



Presentasi
Djikstra ALGORIHM

Oleh: Syahrul Akbar Ramdhani (11230940000027)
Dosen Pengampu: M. Irwan Septiar Musti, M.si

Djikstra

Tujuan dan Aplikasi

1. Menemukan jalur terpendek atau termurah dari satu simpul asal ke semua simpul lain dalam graf berbobot positif.
2. Algoritma Dijkstra digunakan dalam berbagai bidang, seperti:
 - Routing jaringan komputer, untuk menentukan rute tercepat pengiriman data.
 - Distribusi logistik, guna mengoptimalkan rute pengiriman barang.
 - Simulasi penyebaran penyakit, untuk memodelkan jalur infeksi dalam suatu wilayah.
 - Jaringan sosial, dalam menganalisis kedekatan atau pengaruh antar pengguna.

Konsep Dasar Djikstra

Djiksra

- Algoritma Djikstra bekerja pada graf berbobot dengan bobot non-negatif.
- Jalur terpendek di sini berarti total bobot terkecil, bukan jumlah simpul yang paling sedikit.
- Djikstra menggunakan pendekatan greedy, yaitu selalu memilih simpul dengan jarak (bobot total) terkecil yang belum dikunjungi pada setiap langkah.

Struktur data yang digunakan

Djikstra

Struktur Data yang Digunakan

- dist[]: jarak minimum sementara dari sumber
- pred[]: pelacak simpul sebelumnya (seperti BFS)
- vSet atau visited[]: mencatat simpul yang sudah dikunjungi
- Priority queue (min-heap): untuk efisiens

Langkah-langkah Algoritma

Djikstra

1. Inisialisasi $dist[v] = \infty$ untuk semua simpul, dan $dist[src] = 0$
2. Inisialisasi $pred[v] = -1$ dan masukkan semua simpul ke priority queue
3. Selama priority queue tidak kosong:
 - Ambil simpul v dengan $dist[v]$ terkecil
 - Untuk setiap tetangga w dari v :
 - Lakukan edge relaxation: update $dist[w]$ jika jalur melalui v lebih pendek
 - Update $pred[w]$ ke v
 - Tandai v sebagai visited

- $dist[]$ menyimpan jarak terpendek sementara dari simpul asal ke setiap simpul lain.
- $pred[]$ mencatat simpul sebelumnya, berguna untuk merekonstruksi jalur terpendek.
- Priority queue (min-heap) membantu memilih simpul dengan jarak terkecil secara efisien.
- Edge relaxation adalah proses membandingkan dan memperbarui $dist[]$ & $pred[]$ jika ditemukan jalur yang lebih pendek.

Djikstra

Edge Relaxation

Jika $\text{dist}[v] + \text{weight}(v,w) < \text{dist}[w]$, maka:

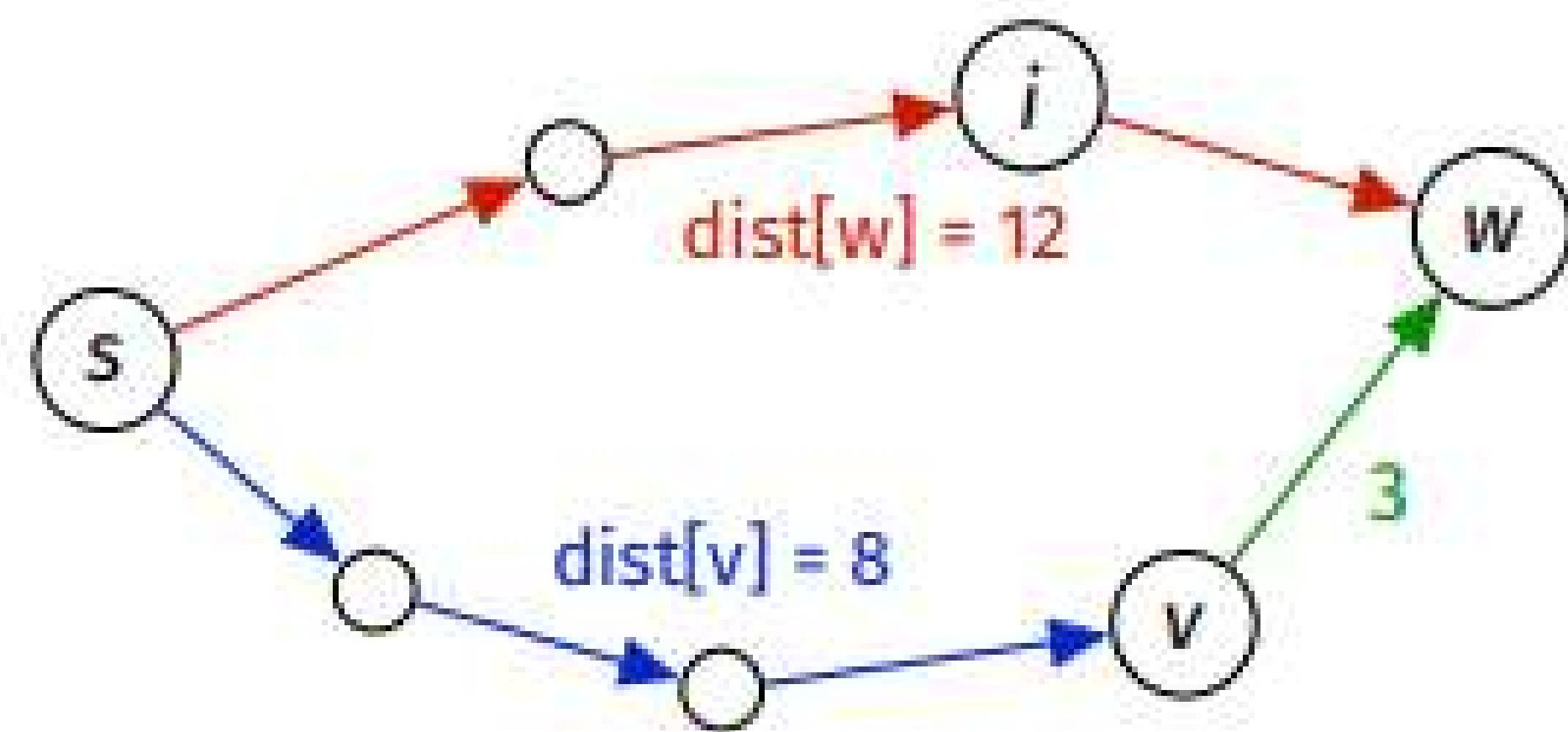
- $\text{dist}[w] = \text{dist}[v] + \text{weight}(v,w)$
- $\text{pred}[w] = v$

Inti dari proses update jarak selama algoritma berlangsung

Djikstra

Contoh Edge Relaxation

Before relaxation along (v, w, 3)

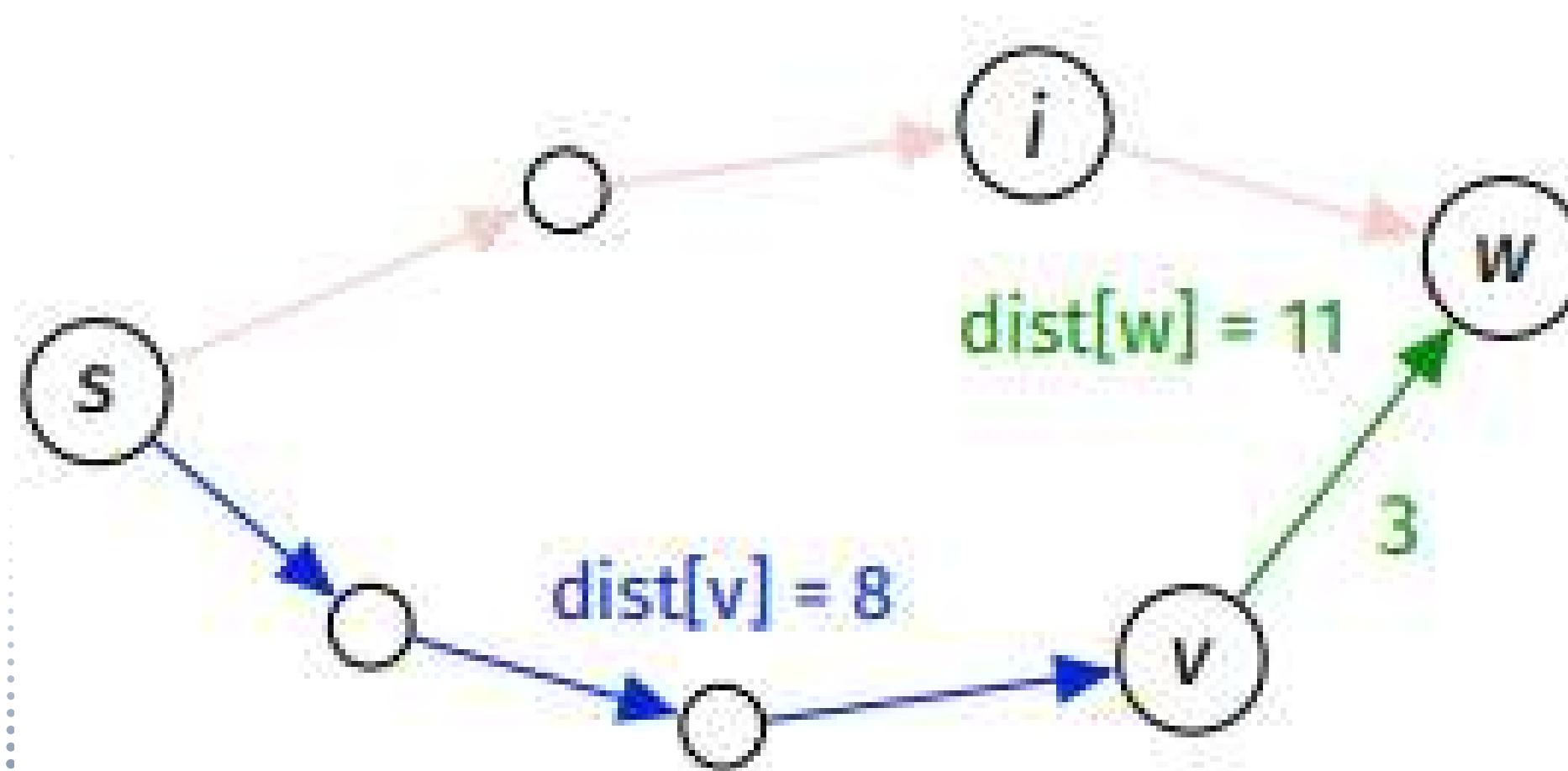


	...	[v]	...	[w]
dist	...	8	...	12
pred	i

Djikstra

Contoh Edge Relaxation

After relaxation along $(v, w, 3)$



	...	[v]	...	[w]
dist	...	8	...	12
pred	w

Pseudo Code

Input:

$G = (V, E)$ → graf berbobot positif

src → simpul asal

Output:

$dist[v]$ → jarak terpendek dari src ke v

$pred[v]$ → simpul sebelumnya dalam jalur terpendek ke v

1. for setiap simpul $v \in V$:

$dist[v] \leftarrow \infty$

$pred[v] \leftarrow \text{undefined}$

2. $dist[src] \leftarrow 0$

3. Buat priority queue Q berisi semua simpul V ,
dengan prioritas berdasarkan $dist[v]$

4. while Q tidak kosong:

$v \leftarrow$ simpul dalam Q dengan $dist[v]$ terkecil
hapus v dari Q

for setiap tetangga w dari v :

if $dist[v] + \text{weight}(v, w) < dist[w]$:

$dist[w] \leftarrow dist[v] + \text{weight}(v, w)$

$pred[w] \leftarrow v$

perbarui prioritas w dalam Q

Djikstra

Implementasi di Python

Djikstra

```
import heapq

def djikstra(graph, src):
    dist = {v: float('inf') for v in graph}
    pred = {v: None for v in graph}
    dist[src] = 0

    # Priority queue: (jarak, simpul)
    queue = [(0, src)]

    while queue:
        current_dist, v = heapq.heappop(queue)

        # Lewati simpul jika ditemukan jarak yang lebih pendek sebelumnya
        if current_dist > dist[v]:
            continue

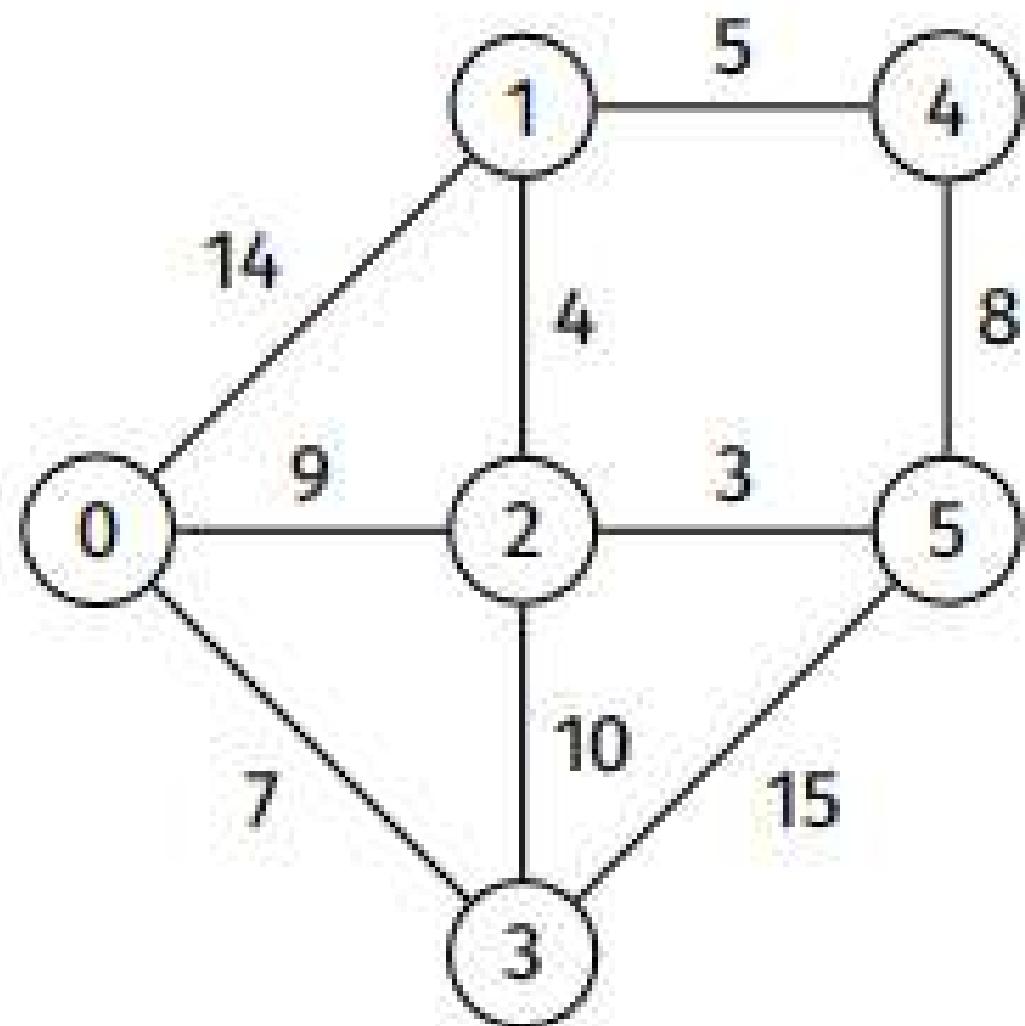
        for neighbor, weight in graph[v]:
            alt = dist[v] + weight
            if alt < dist[neighbor]:
                dist[neighbor] = alt
                pred[neighbor] = v
                heapq.heappush(queue, (alt, neighbor))

    return dist, pred
```

Dijkstra

Example

Dijkstra's algorithm starting at 0

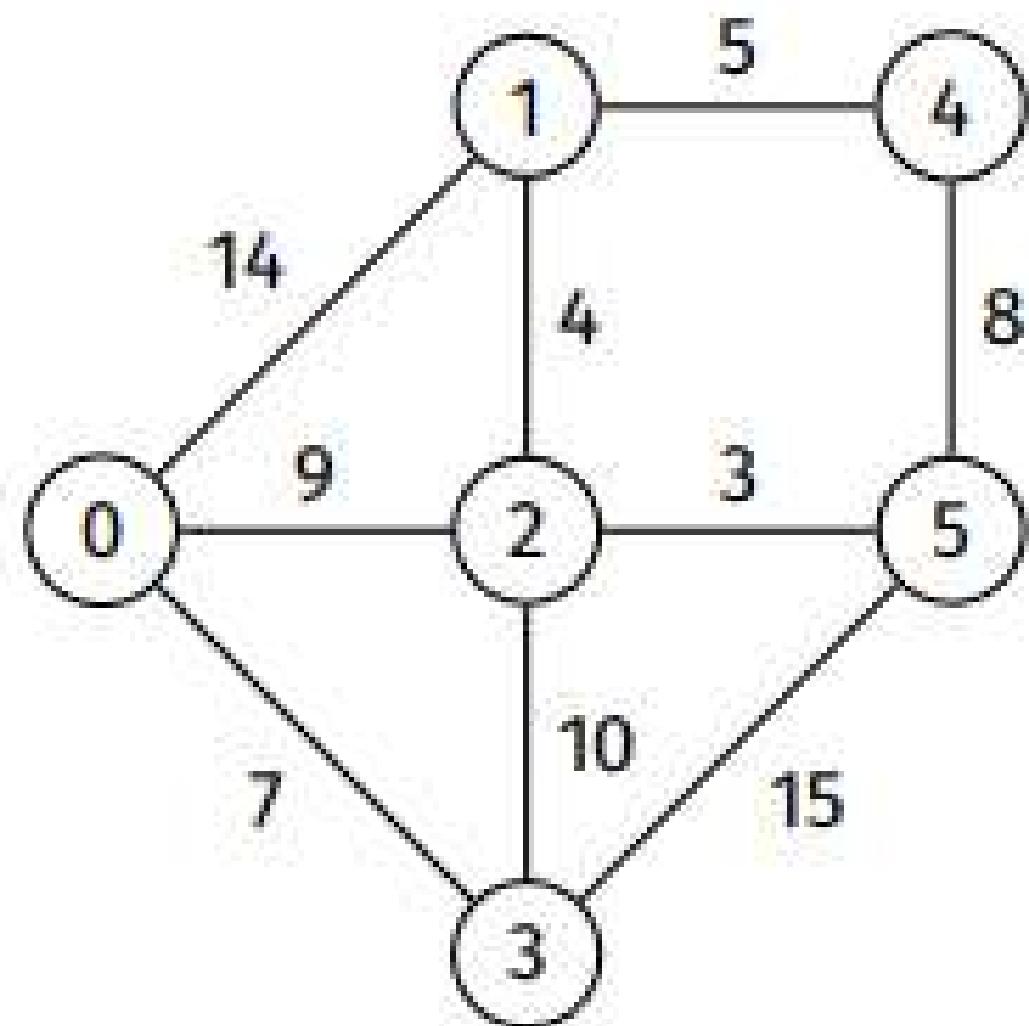


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	∞	∞	∞	∞	∞
pred	-1	-1	-1	-1	-1	-1

Dijkstra

Example

After first iteration ($v = 0$)

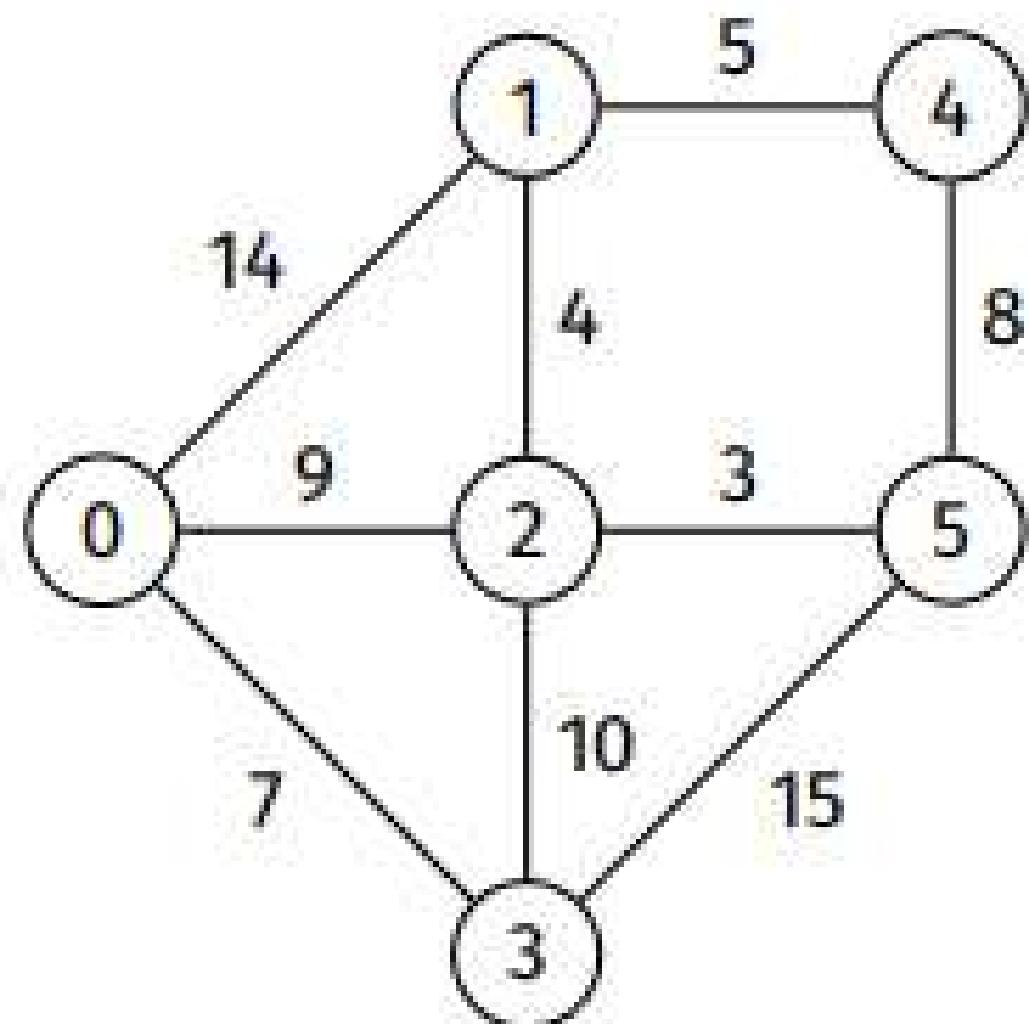


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	14	9	7	∞	∞
pred	-1	0	0	0	-1	-1

Dijkstra

Example

After first iteration ($v = 3$)

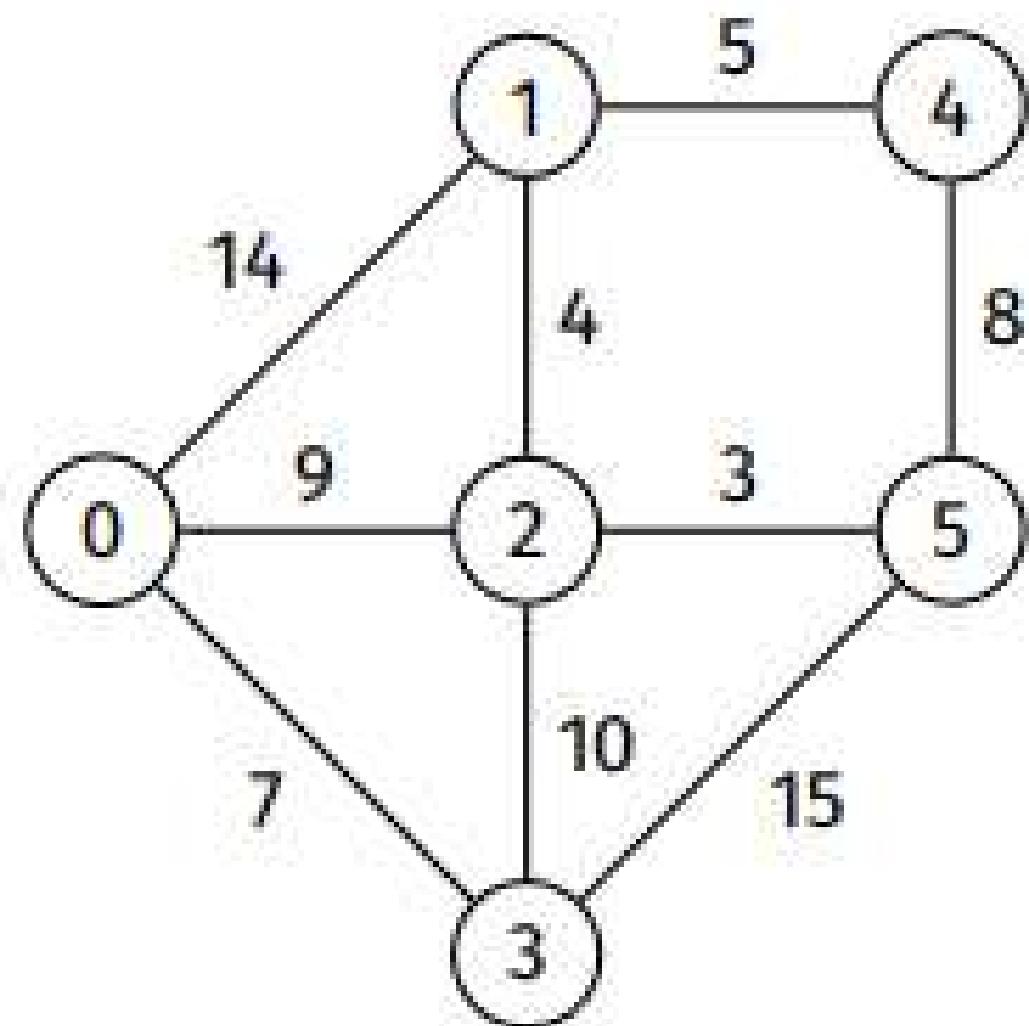


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	14	9	7	∞	22
pred	-1	0	0	0	-1	3

Dijkstra

Example

After first iteration ($v = 2$)

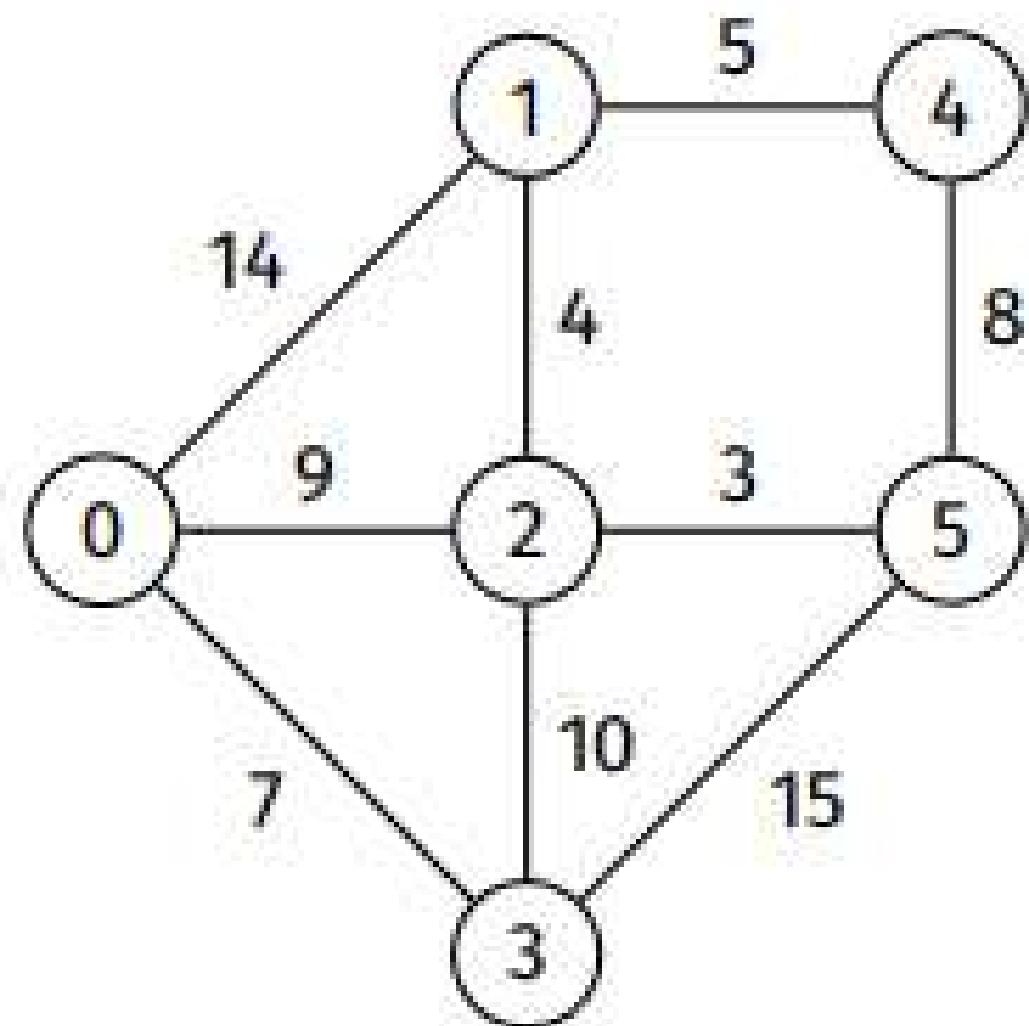


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	∞	12
pred	-1	2	0	0	-1	2

Dijkstra

Example

After first iteration ($v = 5$)

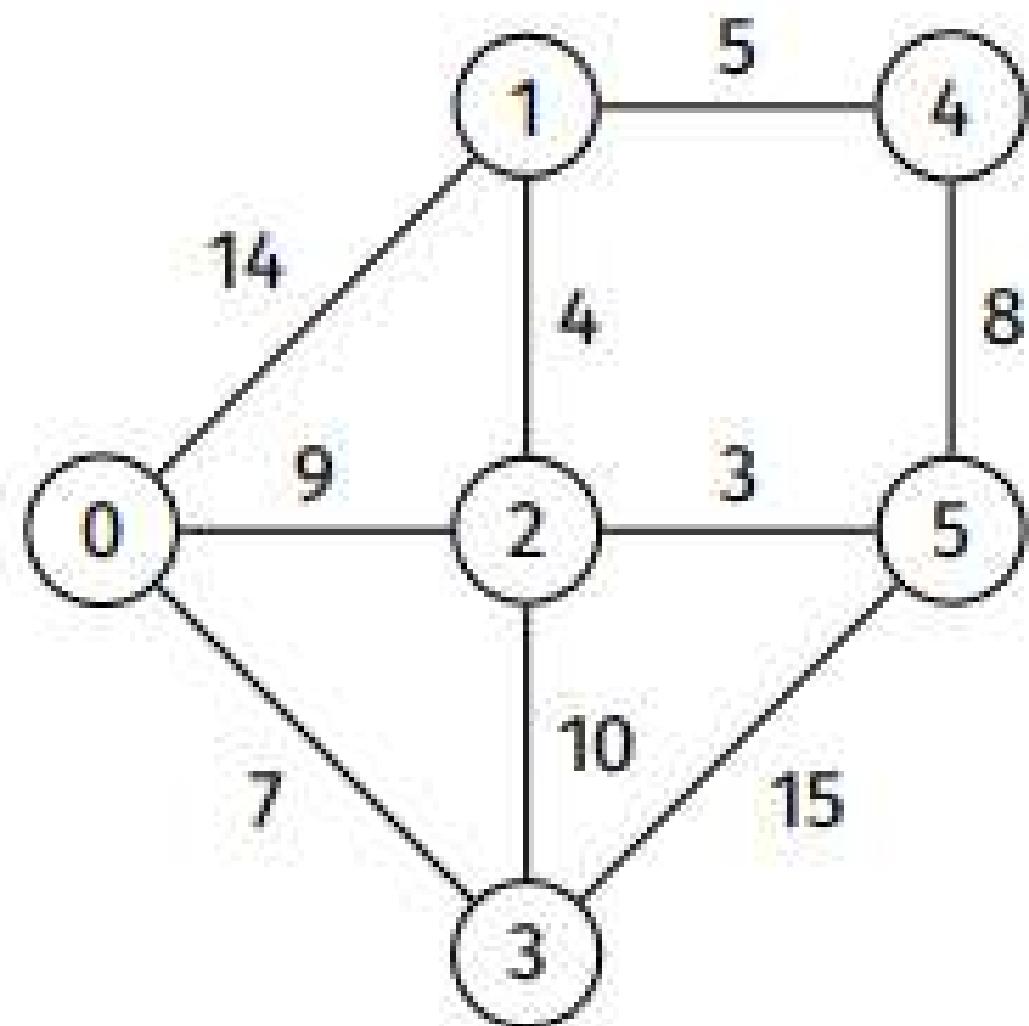


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	20	12
pred	-1	2	0	0	5	2

Dijkstra

Example

After first iteration ($v = 1$)

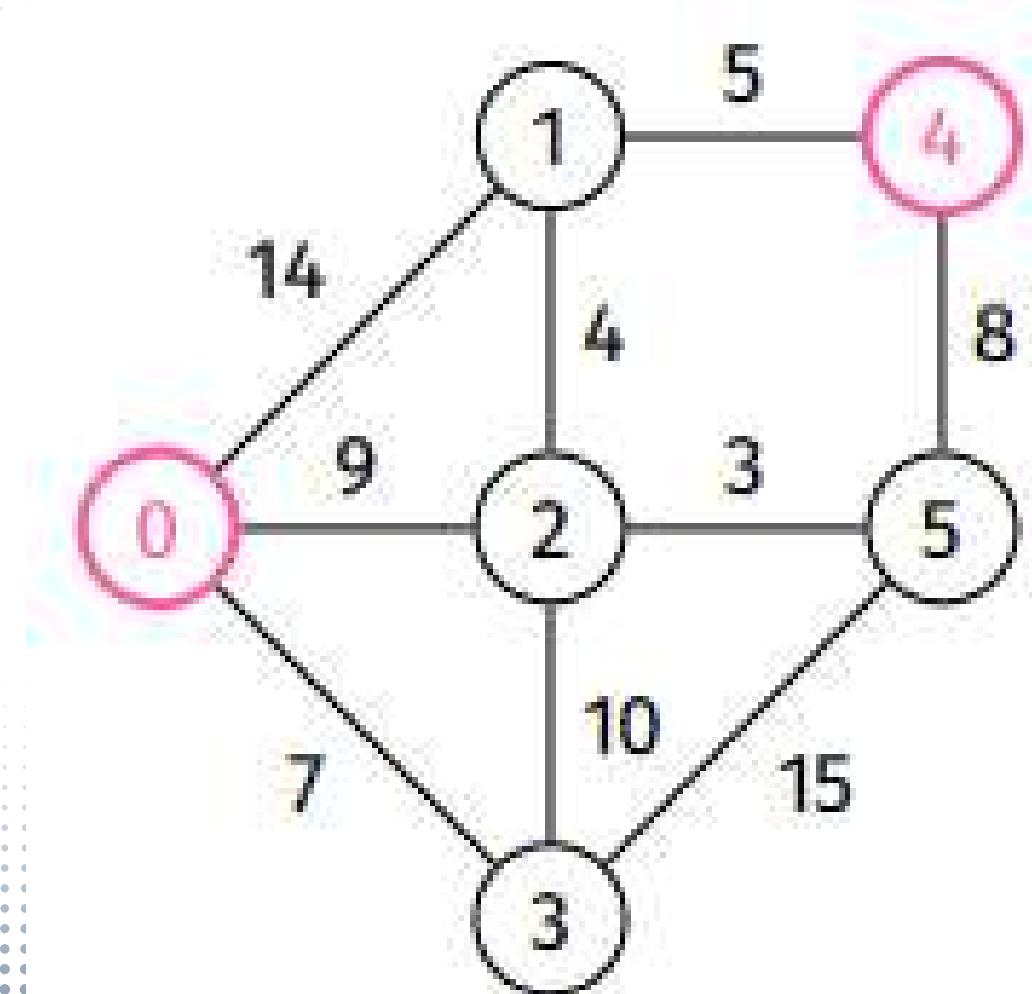


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	18	12
pred	-1	2	0	0	1	2

Dijkstra

Finding Path

Example: Shortest path from 0 to 4

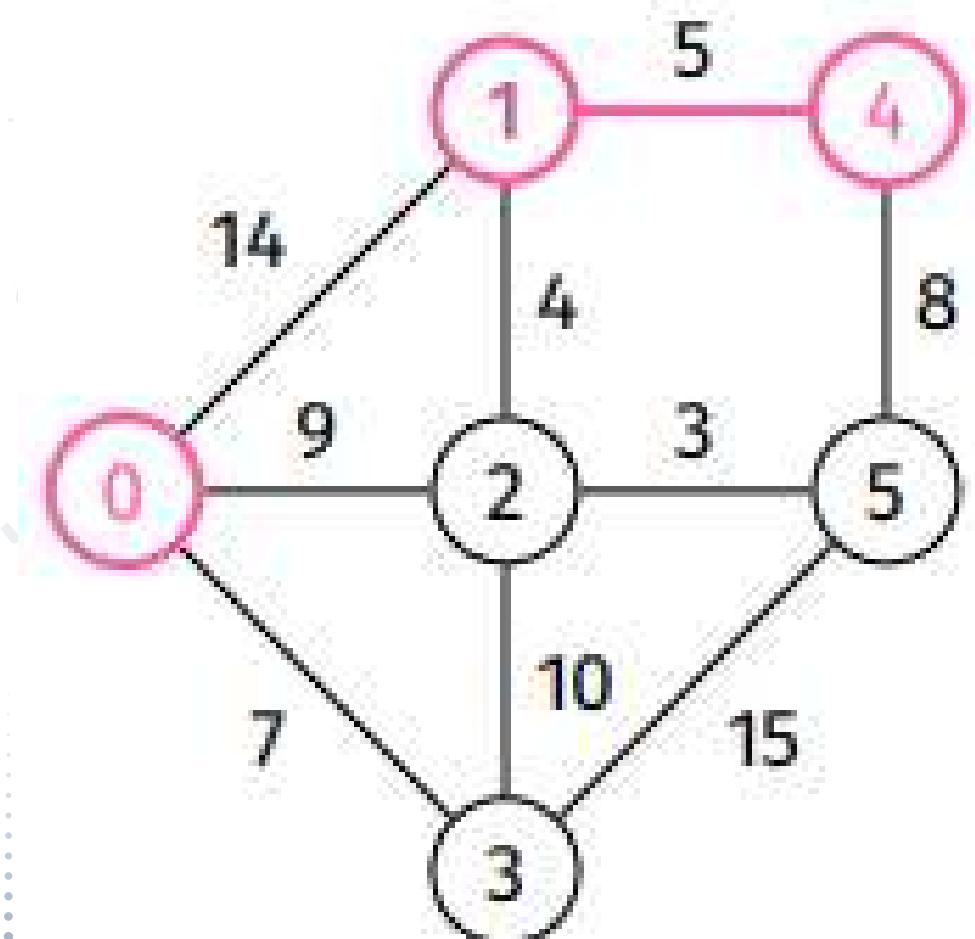


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	18	12
pred	-1	2	0	0	1	2

Dijkstra

Finding Path

Example: Shortest path from 0 to 4

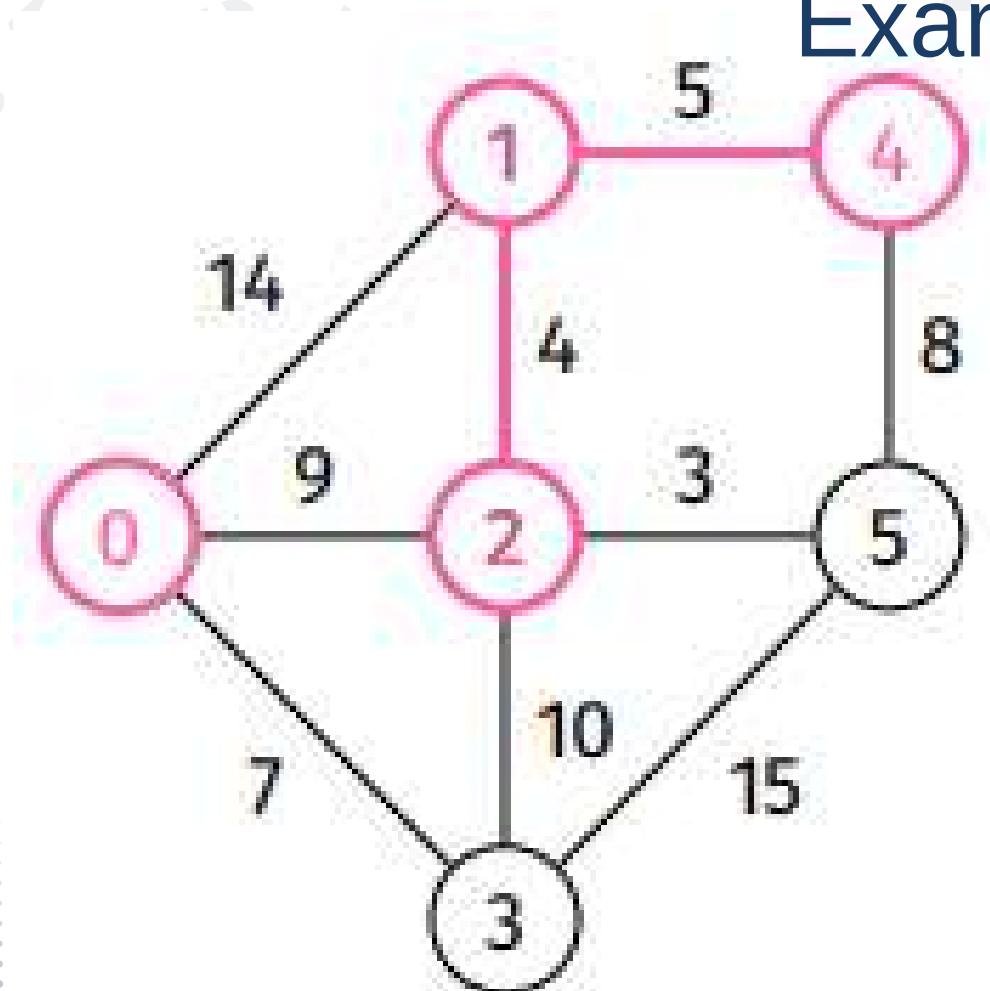


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	18	12
pred	-1	2	0	0	1	2

Dijkstra

Finding Path

Example: Shortest path from 0 to 4

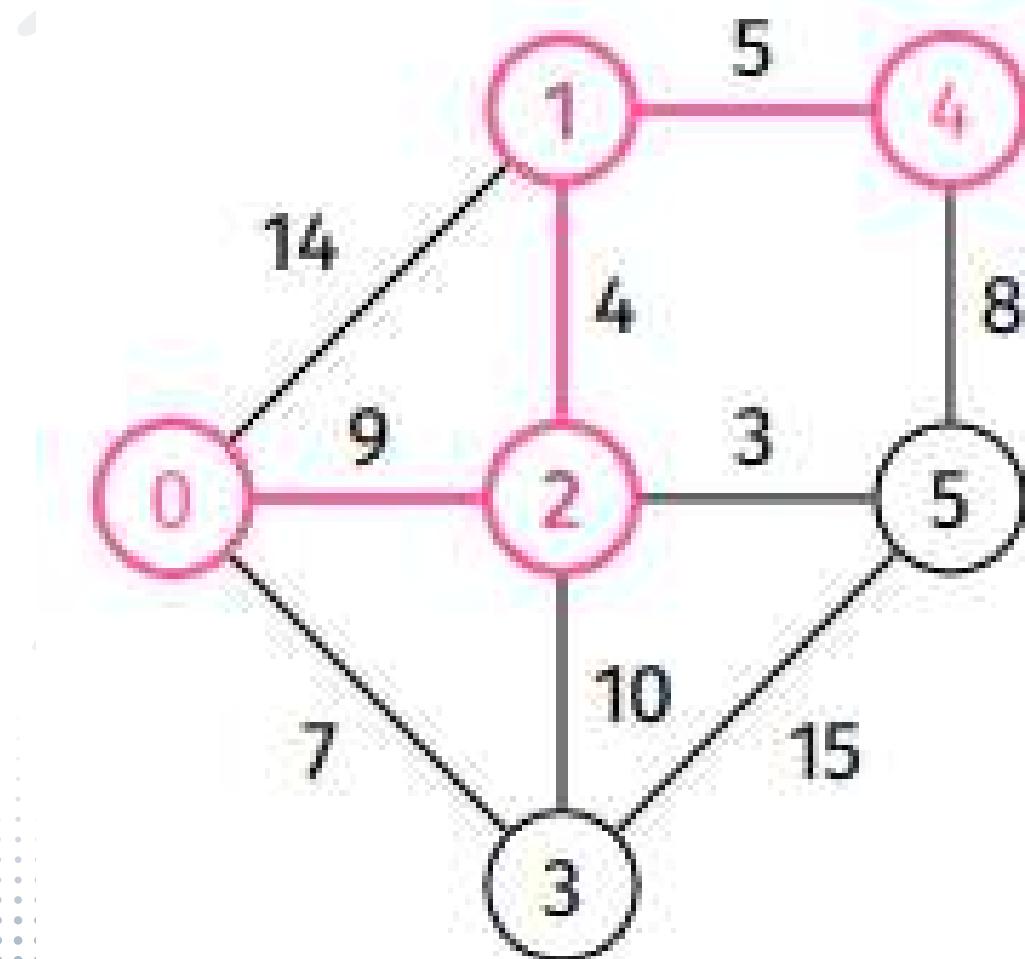


	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	18	12
pred	-1	2	0	0	1	2

Dijkstra

Finding Path

Example: Shortest path from 0 to 4



	[0]	[1]	[2]	[3]	[4]	[5]
dist	0	13	9	7	18	12
pred	-1	2	0	0	1	2

Implementasi vSet

Djikstra

Vissited Array

- mirip dengan visited[] di BFS/DFS
- Gunakan array boolean sebanyak V simpul, inisialisasi ke false.
- Setelah simpul v diproses, tandai visited[v] = true.
- Di tiap iterasi, cari simpul dengan visited = false dan dist terkecil.
- ⏱ Kompleksitas pencarian minimum: O(V)

Array / List Simpul

- Simpan semua simpul dalam array atau linked list.
- Setelah simpul v diproses, hapus v dari array/list.
- Di tiap iterasi, cari simpul di list dengan dist terkecil.
- ⏱ Kompleksitas pencarian minimum: O(V)

Implementasi vSet

Priority Queue / Min-Heap

Djikstra

- Struktur data di mana setiap item punya prioritas.
- Dua operasi utama:
 - Insert(item, priority)
 - Delete() item dengan prioritas tertinggi (jarak terkecil)
- Dalam Djikstra:
 - Simpul disimpan dalam priority queue
 - Prioritas = $\text{dist}[v]$ (semakin kecil, semakin tinggi prioritas)
- Heap akan otomatis mengatur agar simpul dengan dist terkecil selalu di atas.



Terima Kasih