Dijkstra’s Algorithm

Praktik Kecerdasan Buatan

Algoritma Djikstra

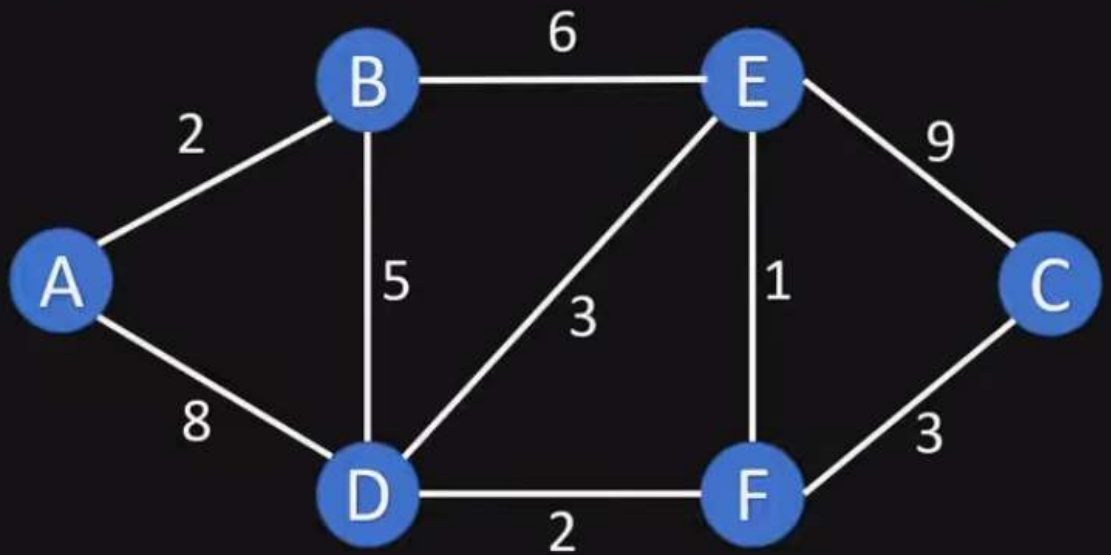
- **Dijkstra** adalah algoritma pencarian jalur terpendek dalam graf berbobot positif (semua bobot edge positif). Algoritma ini dinamai sesuai dengan nama ilmuwan komputer asal Belanda, Edsger W. Dijkstra, yang mengembangkannya pada tahun 1956. Dijkstra menghitung jalur terpendek dari satu simpul (simpul awal) ke semua simpul lain dalam graf.

Prosedur Djikstra

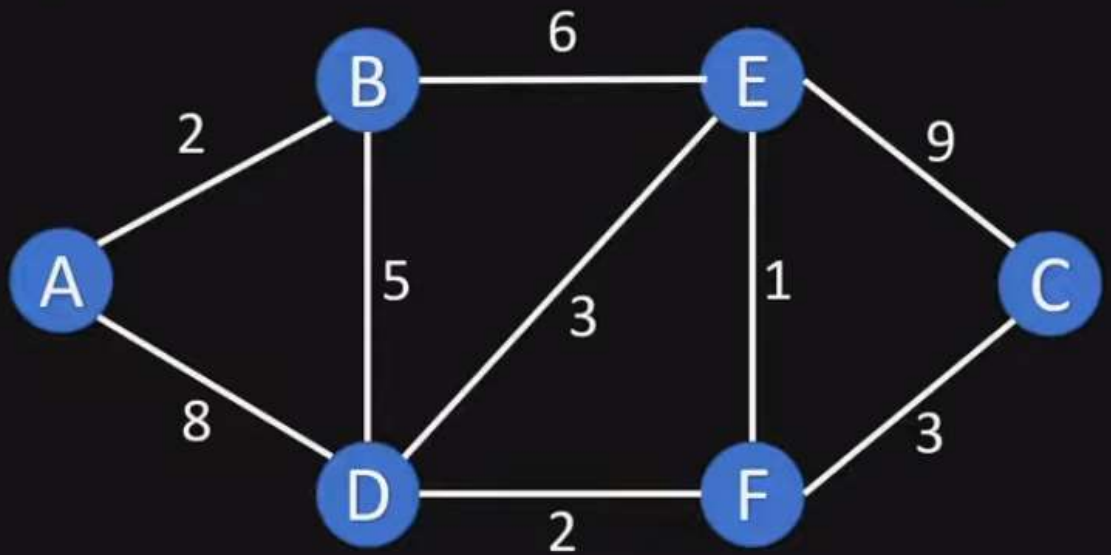
Slide tentang prosedur djikstra ini bersumber dari video:

https://www.youtube.com/watch?v=bZkzH5x0SKU

Dijkstra's Shortest Path Algorithm

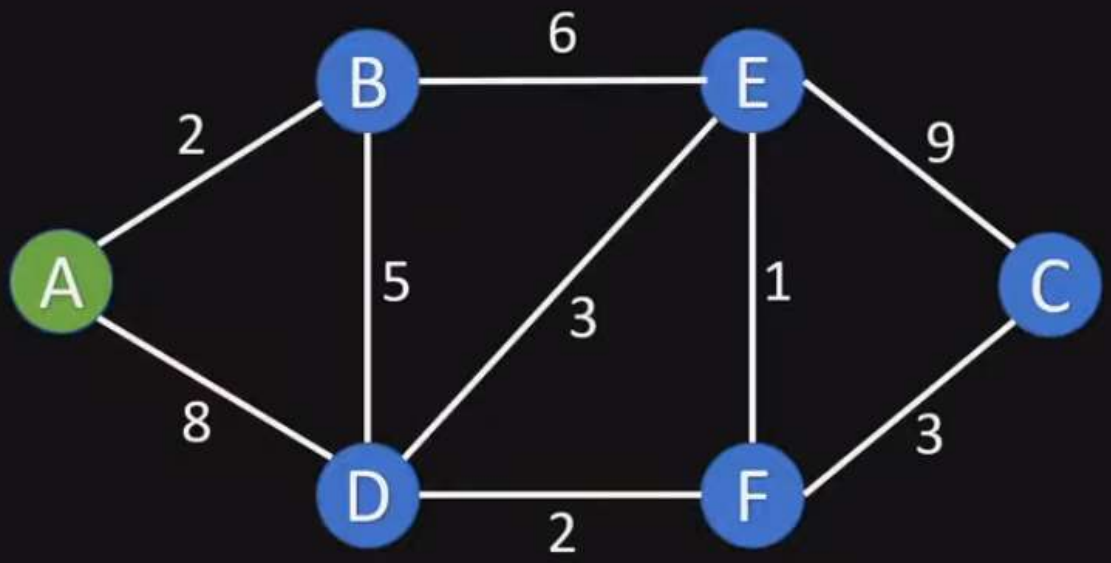


Dijkstra's Shortest Path Algorithm



- Shortest path from a fixed node to every other node
- e.g. Cities and routes between them

1. Mark all nodes as unvisited

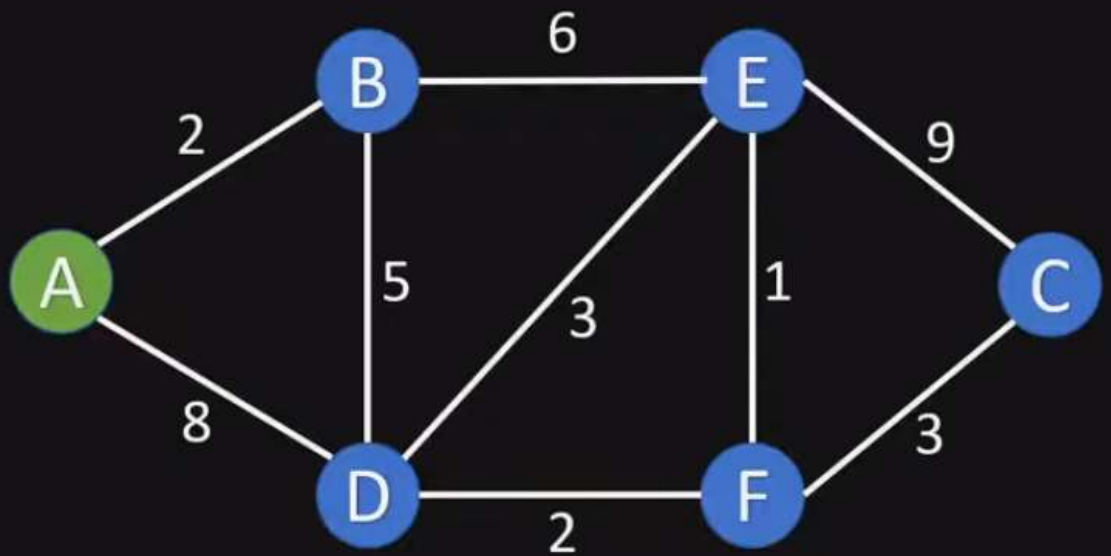


Visited Nodes: []

Unvisited Nodes: [A, B, C, D, E, F]

Dijkstra's Algorithm

2. Assign to all nodes a tentative distance value



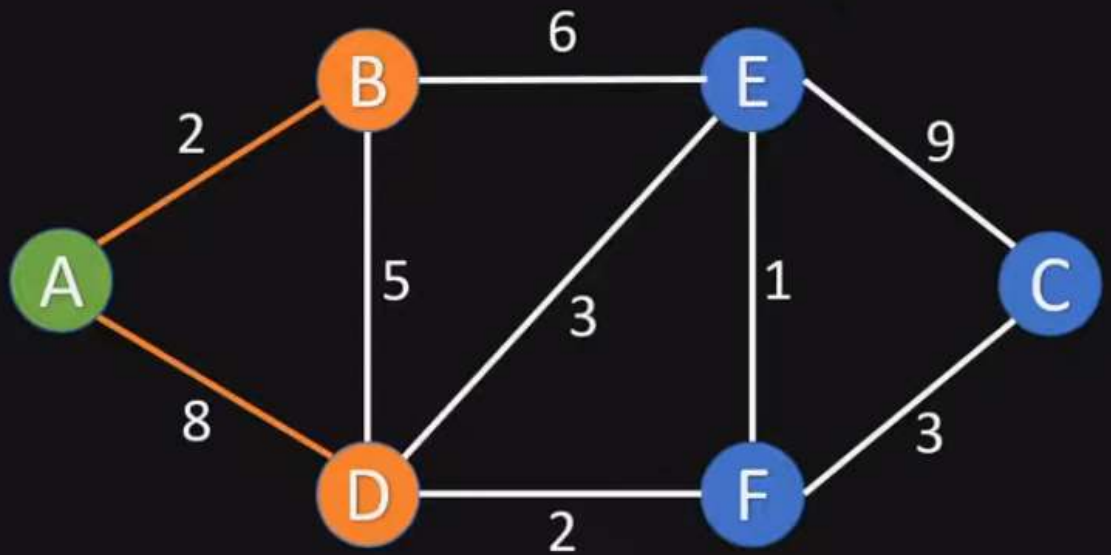
Node	Shortest Distance	Previous Node
A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	

Visited Nodes: []

Unvisited Nodes: [A, B, C, D, E, F]

Dijkstra's Algorithm

4. Mark current node as visited



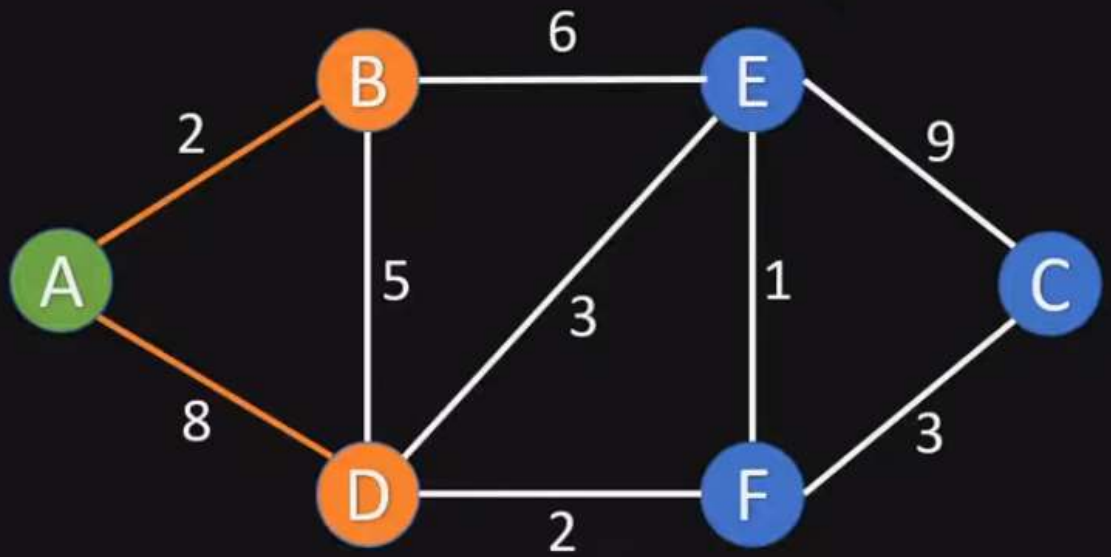
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**8**	A
E	∞	
F	∞	

Visited Nodes: []

Unvisited Nodes: [A, B, C, D, E, F]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours
3.1. Update shortest distance, if new distance is shorter than old distance



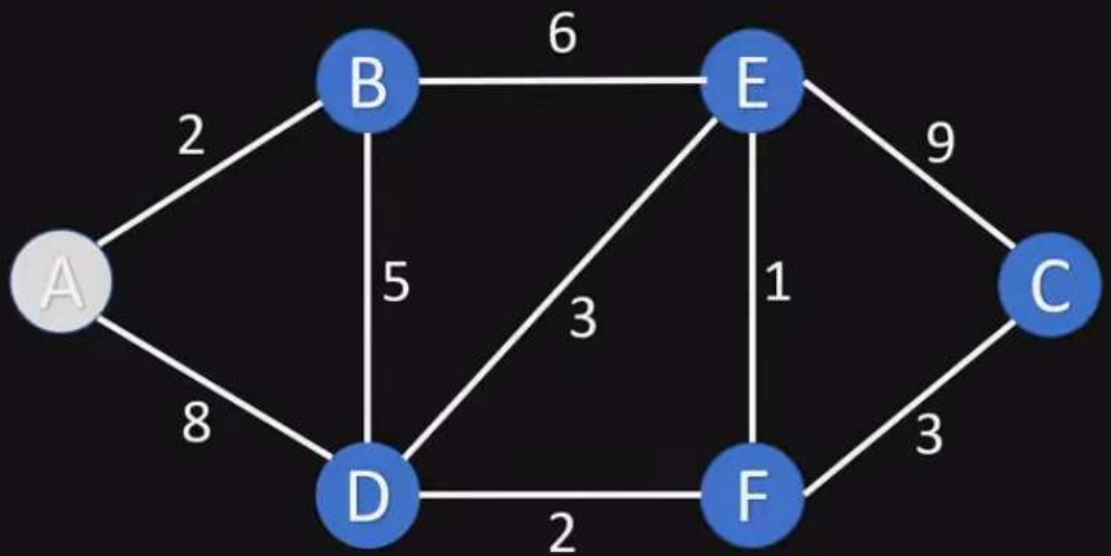
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**8**	A
E	∞	
F	∞	

Visited Nodes: []

Unvisited Nodes: [A, B, C, D, E, F]

Dijkstra's Algorithm

4. Mark current node as visited



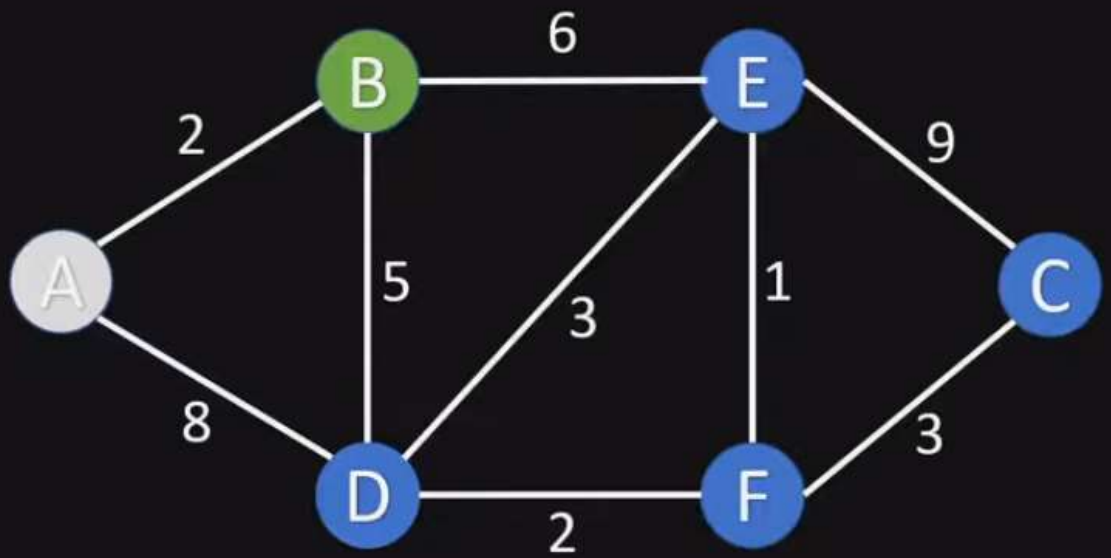
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**8**	A
E	∞	
F	∞	

Visited Nodes: [A]

Unvisited Nodes: [B, C, D, E, F]

Dijkstra's Algorithm

5. Choose new current node from unvisited nodes with minimal distance



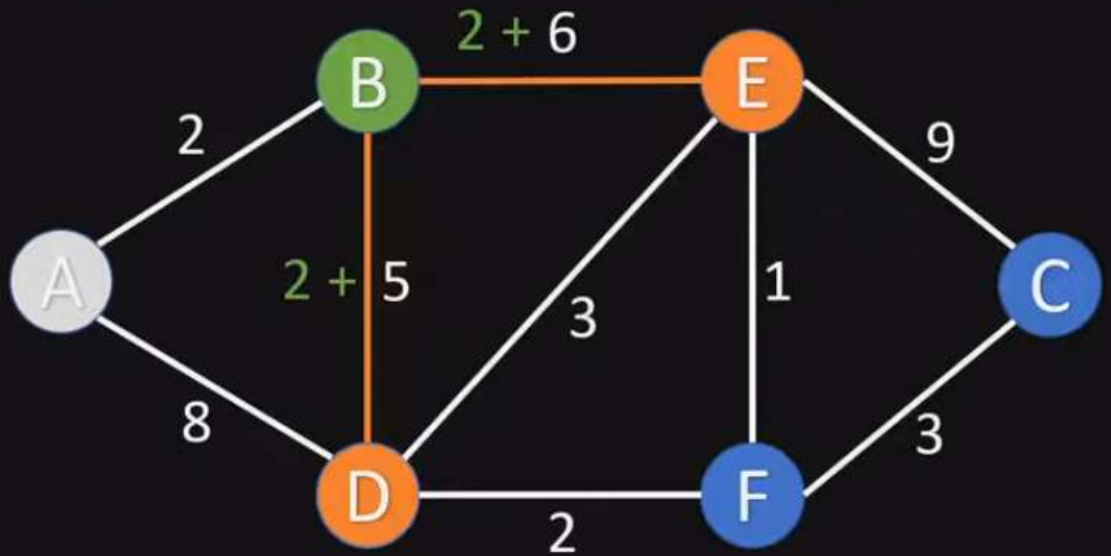
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**8**	A
E	∞	
F	∞	

Visited Nodes: [A]

Unvisited Nodes: [B, C, D, E, F]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours
3.1. Update shortest distance, if new distance is shorter than old distance



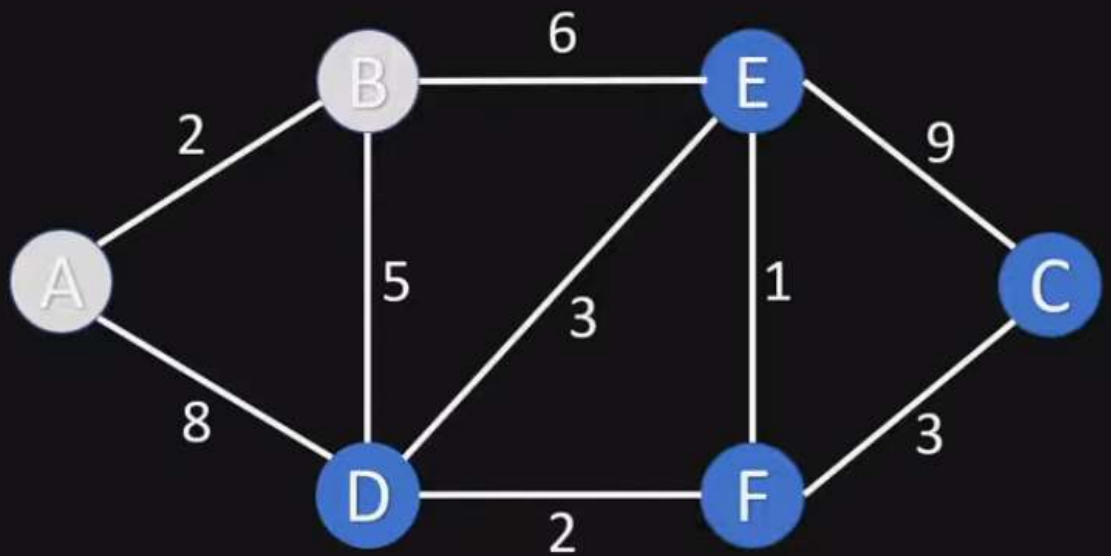
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	∞	

Visited Nodes: [A]

Unvisited Nodes: [B, C, D, E, F]

Dijkstra's Algorithm

4. Mark current node as visited

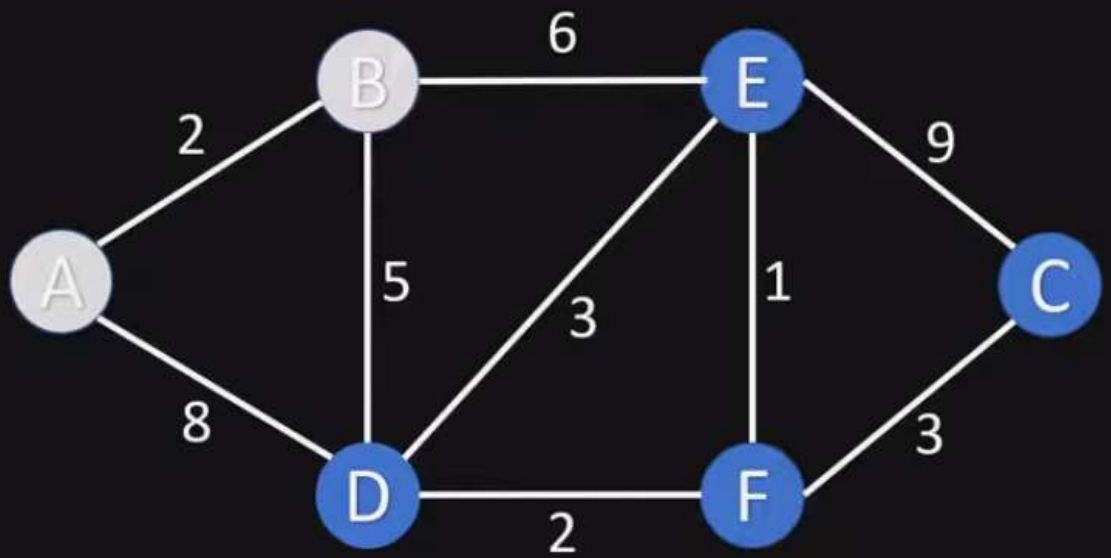


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	∞	

Visited Nodes: [A, B] Unvisited Nodes: [C, D, E, F]

Dijkstra's Algorithm

5. Choose new current node from unvisited nodes with minimal distance

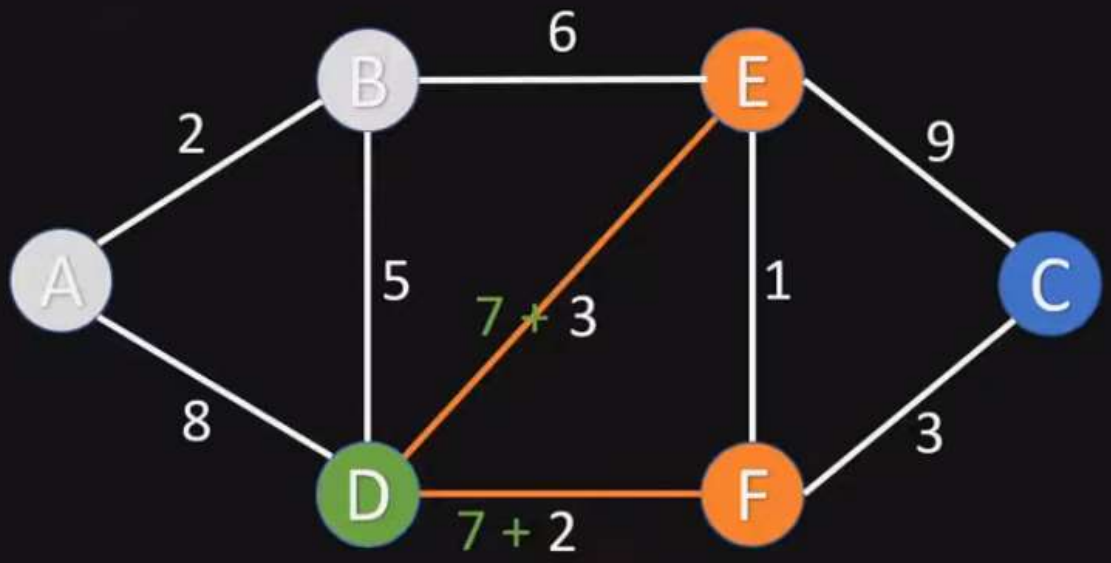


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	∞	

Visited Nodes: [A, B] Unvisited Nodes: [C, D, E, F]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours
3.1. Update shortest distance, if new distance is shorter than old distance

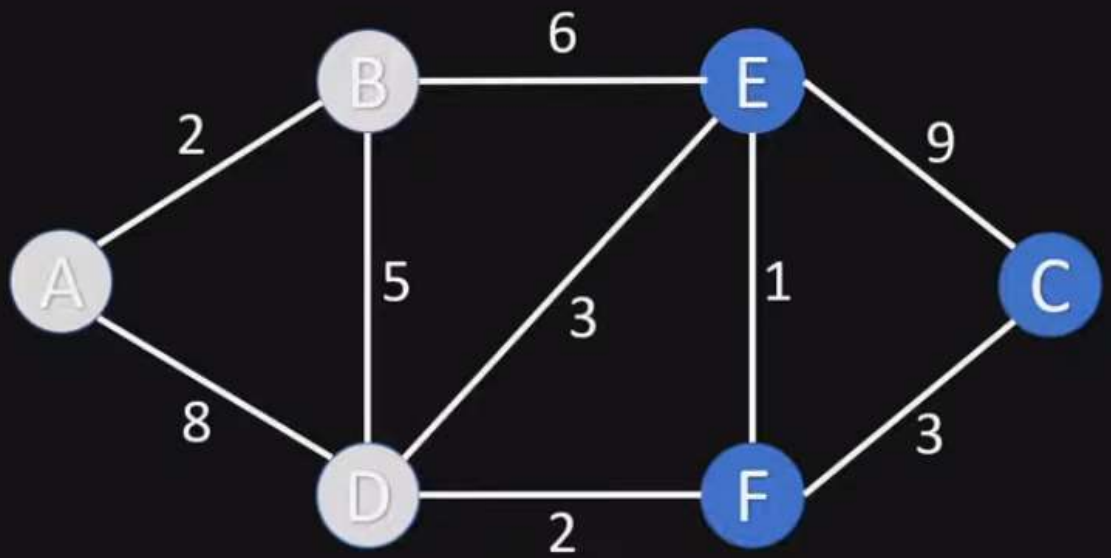


Visited Nodes: [A, B] Unvisited Nodes: [C, D, E, F]

Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	**9**	D

Dijkstra's Algorithm

4. Mark current node as visited

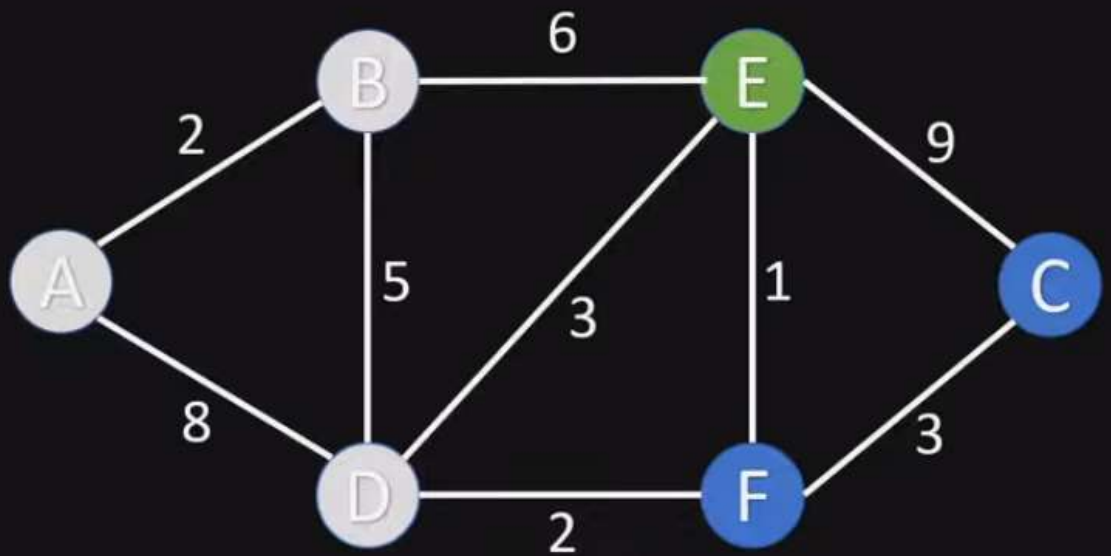


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D] Unvisited Nodes: [C, D, E, F]

Dijkstra's Algorithm

5. Choose new current node from unvisited nodes with minimal distance

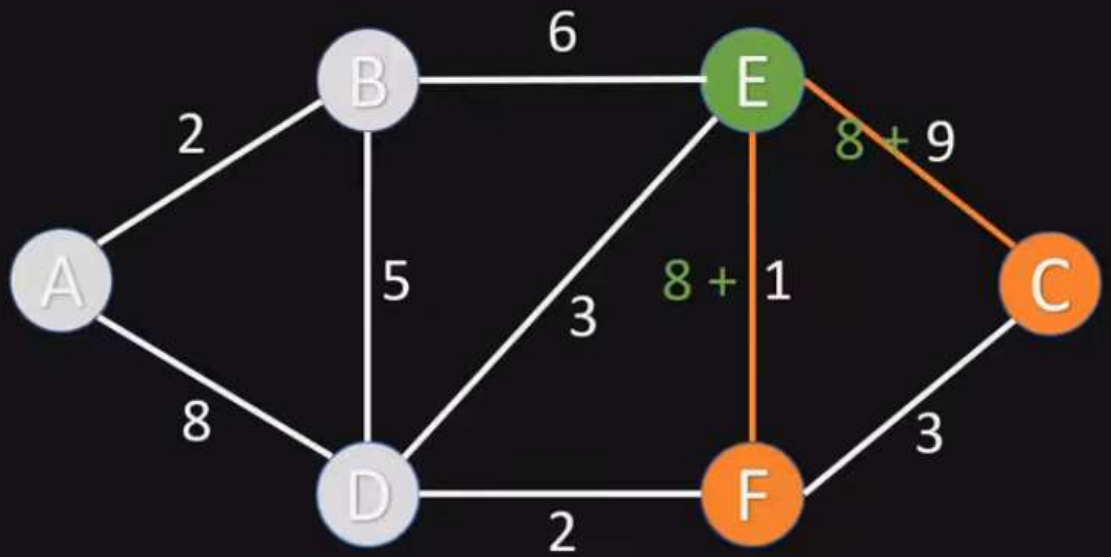


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	∞	
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D] Unvisited Nodes: [C, E, F]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours
3.1. Update shortest distance, if new distance is shorter than old distance

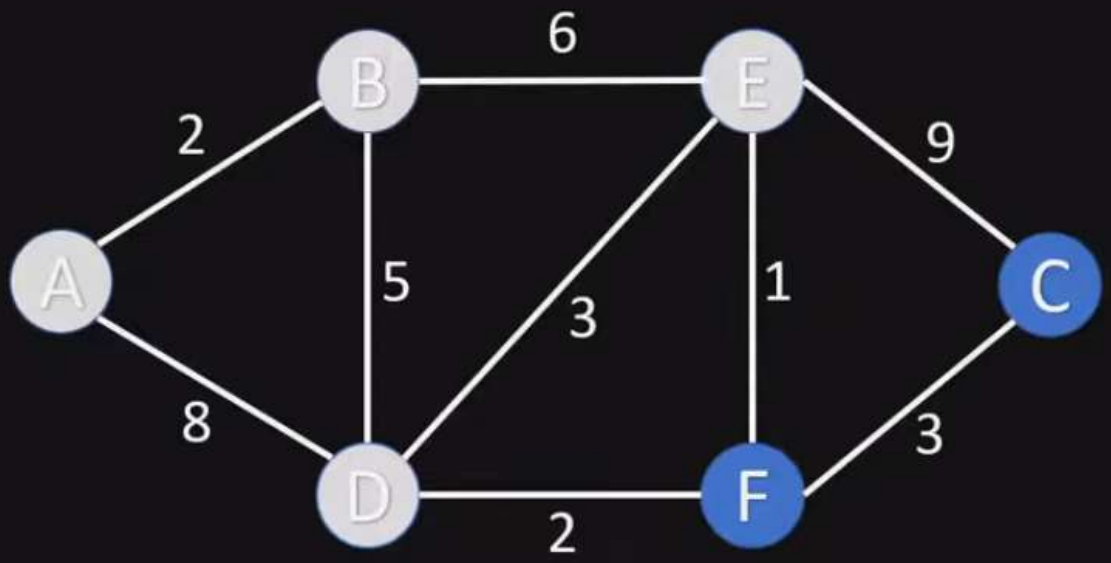


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**17**	E
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D] Unvisited Nodes: [C, E, F]

Dijkstra's Algorithm

4. Mark current node as visited

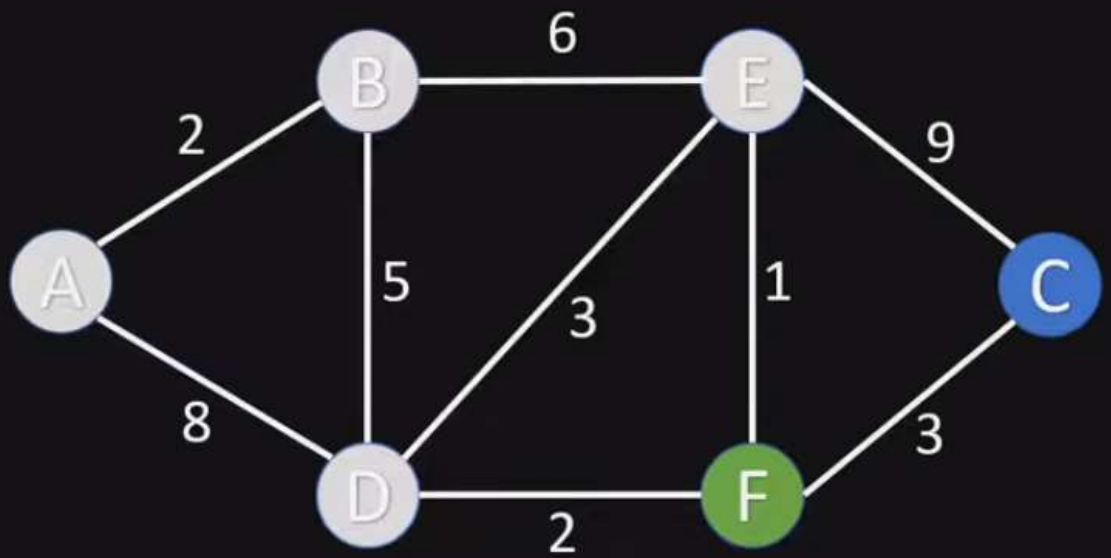


Visited Nodes: [A, B, D, E] Unvisited Nodes: [C, F]

Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**17**	E
D	**7**	B
E	**8**	B
F	**9**	D

Dijkstra's Algorithm

5. Choose new current node from unvisited nodes with minimal distance



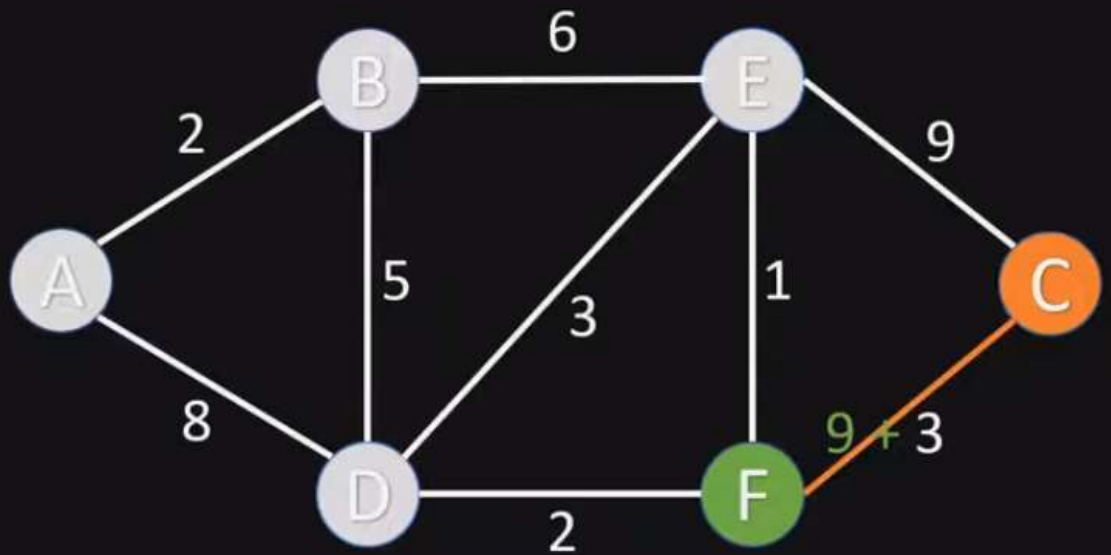
Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**17**	E
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D, E] Unvisited Nodes: [C, F]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours

3.1. Update shortest distance, if new distance is shorter than old distance

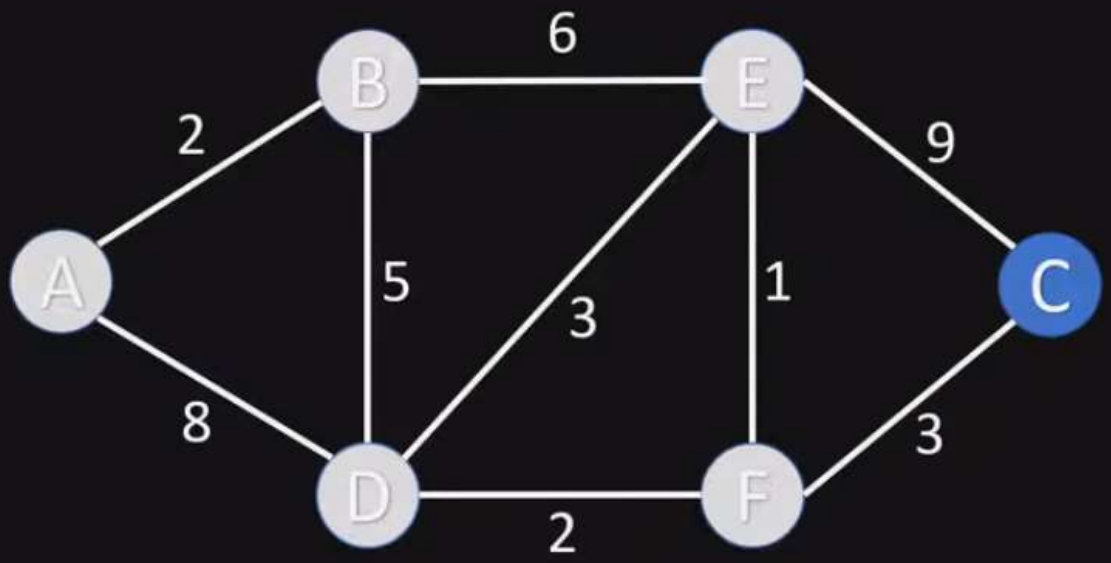


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D, E] Unvisited Nodes: [C, F]

Dijkstra's Algorithm

4. Mark current node as visited

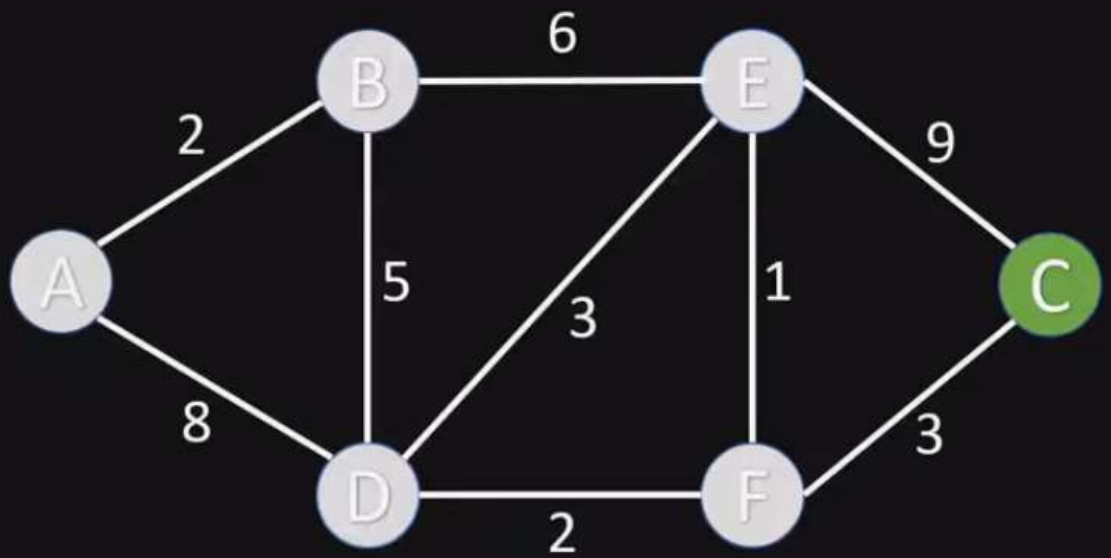


Visited Nodes: [A, B, D, E, F] Unvisited Nodes: [C]

Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Dijkstra's Algorithm

5. Choose new current node from unvisited nodes with minimal distance

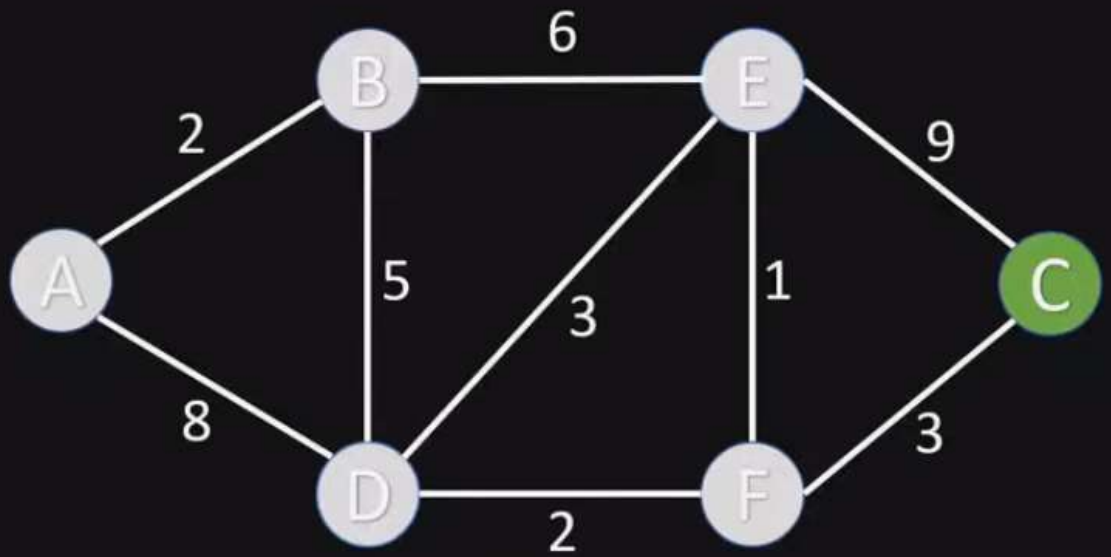


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D, E, F] Unvisited Nodes: [C]

Dijkstra's Algorithm

3. For the current node calculate the distance to all unvisited neighbours

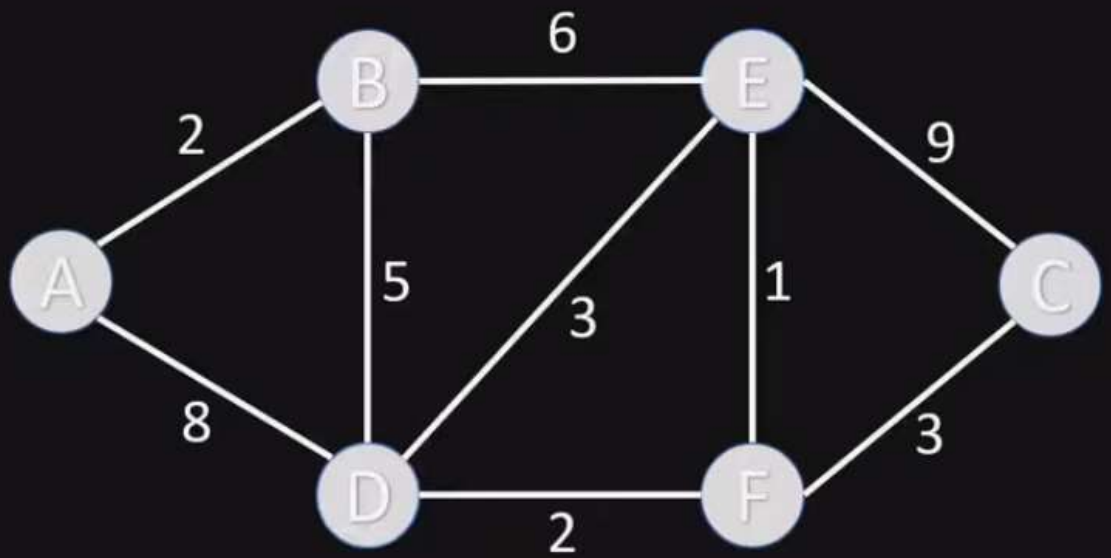


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D, E, F] Unvisited Nodes: [C]

Dijkstra's Algorithm

4. Mark current node as visited

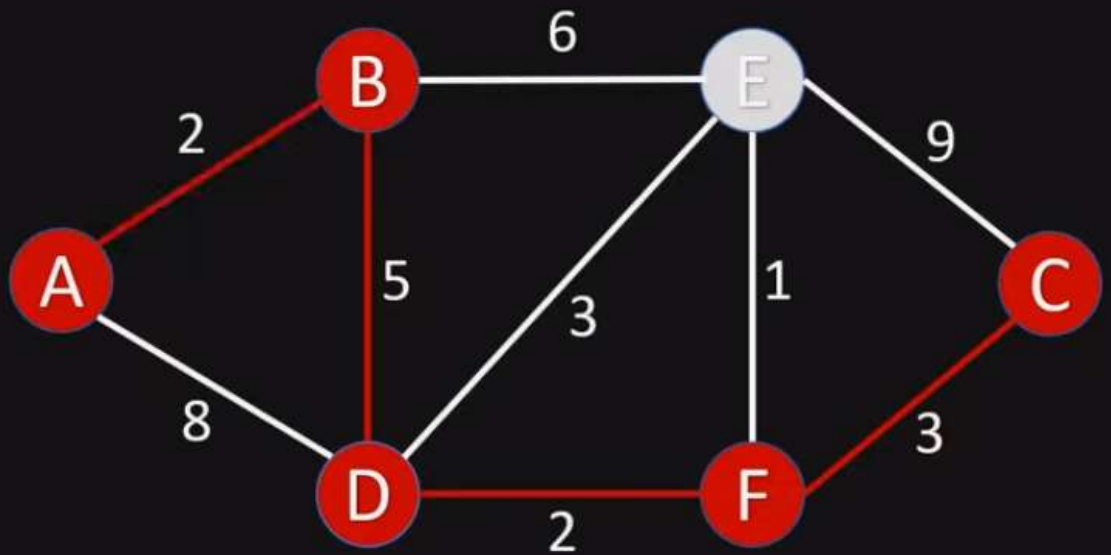


Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Visited Nodes: [A, B, D, E, F, C] Unvisited Nodes: []

Dijkstra's Algorithm

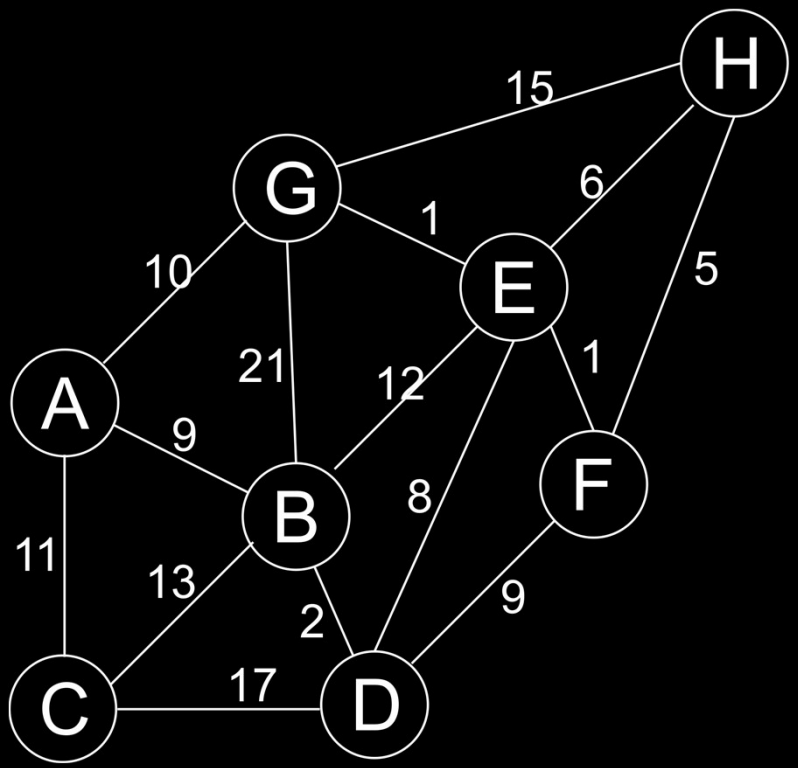
Get shortest path from A to C



Node	Shortest Distance	Previous Node
A	0	
B	**2**	A
C	**12**	F
D	**7**	B
E	**8**	B
F	**9**	D

Dijkstra's Algorithm

Praktik Djikstra



Tentukan jarak terdekat dari:

1. C ke F
2. C ke H

Gunakan program pada https://github.com/taufiqmus/Praktik-Kecerdasan-Buatan/tree/main/Searching

Bandingkan dengan perhitungan manual dan gambarlah hasil jalur yang tercipta ☺

Kesimpulan (Algoritma Searching)

Tujuan Utama:

- **Dijkstra:** Menemukan jalur terpendek dari satu simpul awal ke semua simpul lain dalam graf berbobot positif.
- **BFS (Breadth-First Search):** Menemukan jalur terpendek dari satu simpul awal ke satu simpul tujuan dalam graf tak berbobot.
- **DFS (Depth-First Search):** Tidak khusus untuk mencari jalur terpendek, melainkan digunakan untuk eksplorasi dalam graf.

Terima Kasih