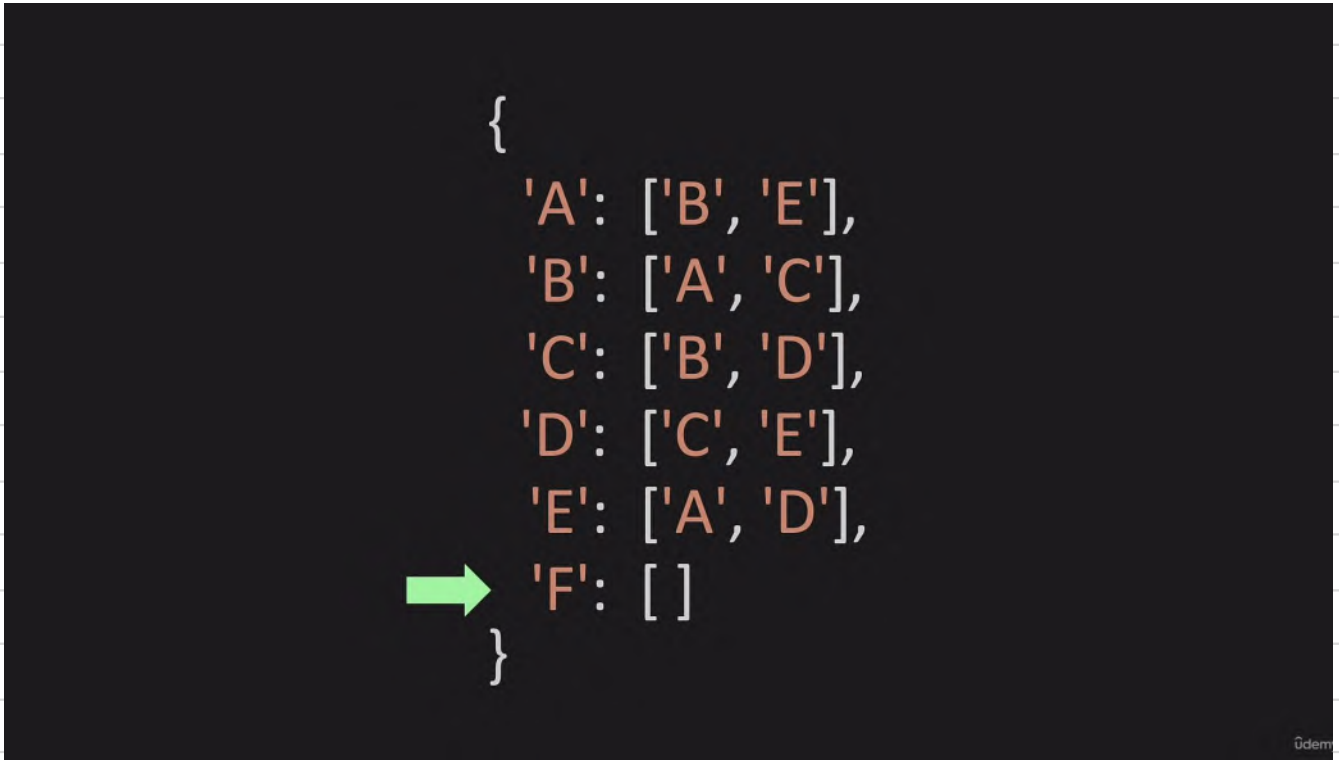
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Hal yang sama dilakukan pada list F yaitu mengunjungi list vertex F dan melakukan penelusuran untuk menemukan vertex B dan menghapusnya

Graph Big O:
Operasi Remove Edge

Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Hal yang sama dilakukan pada list F yaitu mengunjungi list vertex F dan melakukan penelusuran untuk menemukan vertex B dan menghapusnya

Graph Big O:
Operasi Remove Edge

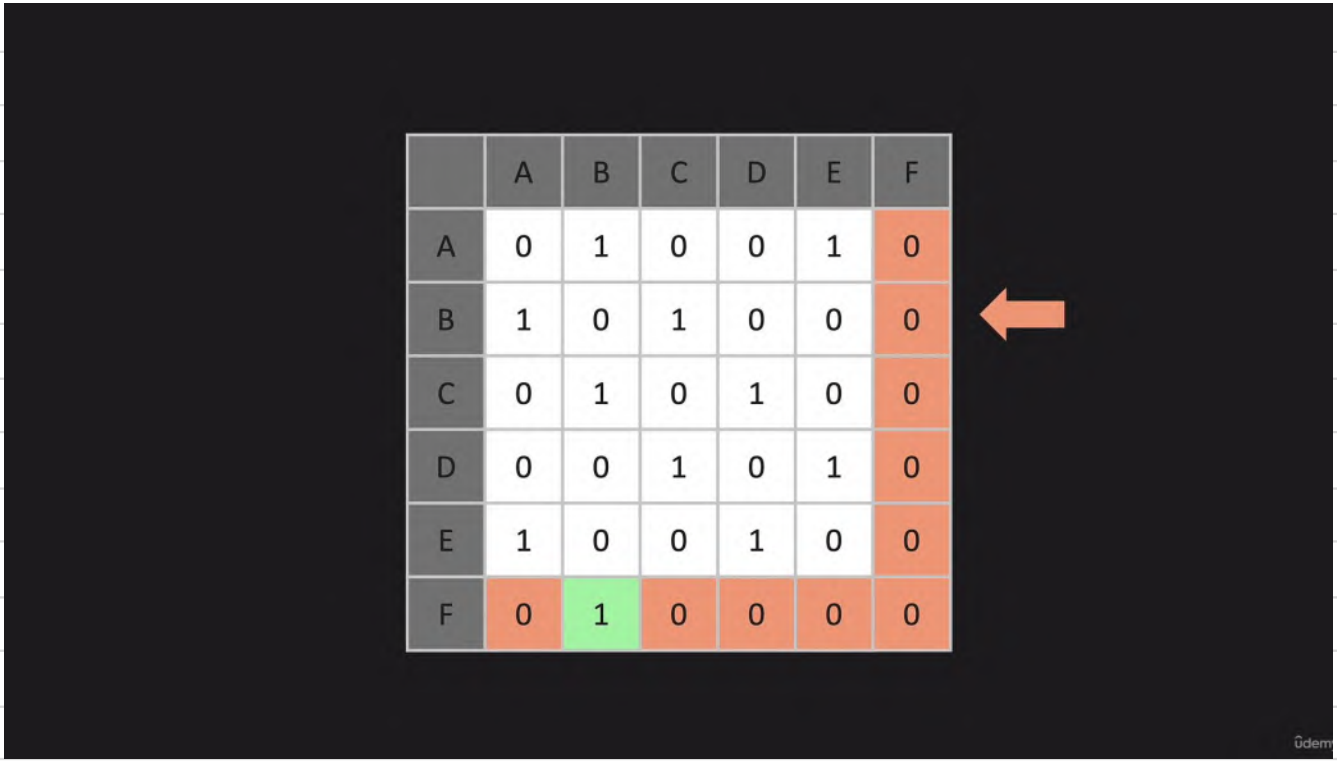
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B

	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	1	0	0	1
C	0	1	0	1	0	0
D	0	0	1	0	1	0
E	1	0	0	1	0	0
F	0	1	0	0	0	0

Jika menggunakan adj matrix kita hanya perlu mengubah nilai 1 menjadi 0 pada kolom dan baris B & F

Graph Big O:
Operasi Remove Edge

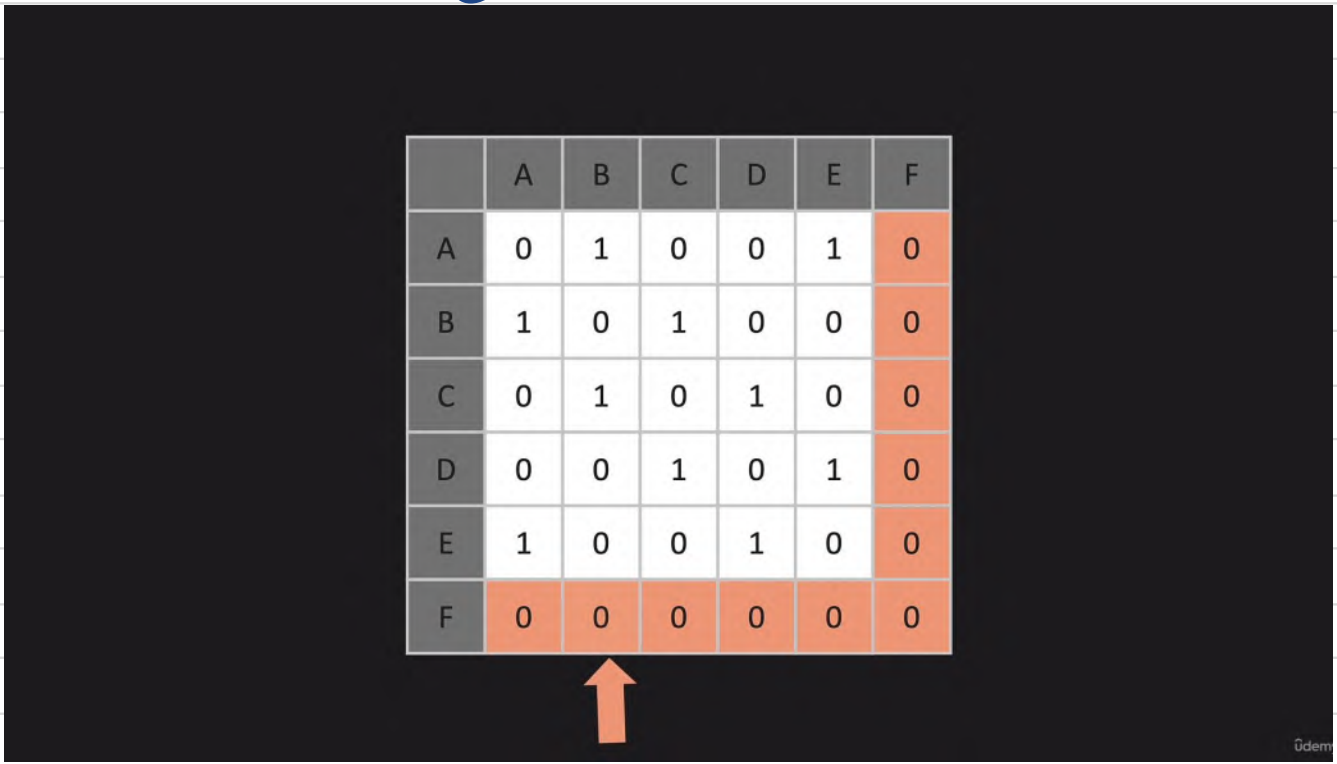
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Jika menggunakan adj matrix kita hanya perlu mengubah nilai 1 menjadi 0 pada kolom dan baris B & F

Graph Big O:
Operasi Remove Edge

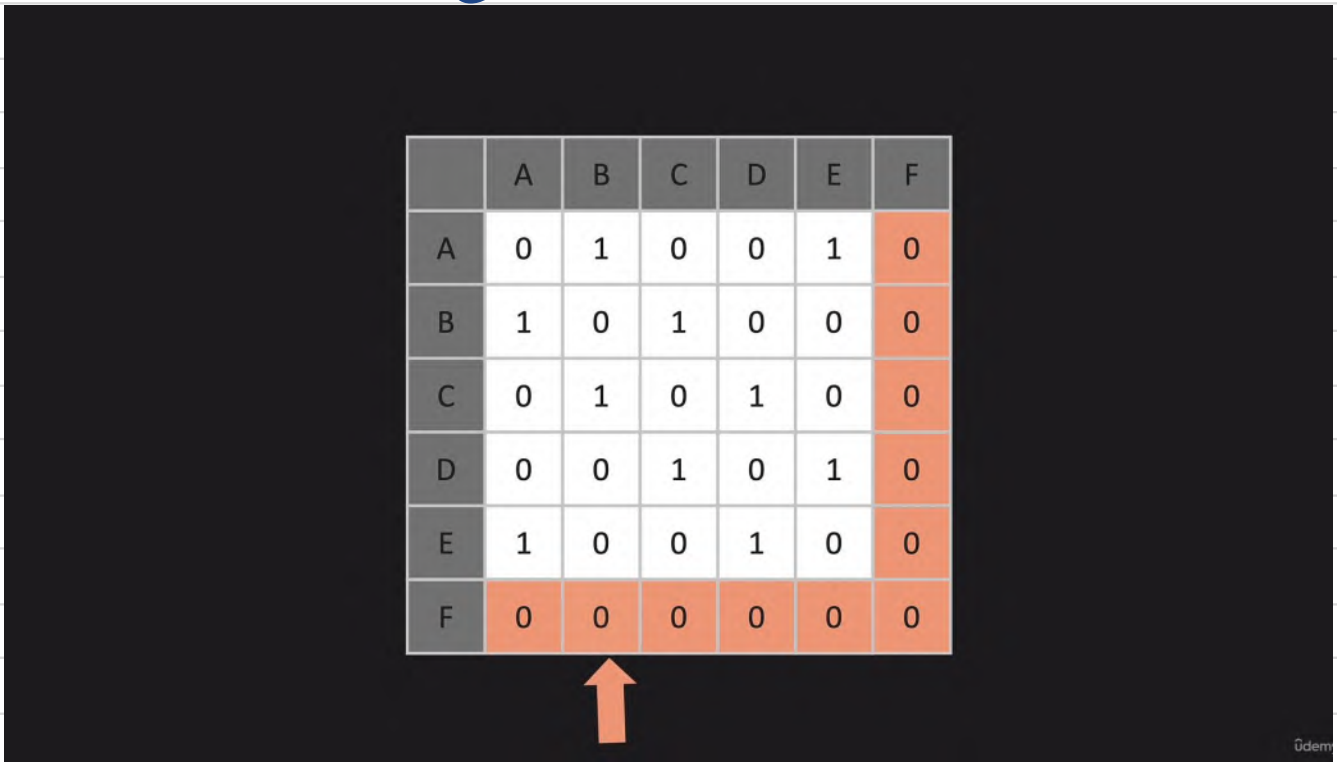
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Jika menggunakan adj matrix kita hanya perlu mengubah nilai 1 menjadi 0 pada kolom dan baris B & F

Graph Big O:
Operasi Remove Edge

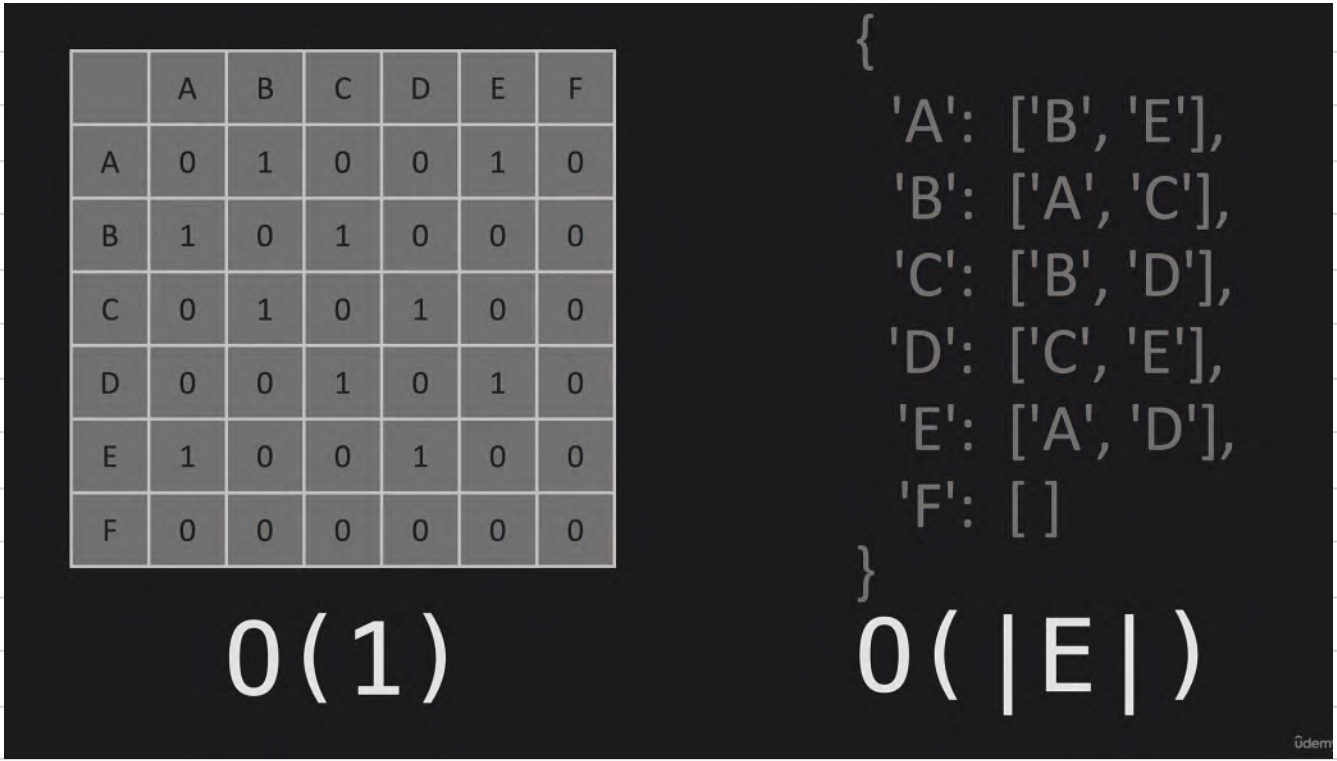
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Jika menggunakan adj matrix kita hanya perlu mengubah nilai 1 menjadi 0 pada kolom dan baris B & F

Graph Big O:
Operasi Remove Edge

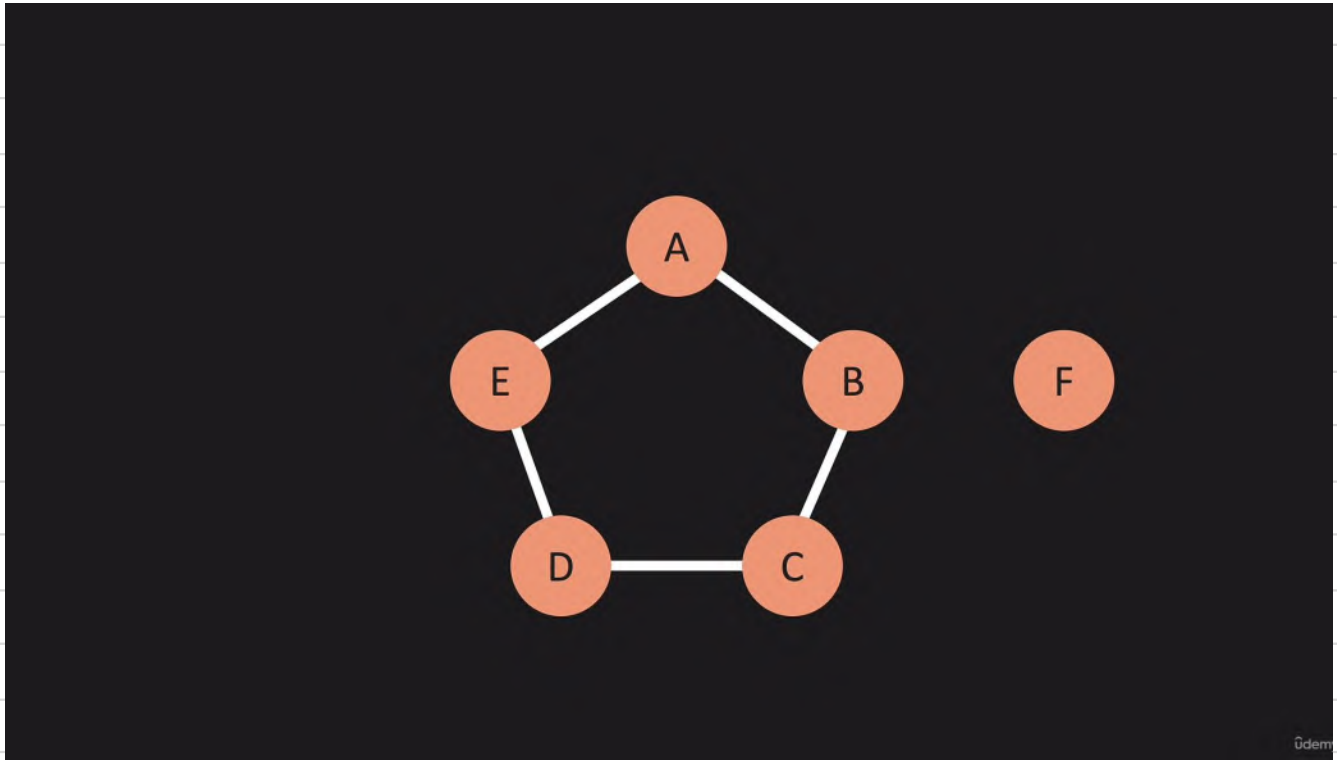
Diketahui suatu graph dan kita ingin
Menghapus edge antara vertex F dan B



Untuk operasi Remove Edge adj matrix lebih efisien dibandingkan adj list

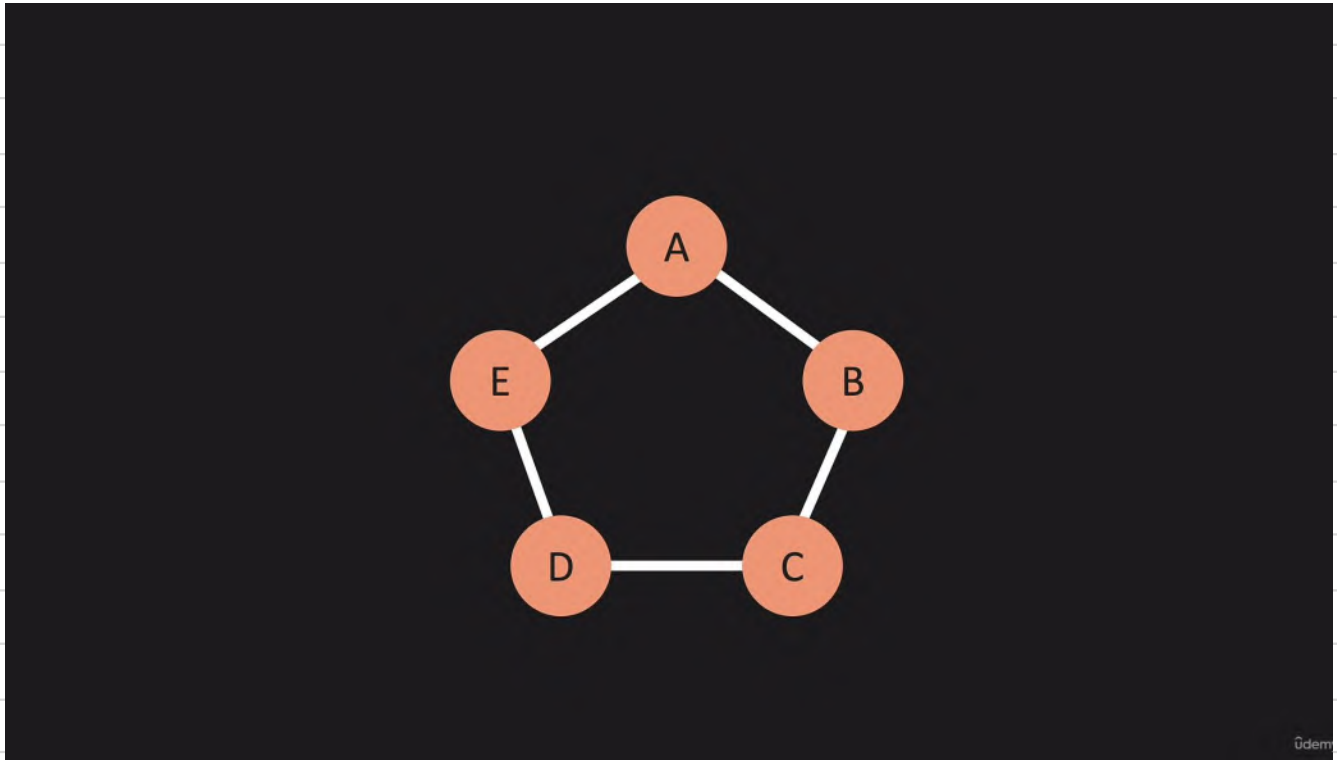
Graph Big O:
Operasi Remove Vertex

Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



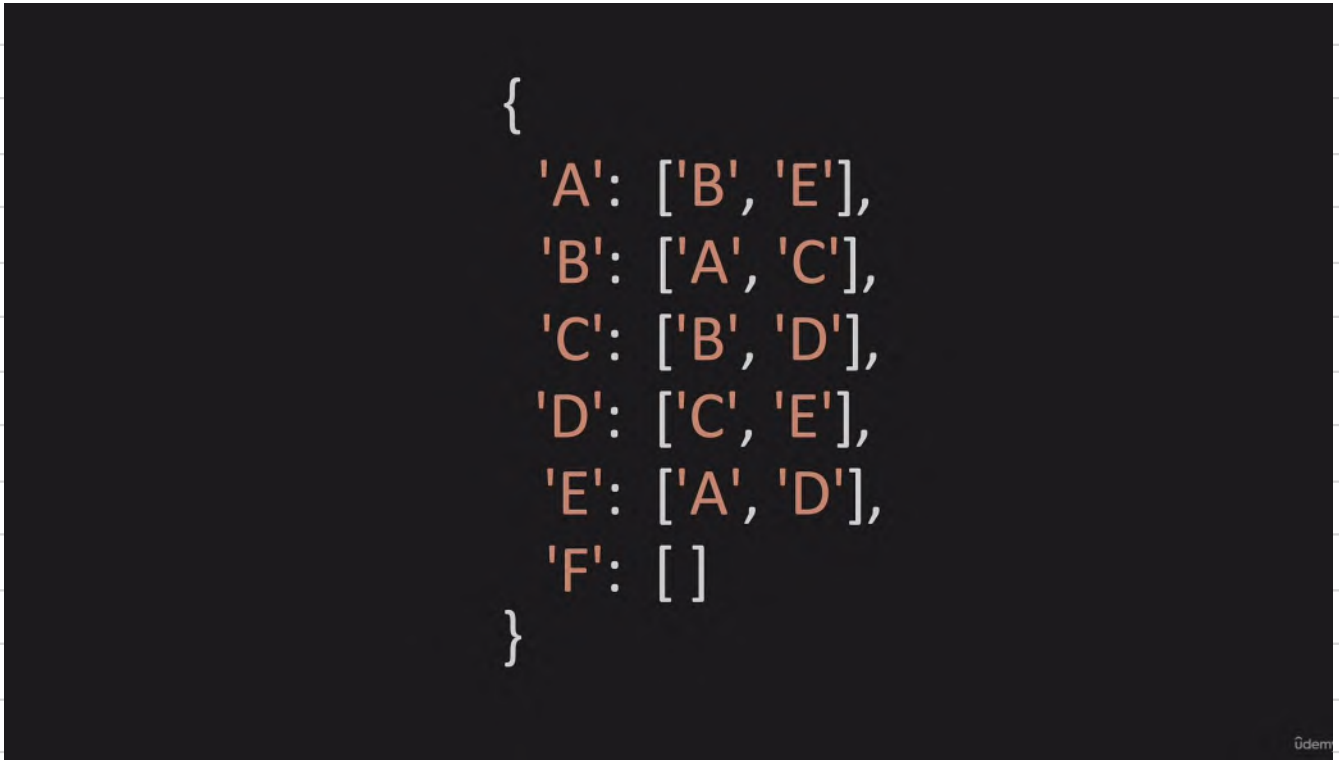
Graph Big O:
Operasi Remove Vertex

Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Graph Big O:
Operasi Remove Vertex

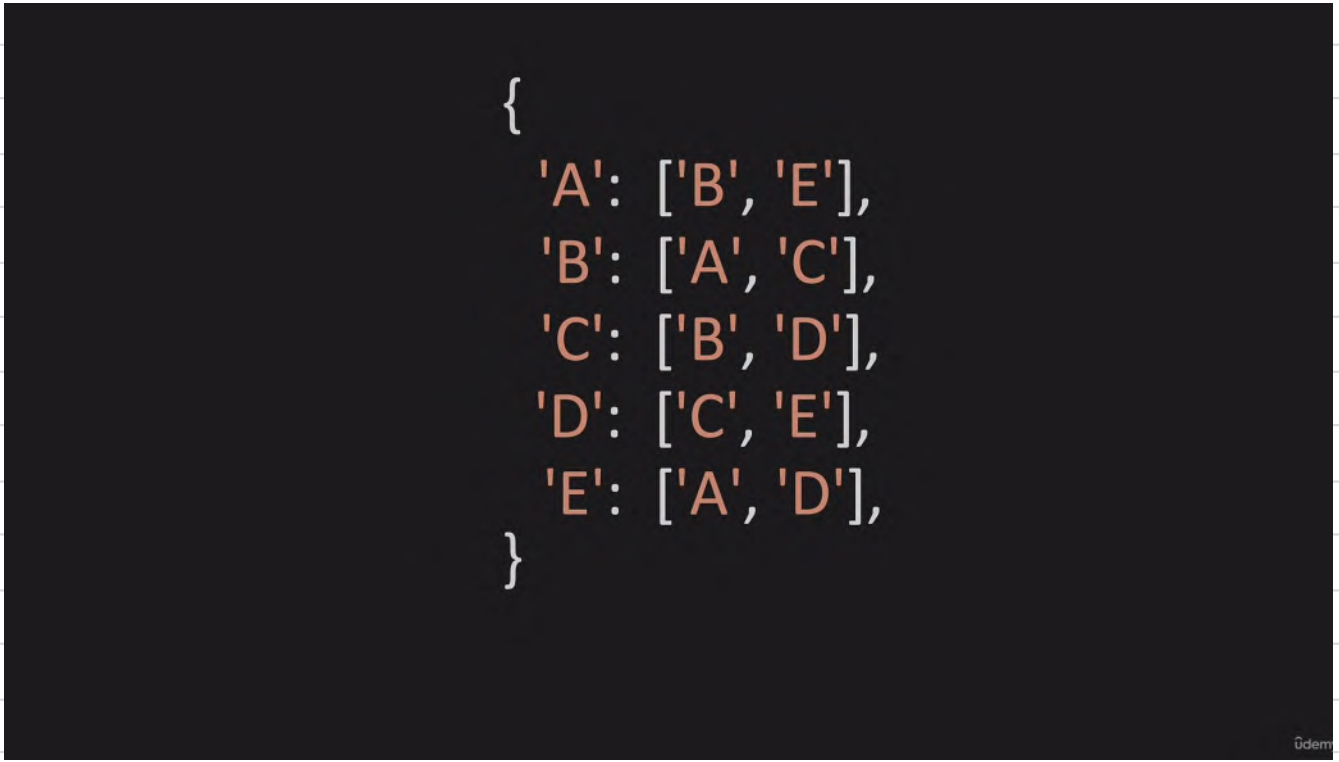
Diketahui suatu graph dan kita ingin Menghapus vertex F dari Graph



Jika menggunakan adj list maka kita akan menghapus vertex F beserta list nya di bagian paling bawah

Graph Big O:
Operasi Remove Vertex

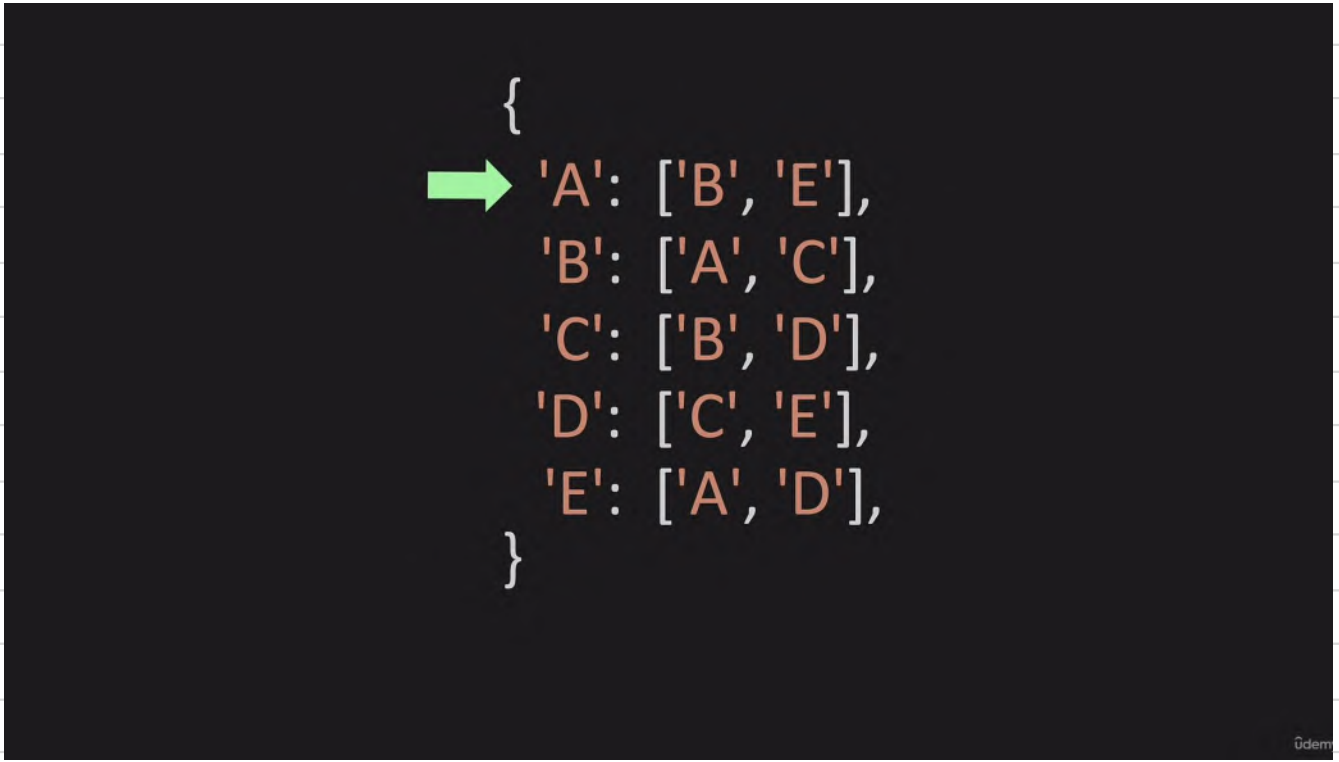
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Namun selain itu, kita perlu mengunjungi semua vertex yang lain satu per satu
dan menelusuri seluruh listnya untuk menemukan F dan menghapusnya

Graph Big O:
Operasi Remove Vertex

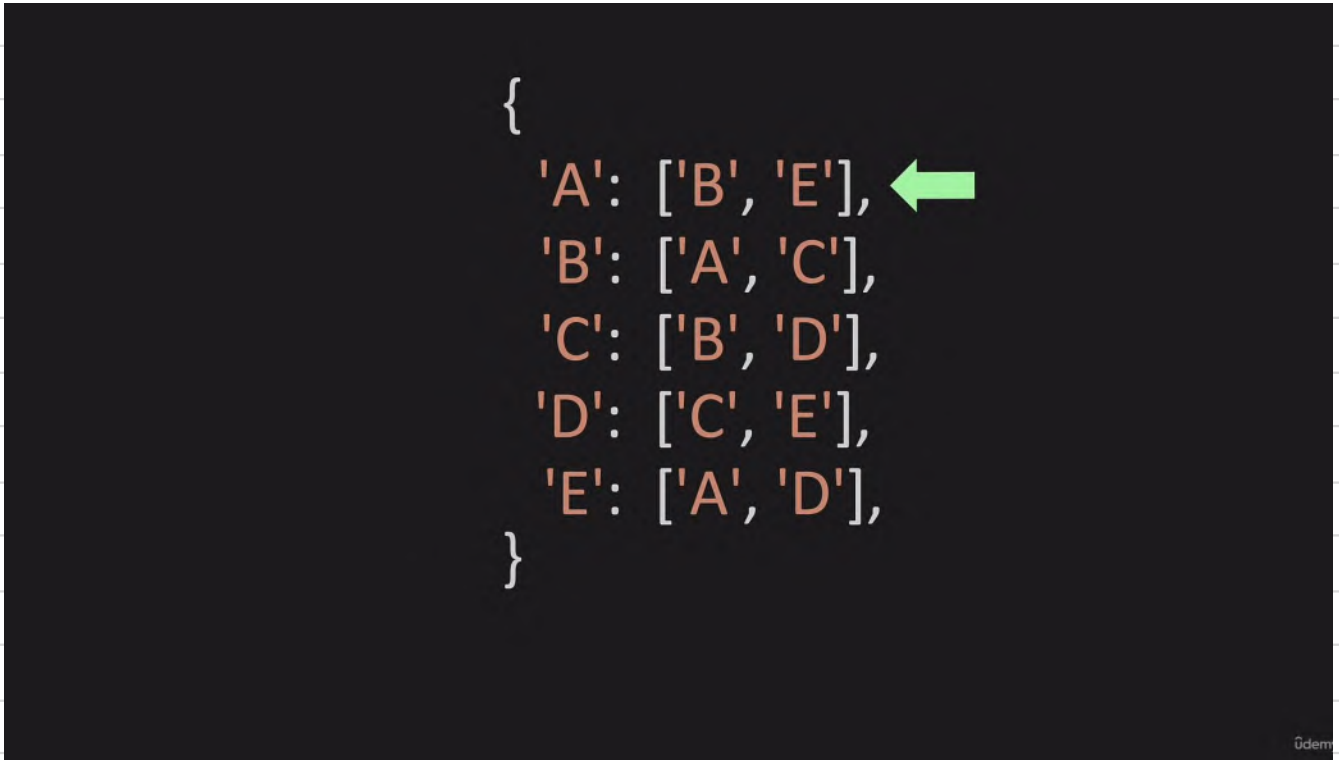
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Namun selain itu, kita perlu mengunjungi semua vertex yang lain satu per satu
dan menelusuri seluruh listnya untuk menemukan F dan menghapusnya

Graph Big O:
Operasi Remove Vertex

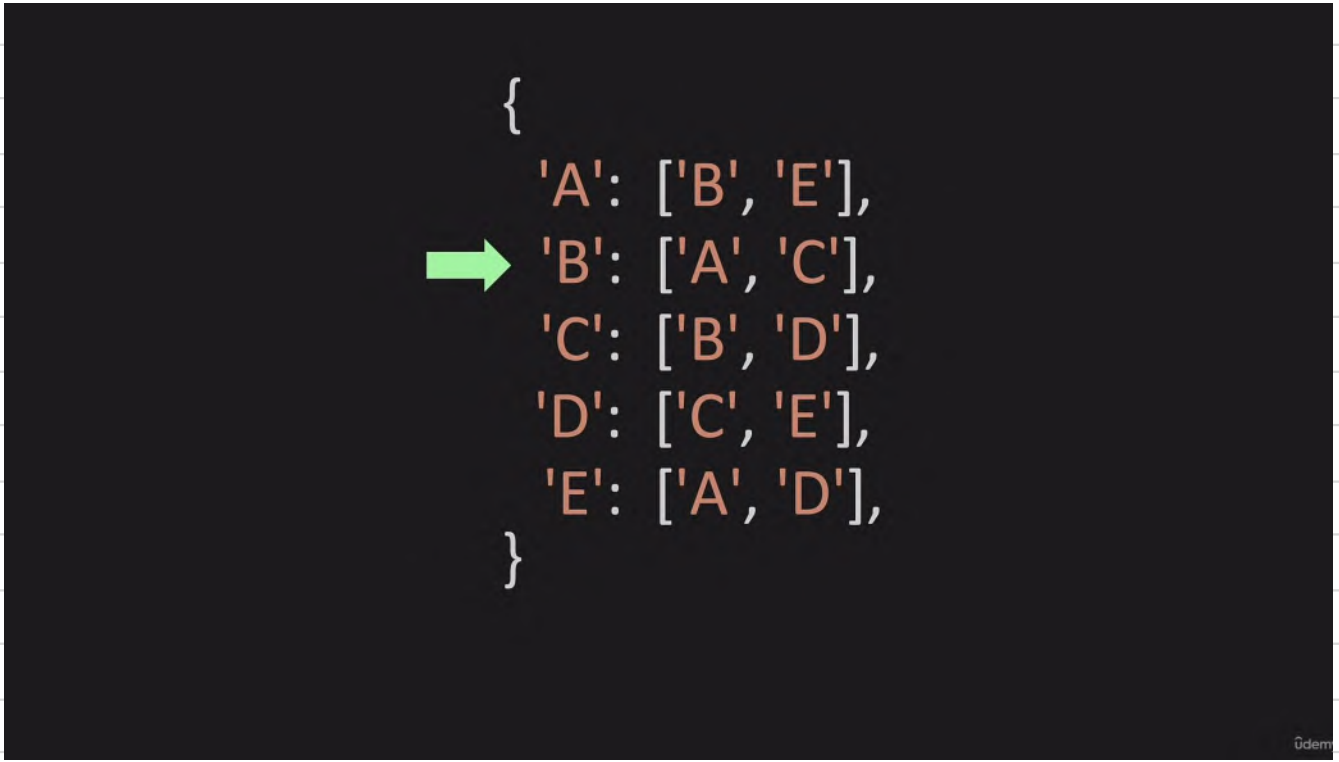
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Namun selain itu, kita perlu mengunjungi semua vertex yang lain satu per satu
dan menelusuri seluruh listnya untuk menemukan F dan menghapusnya

Graph Big O:
Operasi Remove Vertex

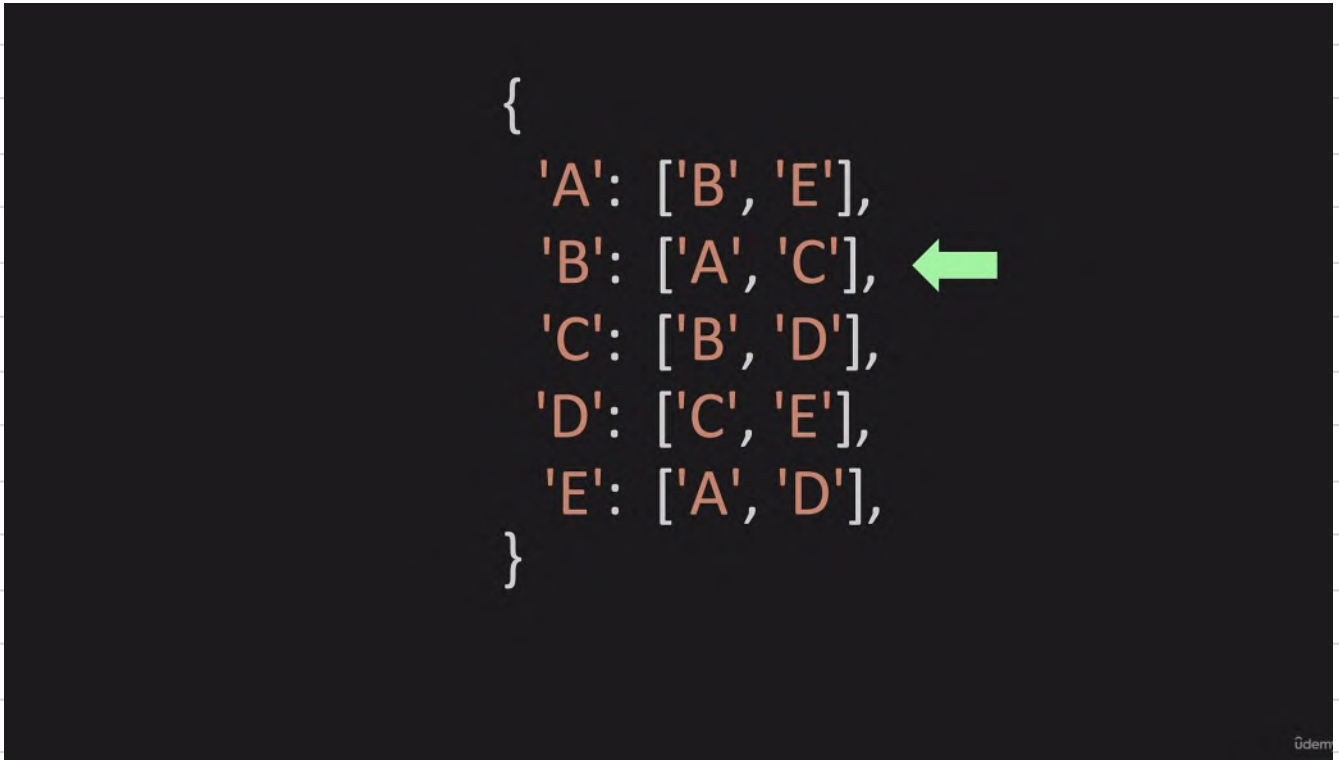
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Namun selain itu, kita perlu mengunjungi semua vertex yang lain satu per satu
dan menelusuri seluruh listnya untuk menemukan F dan menghapusnya

Graph Big O:
Operasi Remove Vertex

Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Namun selain itu, kita perlu mengunjungi semua vertex yang lain satu per satu
dan menelusuri seluruh listnya untuk menemukan F dan menghapusnya

Graph Big O:
Operasi Remove Vertex

Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph

	A	B	C	D	E	F
A	0	1	0	0	1	0
B	1	0	1	0	0	0
C	0	1	0	1	0	0
D	0	0	1	0	1	0
E	1	0	0	1	0	0
F	0	0	0	0	0	0

Jika menggunakan adj matrix kita perlu menulis ulang seluruh matrix karena mengubah ukuran matrix dari 6x6 menjadi 5x5

Graph Big O:
Operasi Remove Vertex

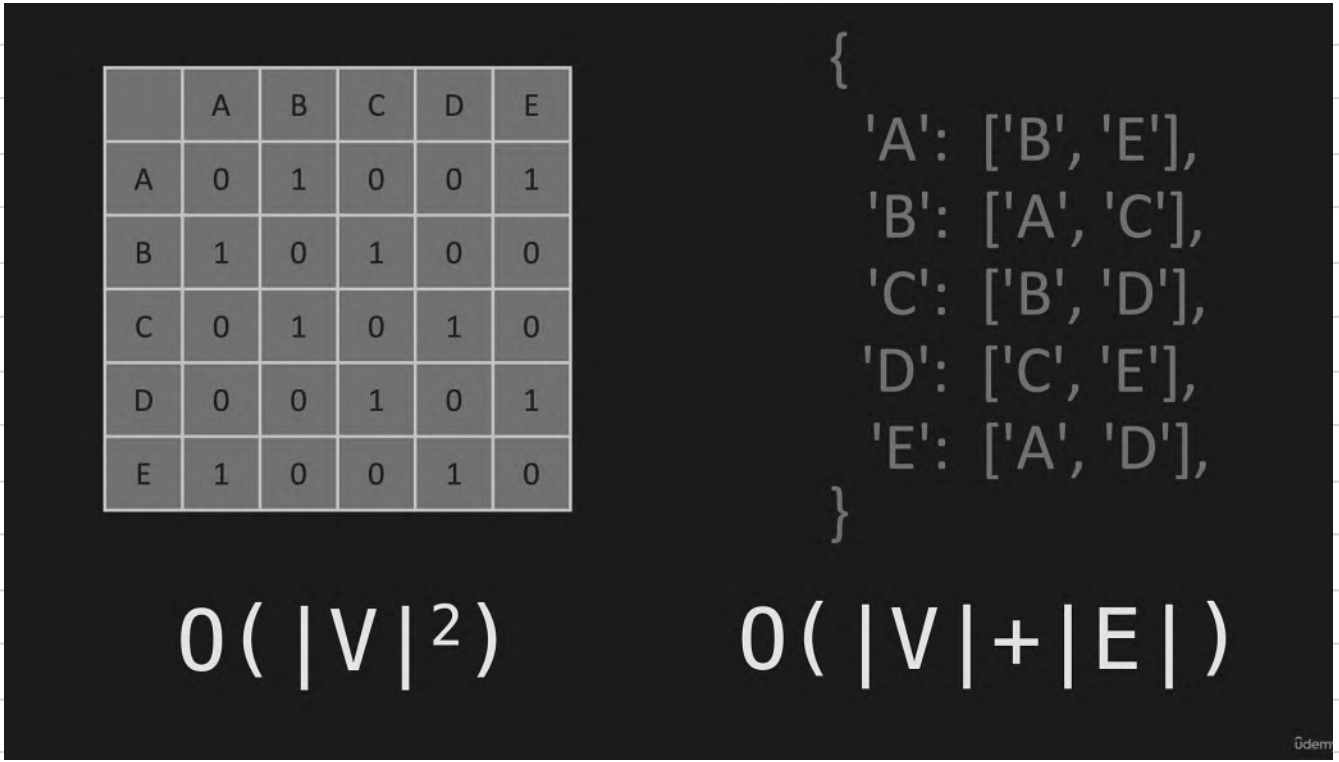
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	1	0
D	0	0	1	0	1
E	1	0	0	1	0

Jika menggunakan adj matrix kita perlu menulis ulang seluruh matrix karena mengubah ukuran matrix dari 6x6 menjadi 5x5

Graph Big O:
Operasi Remove Vertex

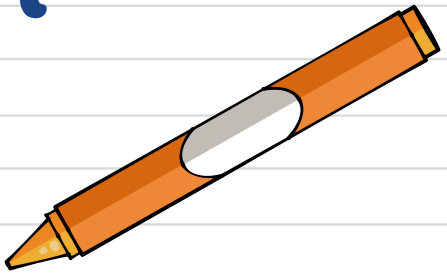
Diketahui suatu graph dan kita ingin
Menghapus vertex F dari Graph



Untuk operasi Remove Vertex adj List lebih efisien dibandingkan adj matrix

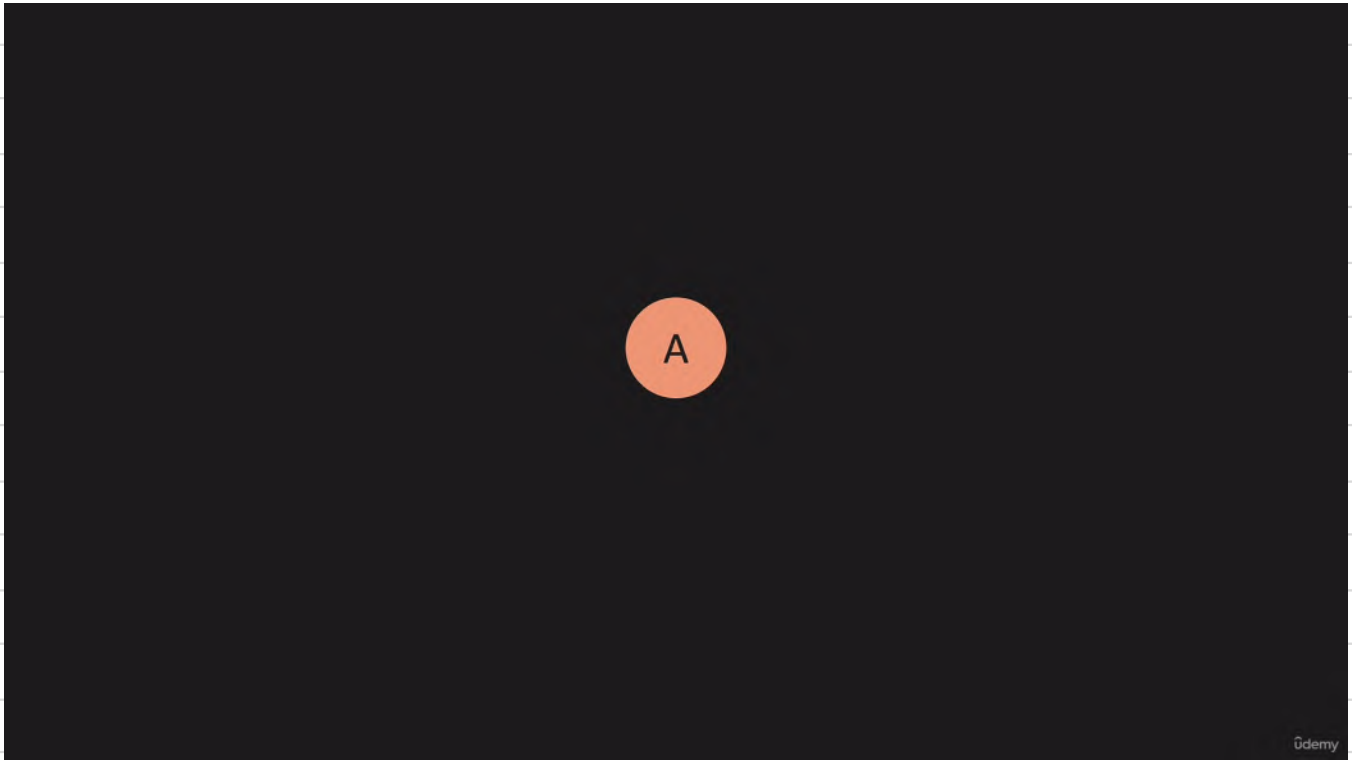
04

Add Vertex Implementation



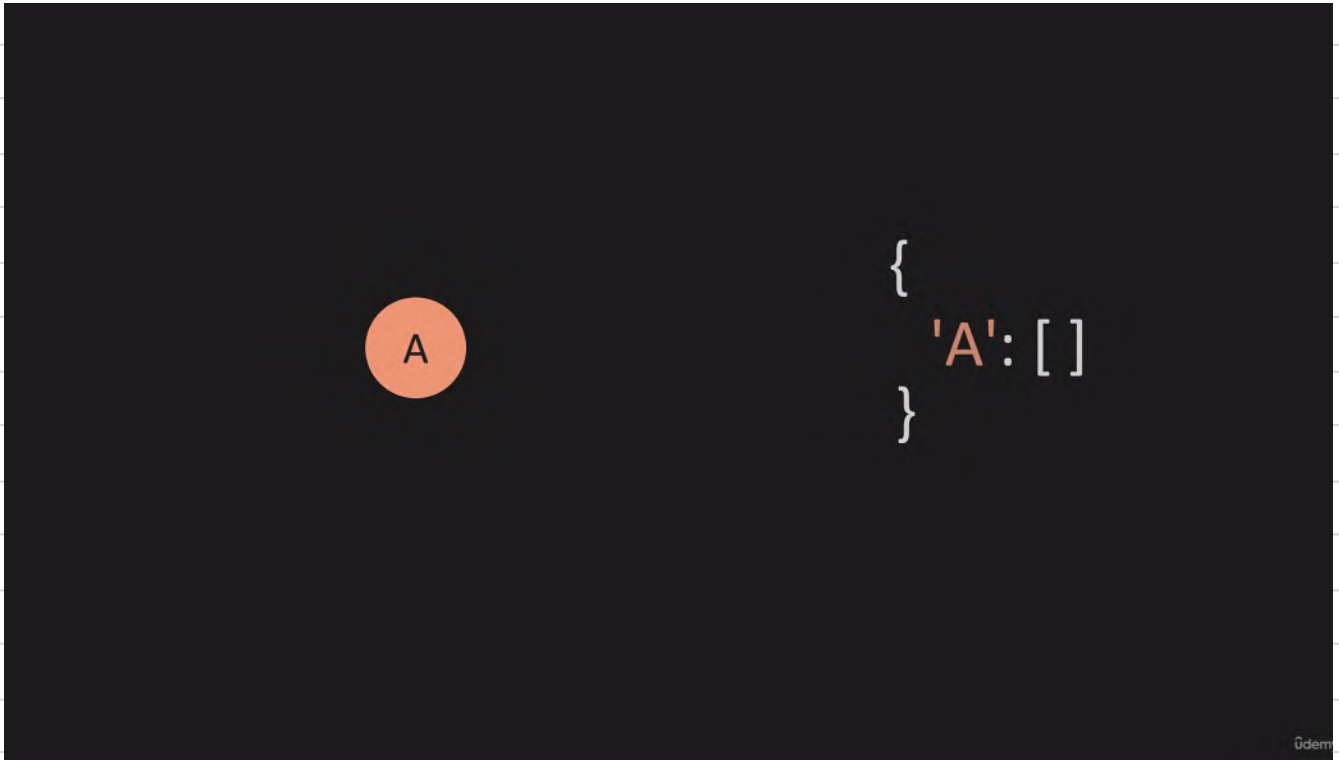
Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A



Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A



Secara logika, seharusnya akan terbuat adj list dengan key ‘A’ seperti di kanan

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

```
class Graph:
    def __init__(self):
        self.adj_list = {}
```

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

Secara logika, seharusnya akan terbuat adj list kosong seperti gambar

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

```
class Graph:
    def __init__(self):
        self.adj_list = {}

    def add_vertex(self, vertex):
```

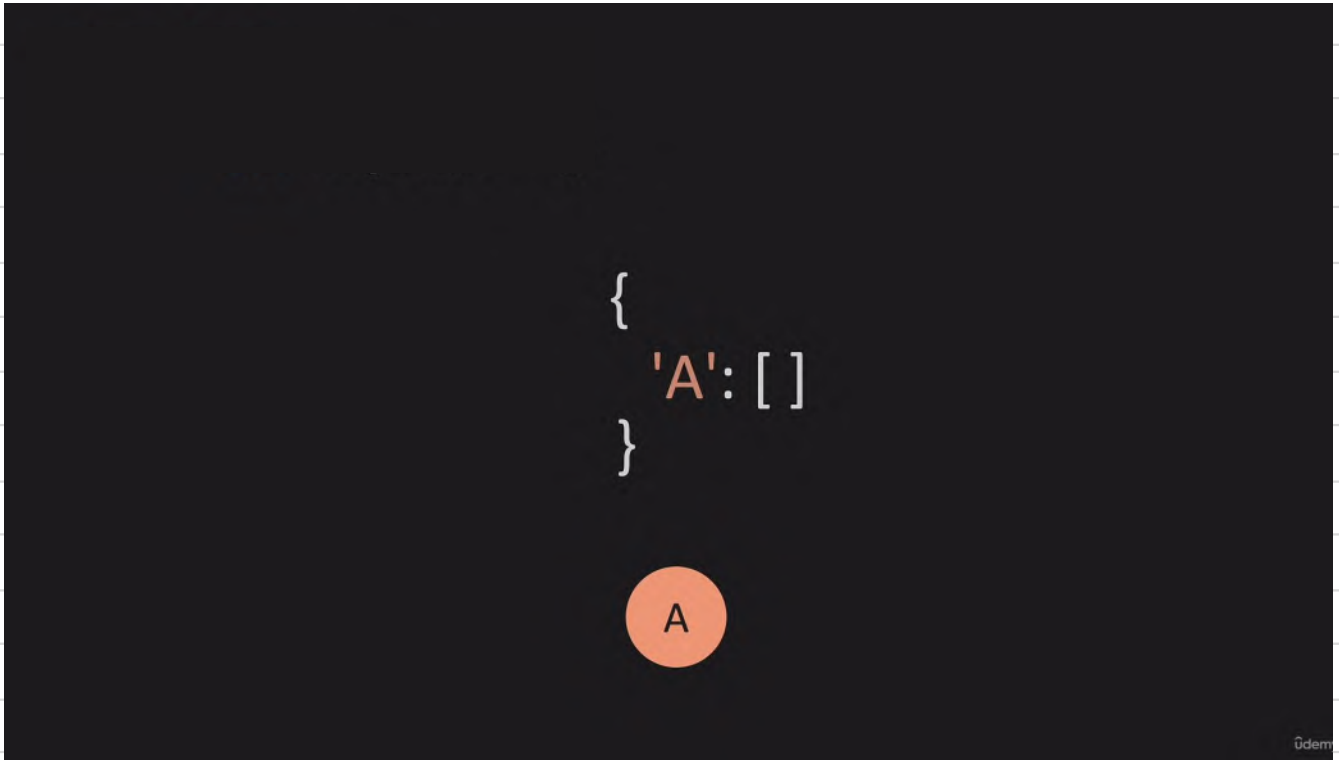
Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

```
class Graph:
    def __init__(self):
        self.adj_list = {}

    def add_vertex(self, vertex):
        if vertex not in self.adj_list.keys():
            self.adj_list[vertex] = []
            return True
        return False
```

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A



Secara logika, seharusnya akan terbuat adj list kosong dengan key ‘A’ seperti pada gambar

Add Vertex

Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

```python
class Graph:
    def __init__(self):
        self.adj_list = {}

    def print_graph(self):
        for vertex in self.adj_list:
            print(vertex, ':', self.adj_list[vertex])

    def add_vertex(self, vertex):
        if vertex not in self.adj_list.keys():
            self.adj_list[vertex] = []
            return True
        return False


my_graph = Graph()

my_graph.add_vertex('A')

my_graph.print_graph()
```

Add Vertex

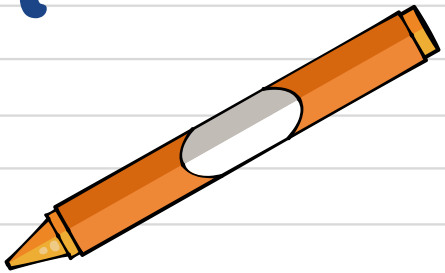
Misal kita ingin membuat 1 vertex di dalam Graph yaitu vertex A

```
    def add_vertex(self, vertex):
        if vertex not in self.adj_list.keys():
            self.adj_list[vertex] = []
            return True
        return False


my_graph = Graph()

my_graph.add_vertex('A')

my_graph.print_graph()
```

```
A : []
```

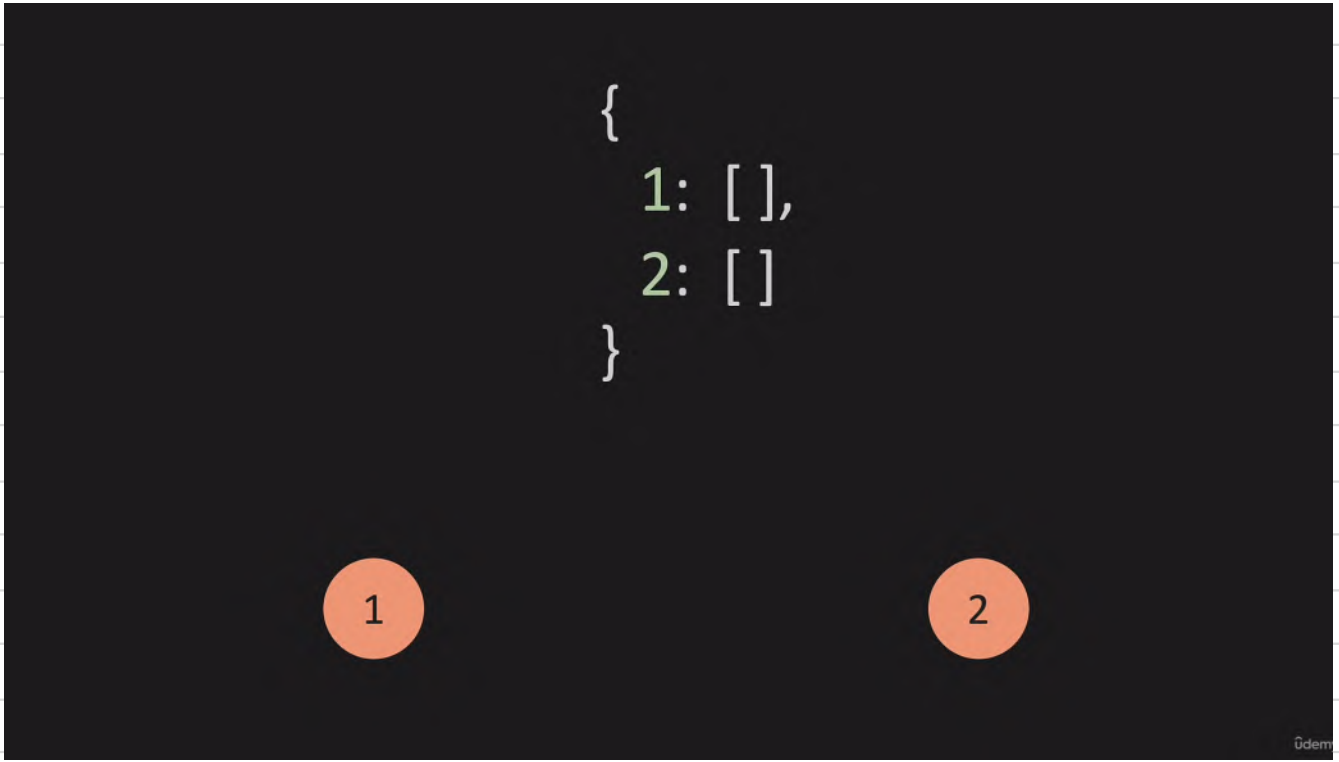
05

Add Edge Implementation



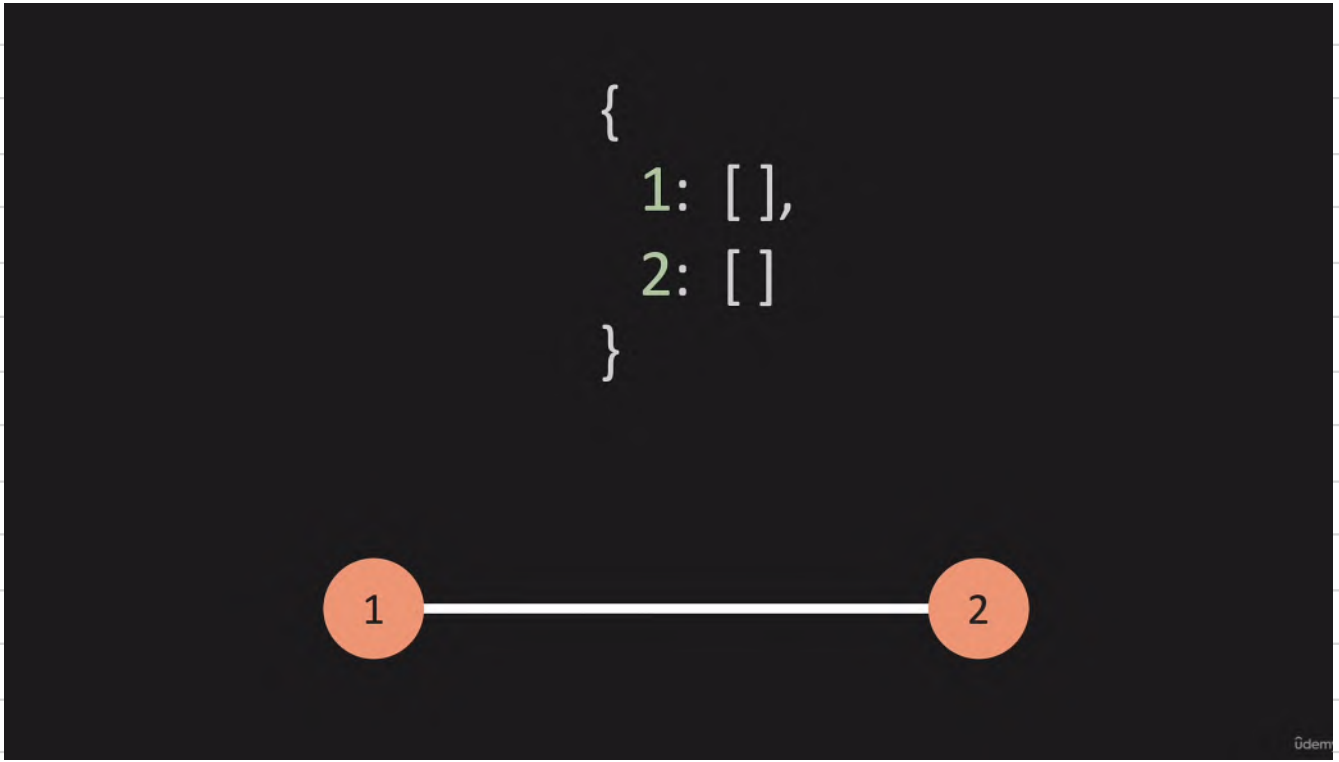
Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge



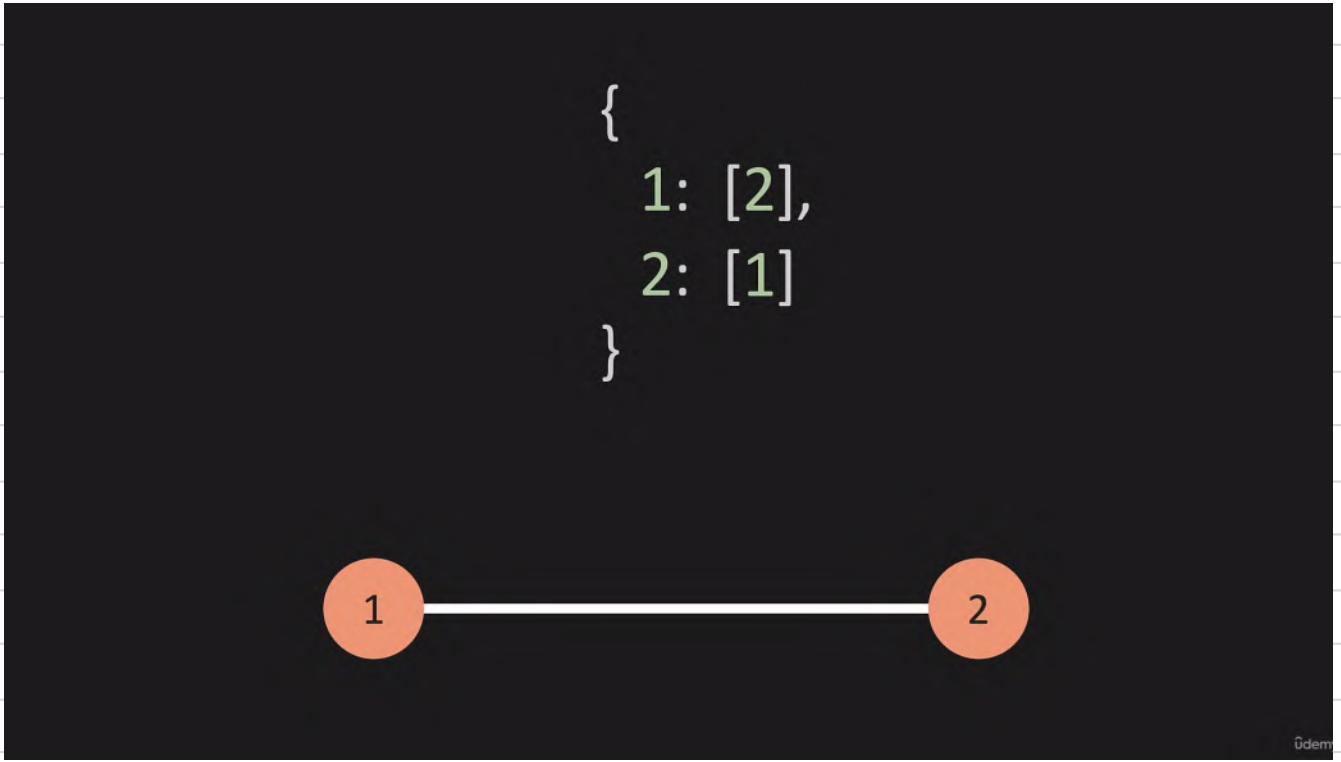
Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge



Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge



Secara logika, setelah kedua vertex disambungkan, seharusnya adj list masing-masing vertex telah terisi

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
def add_edge(self, v1, v2):

{
  1: [ ],
  2: [ ]
}
```

Kita mulai dengan membuat kerangka add_edge dimana v1 dan v2 adalah 2 vertex yang akan dihubungkan

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
def add_edge(self, v1, v2):
    self.adj_list[v1].append(v2)
```

```
{
  1: [2],
  2: [ ]
}
```

Baris kode program di atas akan meng-insert-kan 2 pada adj list key ‘1’

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
def add_edge(self, v1, v2):
    self.adj_list[v1].append(v2)
    self.adj_list[v2].append(v1)
```

```
{
  1: [2],
  2: [1]
}
```

Baris kode program di berikutnya akan meng-insert-kan 1 pada adj list key ‘2’

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
def add_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].append(v2)
        self.adj_list[v2].append(v1)
```

```
{
  1: [2],
  2: [1]
}
```

Janga lupa bahwa aksi ini hanya bisa dilakukan jika v1 dan v2 sudah menjadi key di adj list graph yang kita buat

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
def add_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].append(v2)
        self.adj_list[v2].append(v1)
        return True
    return False
```

Return kan True jika berhasil append ke adj list, false jika v1 atau v2 ada yg belum menjadi key di adj list

Add Edge

Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
    def add_edge(self, v1, v2):
        if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
            self.adj_list[v1].append(v2)
            self.adj_list[v2].append(v1)
            return True
        return False


my_graph = Graph()

my_graph.add_vertex(1)
my_graph.add_vertex(2)

my_graph.add_edge(1,2)

my_graph.print_graph()
```

Add Edge

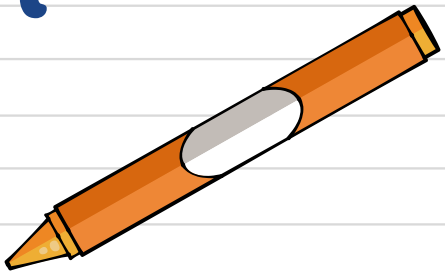
Misal kita memiliki 2 vertex dan kita akan menyambungkan keduanya dengan edge

```
    def add_edge(self, v1, v2):
        if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
            self.adj_list[v1].append(v2)
            self.adj_list[v2].append(v1)
            return True
        return False


my_graph = Graph()

my_graph.add_vertex(1)
my_graph.add_vertex(2)

my_graph.add_edge(1,2)

my_graph.print_graph()
```

```
1 : [2]
2 : [1]
```

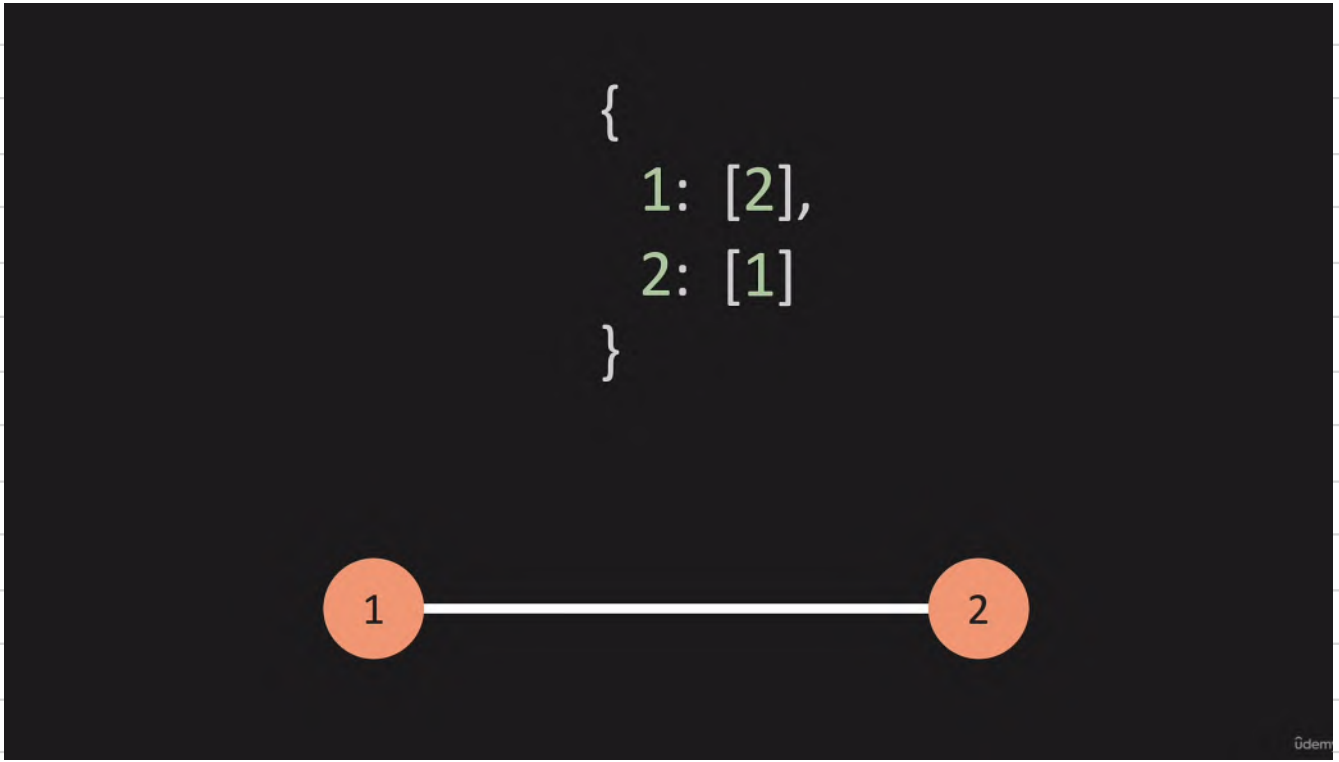
06

Remove Edge Implementation



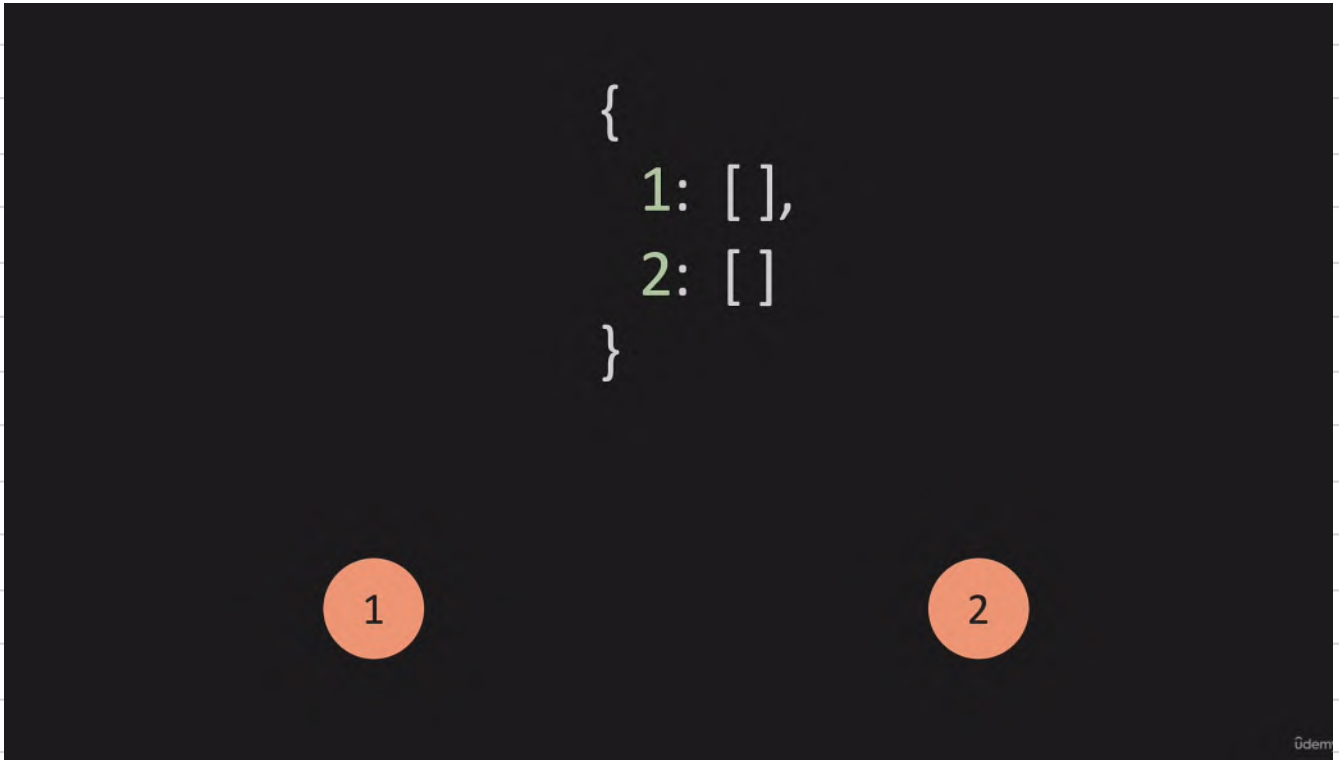
Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya



Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya



Secara logika, setelah edge kedua vertex dihapus, seharusnya adj list masing-masing vertex juga terhapus

Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya

```
def remove_edge(self, v1, v2):
```

```
{
  1: [2],
  2: [1]
}
```

Kita mulai dengan membuat kerangka remove_edge dimana v1 dan v2 adalah 2 vertex yang akan dihapus edge-nya

Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya

```
def remove_edge(self, v1, v2):
    self.adj_list[v1].remove(v2)
```

```
{
  1: [ ],
  2: [1]
}
```

Baris kode program di atas akan meng-hapus-kan 2 pada adj list key ‘1’

Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya

```
def remove_edge(self, v1, v2):
    self.adj_list[v1].remove(v2)
    self.adj_list[v2].remove(v1)
```

```
{
  1: [ ],
  2: [ ]
}
```

Baris kode program di berikutnya akan meng-hapus-kan 1 pada adj list key ‘2’

Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya

```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].remove(v2)
        self.adj_list[v2].remove(v1)
```

```
{
  1: [ ],
  2: [ ]
}
```

Janga lupa bahwa aksi ini hanya bisa dilakukan jika v1 dan v2 sudah menjadi key di adj list graph yang kita buat

Remove Edge

Misal kita memiliki 2 vertex yang terhubung dan kita akan menghapus koneksi edge antar keduanya

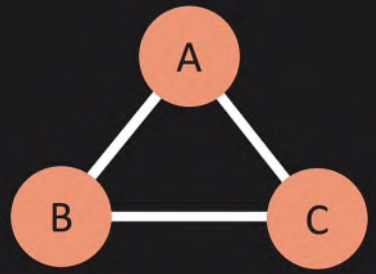
```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].remove(v2)
        self.adj_list[v2].remove(v1)
        return True
    return False
```

Return kan True jika berhasil di-remove dari adj list, false jika v1 atau v2 ada yg belum menjadi key di adj list

Remove Edge
Skenario 1

Misa kita memiliki 3 vertex yang saling tersambung ABC

```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].remove(v2)
        self.adj_list[v2].remove(v1)
        return True
    return False
```

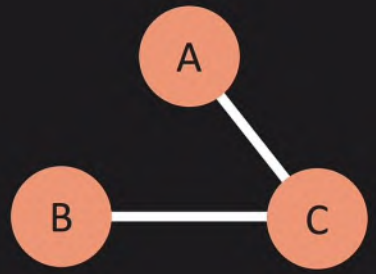


Sambungan antara A dan B akan dihapus

Remove Edge
Skenario 1

Misa kita memiliki 3 vertex yang saling tersambung ABC

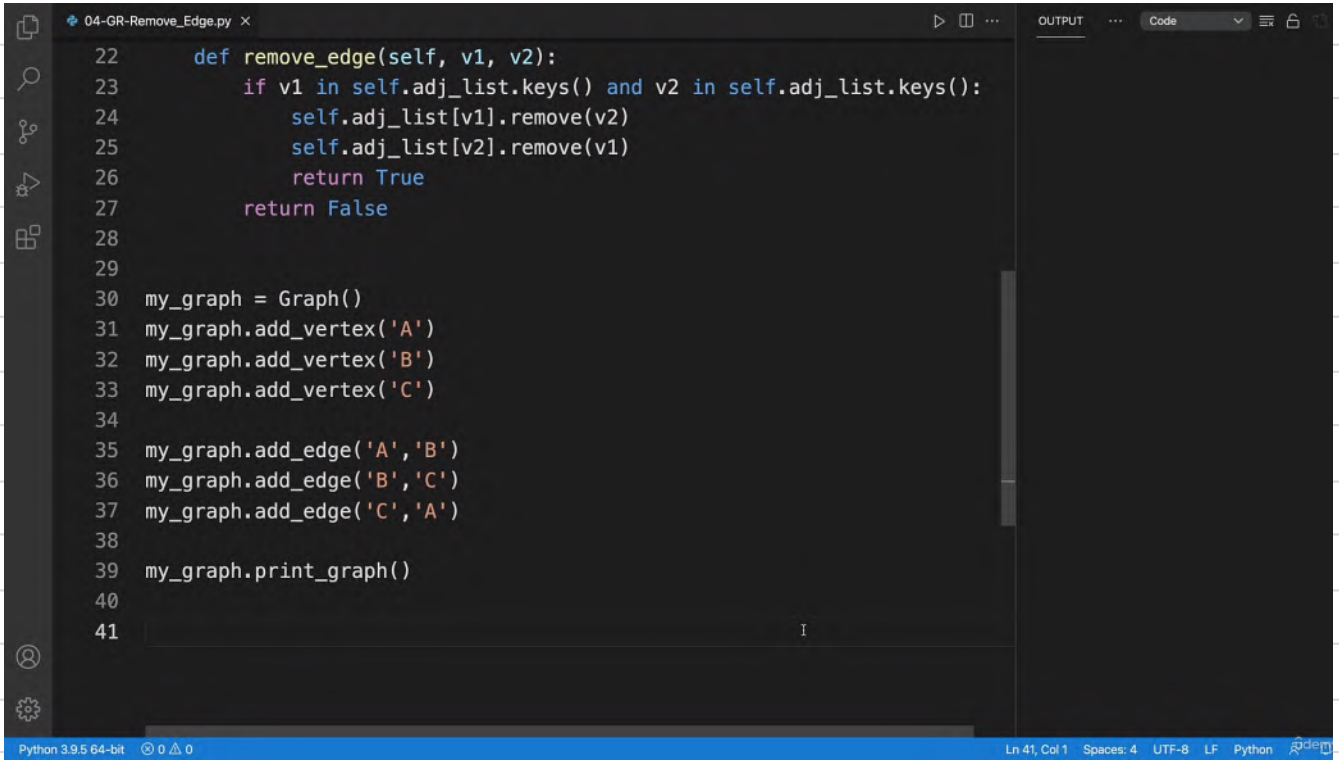
```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        self.adj_list[v1].remove(v2)
        self.adj_list[v2].remove(v1)
        return True
    return False
```



Sambungan antara A dan B akan dihapus

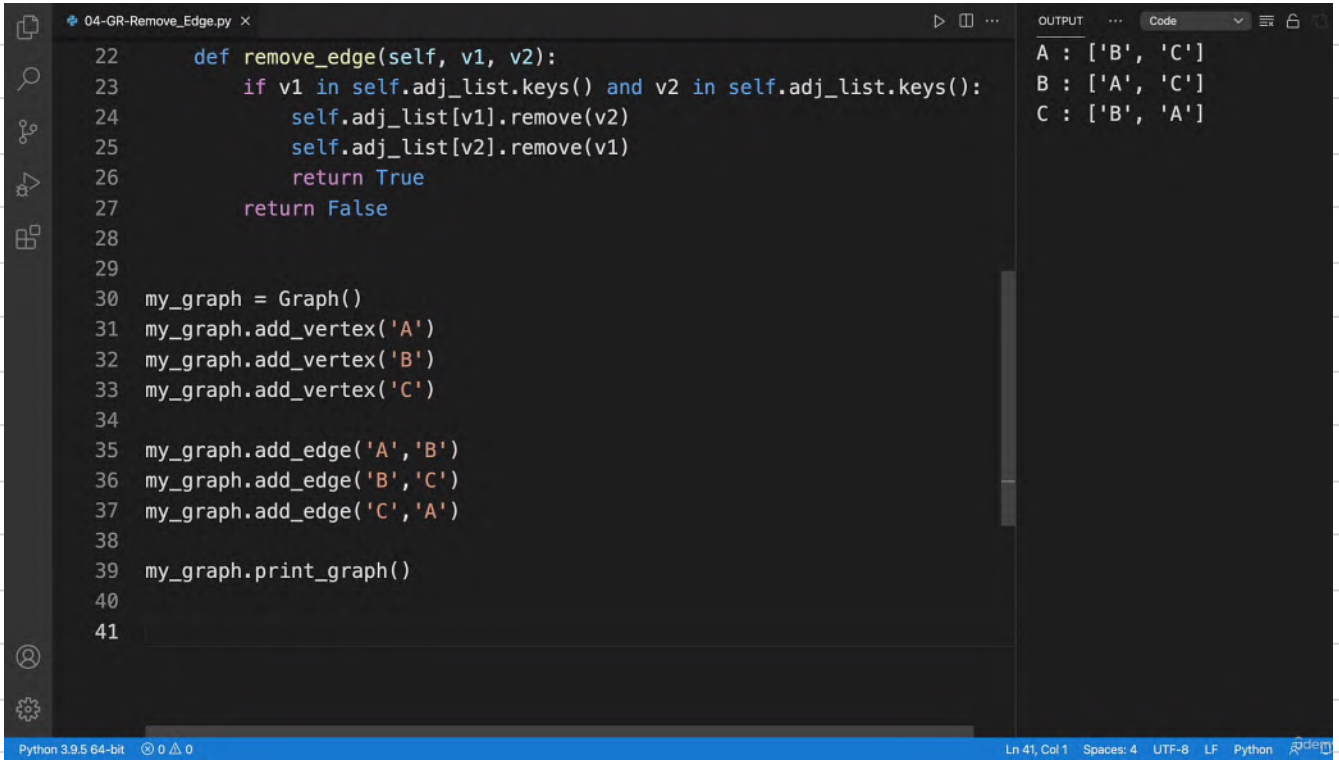
Remove Edge

Misa kita memiliki 3 vertex yang saling tersambung ABC



Remove Edge

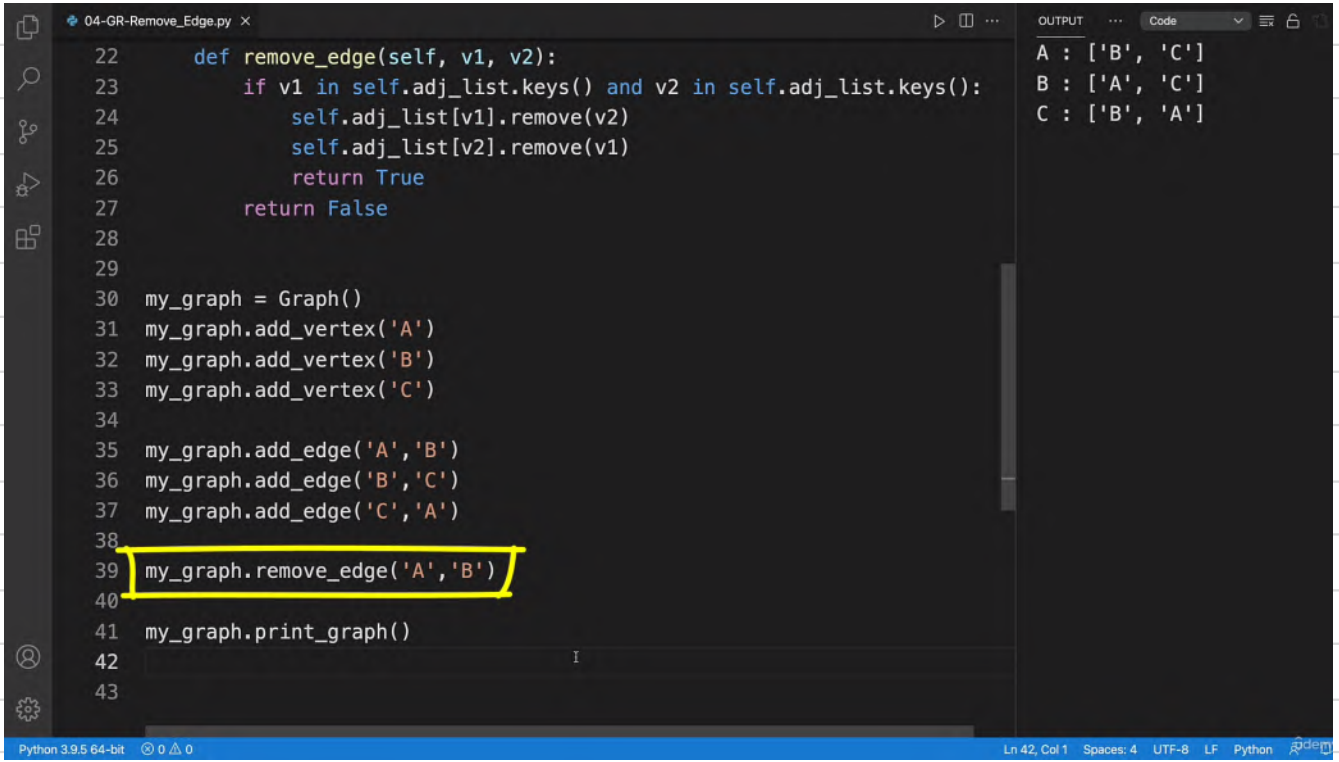
Misa kita memiliki 3 vertex yang saling tersambung ABC



Sebelum penghapusan masing-masing vertex masih tersambung satu sama lain

Remove Edge

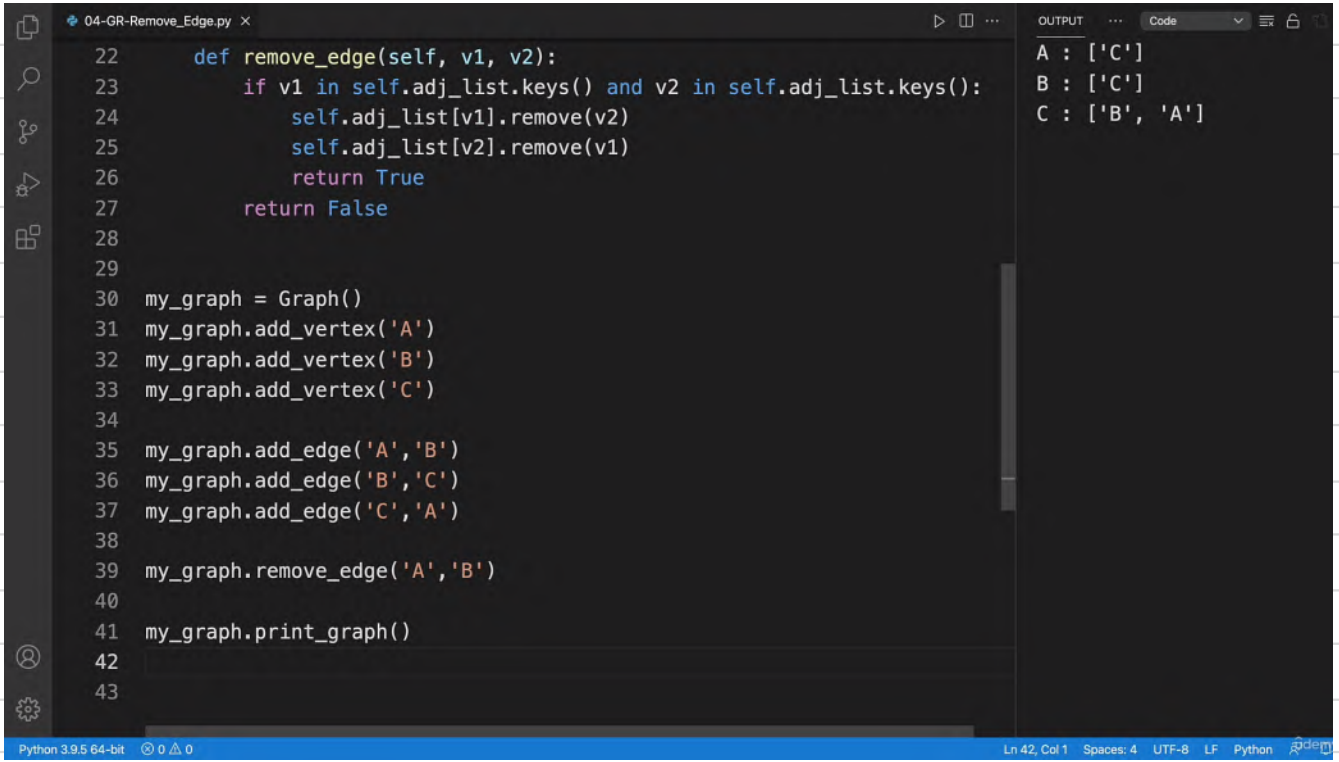
Misa kita memiliki 3 vertex yang saling tersambung ABC



Kotak kuning adalah sintaks untuk menghapus edge antara A dan B

Remove Edge

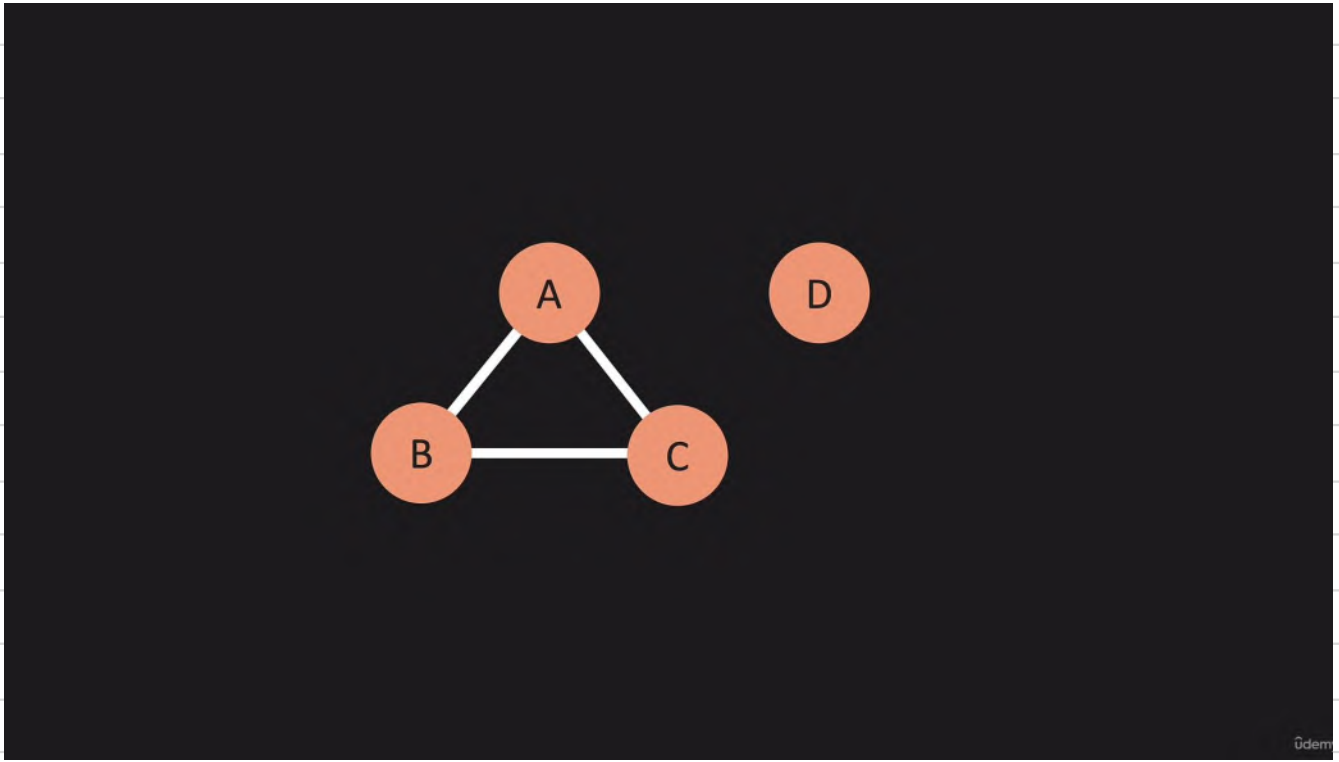
Misa kita memiliki 3 vertex yang saling tersambung ABC



Setelah penghapusan adj list key A dan B sama-sama tidak memuat satu sama lain

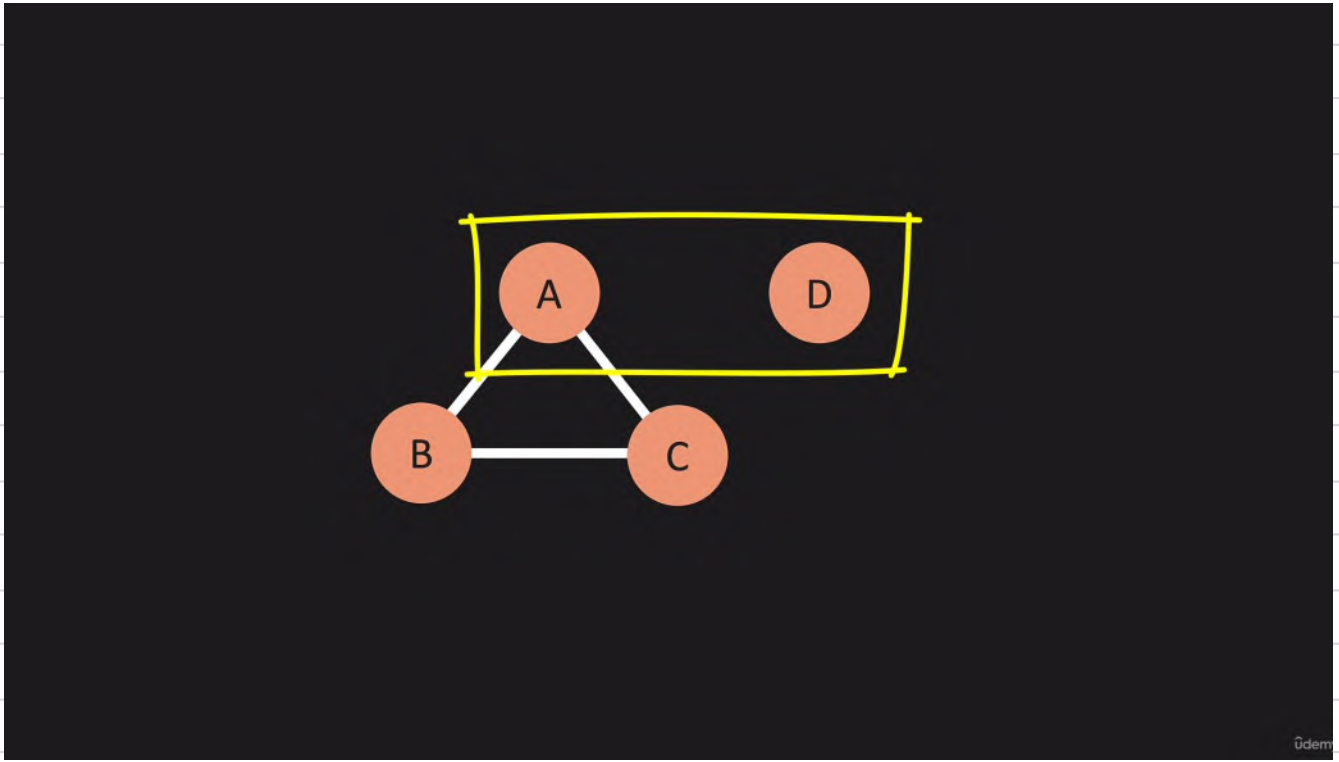
Remove Edge
Skenario 2

Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Remove Edge Skenario 2

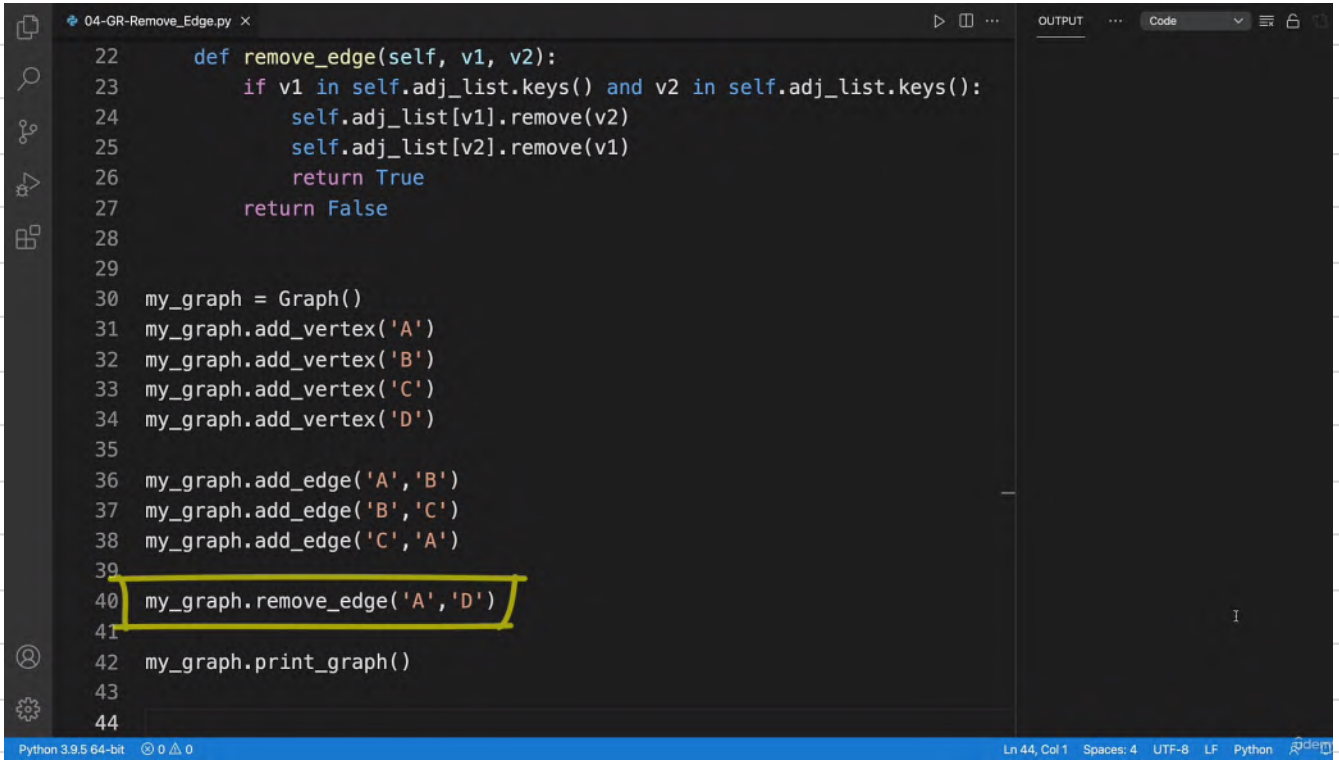
Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Kita akan menghapus Edge antara A dan D (yang tidak pernah ada di dalam Graph)

Remove Edge Skenario 2

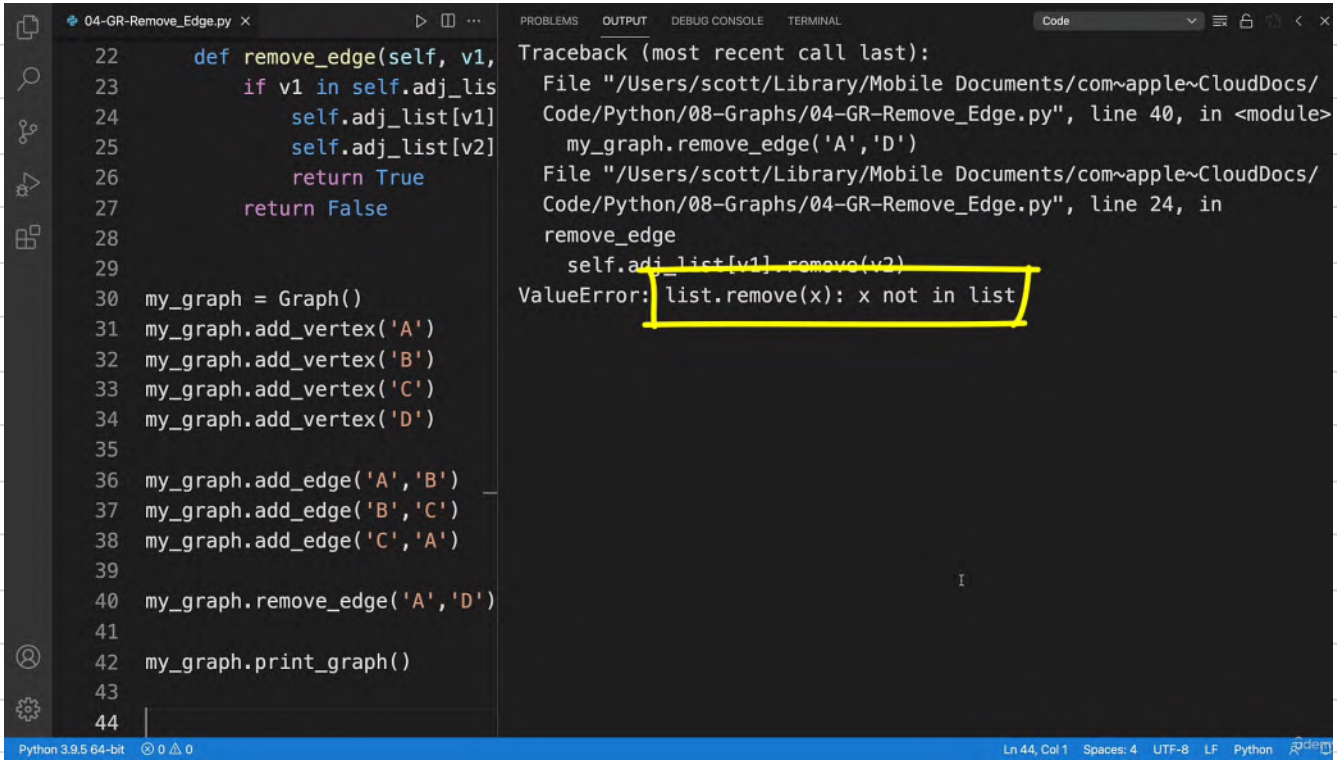
Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Kota kuning adalah sintaks untuk menghapus edge antara A dan D

Remove Edge
Skenario 2

Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Setelah eksekusi muncul error x not in list

Remove Edge Skenario 2

Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung

Ternyata permasalahan ada di baris 24

Remove Edge Skenario 2

Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung

```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        try:
            self.adj_list[v1].remove(v2)
            self.adj_list[v2].remove(v1)
        except ValueError:
            pass
        return True
    return False
```

Kita modifikasi baris tersebut dengan menambahkan try-except untuk penanganan exception yang lebih baik

Remove Edge Skenario 2

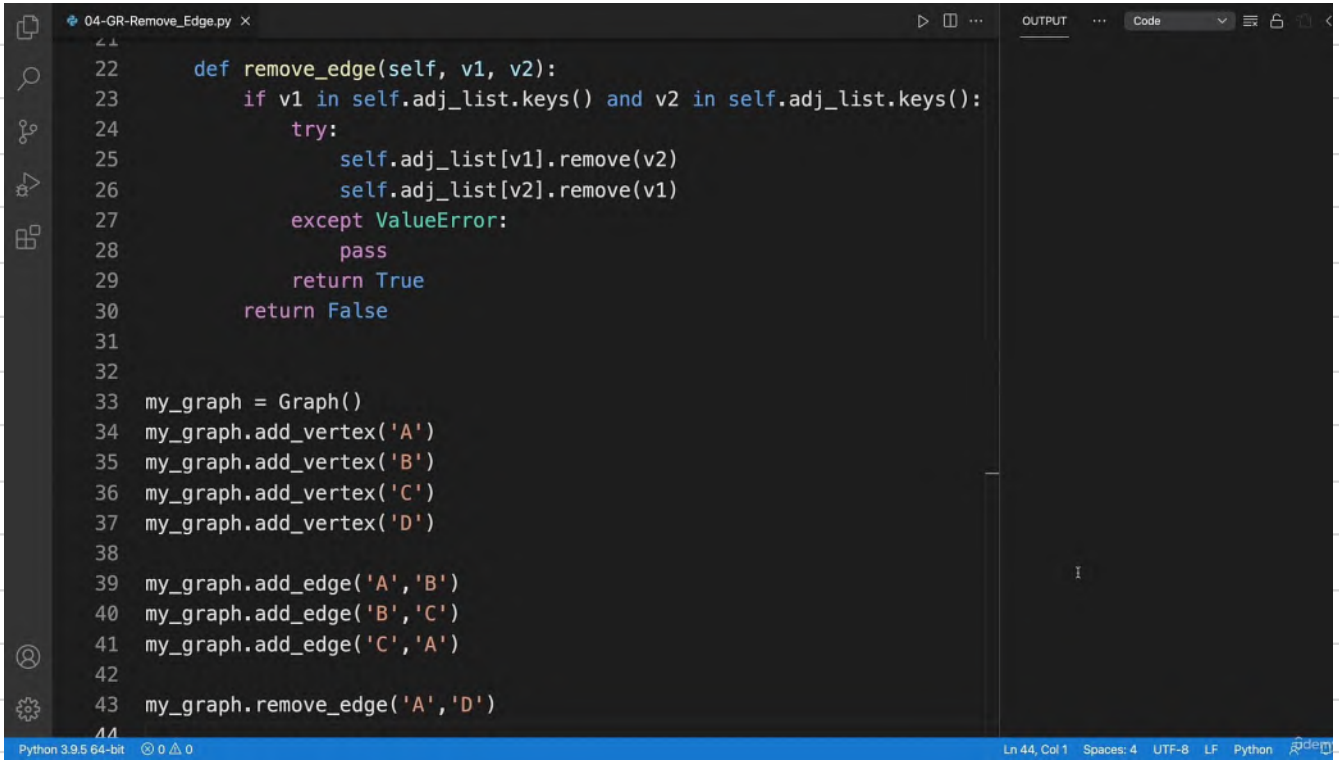
Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung

```
def remove_edge(self, v1, v2):
    if v1 in self.adj_list.keys() and v2 in self.adj_list.keys():
        try:
            self.adj_list[v1].remove(v2)
            self.adj_list[v2].remove(v1)
        except ValueError:
            pass
        return True
    return False
```

Kode program setelah diperbarui

Remove Edge Skenario 2

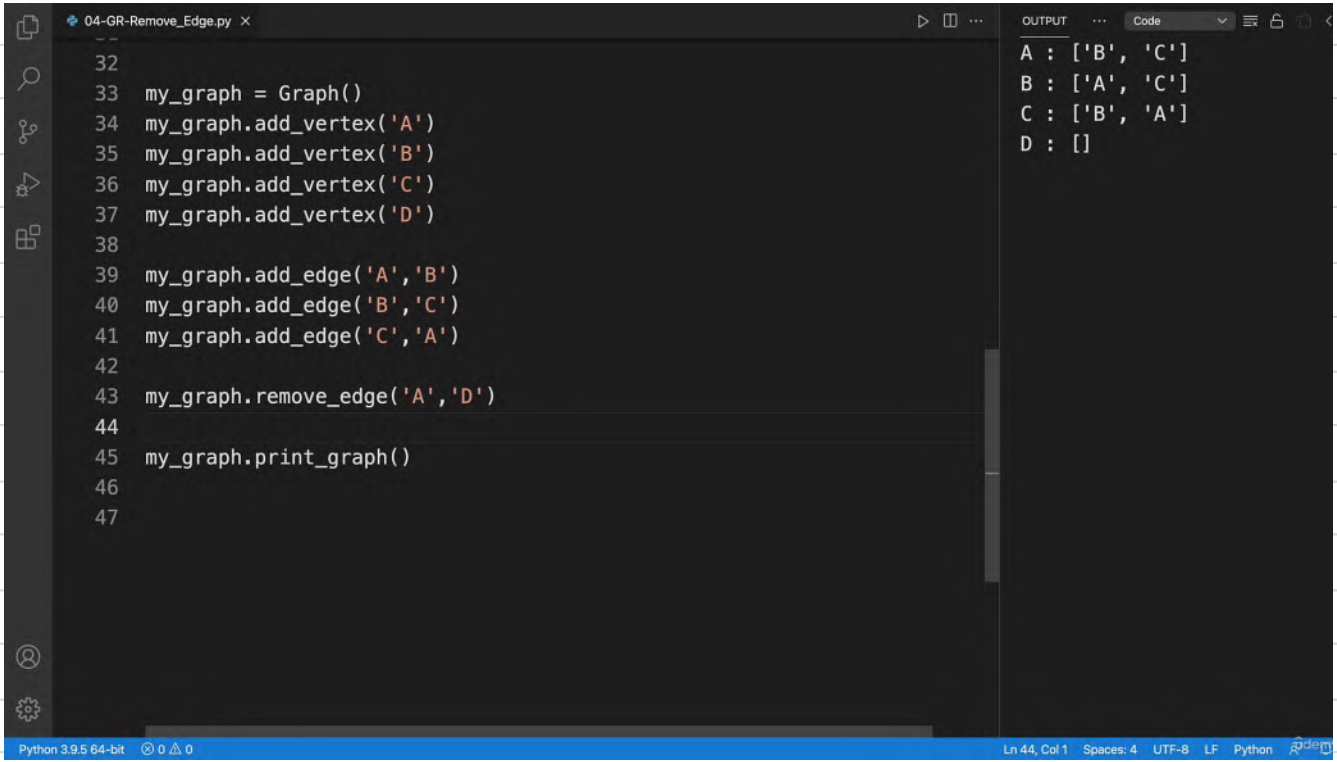
Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Dilakukan pengujian ulang dengan cara menghapus edge antara A dan D

Remove Edge
Skenario 2

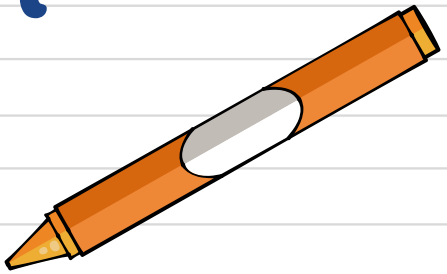
Misa kita memiliki 3 vertex yang saling tersambung ABC dan D yang tidak tersambung



Hasilnya program tetap berjalan dan tidak ada value error yang muncul

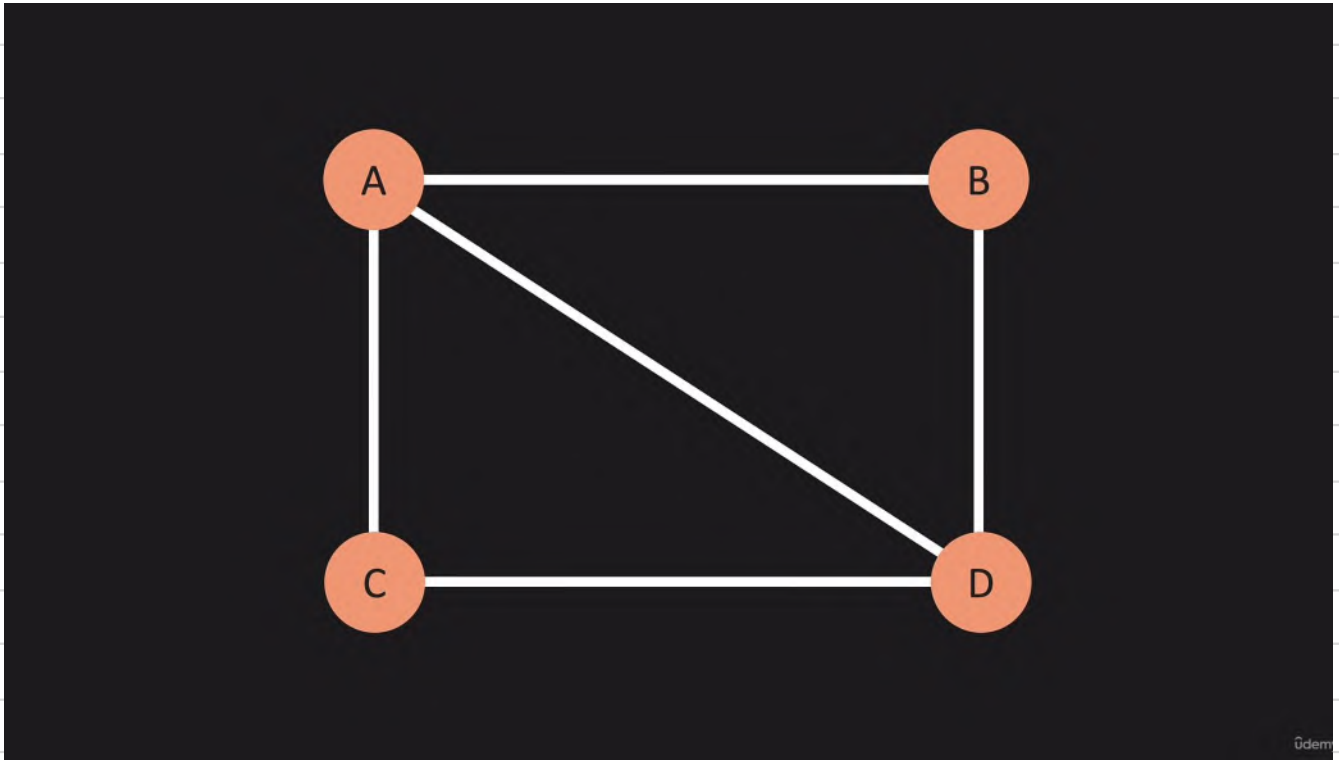
07

Remove Vertex Implementation



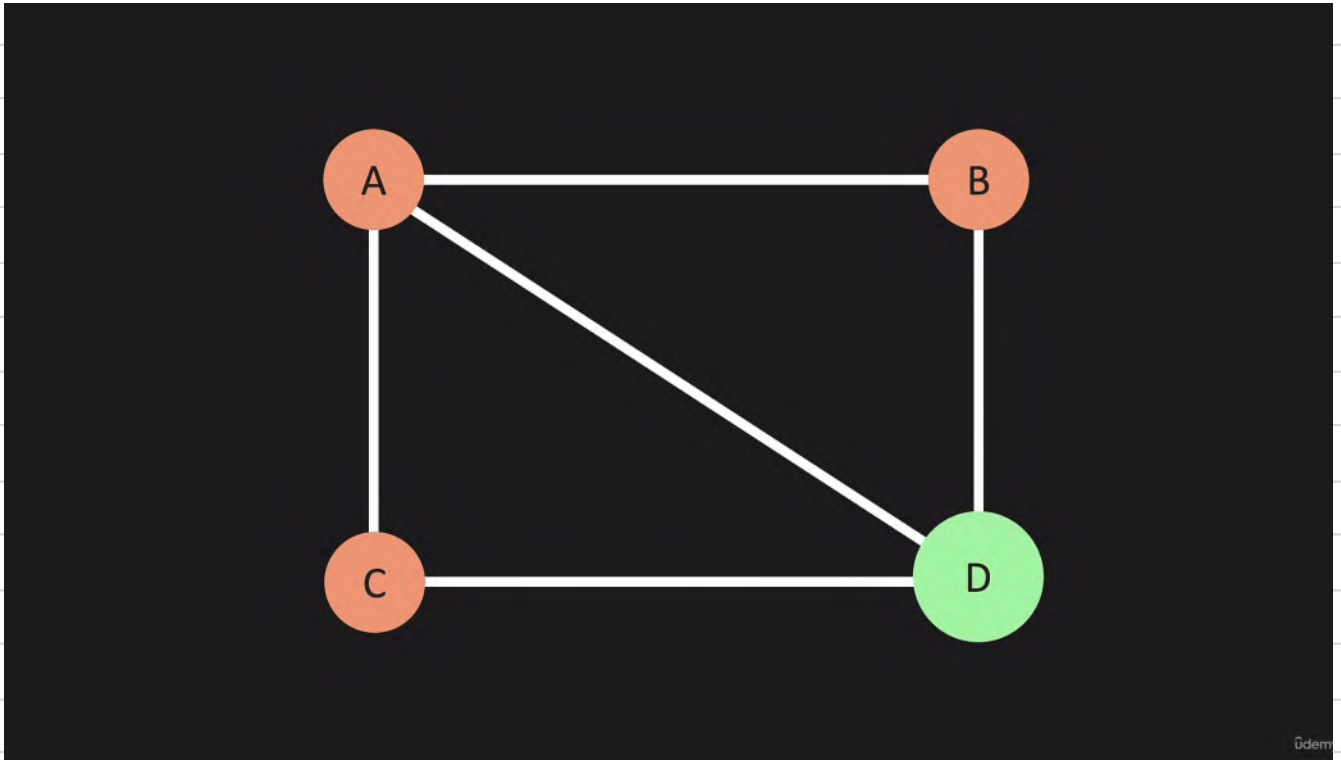
Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Remove Vertex

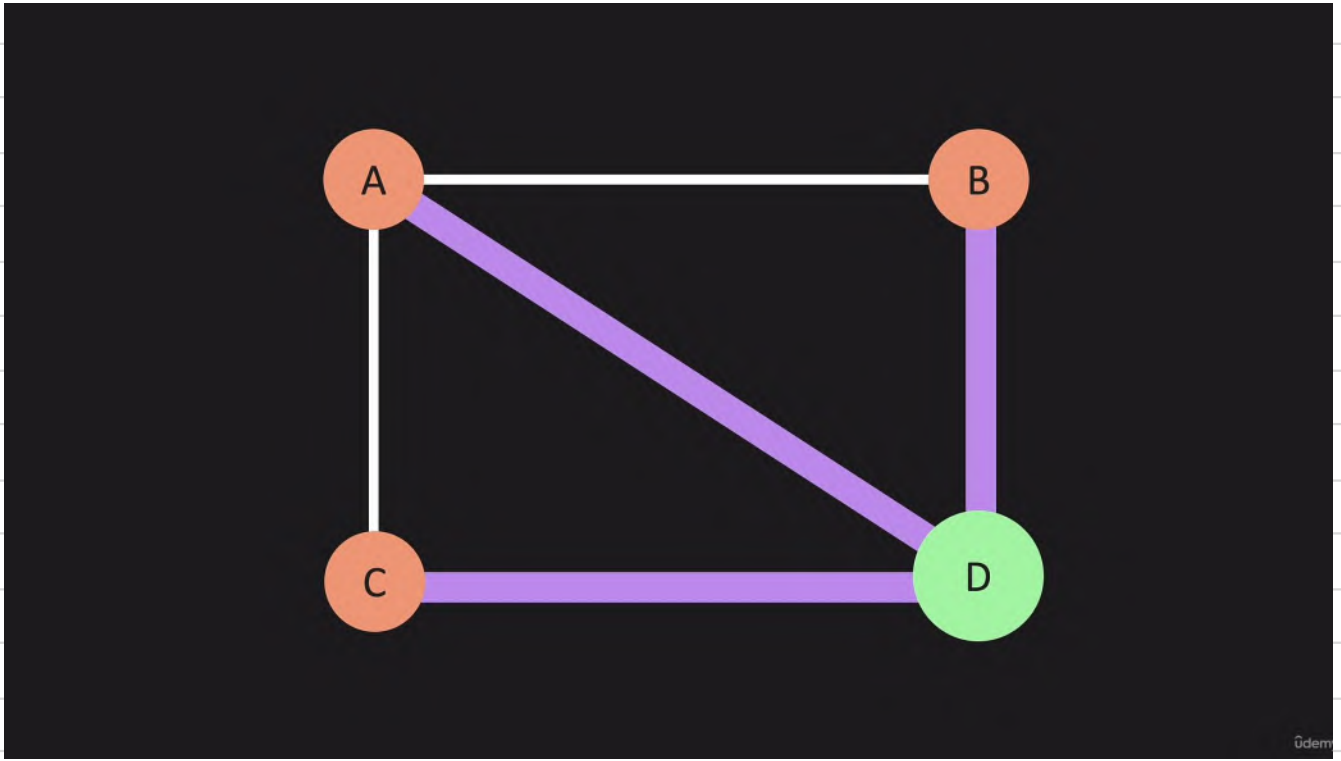
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Kita akan menghapus vertex D

Remove Vertex

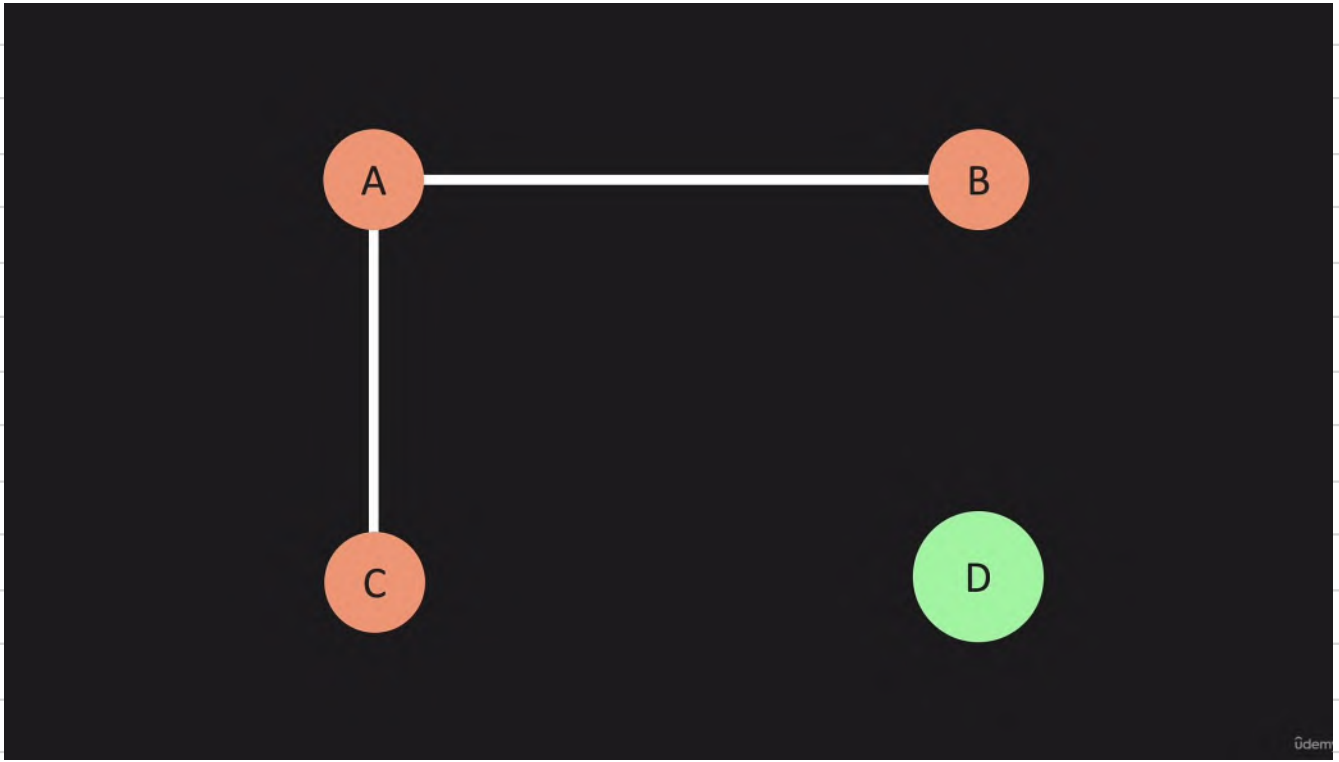
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Secara logika, seharusnya edge yang tersambung ke vertex D harus dihapuskan terlebih dahulu

Remove Vertex

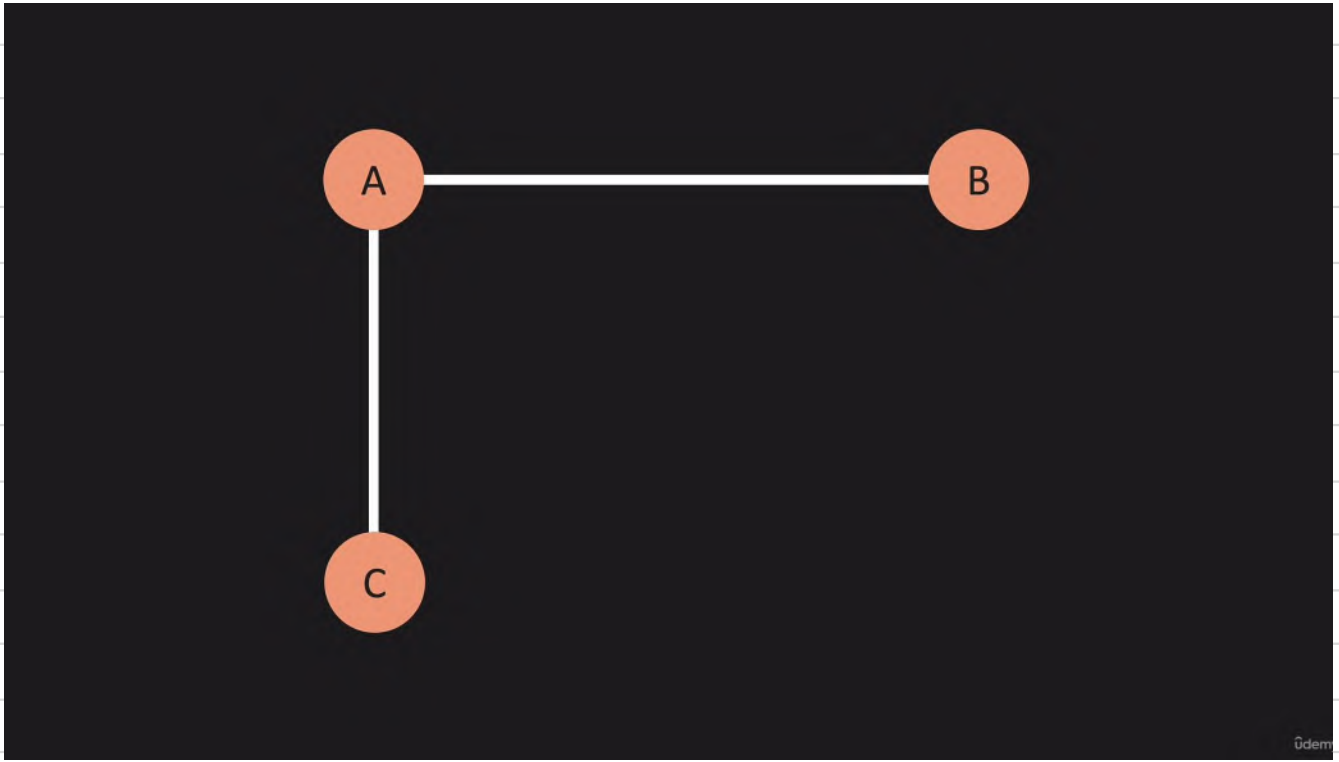
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Secara logika, seharusnya edge yang tersambung ke vertex D harus dihapuskan terlebih dahulu

Remove Vertex

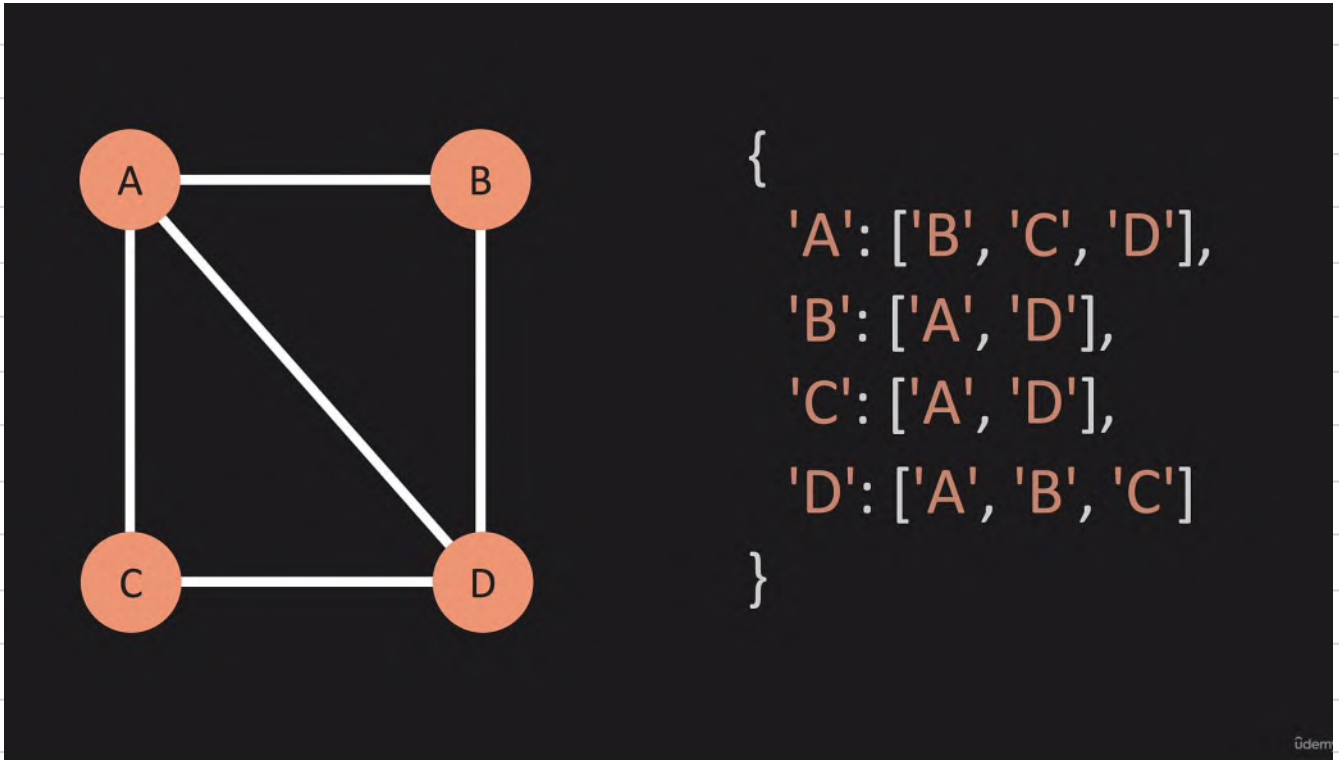
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Setelah itu baru kita bisa menghapus vertex D

Remove Vertex

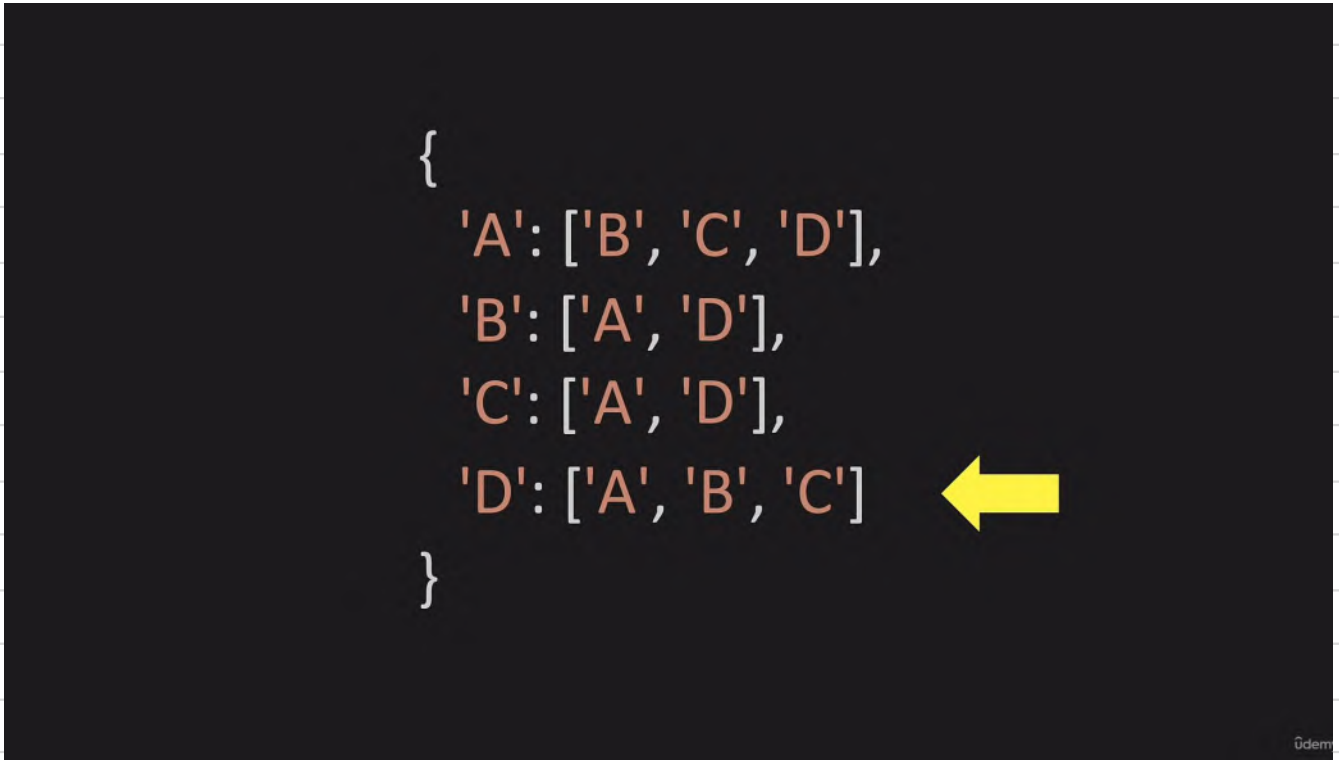
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Secara logika, graph kita diwujudkan menggunakan adj list seperti gambar di kanan

Remove Vertex

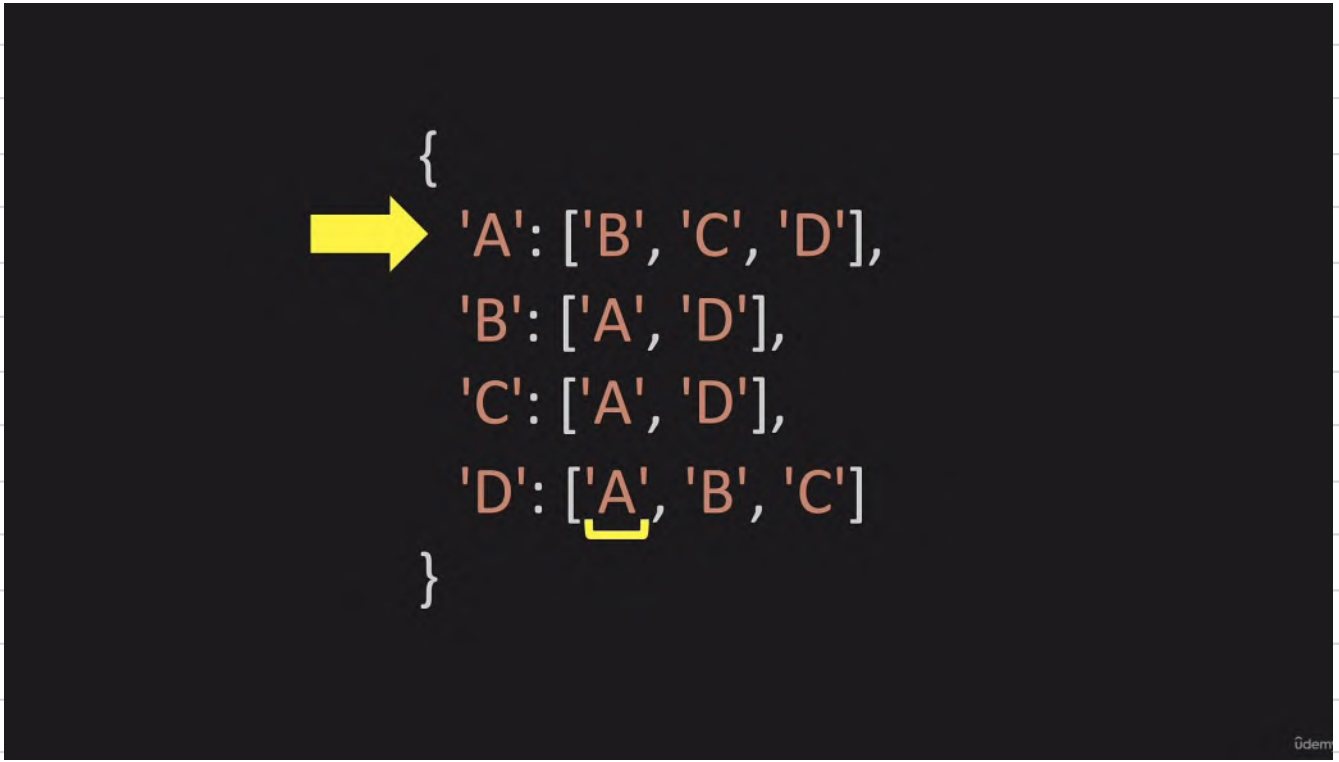
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Untuk dapat menghapus vertex D kita perlu melihat seluruh vertex yang terhubung dengan D

Remove Vertex

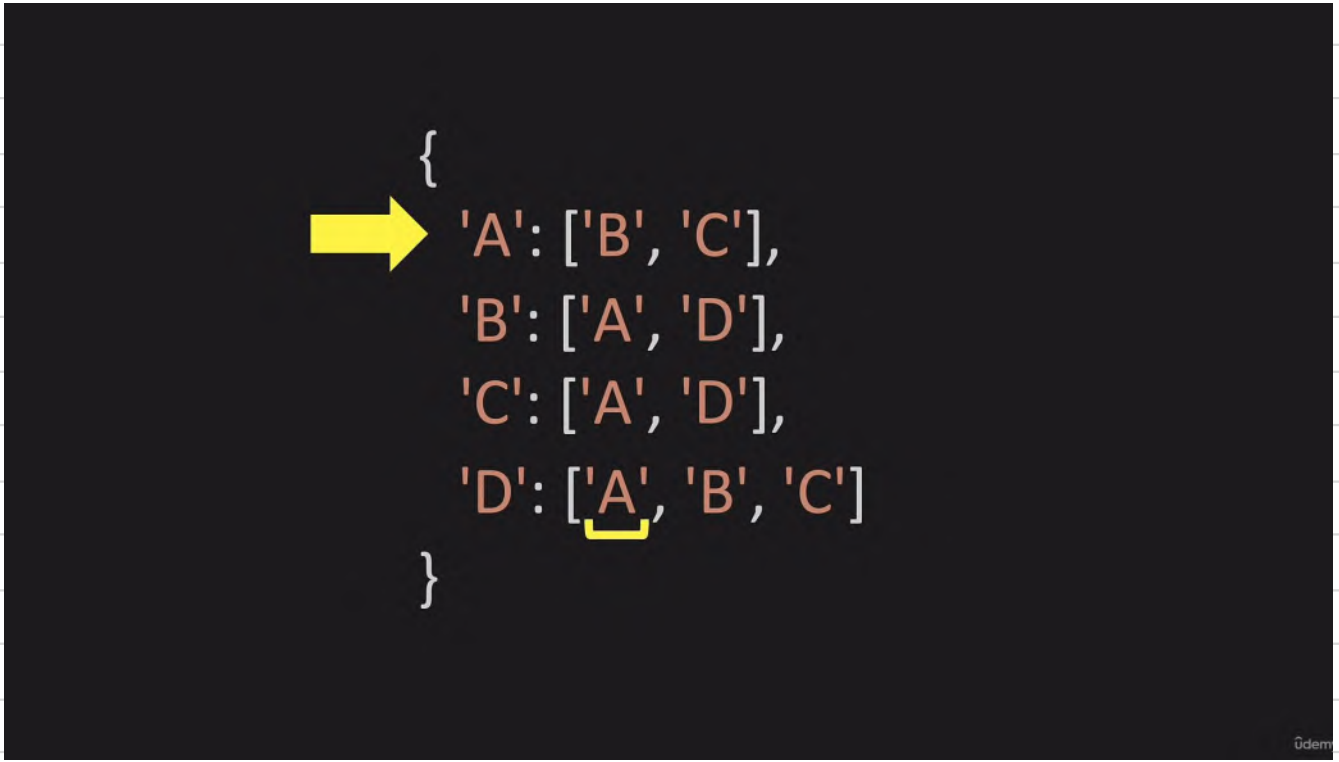
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Kemudian kita mengunjungi setiap vertex yang terhubung dengan D satu per satu mulai dari yg paling awal

Remove Vertex

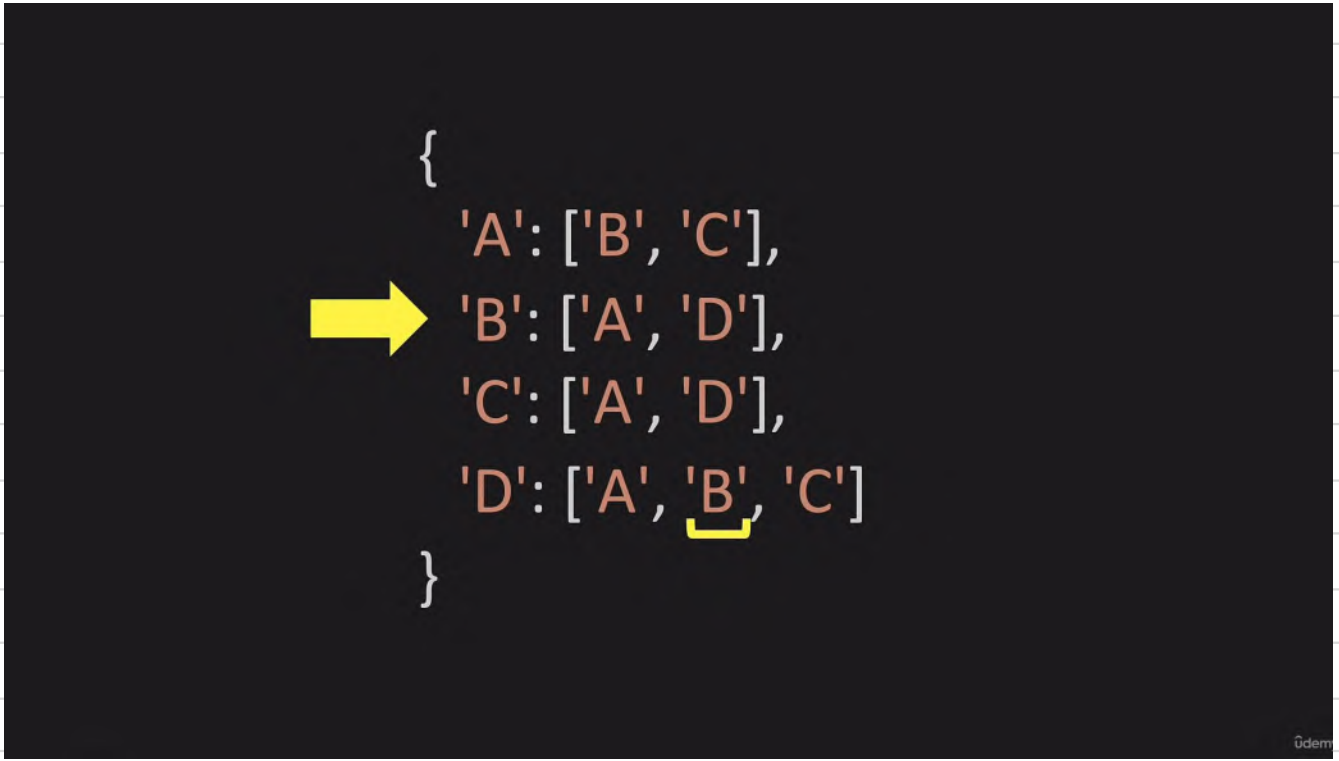
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Di setiap kita mengunjungi vertex yang terhubung itu, kita akan menghapus vertex D pada adj list vertex tersebut

Remove Vertex

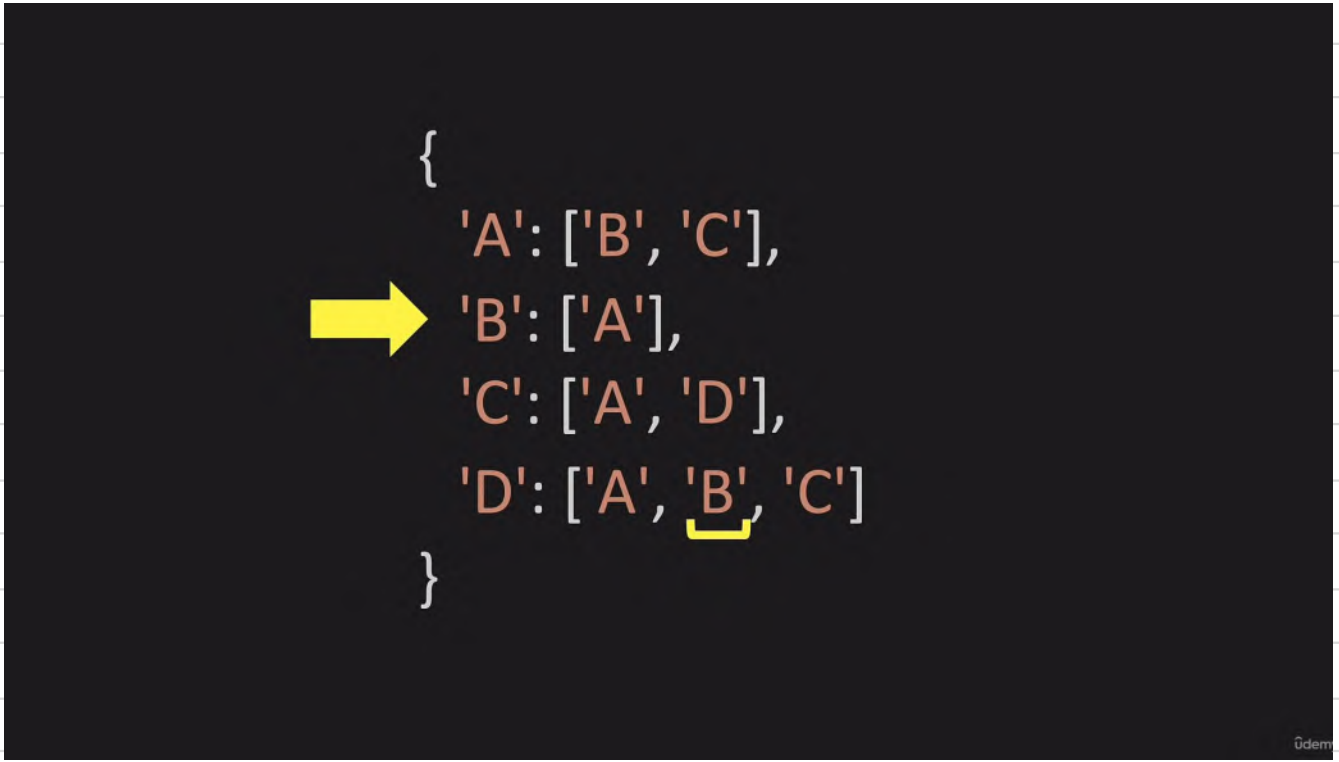
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Begitu juga pada vertex yang terhubung berikutnya

Remove Vertex

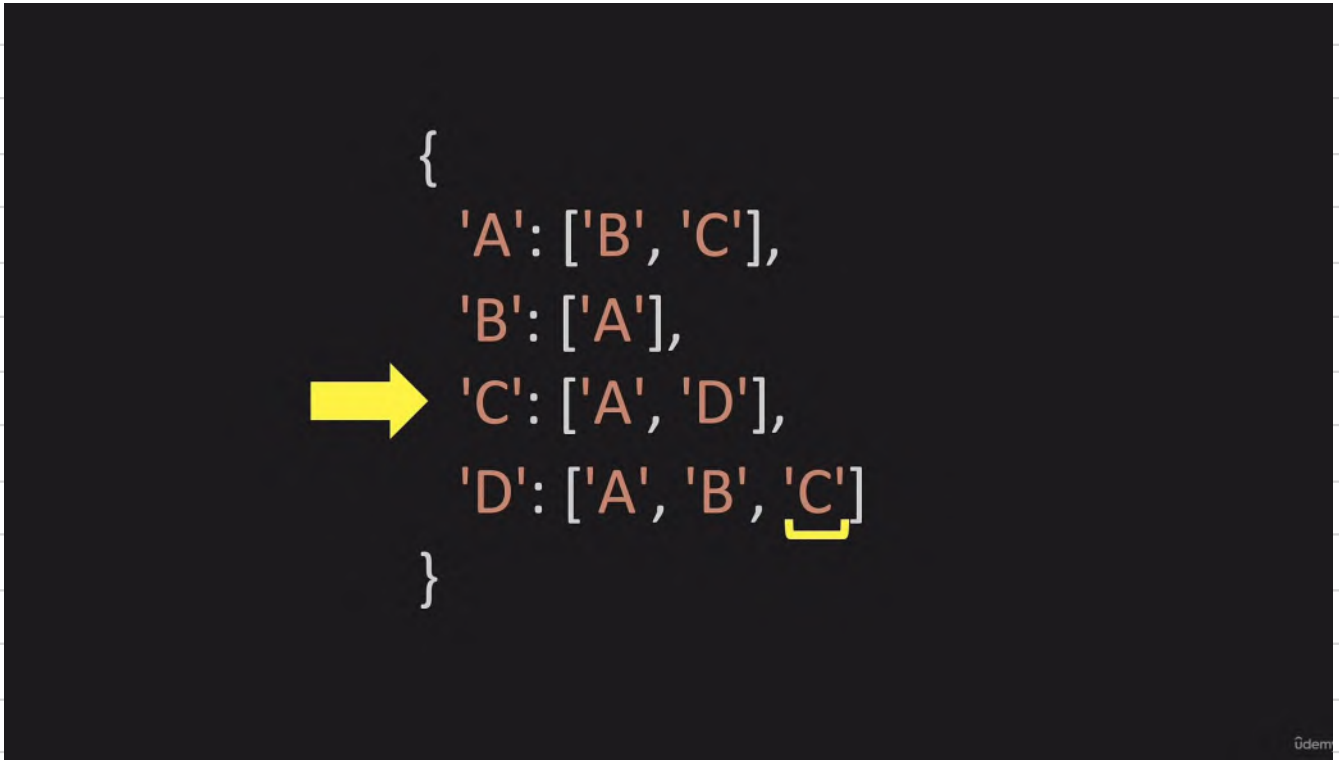
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Telusuri satu per satu isi adj list vertex tersebut untuk menemukan vertex D dan hapus dari adj list tersebut

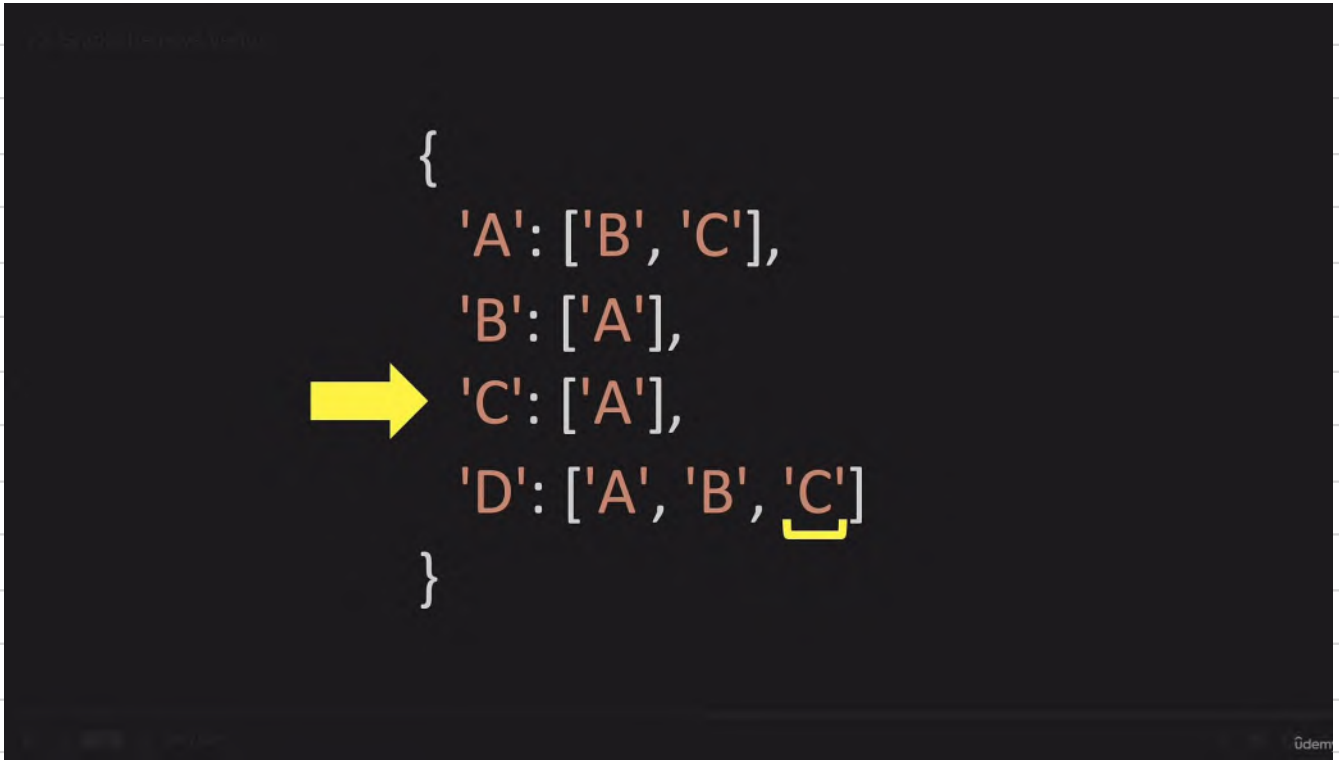
Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Remove Vertex

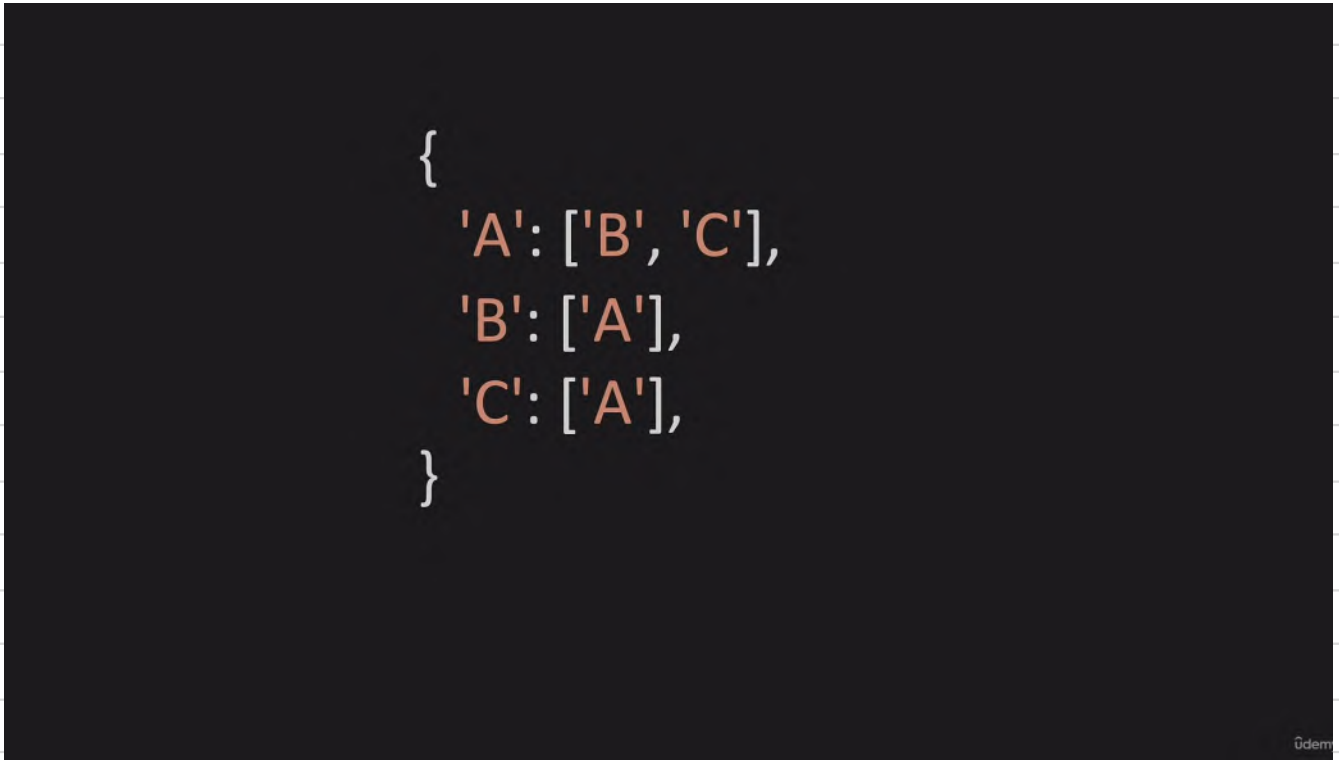
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Setelah seluruh sambungan ke vertex D dihapus, langkah terakhir adalah menghapus vertex D itu sendiri

Remove Vertex

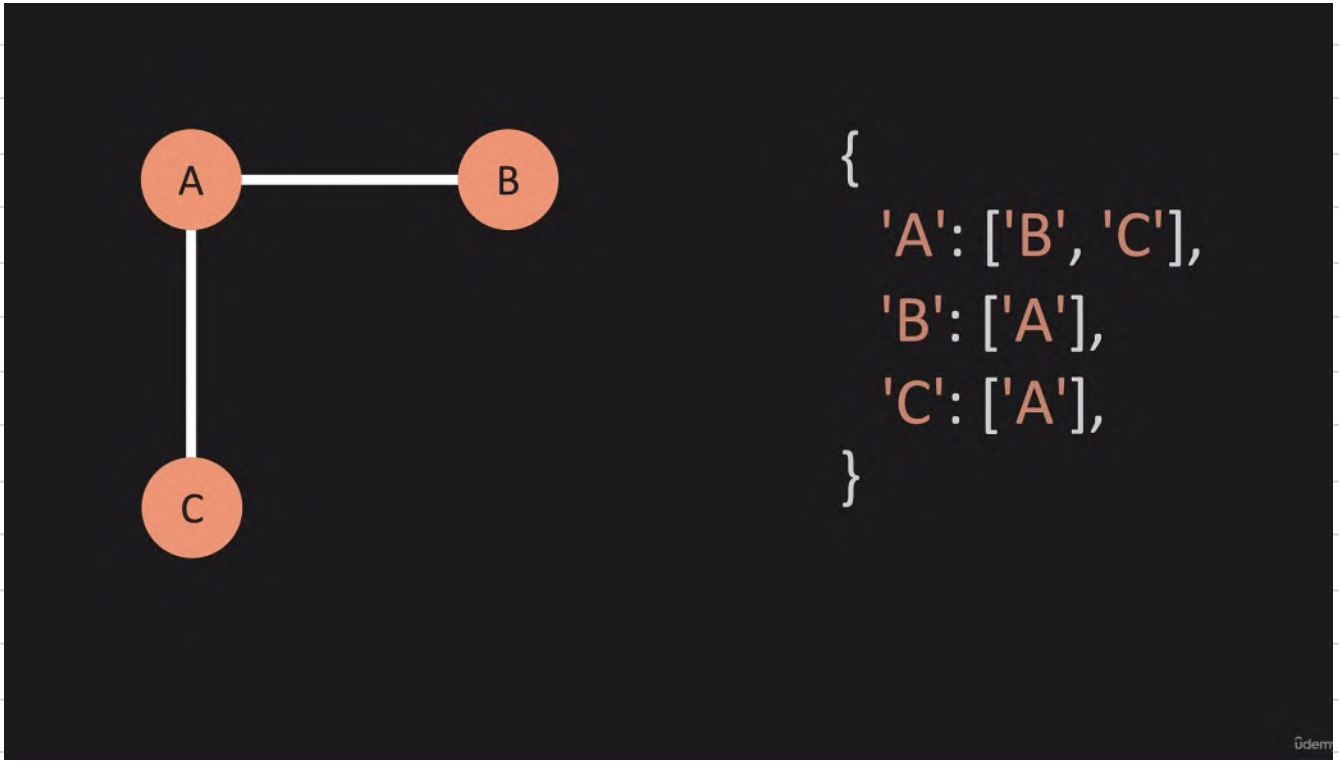
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Pada akhirnya Graph kita menyisakan adj list seperti pada bagian atas

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Jika digambarkan maka Graph kita sekarang sudah tidak memiliki Vertex D

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
```

Kita mulai dengan membuat kerangka remove_vertex dimana parameter vertex adalah vertex yang akan dihapus

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
    if vertex in self.adj_list.keys():
```

Lakukan pemeriksaan jika vertex yang akan dihapus sudah menjadi key di dalam adj list

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
    if vertex in self.adj_list.keys():
        for other_vertex in self.adj_list[vertex]:
```

Lakukan perulangan untuk mengunjungi seluruh vertex yang tersambung dengan vertex yang akan dihapus

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
    if vertex in self.adj_list.keys():
        for other_vertex in self.adj_list[vertex]:
            self.adj_list[other_vertex].remove(vertex)
```

Pada setiap vertex yang dikunjungi saat pencarian, cari vertex yang akan dihapus dan hapus dari adj_list

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
    if vertex in self.adj_list.keys():
        for other_vertex in self.adj_list[vertex]:
            self.adj_list[other_vertex].remove(vertex)
        del self.adj_list[vertex]
        return True
```

Setelah selesai mengunjungi seluruh vertex yang terhubung dengan vertex yang akan dihapus, hapus vertex sesuai parameter menggunakan fungsi del dan return True

Remove Vertex

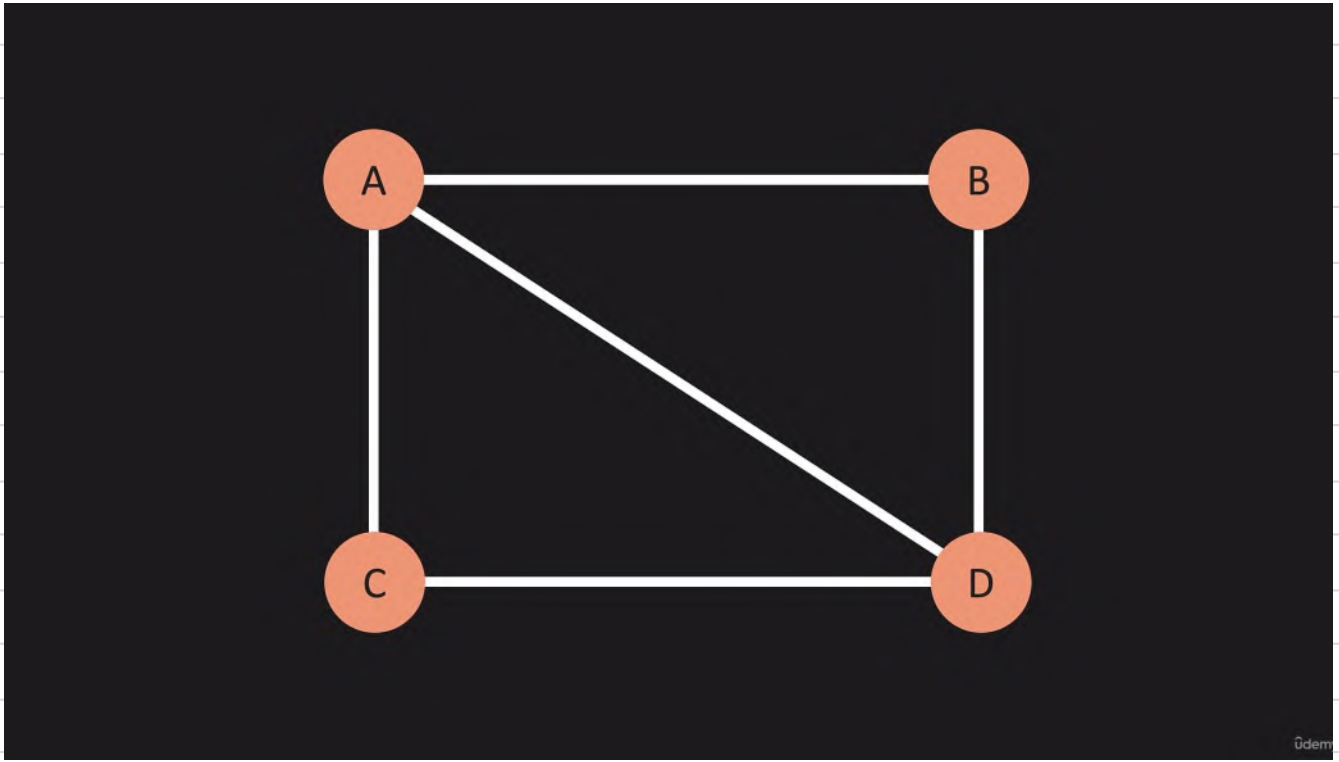
Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
def remove_vertex(self, vertex):
    if vertex in self.adj_list.keys():
        for other_vertex in self.adj_list[vertex]:
            self.adj_list[other_vertex].remove(vertex)
        del self.adj_list[vertex]
        return True
    return False
```

Return kan False jika vertex yang akan di cari belum ada di dalam Adj List

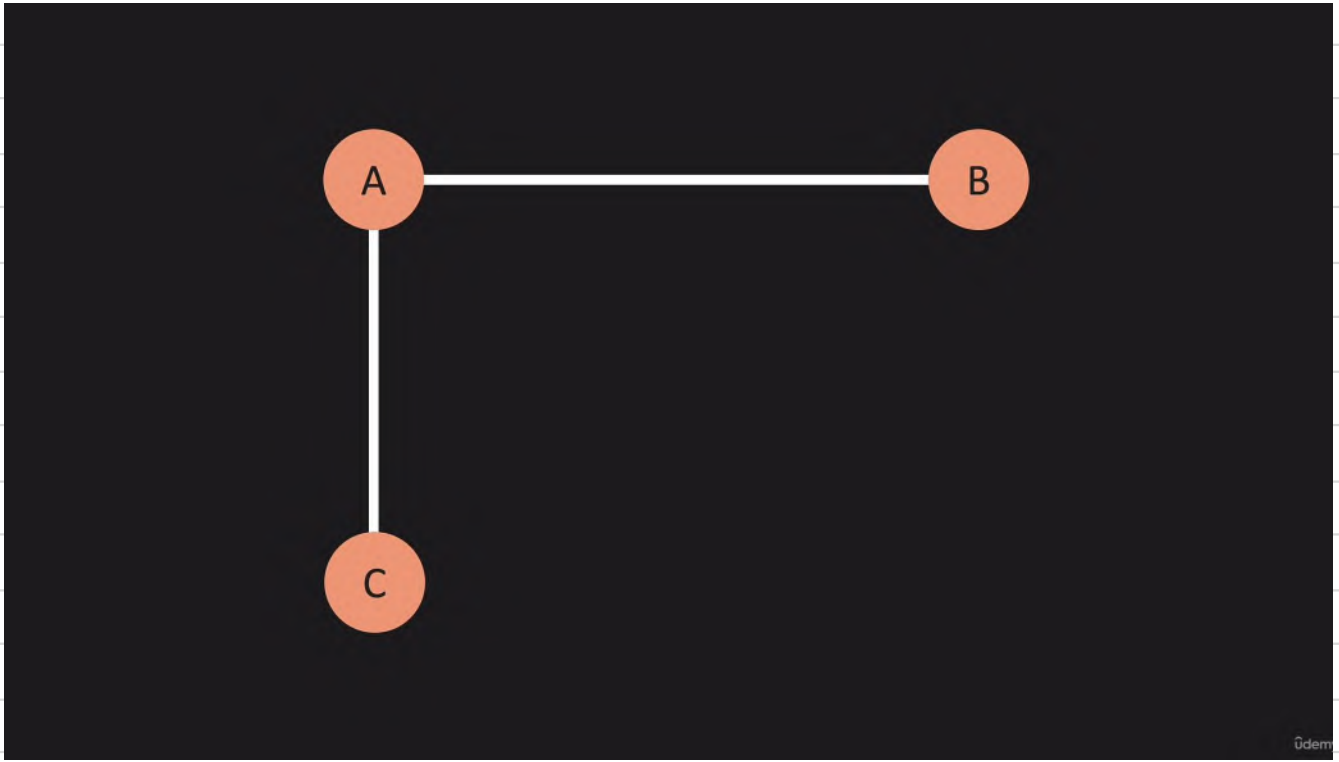
Remove Vertex
Skenario Percobaan

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



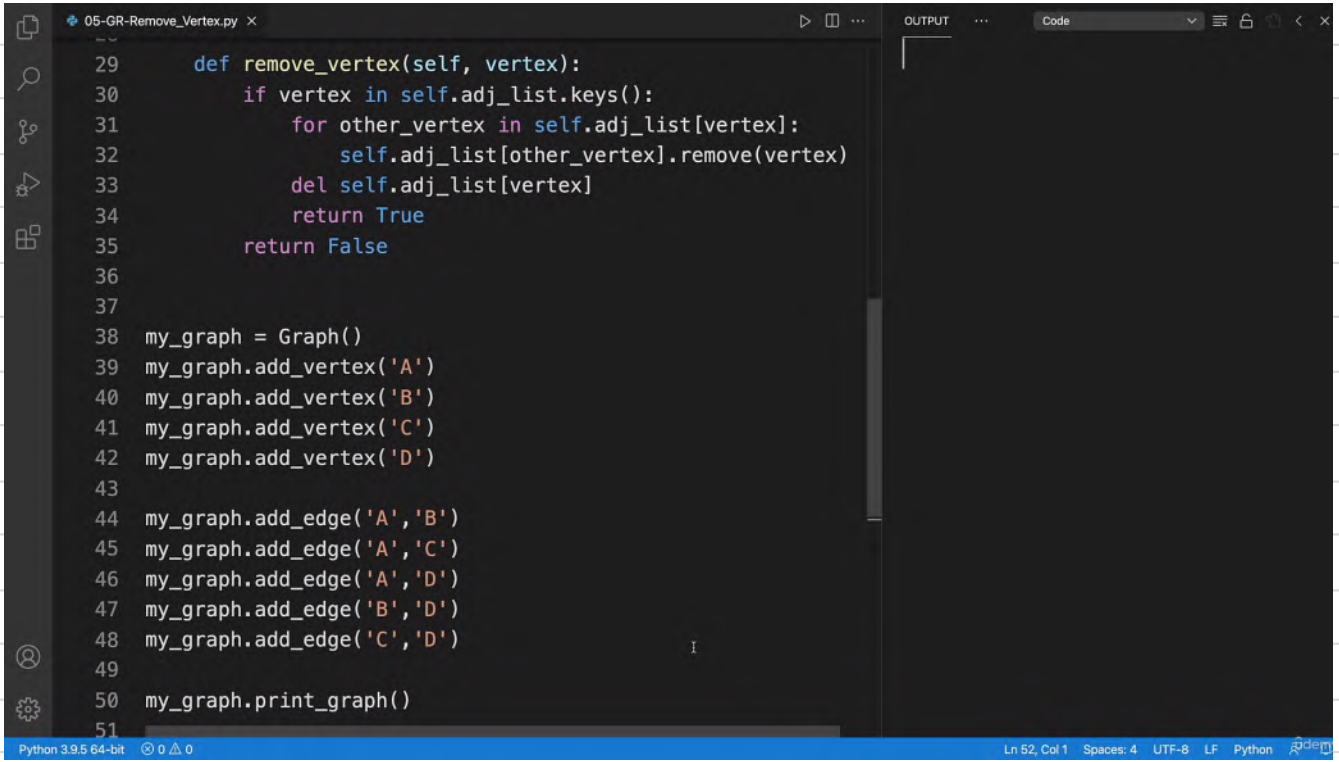
Remove Vertex
Skenario Percobaan

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex



Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
my_graph = Graph()
my_graph.add_vertex('A')
my_graph.add_vertex('B')
my_graph.add_vertex('C')
my_graph.add_vertex('D')

my_graph.add_edge('A','B')
my_graph.add_edge('A','C')
my_graph.add_edge('A','D')
my_graph.add_edge('B','D')
my_graph.add_edge('C','D')

my_graph.print_graph()
```

```
A : ['B', 'C', 'D']
B : ['A', 'D']
C : ['A', 'D']
D : ['A', 'B', 'C']
```

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
my_graph = Graph()
my_graph.add_vertex('A')
my_graph.add_vertex('B')
my_graph.add_vertex('C')
my_graph.add_vertex('D')

my_graph.add_edge('A','B')
my_graph.add_edge('A','C')
my_graph.add_edge('A','D')
my_graph.add_edge('B','D')
my_graph.add_edge('C','D')

my_graph.remove_vertex('D')

my_graph.print_graph()
```

```
A : ['B', 'C', 'D']
B : ['A', 'D']
C : ['A', 'D']
D : ['A', 'B', 'C']
```

Remove Vertex

Misal kita memiliki 4 vertex yang terhubung dan kita akan menghapus 1 vertex

```
my_graph = Graph()
my_graph.add_vertex('A')
my_graph.add_vertex('B')
my_graph.add_vertex('C')
my_graph.add_vertex('D')

my_graph.add_edge('A','B')
my_graph.add_edge('A','C')
my_graph.add_edge('A','D')
my_graph.add_edge('B','D')
my_graph.add_edge('C','D')

my_graph.remove_vertex('D')

my_graph.print_graph()
```

```
A : ['B', 'C']
B : ['A']
C : ['A']
```

Terima Kasih

Ada **Pertanyaan?**