Algoritma Djikstra

Pendahuluan

- Algoritme Dijkstra, (sesuai penemunya Edsger Dijkstra), adalah sebuah algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah (directed graph).
- Algoritma ini dioublikasikan pada tahun 1959 jurnal Numerische Mathematik yang berjudul “A Note on Two Problems in Connexion with Graphs” dan dianggap sebagai algoritma greedy.
- Permasalahan rute terpendek dari sebuah titik ke akhir titik lain adalah sebuah masalah klasik optimasi yang banyak digunakan untuk menguji sebuah algoritma yang diusulkan. Permasalahan rute terpendek dianggap cukup baik untuk mewakili masalah optimisasi, karena permasalahannya mudah dimengerti (hanya menjumlahkan seluruh edge yang dilalui) namun memiliki banyak pilihan solusi.

Pendahuluan

- Menurut Andrew Goldberg peneliti Microsoft Research Silicon Valley, mengatakan ada banyak alasan mengapa peneliti terus mempelajari masalah pencarian jalan terpendek. “Jalan terpendek adalah masalah optimasi yang relevan untuk berbagai macam aplikasi, seperti jaringan routing, game, desain sirkuit, dan pemetaan”.
- Diskripsi matematis untuk grafik dapat diwakili G = {V. E}, yang berarti sebuah grafik (G) didefenisikan oleh satu set simpul (Vertex = V) dan koleksi Edge (E).
- Algoritma Dijkstra bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke n, setidaknya ada n node yang sudah kita tahu jalur terpendek.

Langkah Penyelesaian

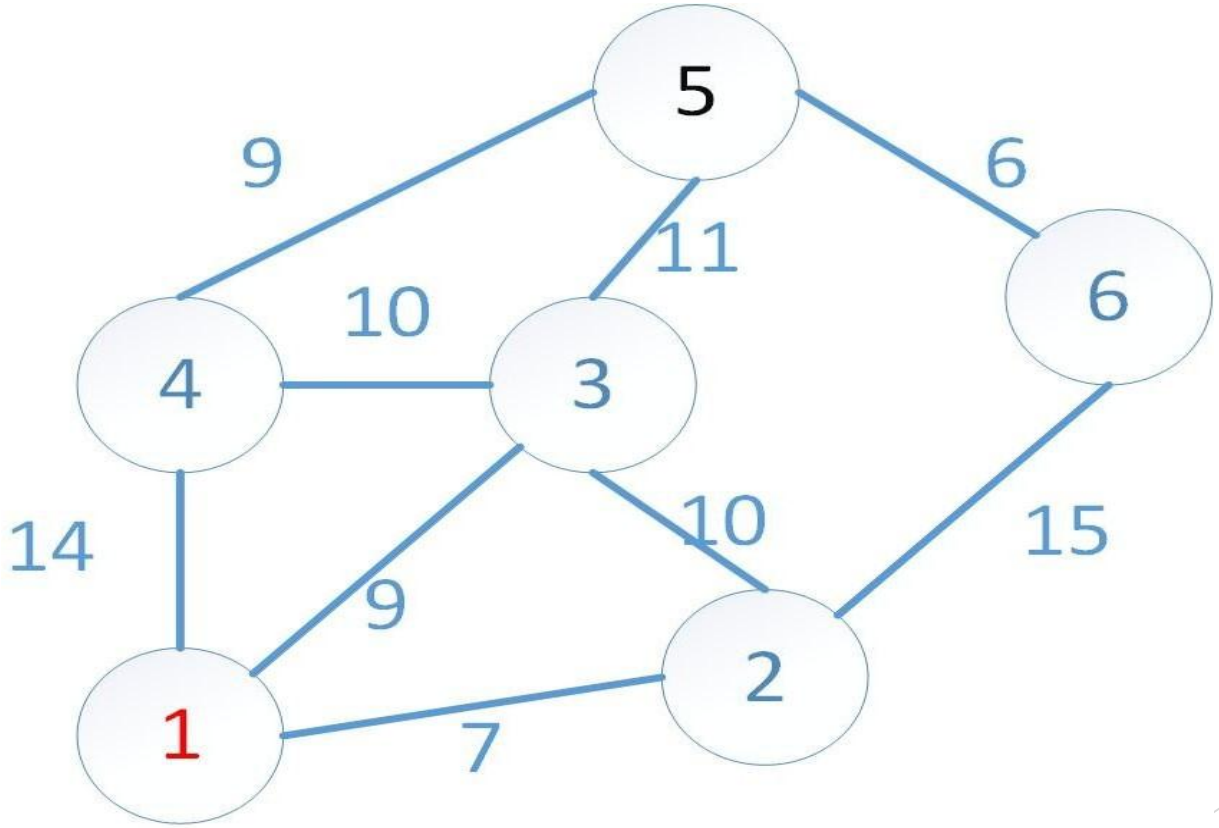
1. Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.
2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi) 2.
3. Set semua node yang belum dilalui dan set node awal sebagai “Node keberangkatan”

Langkah Penyelesaian

4. Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru
5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai “Node dilewati”. Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. Set “Node belum dilewati” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan ulangi langkah 4.

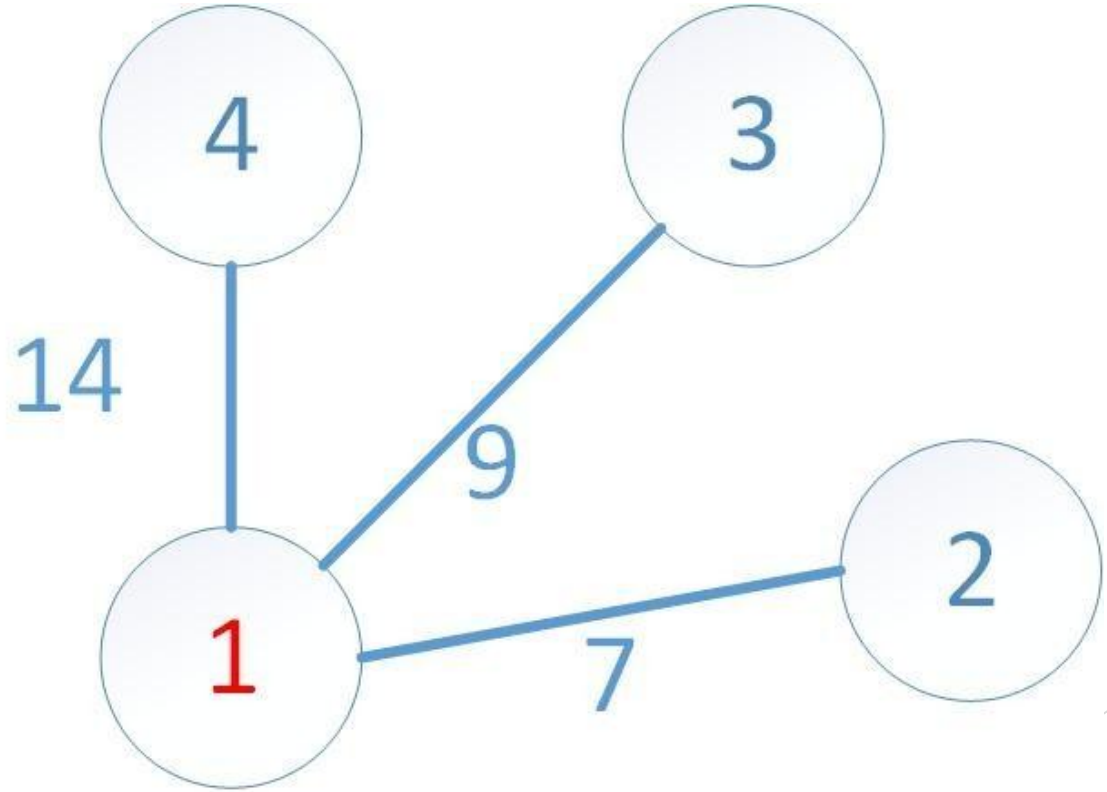
Contoh Algoritma Dijkstra

Node awal 1, Node tujuan 5. Setiap edge yang terhubung antar node telah diberi nilai



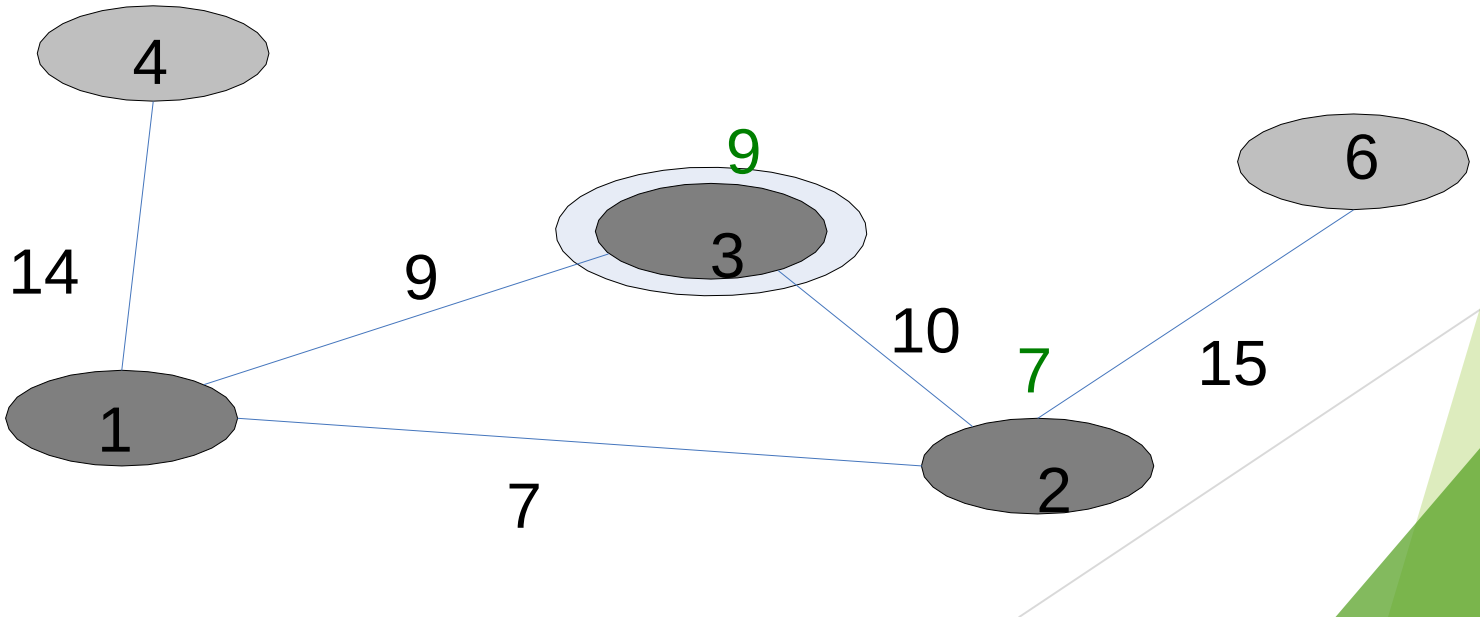
Contoh Algoritma Dijkstra

Dijkstra melakukan kalkulasi terhadap node tetangga yang terhubung langsung dengan node keberangkatan (node 1), dan hasil yang didapat adalah node 2 karena bobot nilai node 2 paling kecil dibandingkan nilai pada node lain, nilai = 7 (0+7).



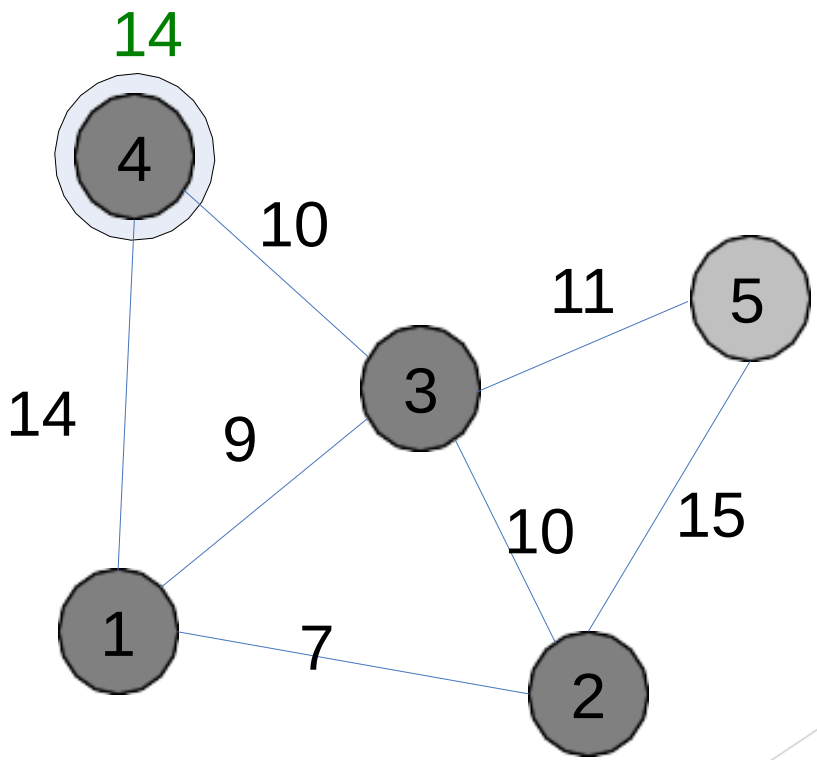
Contoh Algoritma Dijkstra

Node 2 diset menjadi node keberangkatan dan ditandai sebagi node yang telah terjamah. Dijkstra melakukan kalkulasi kembali terhadap node-node tetangga yang terhubung langsung dengan node yang telah terjamah. Dan kalkulasi dijkstra menunjukan bahwa node 3 yang menjadi node keberangkatan selanjutnya karena bobotnya yang paling kecil dari hasil kalkulasi terakhir, nilai 9 (0+9).



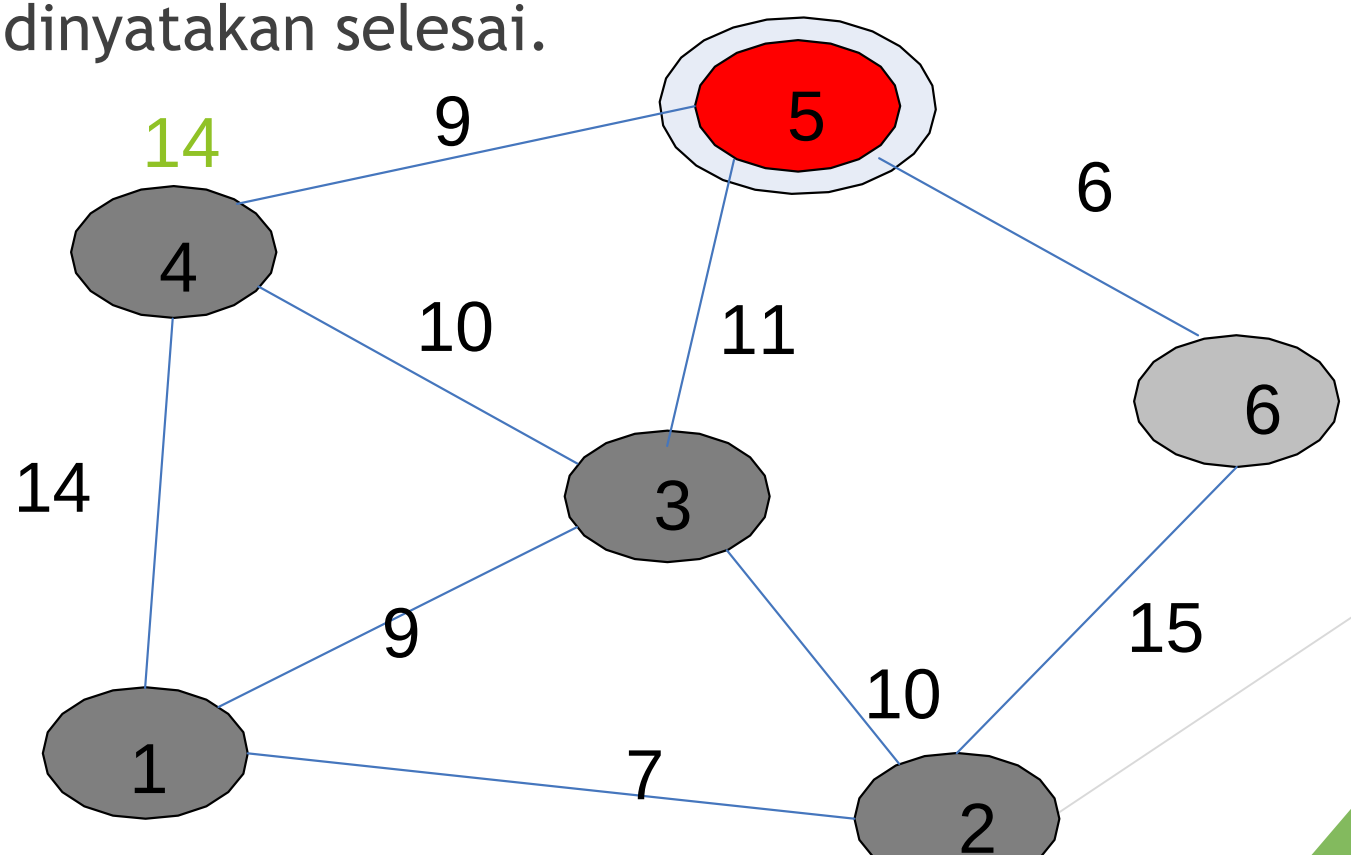
Contoh Algoritma Dijkstra

Perhitungan berlanjut dengan node 3 ditandai menjadi node yang telah terjamah. Dari semua node tetangga belum terjamah yang terhubung langsung dengan node terjamah, node selanjutnya yang ditandai menjadi node terjamah adalah node 4 karena nilai bobot yang terkecil, nilai 14

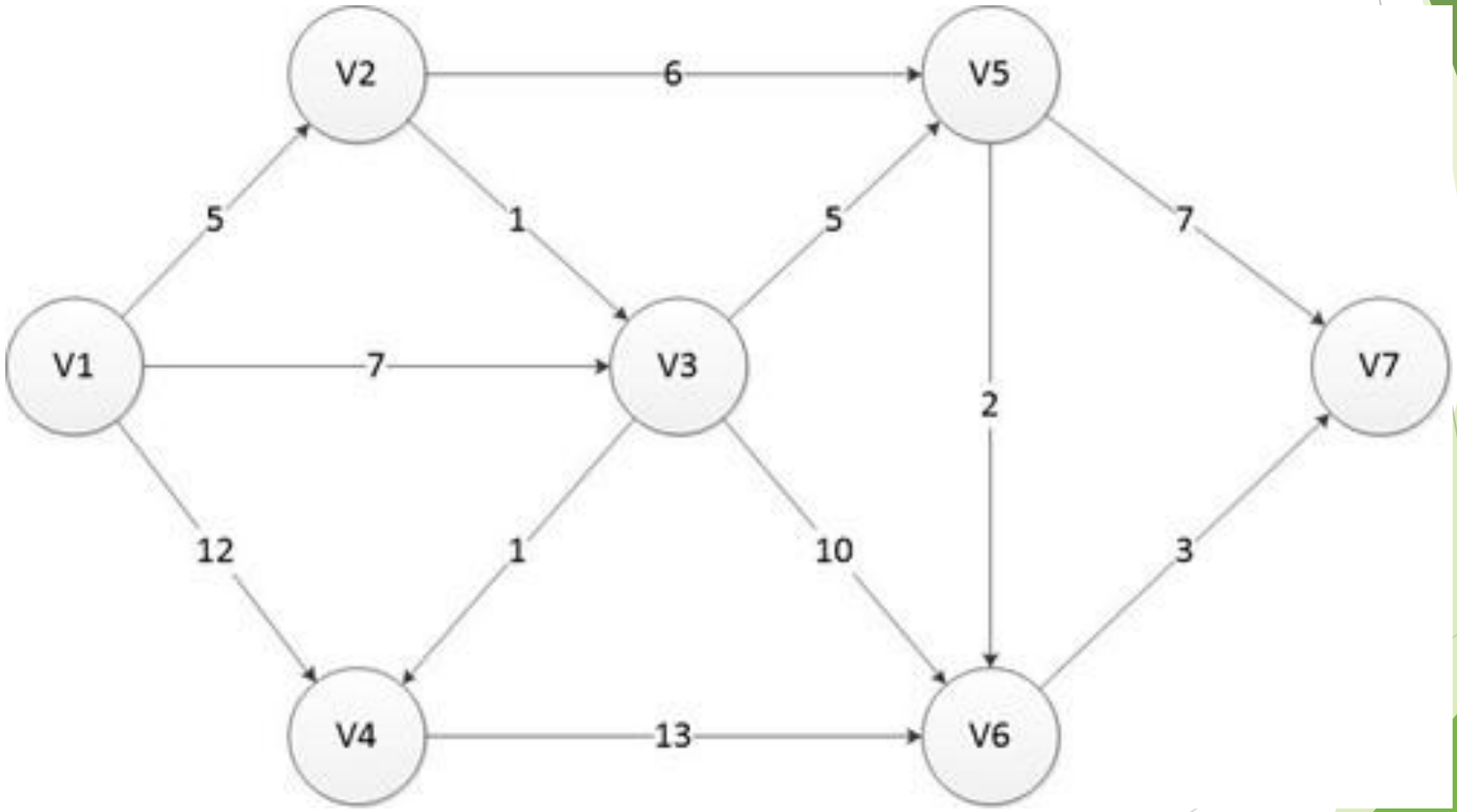


Contoh Algoritma Dijkstra

Node 4 menjadi node terjamah, dijkstra melakukan kalkulasi kembali, dan menemukan bahwa node 5 (node tujuan) telah tercapai lewat node 4. Jalur terpendeknya adalah 1-3-5, dan niilai bobot yang didapat adalah 20 (11+9). Bila node tujuan telah tercapai maka kalkulasi dijkstra dinyatakan selesai.



Contoh hitunglah Jarak terdekat dari V1 ke V7 pada gambar berikut ini



Hasil setiap stepnya dapat dilihat pada tabel berikut ini

Iteration	Unvisited (Q)	Visited (S)	Current	Node : Min = (dist[node], prev[node])iteration						
				V1	V2	V3	V4	V5	V6	V7
	Initialization {V1,V2,V3,V4,V5,V6,V7}	{-}		(0, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0
1	{V2,V3,V4,V5,V6,V7}	{V1}	V1		(5,V1)1	(7,V1)1	(12,V1)1	(∞,V1)1	(∞,V1)1	(∞,V1)1
2	{V3,V4,V5,V6,V7}	{V1,V2}	V2			(6,V2)2	(12,V1)1	(11,V2)2	(∞,V2)2	(∞,V2)2
3	{V4,V5,V6,V7}	{V1,V2, V3}	V3				(7,V3)3	(11,V3)3	(16,V3)3	(∞,V3)3
4	{V5,V6,V7}	{V1,V2, V3,V4}	V4					(11,V3)3	(16,V3)3	(∞,V3)3
5	{V6,V7}	{V1,V2, V3,V4,V5}	V5						(13,V5)5	(18,V5)5
6	{V7}	{V1,V2, V3,V4,V5.V6}	V6							(16,V6)6

Dengan demikian jarak terpendek dari V1 ke V7 adalah 16 dengan jalur V1->V2->V3->V6->V7

Link Video Penjelasan

https://drive.google.com/file/d/1DfVsKFnFDbXTl2d6CYVQj-l612PV2w_s/view?usp=sharing

Terimakasih