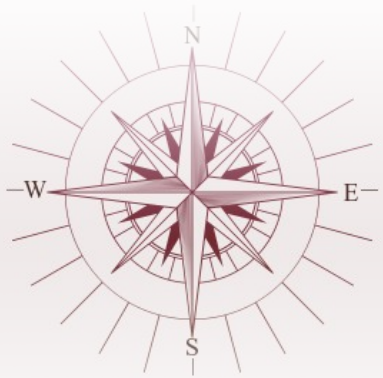


Day 9
Paths for an odyssey

Lecturer: Msc. Minh Tan Le

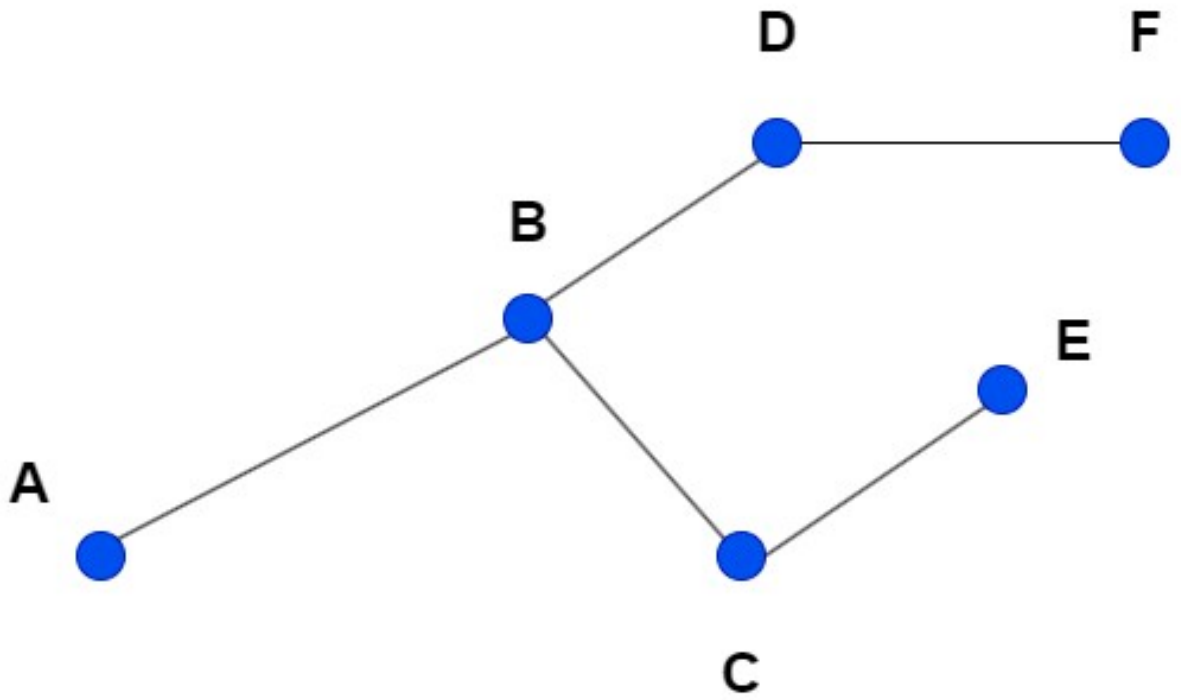
Today’s lesson includes…

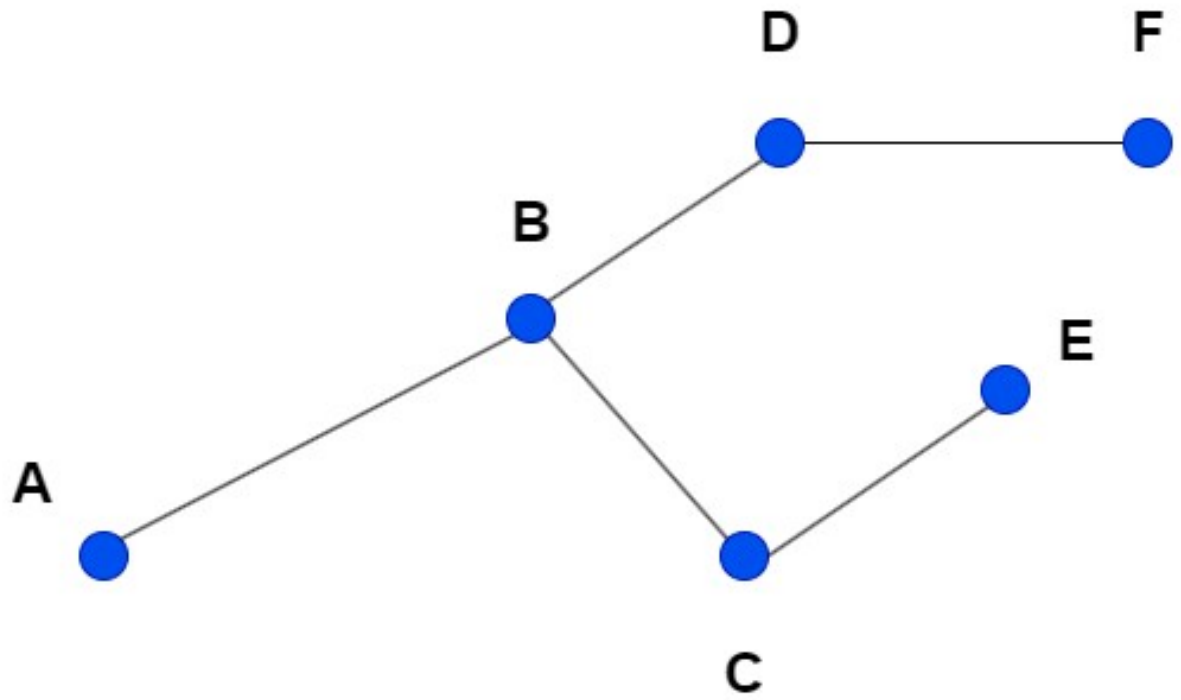
I. Path demonstration on paper

II. Fleury algorithm for Euler path searching

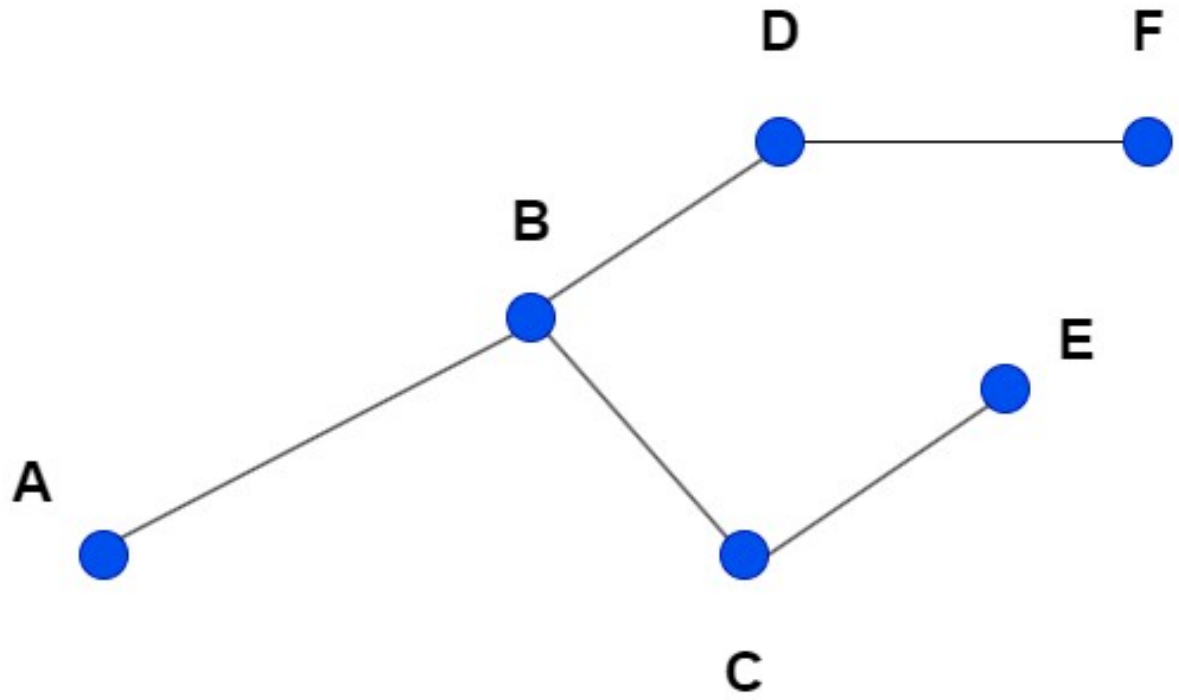
III. Dijkstra for shortest path searching

I. Path demonstration on paper



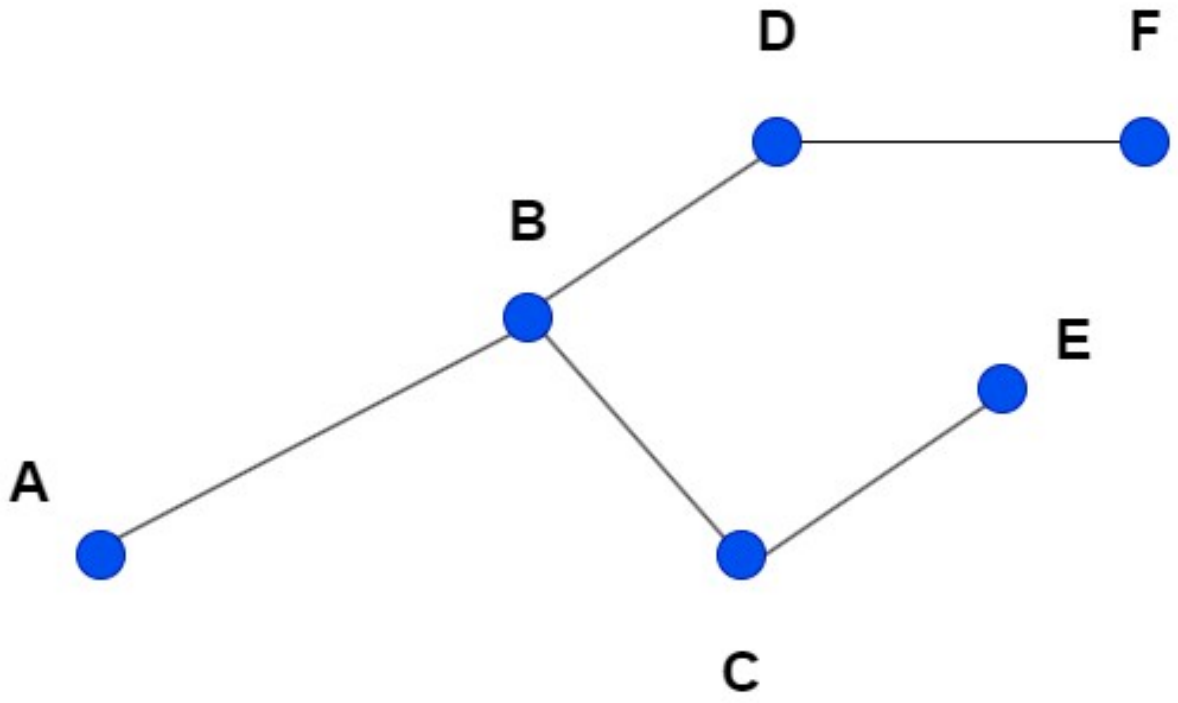


A, B, D, C, F, E

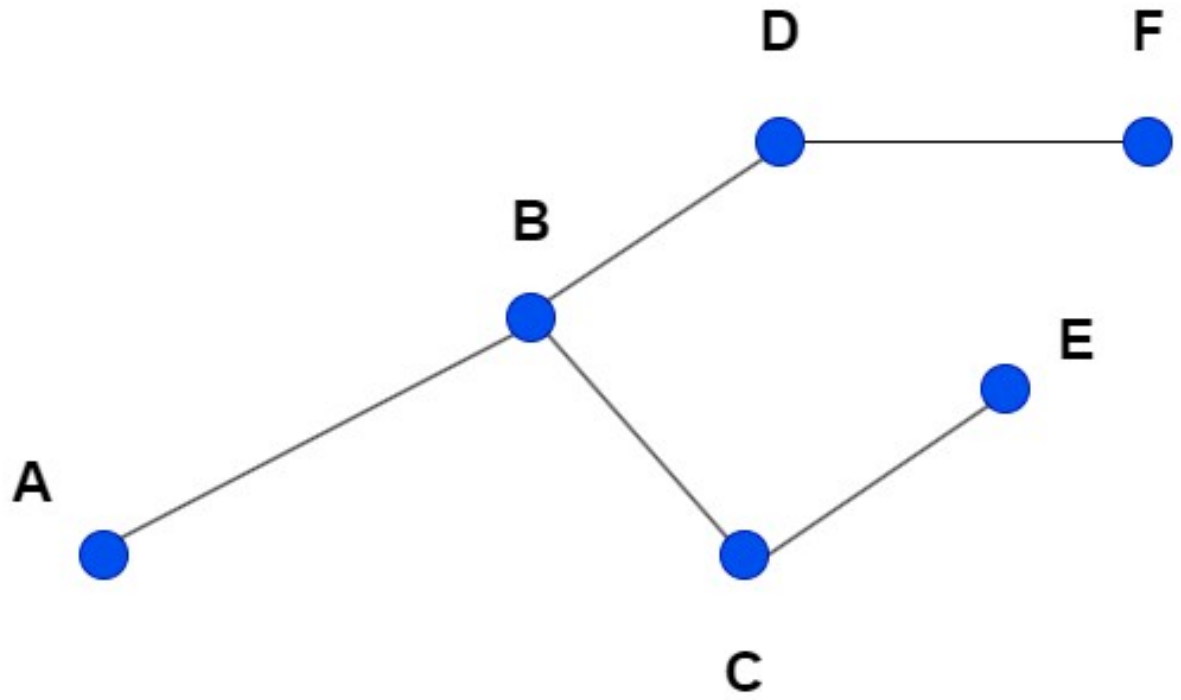


A, B, D, C, F, E

$$A \rightarrow B \rightarrow D \rightarrow F$$
$$\searrow$$
$$C \rightarrow E$$



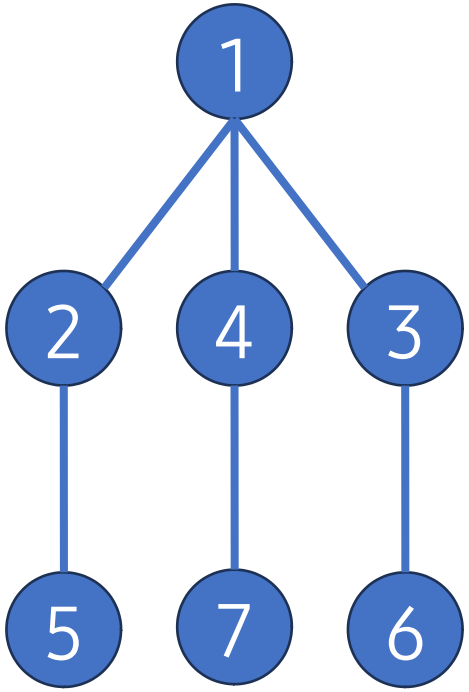
A, B, D, C, F, E ✗ A → B → D → F
↘
C → E



(A,B), (B, D), (B, C), (C, E), (D, F)

Notes

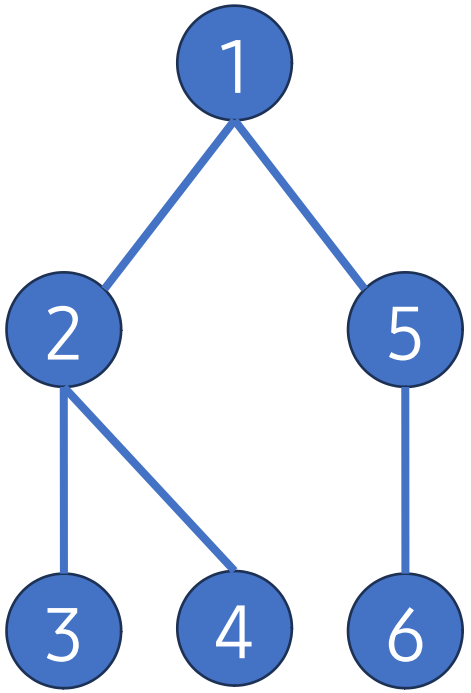
- If it's a path finding from A to B, verify the path.
- Count the frequency of nodes, which equals to *deg.*
- Distinguish between BFS & DFS.



Early visited

(1, 2), (1, 3), (1, 4), (2, 5), (3, 6), (4, 7)

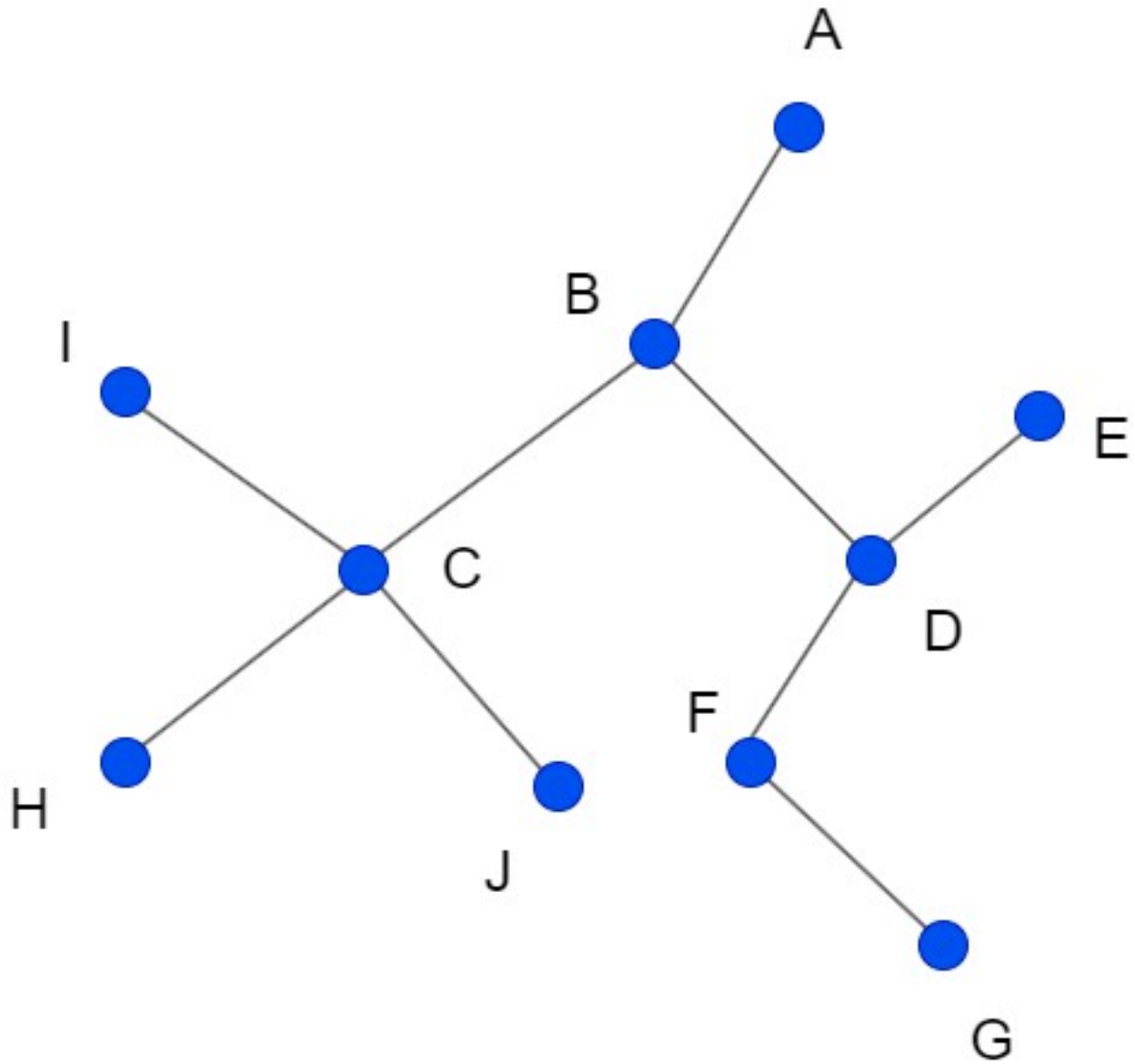
Late visited



(1, 2), (2, 3), (2, 4), (1, 5), (5, 6)

Exercise #1

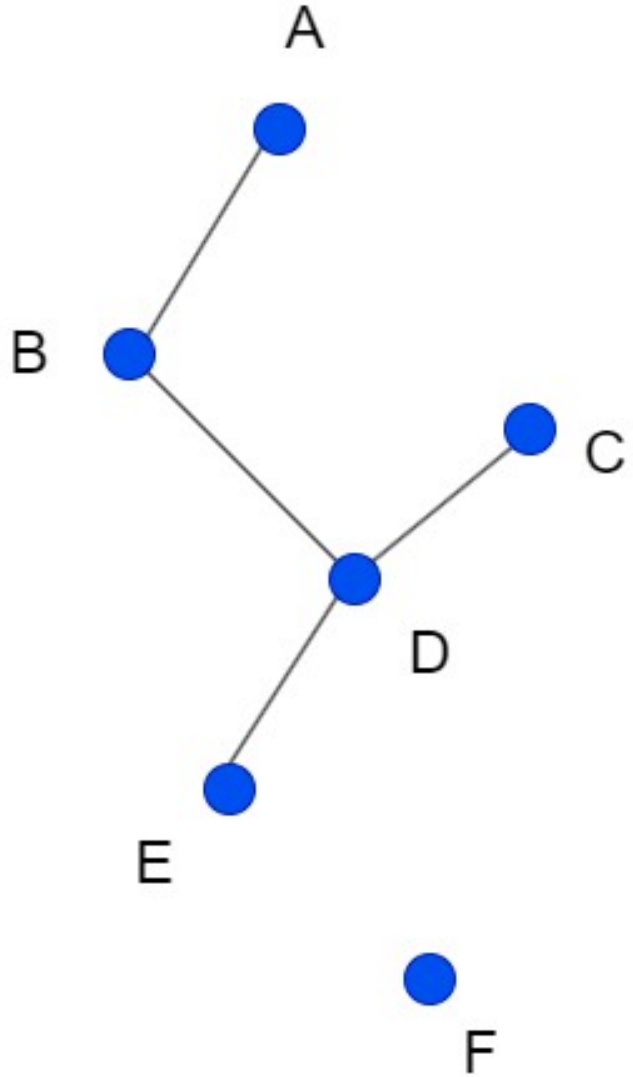
Find the path from A to G using **DFS** that there must be **at least 6 visited nodes**.



Exercise #2

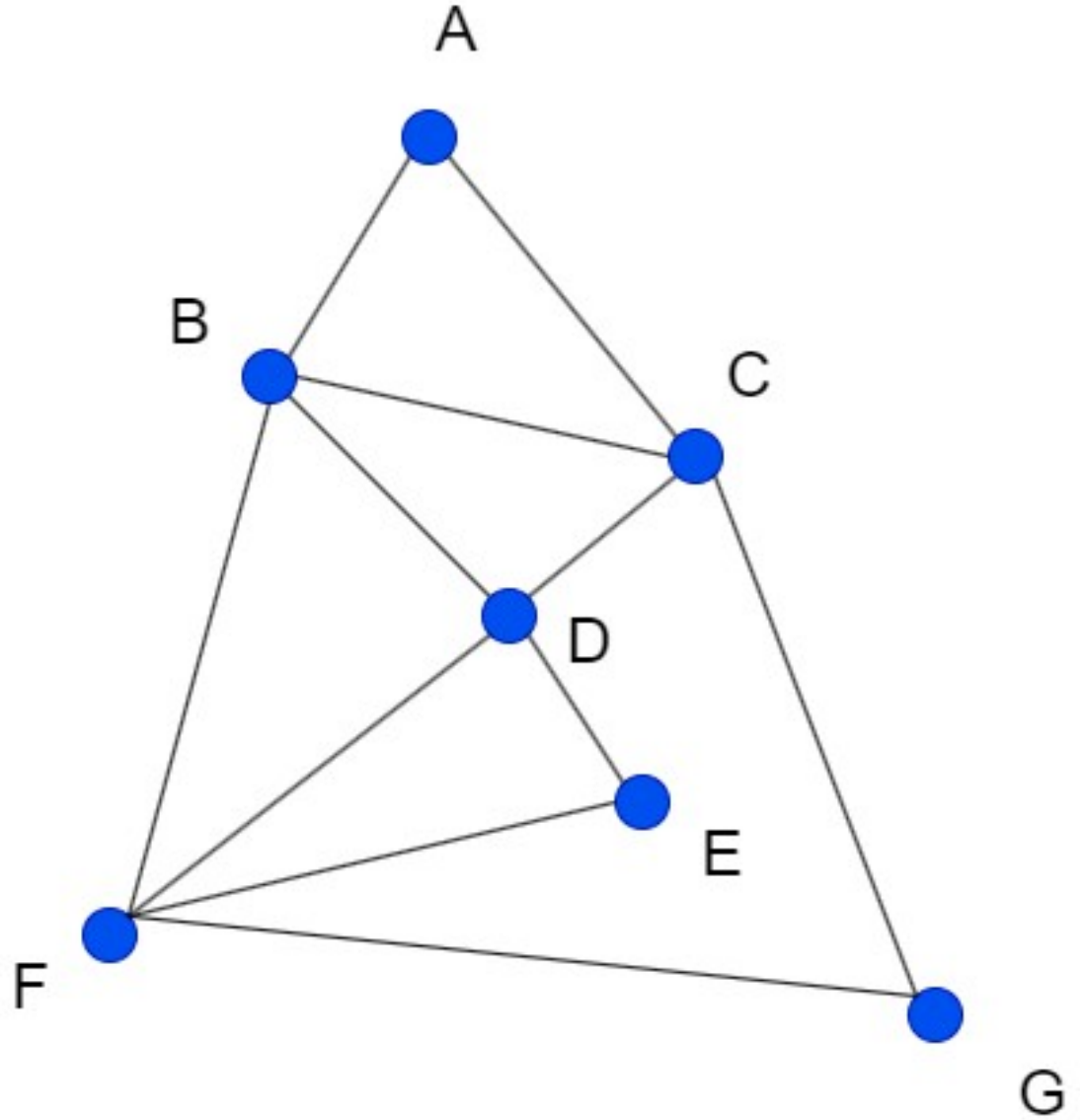
Search for F from A

using DFS.



Exercise #3

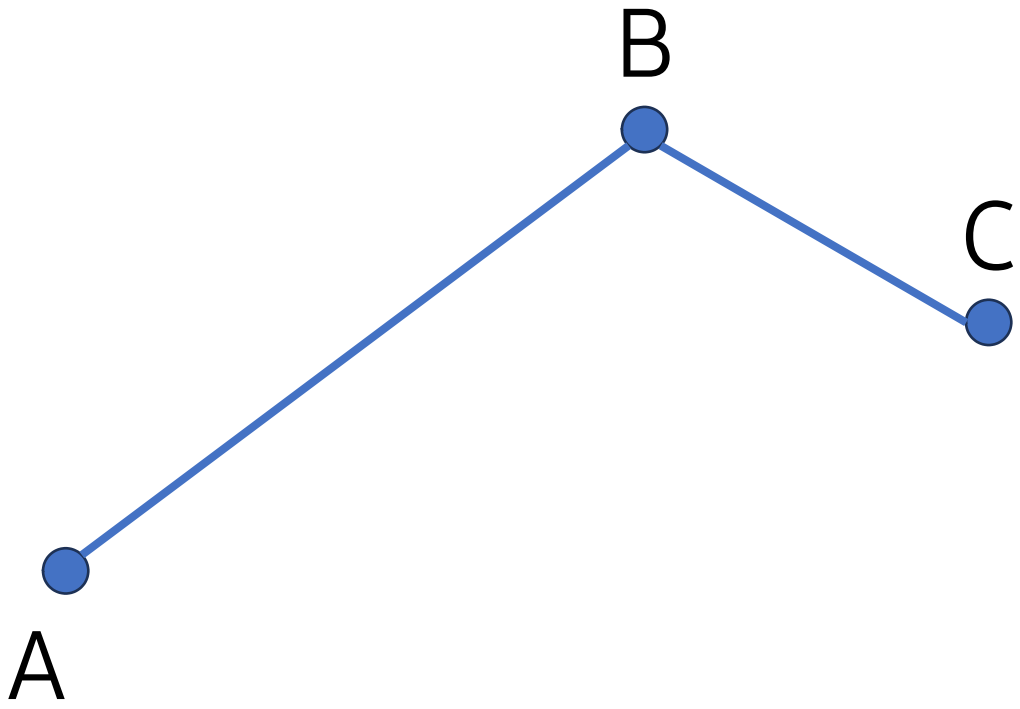
Find the path from A to H using **BFS**. The order is Alphabet.



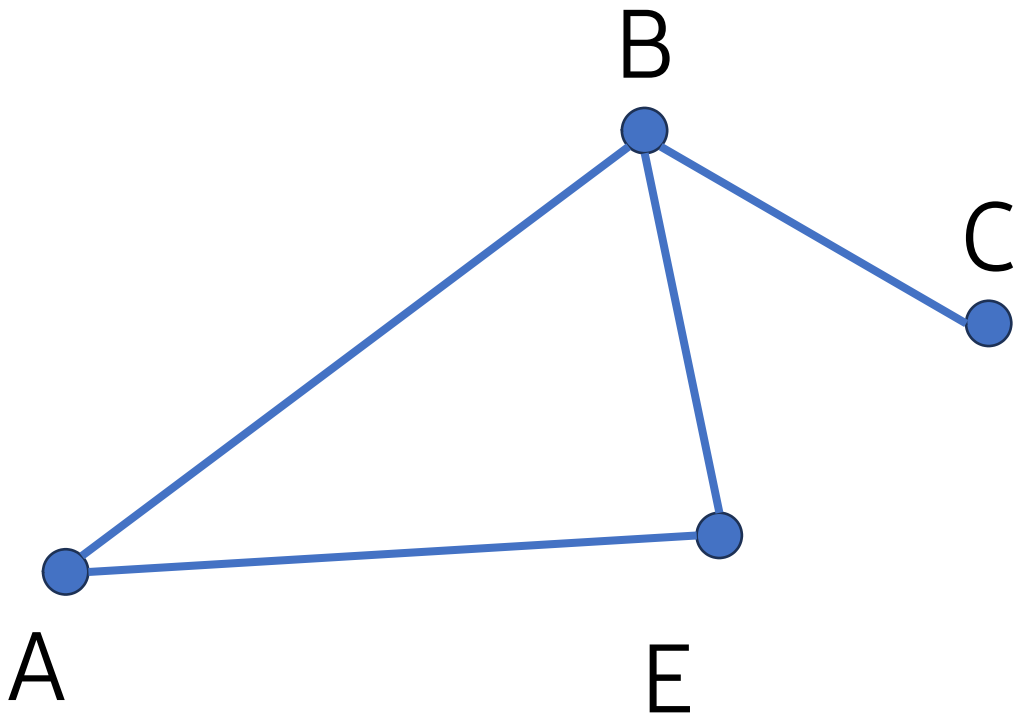
Problems for DFS/BFS

- Scanning
- Searching
- Path finding (from A to B)
- Connectivity check
- Spanning tree search
- Tree detection

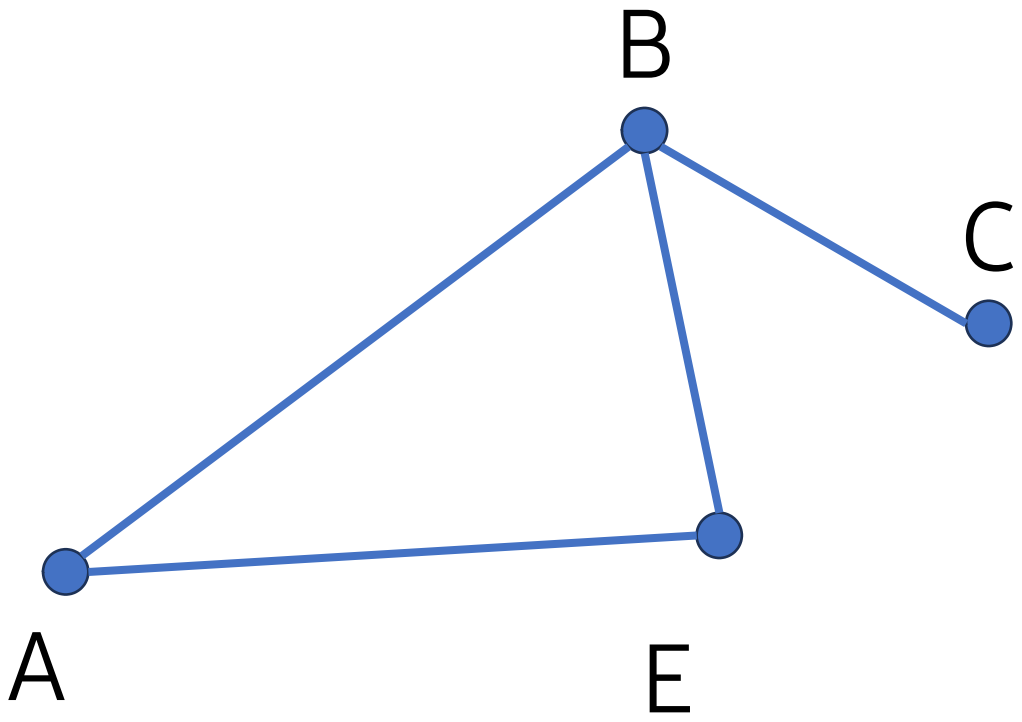
Define a true “Tree” structure?



A.connect(B)
B.connect(C)
G.addNodes({A, B, C})



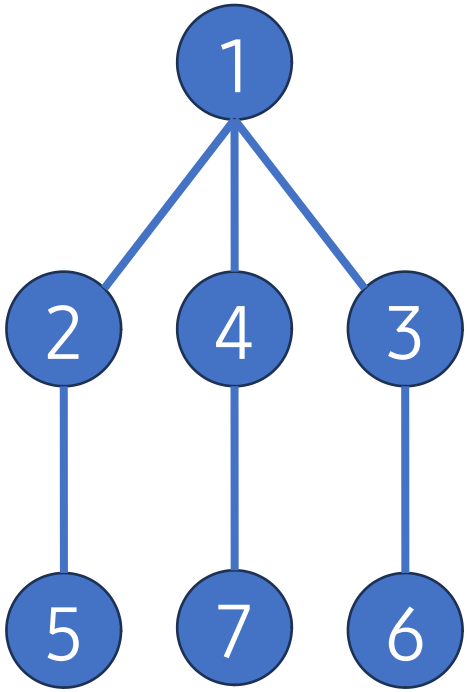
A.connect(B)
B.connect(C)
E.connect(A)
E.connect(B)
G.addNodes({A, B, C, E})



A.connect(B)
B.connect(C)
E.connect(A)
E.connect(B)
G.addNodes({A, B, C, E})

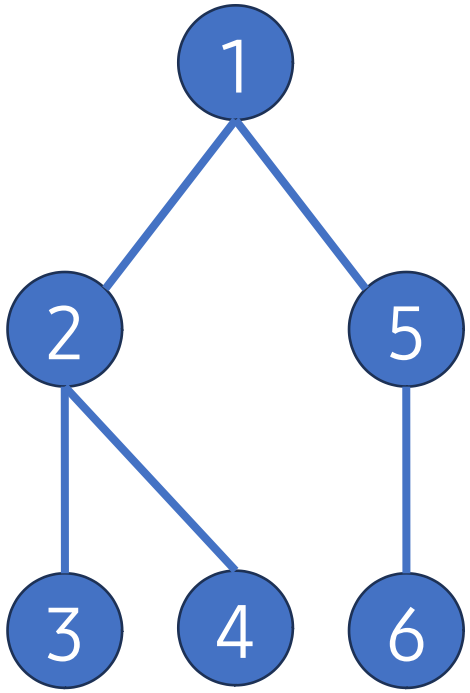
DFS from E?

BFS



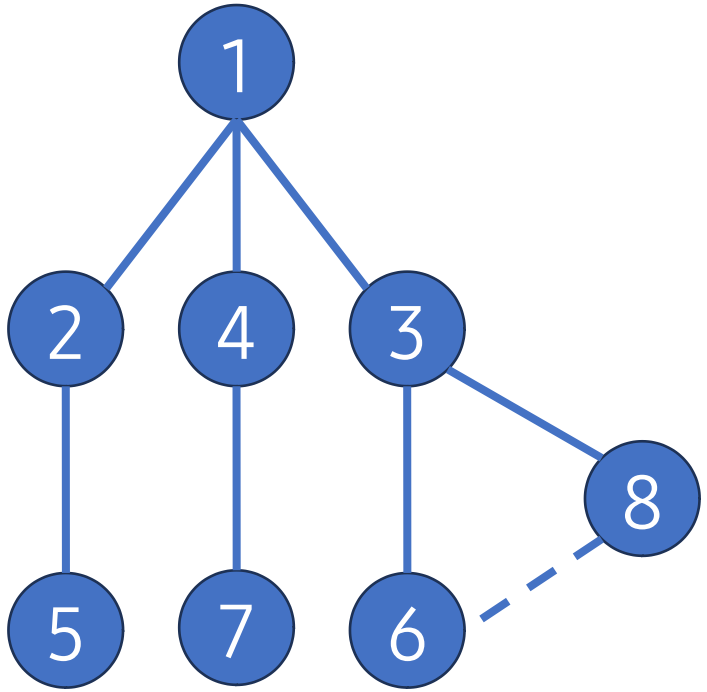
(1, 2), (1, 3), (1, 4), (2, 5), (3, 6), (4, 7)

DFS



(1, 2), (2, 3), (2, 4), (1, 5), (5, 6)

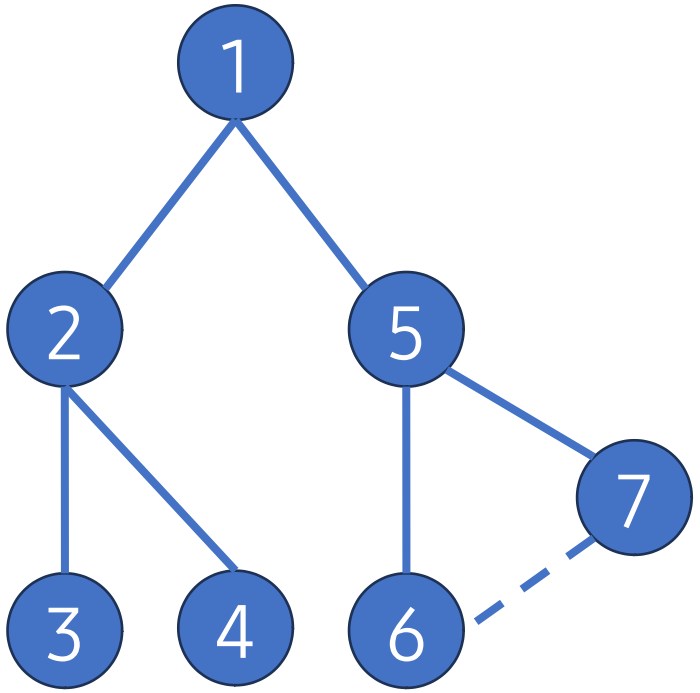
BFS



(1, 2), (1, 3), (1, 4), (2, 5), (3, 6), (4, 7)

(1, 2), (1, 3), (1, 4), (2, 5), (3, 6), (3, 8), (4, 7)

DFS



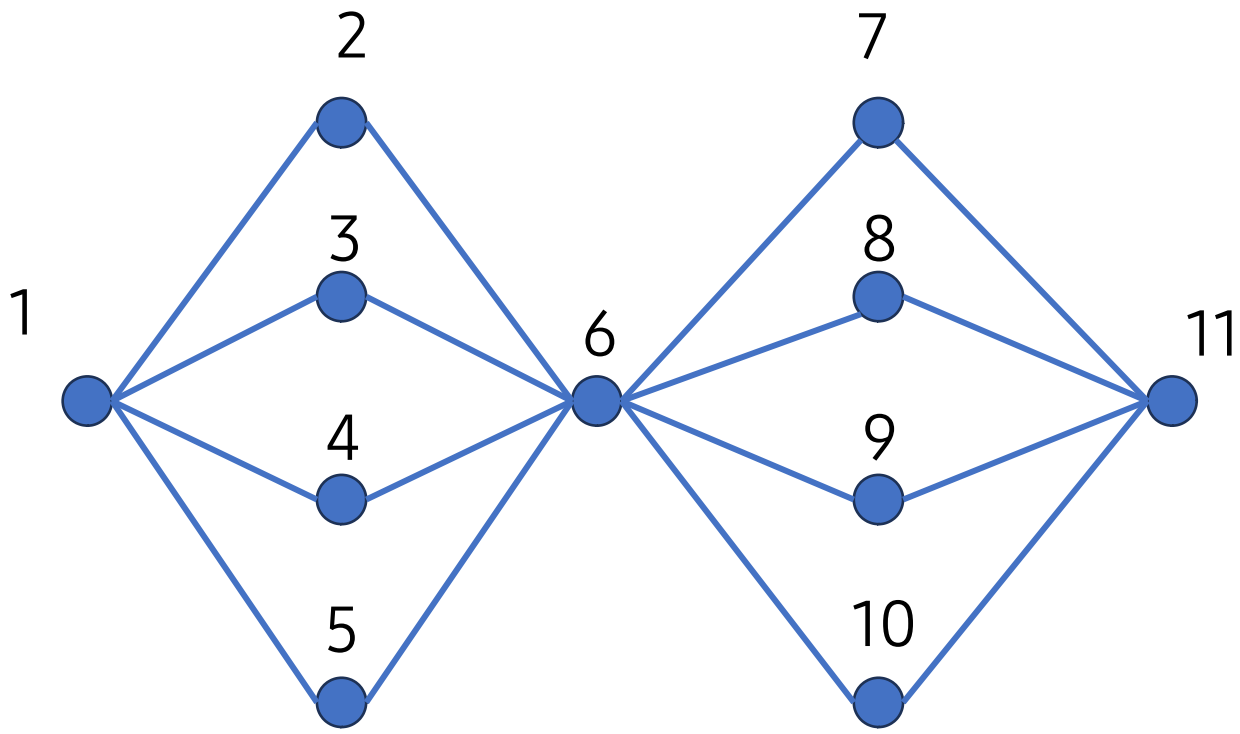
(1, 2), (2, 3), (2, 4), (1, 5), (5, 6)

(1, 2), (2, 3), (2, 4), (1, 5), (1, 6), (5, 7)

II. Euler path finding

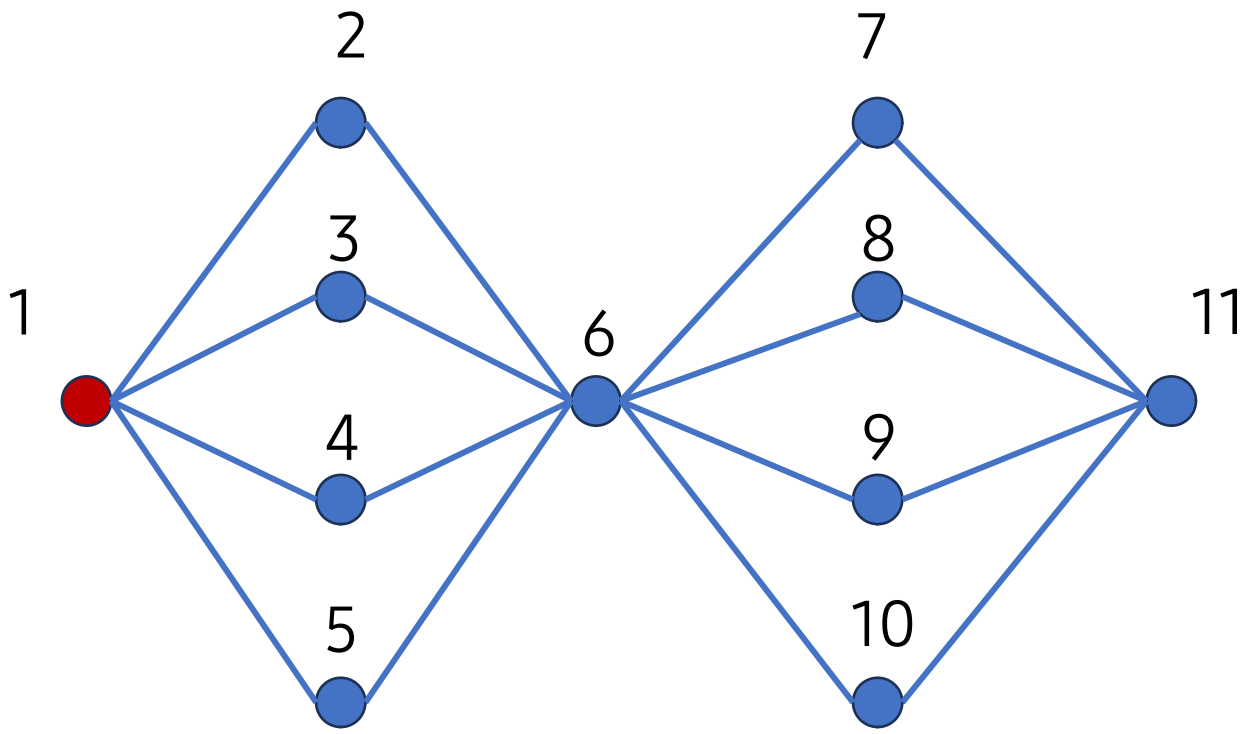
Lesson review

What is an Euler path?


1
2
3
4
5
6
7
8
9
10
11

Preparation

1. Make sure all vertices are even degrees or there are exactly 2 of them with odd degrees.
2. Choose starting vertex as which can be random or one of 2 odd deg vertex.
3. Let as visited vertex.



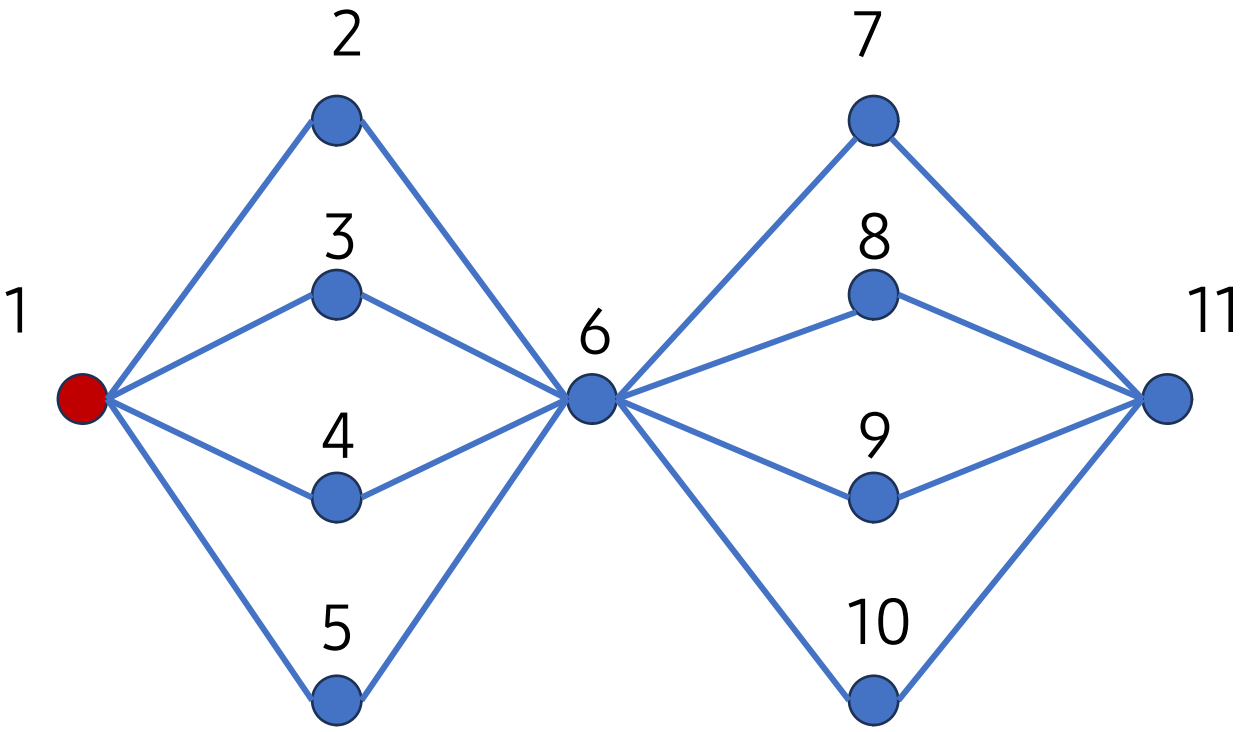
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{ 1 \}$$

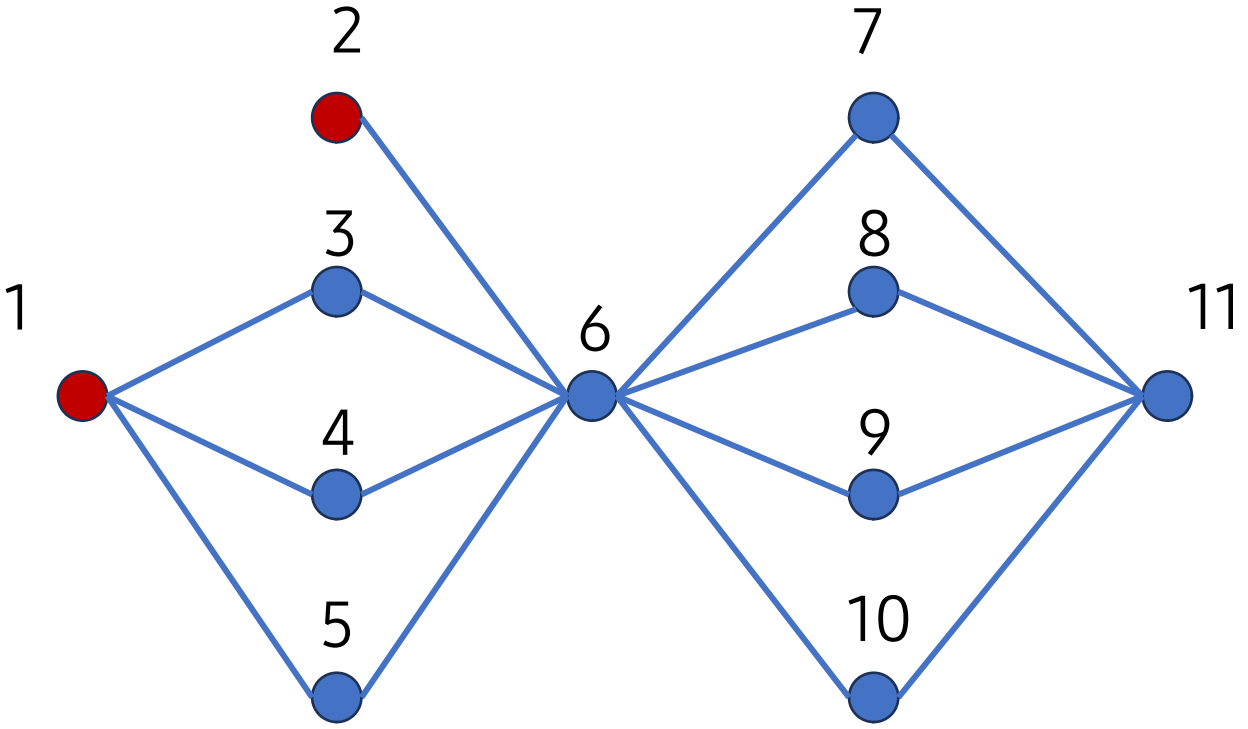
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1, 2\}$$

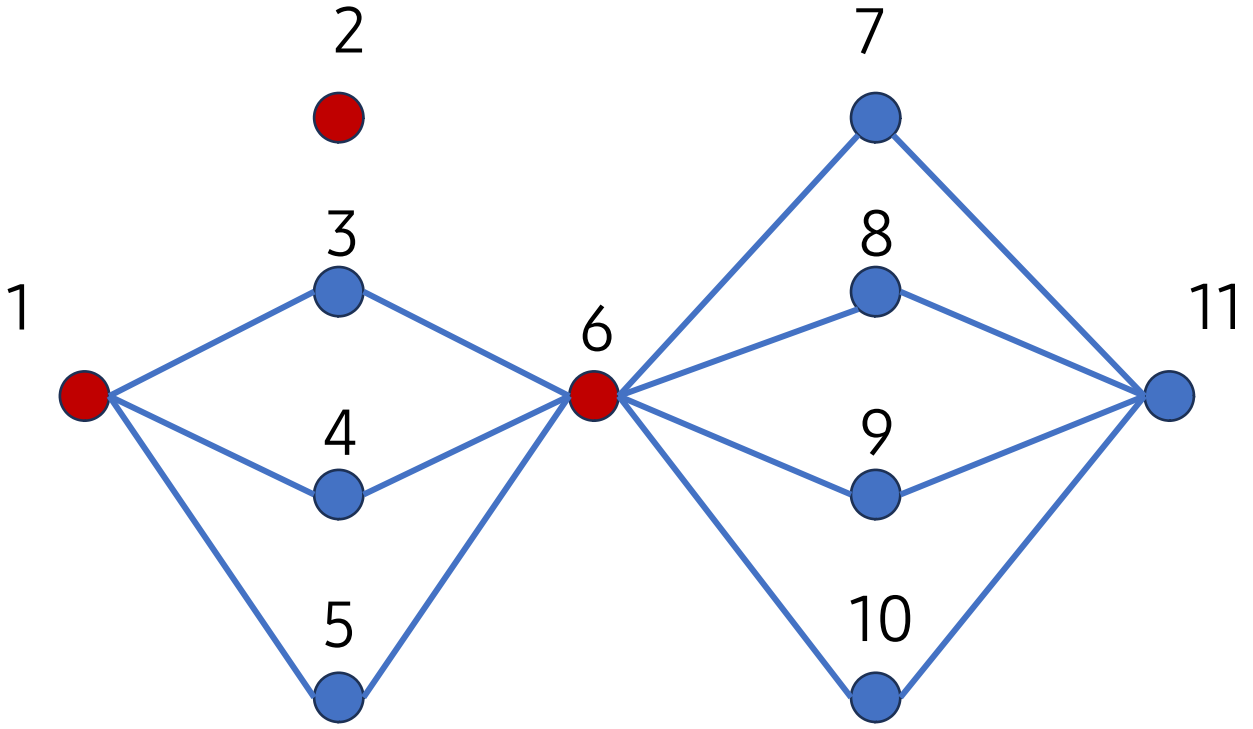
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1, 2, 6\}$$

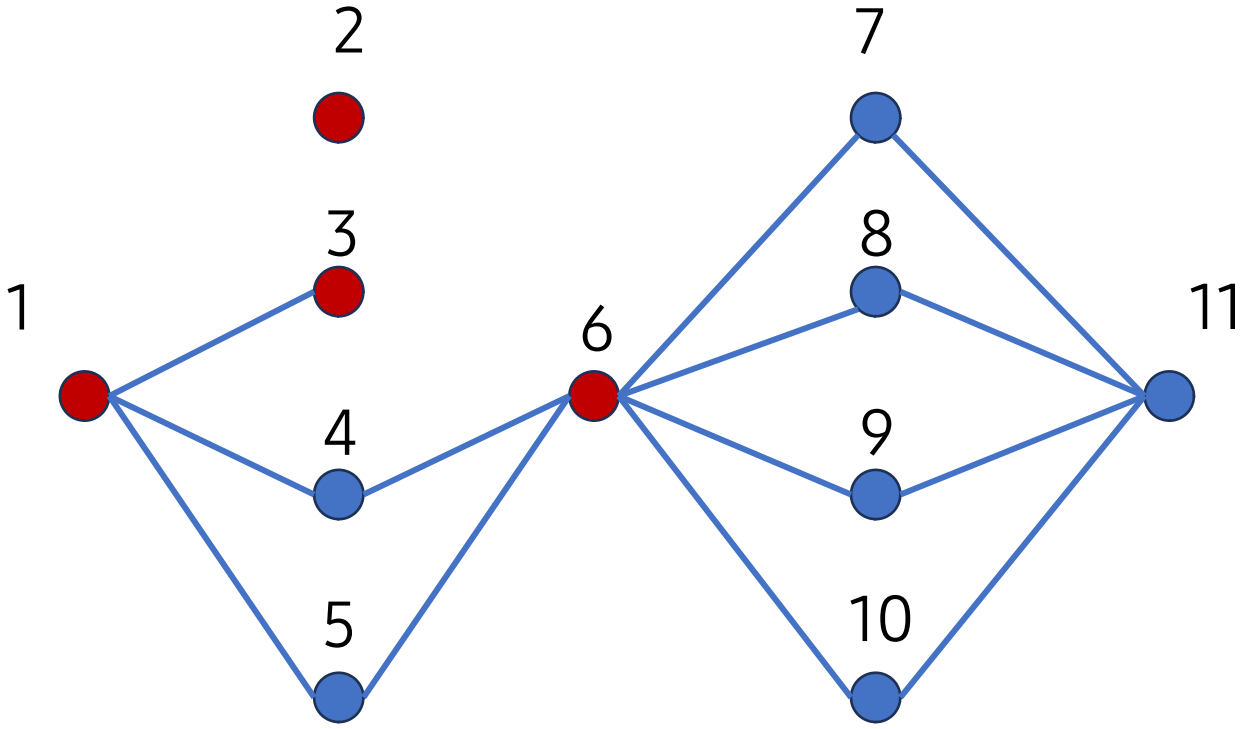
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{ 1,2,6,3 \}$$

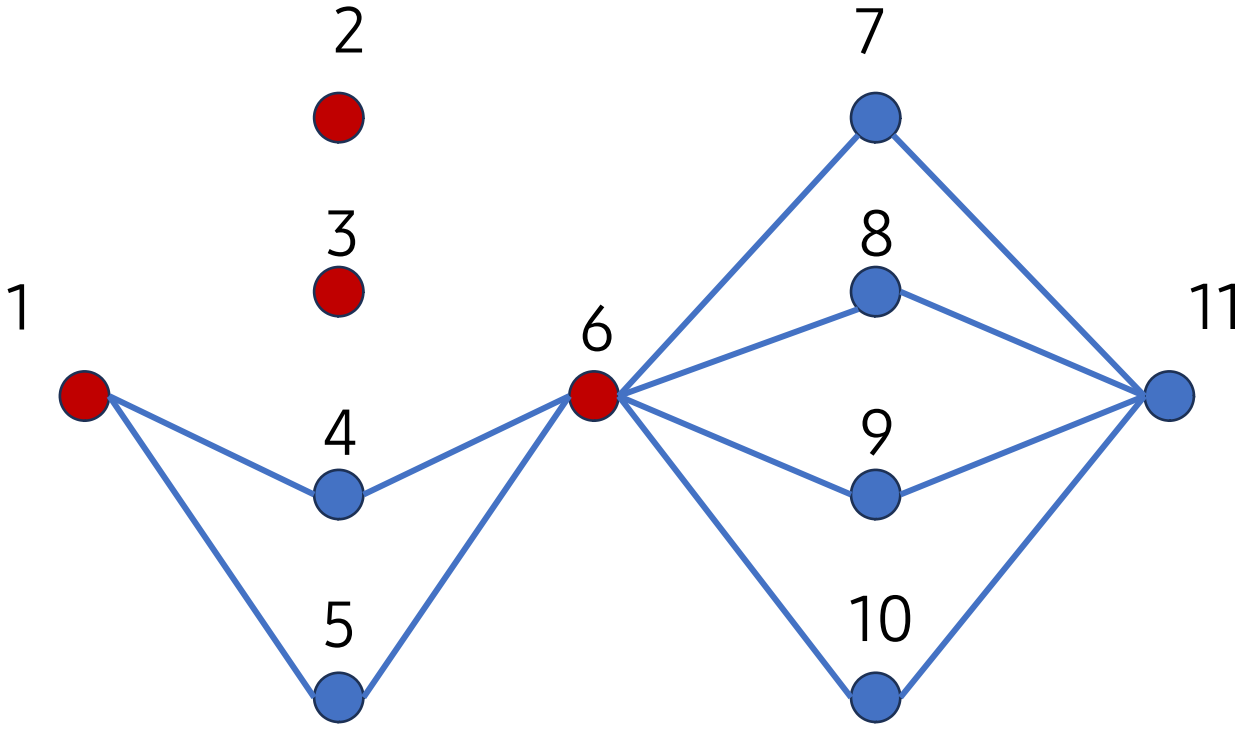
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1,2,6,3,1\}$$

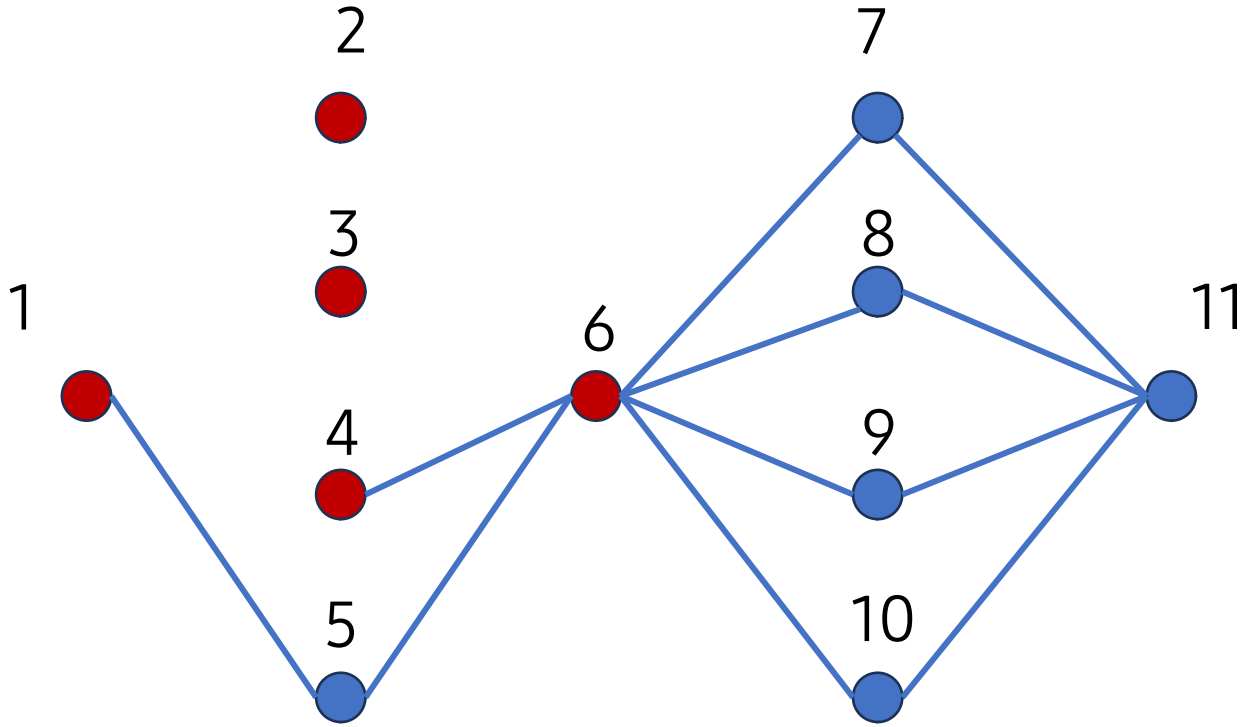
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1,2,6,3,1,4\}$$

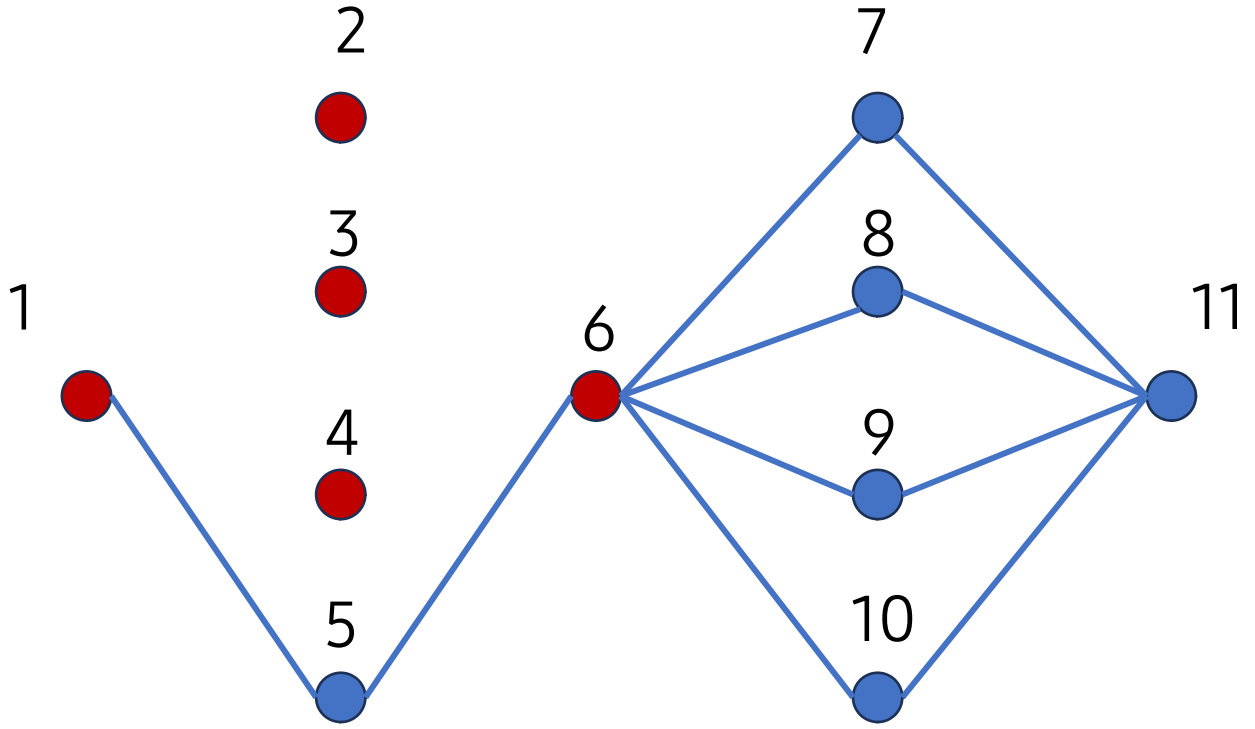
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1,2,6,3,1,4,6\}$$

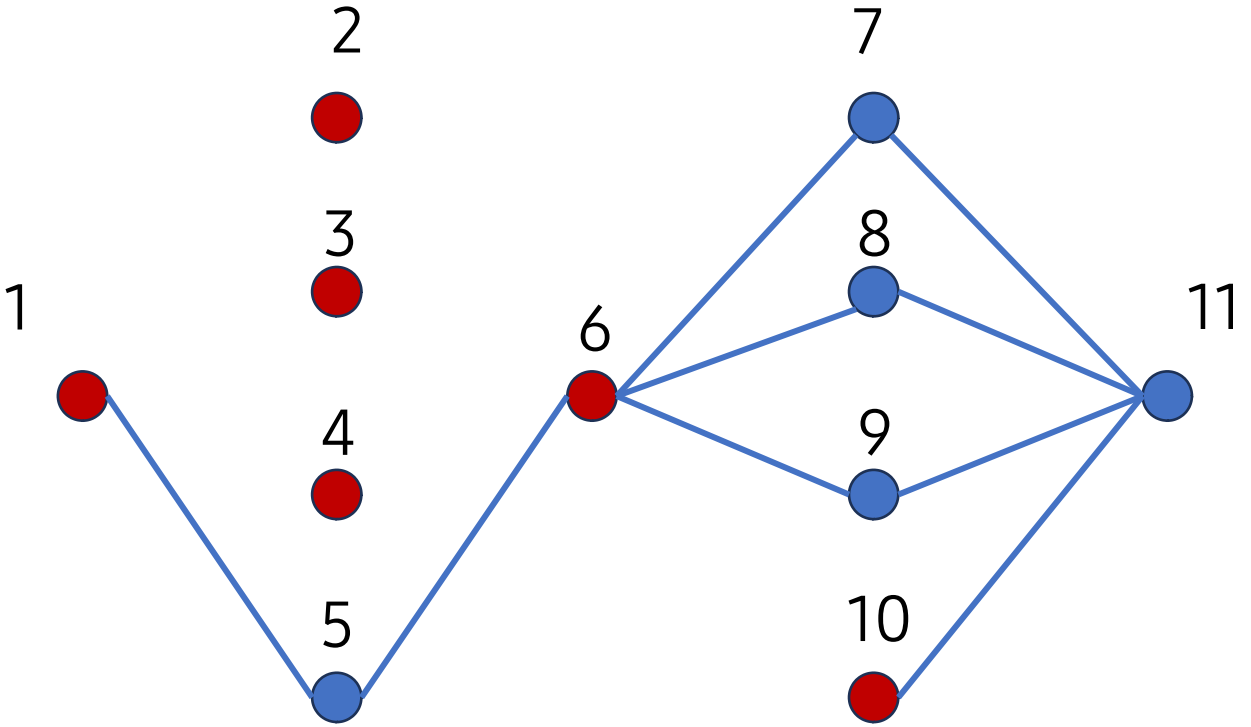
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



$$vs = \{1,2,6,3,1,4,6,10\}$$

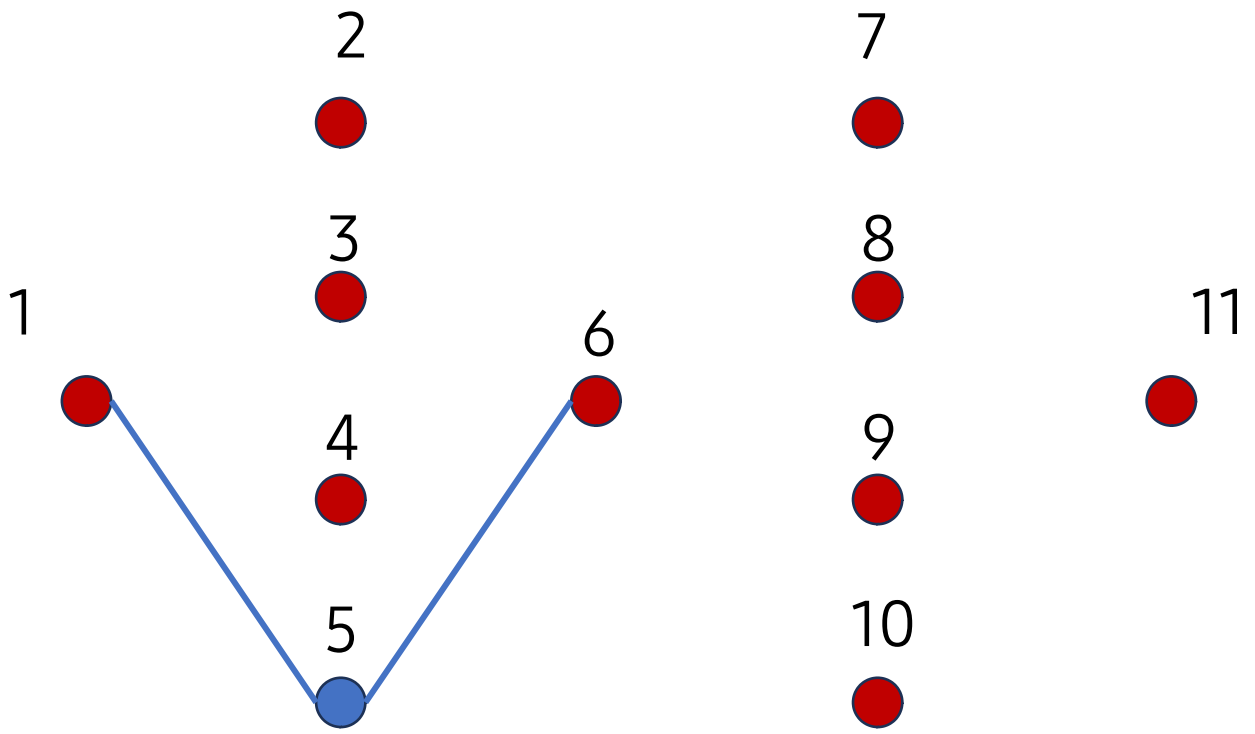
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



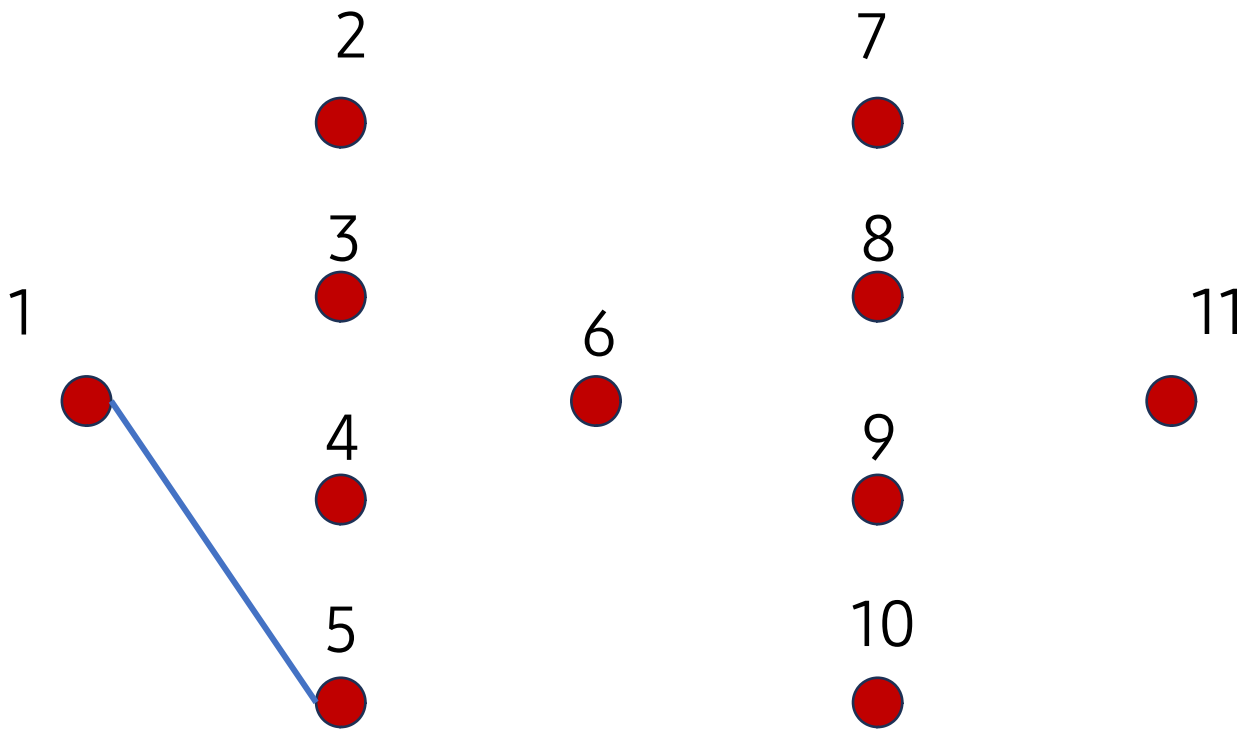
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



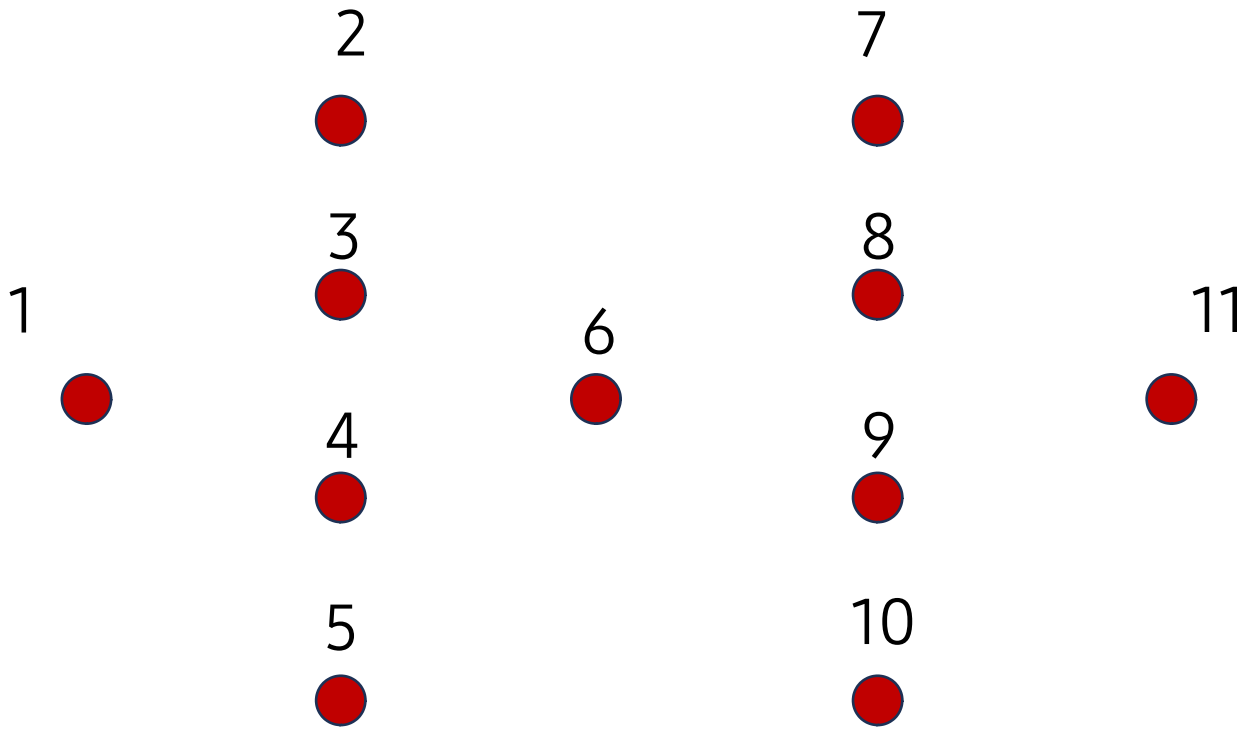
Action!

Step 1: Add to .

Step 2: Find which connects to and the edge between them is not bridge.

Step 3: If exists, remove the edge, set and go to step 1.

Step 4: If not, stop.



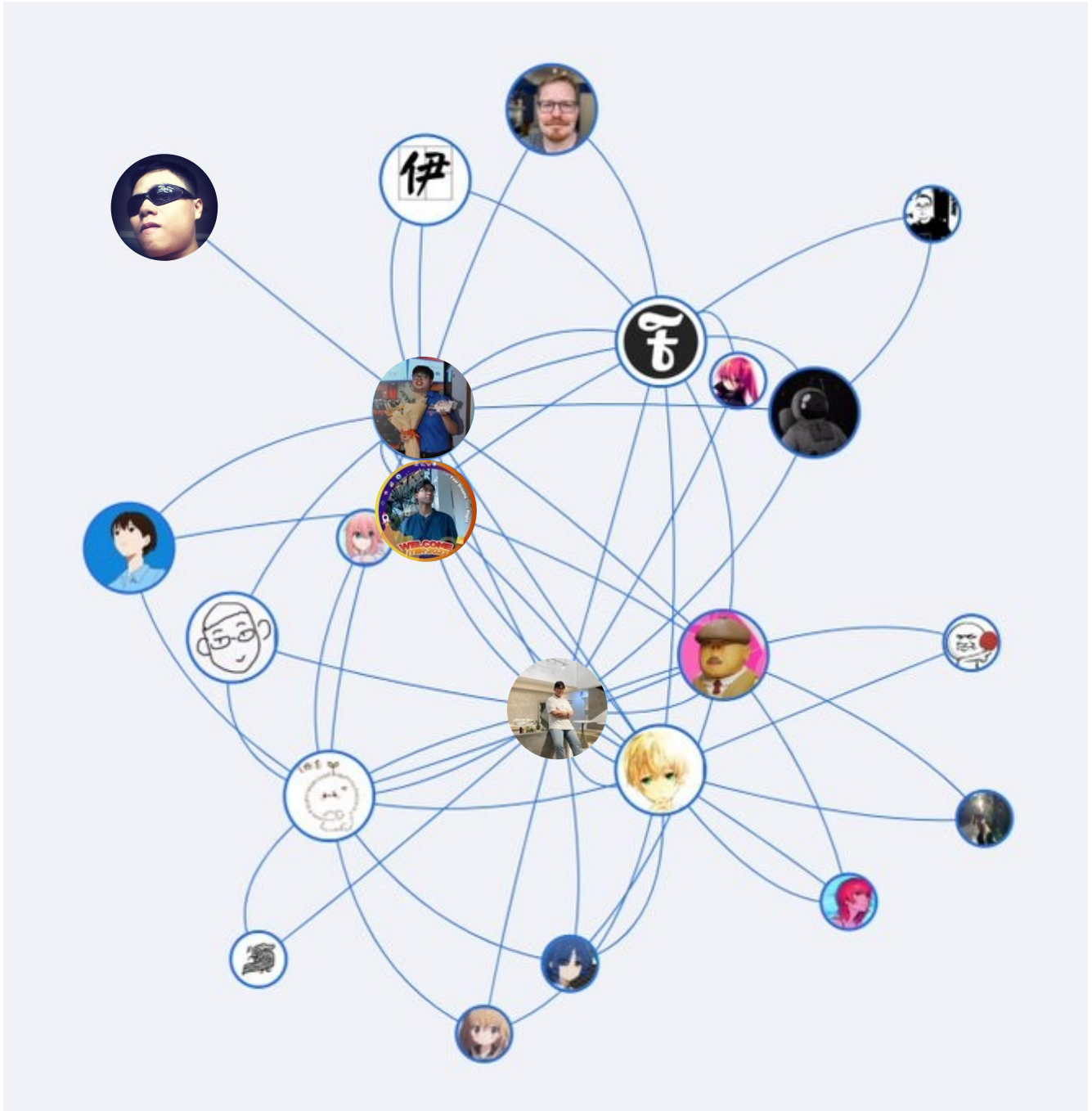
Shortest path searching

A shortest path is a minimum-weight path between two specified vertices and .

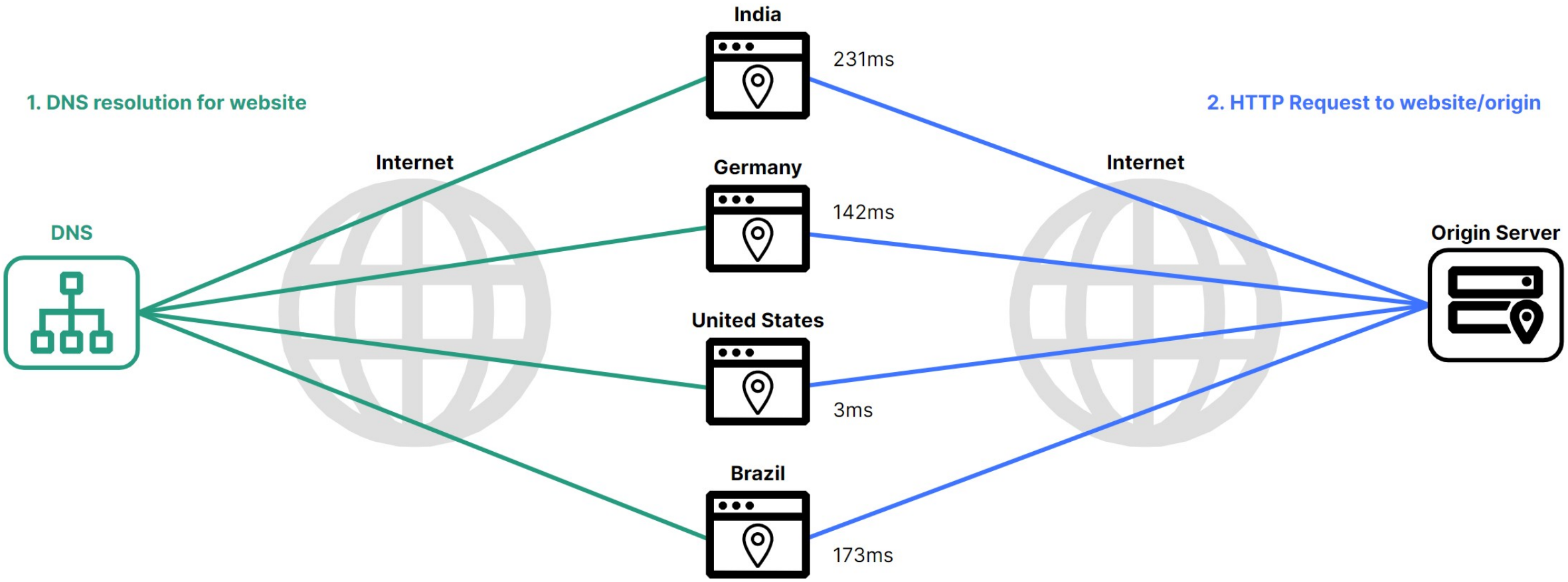
p. 150, Graph Theory by Adrian Bondy

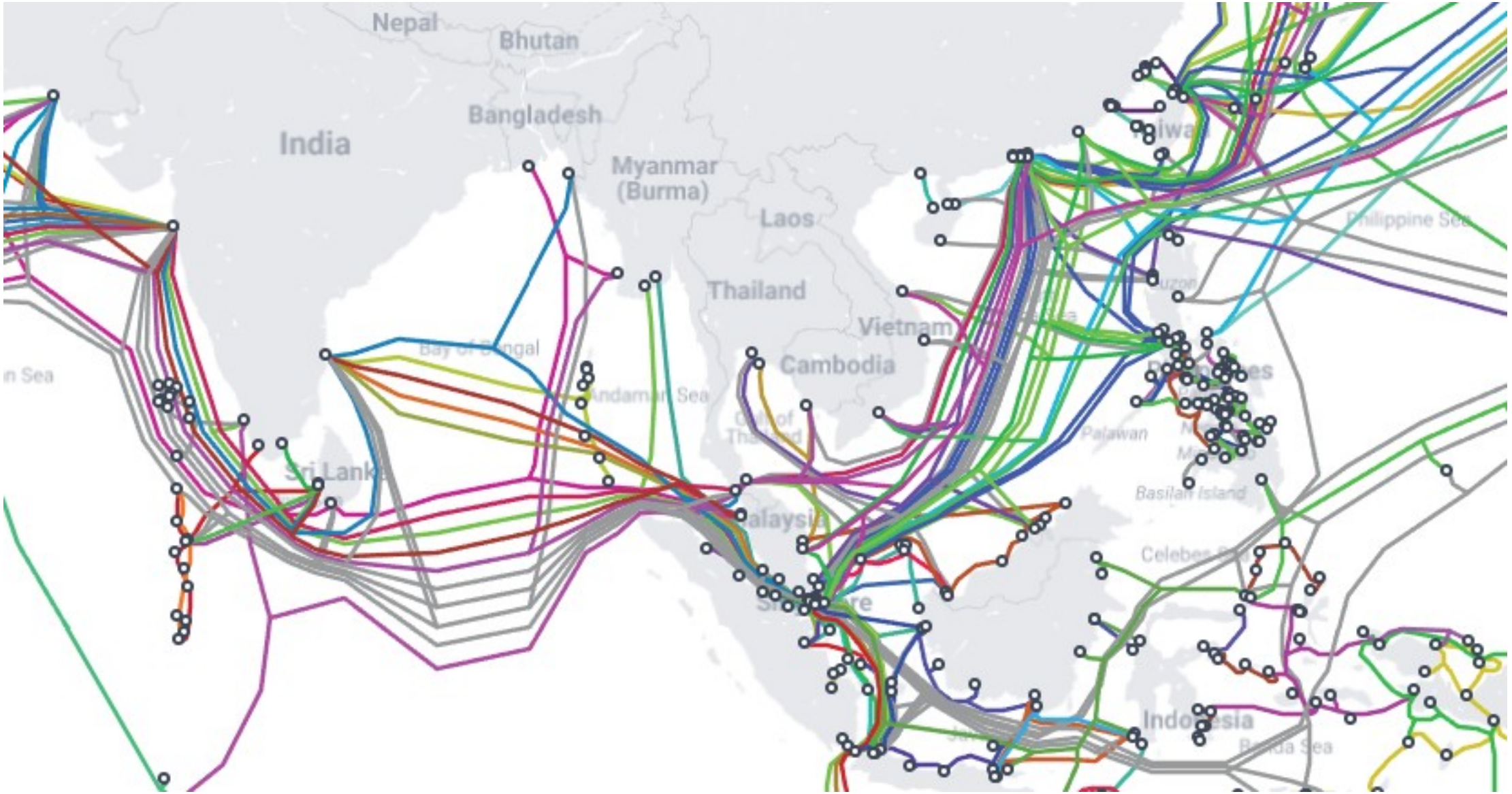
Why shortest?

Friends recommendation



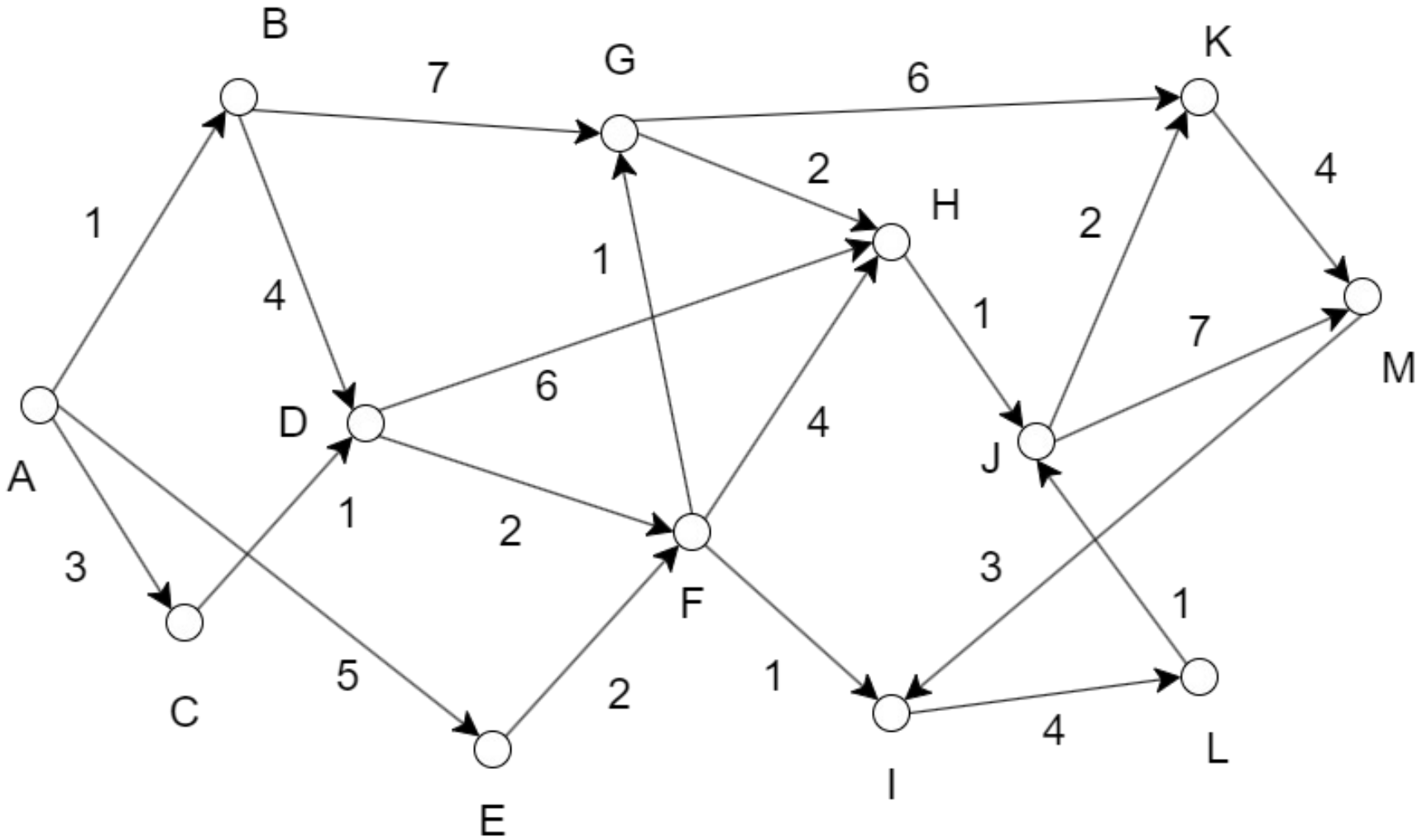
CDN Edge Servers



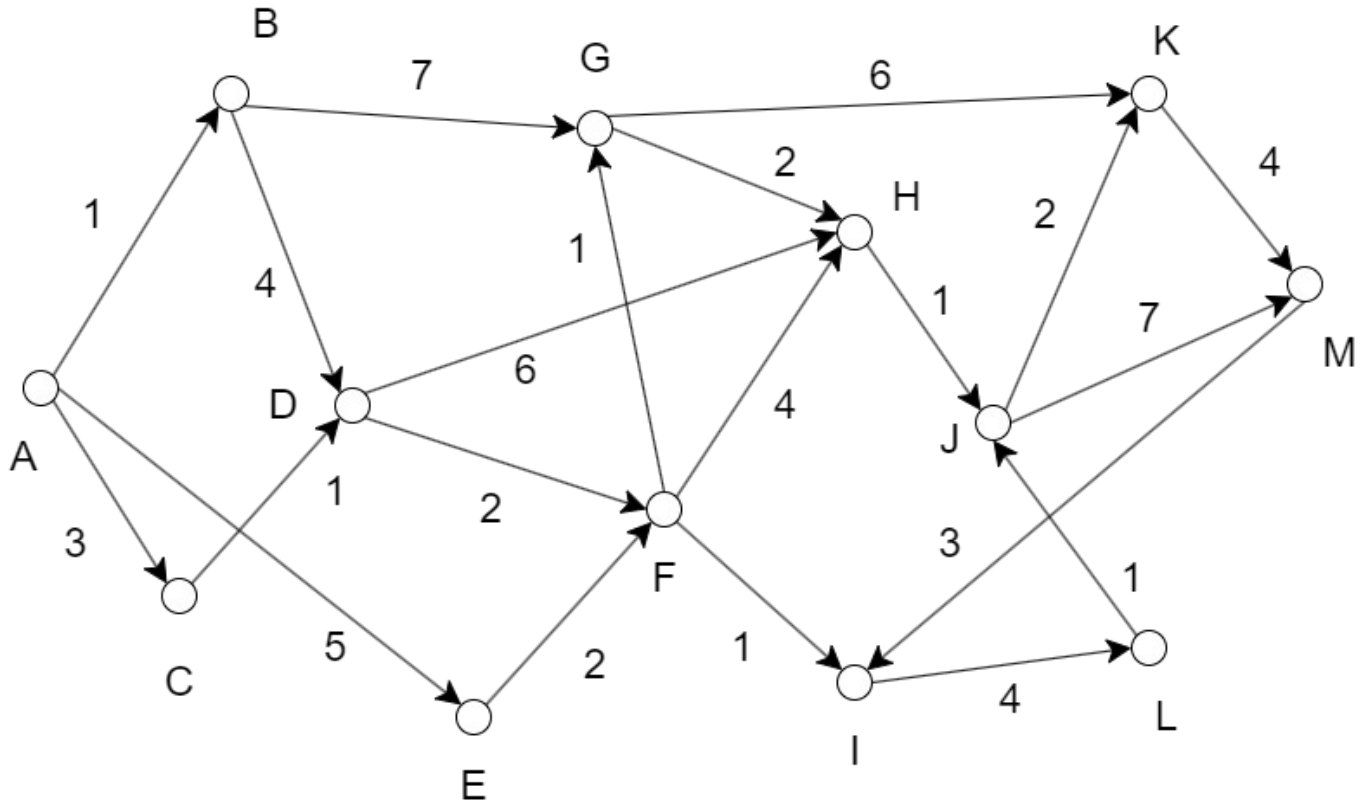


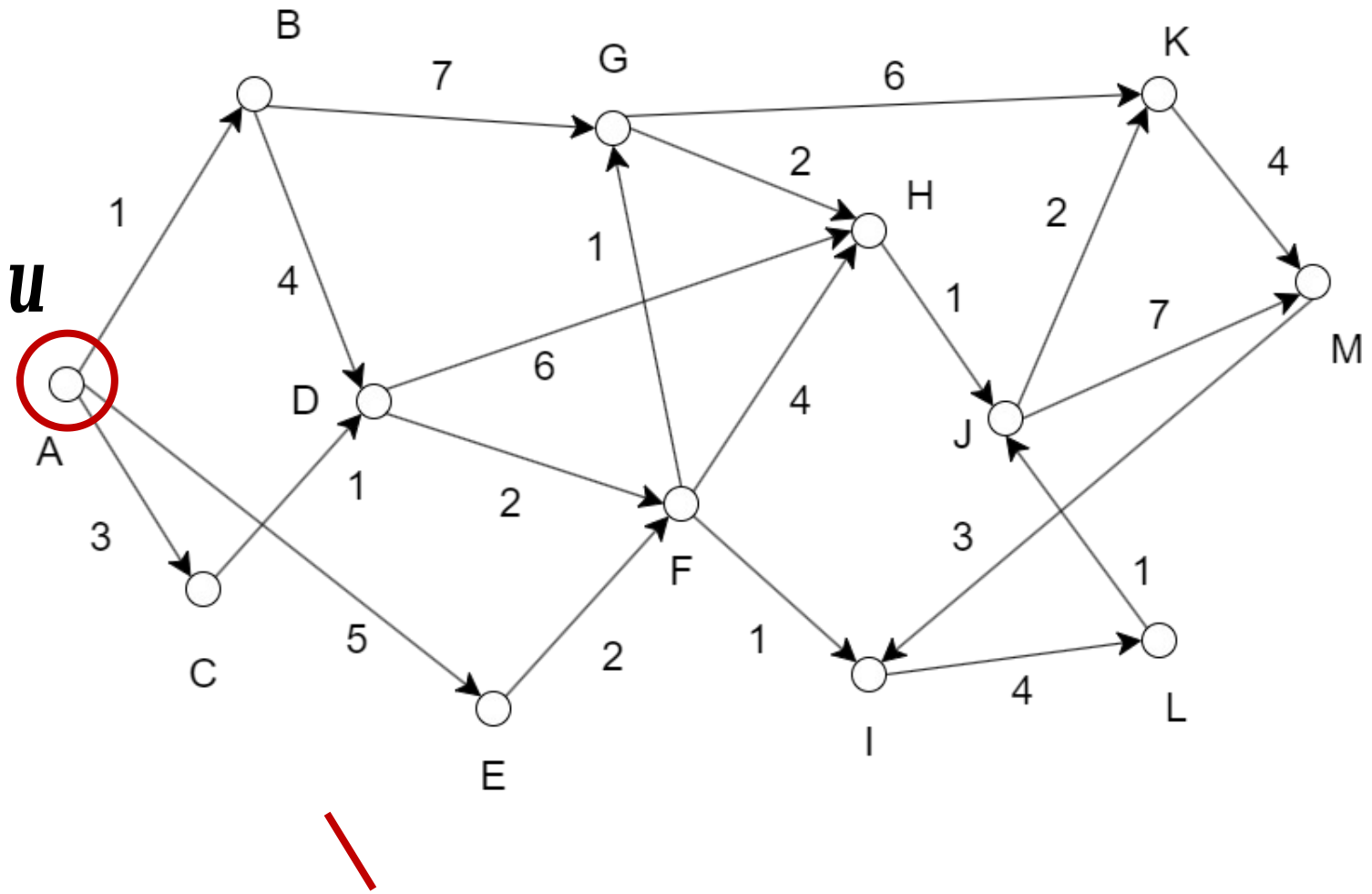
https://www.submarinecablemap.com/

Problem #1: Find shortest path from A to M



Dijkstra method: Find shortest from A to M



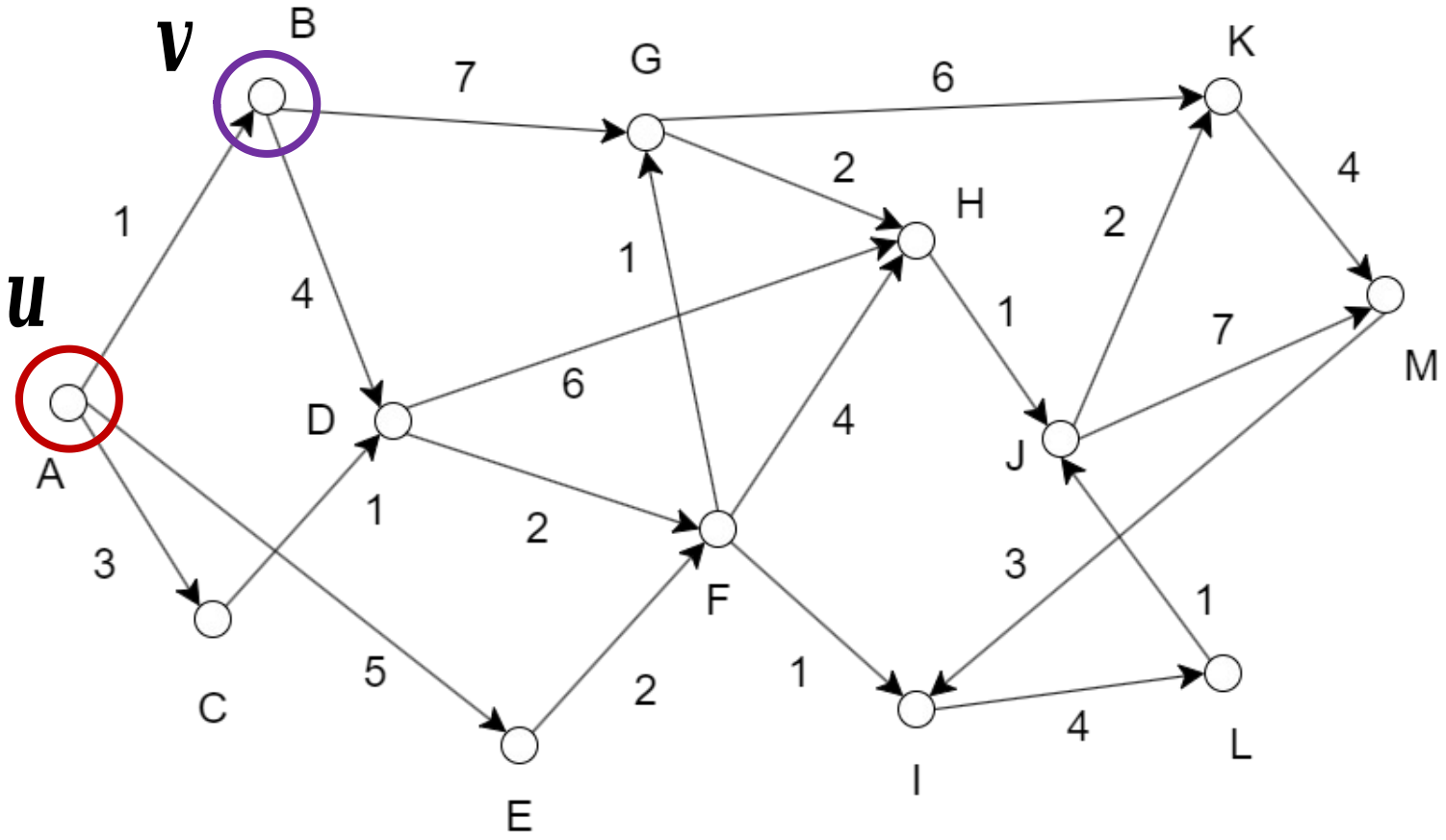


Step 1: Find so that $D[u] = ¿ min\ (D)$

Step 2: If doesn't exist, stop.

Step 3: If $u = ¿\ M$, stop.

Step 4: Remove u from Q.



Step 5: For each connecting to :

1. $d = D[u] + weight_{u,v}$
2. if $d < D[v]$

$$D[v] = d$$

$$P[v] = u$$

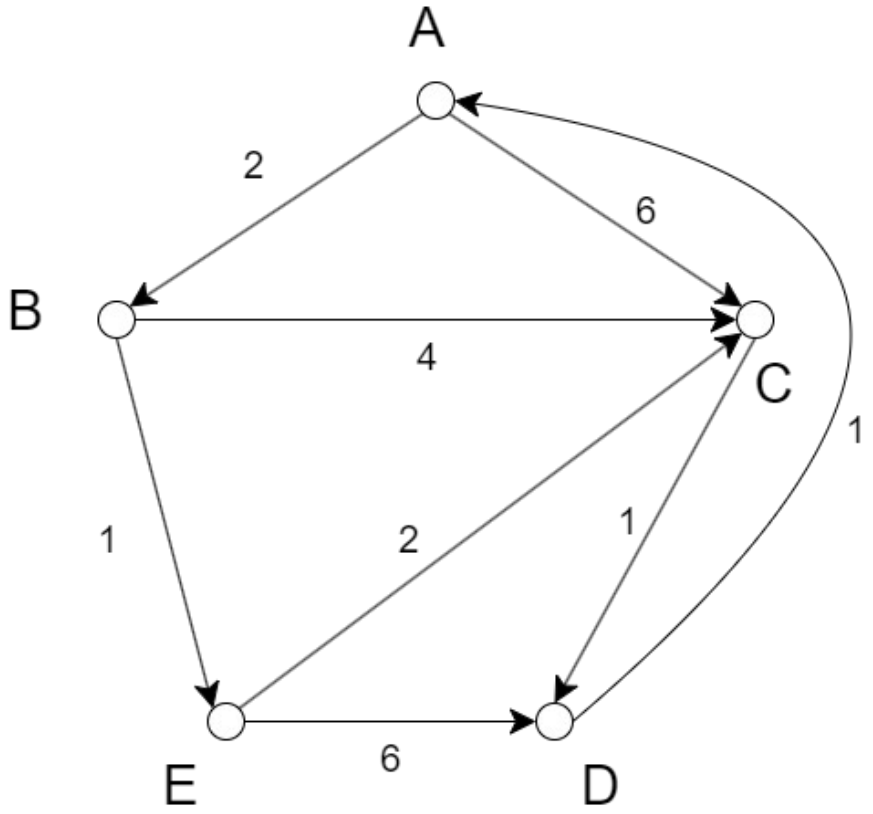
Step 6: Repeat step 1.

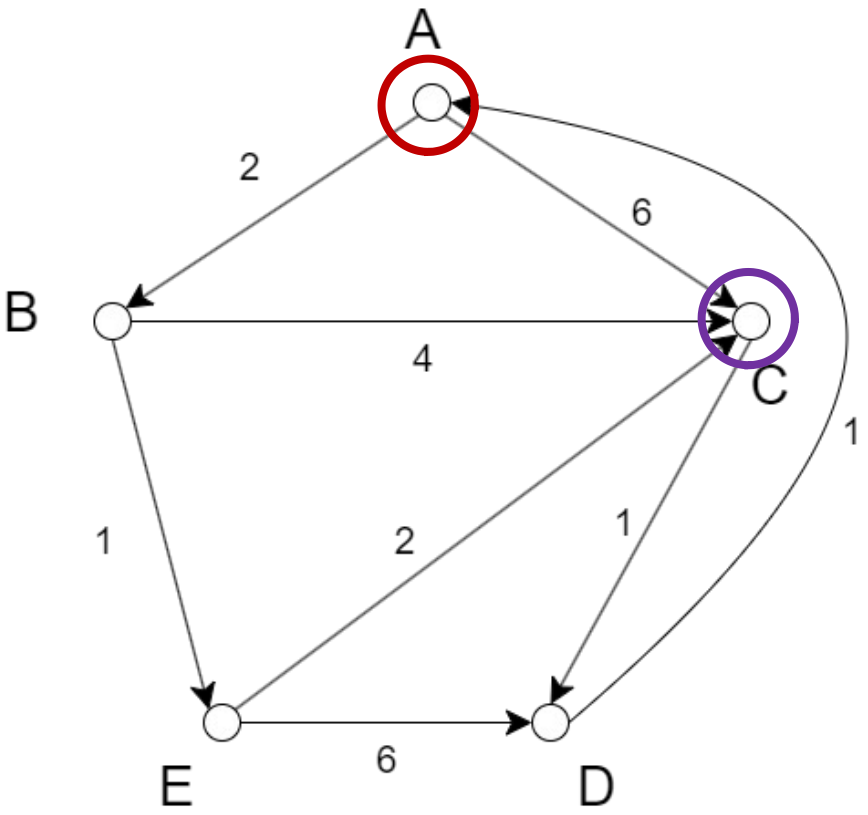
Exercise #2

Complete the below missing values with alphabet order:

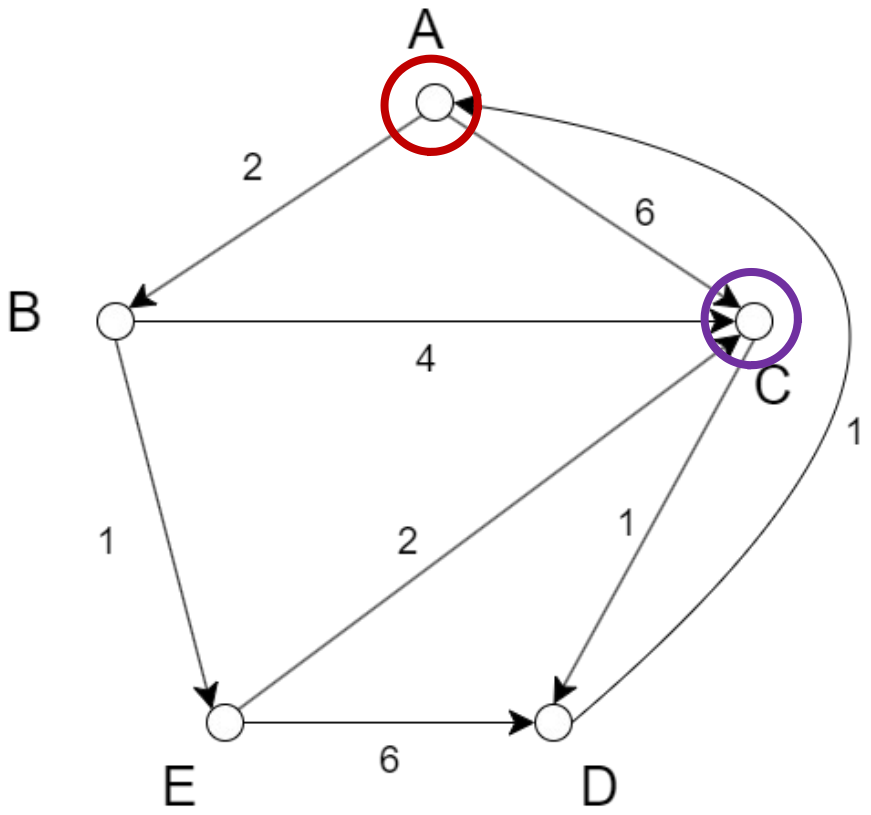
Demonstration on paper

Problem #1: Find the shortest path from A to C.

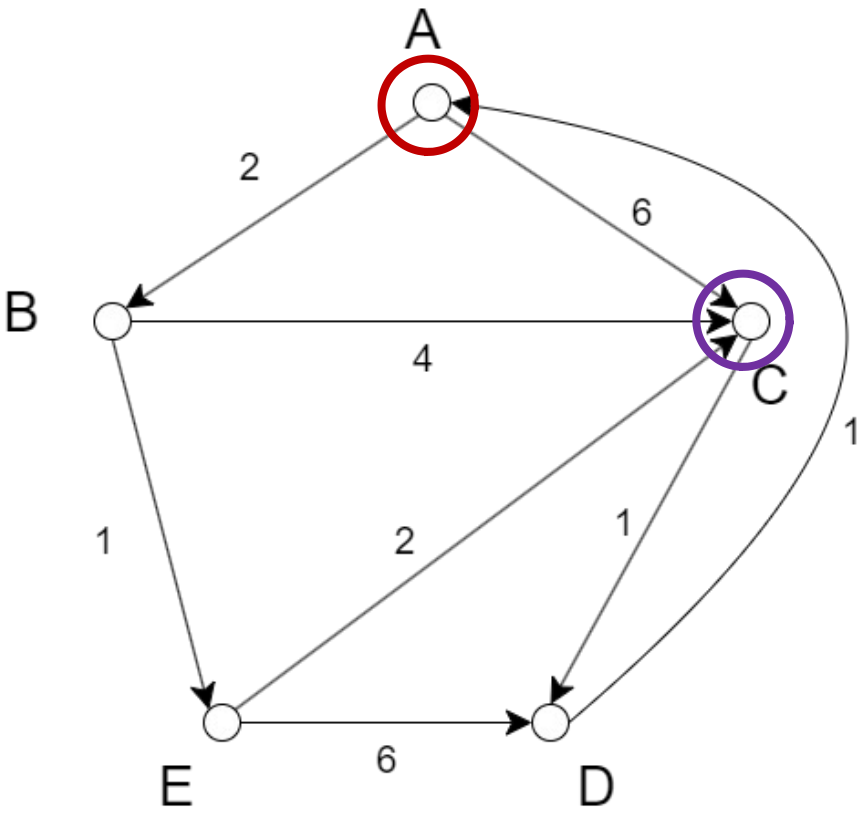




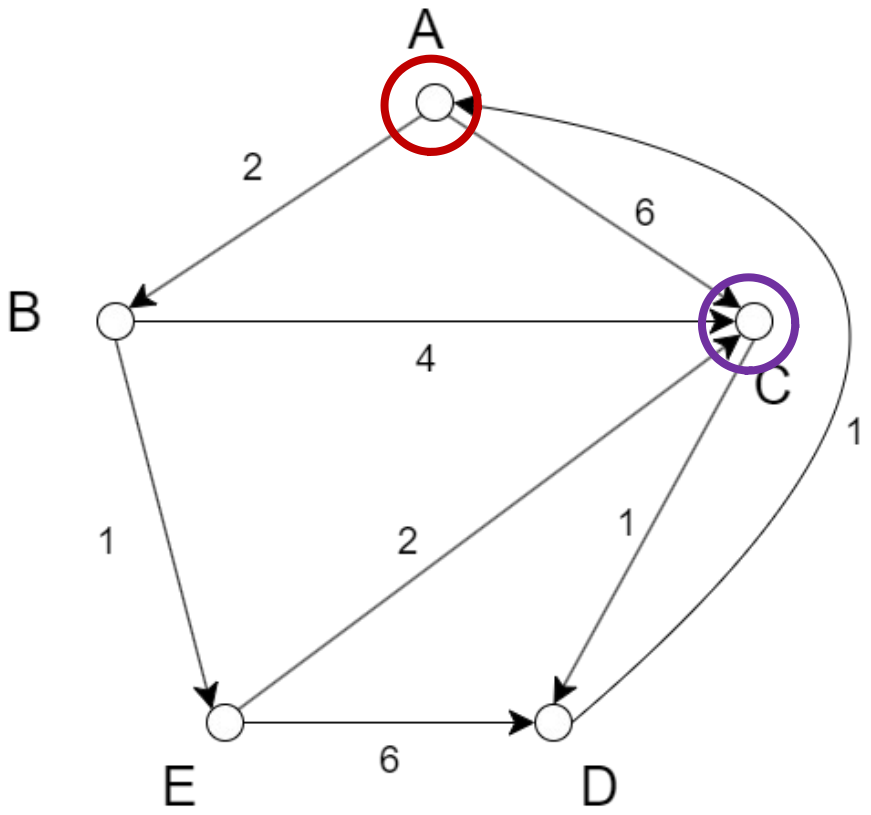
Set Q	B	C	D	E



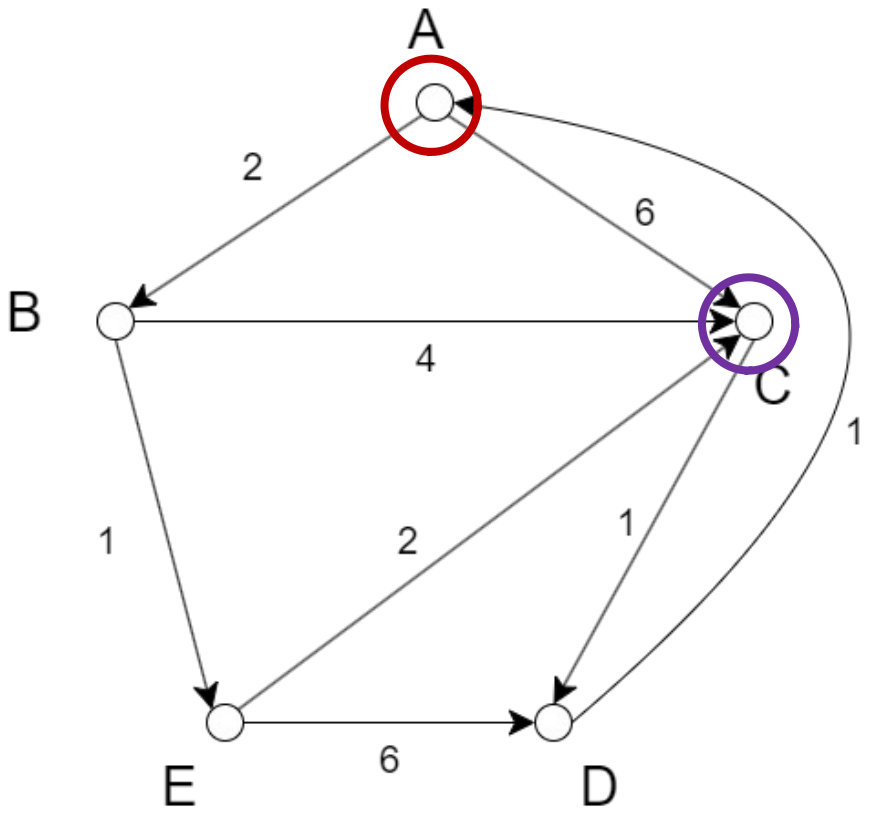
Set Q	B	C	D	E
B, C, D, E	2, A	6, A	Inf, A	Inf, A



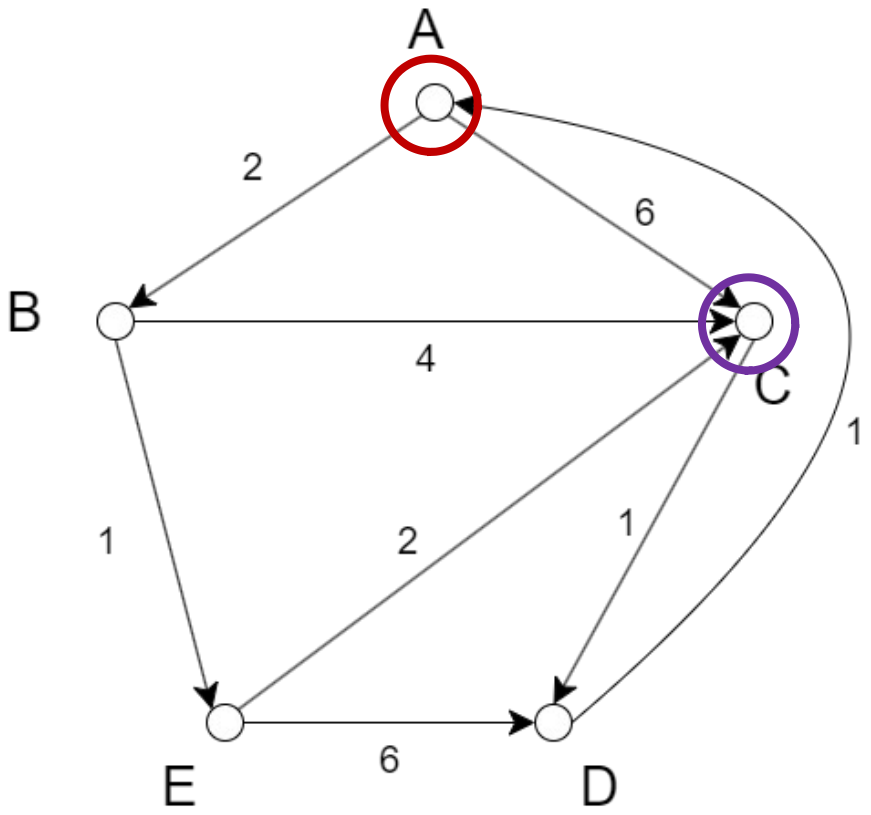
Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, E				3, B



Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, ~~E~~				3, B
C, D		5, E	9, E	



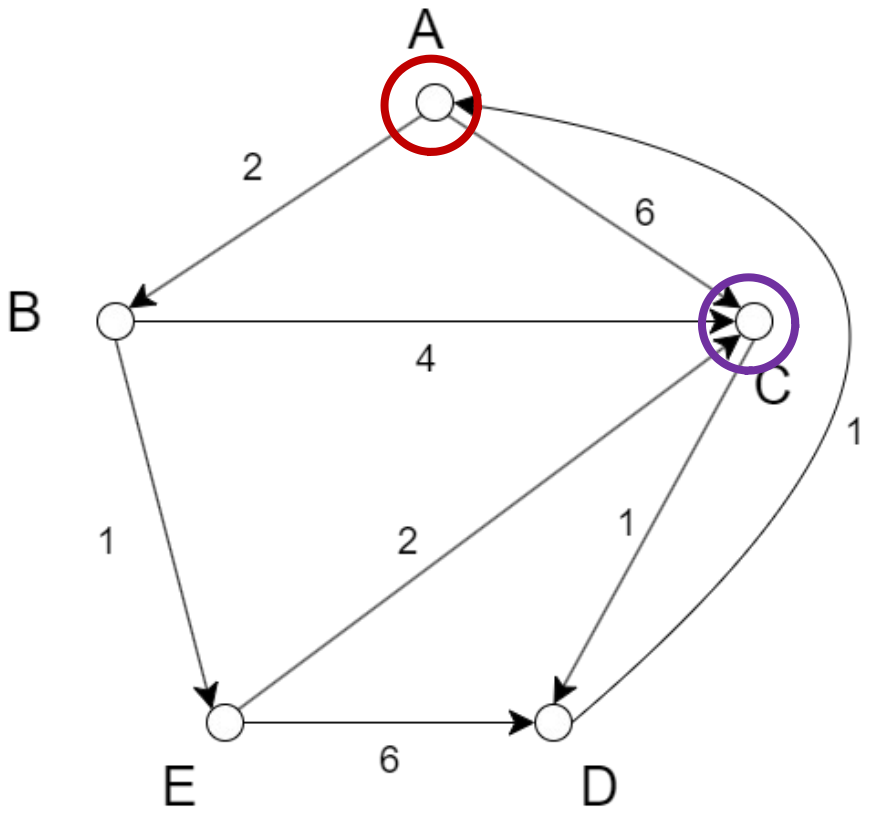
Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, ~~E~~				3, B
~~C~~, D		5, E	9, E	
D				



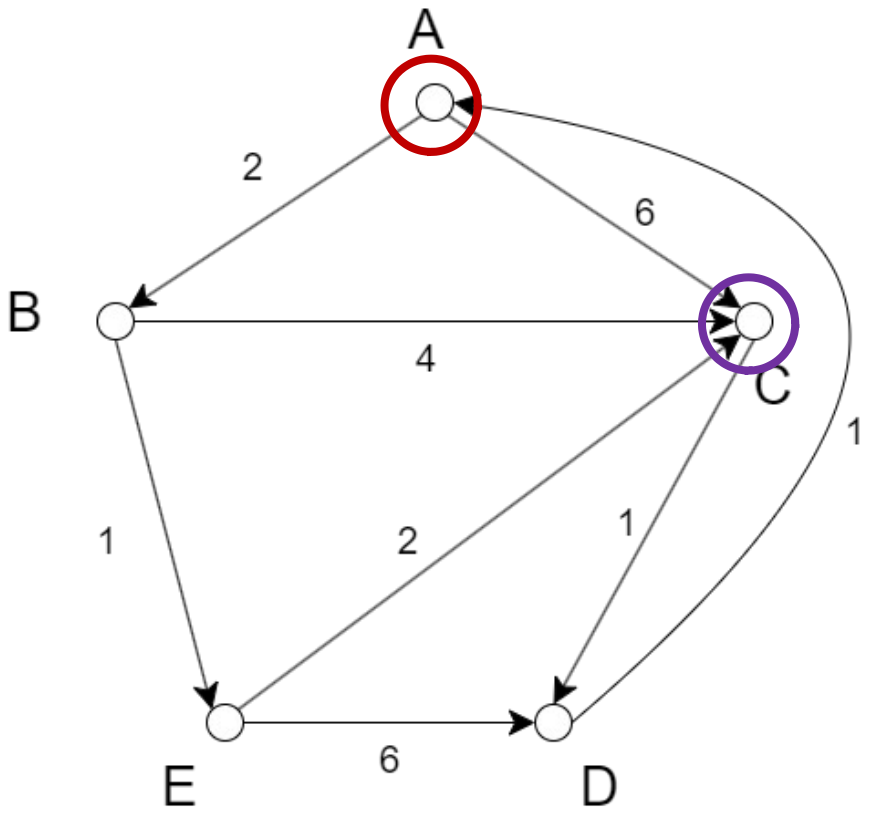
Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, ~~E~~				3, B
~~C~~, D		5, E	9, E	
D	2, A	5, E	9, E	3, B

Problem #3: Find shortest path to all

- **Step 1**: Find so that .
- **Step 2**: If doesn't exist, stop.
- **Step 3**: If , stop.
- **Step 4**: Remove from .

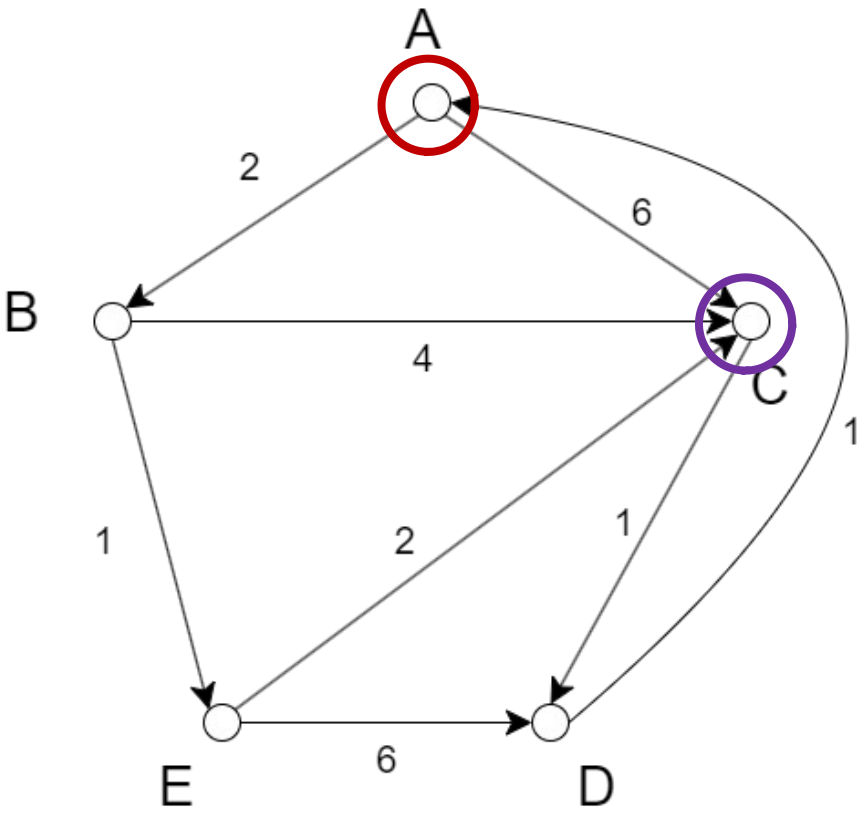


Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
~~C~~, D, E				3, B
~~C~~, D		5, E	9, E	
D			6, C	



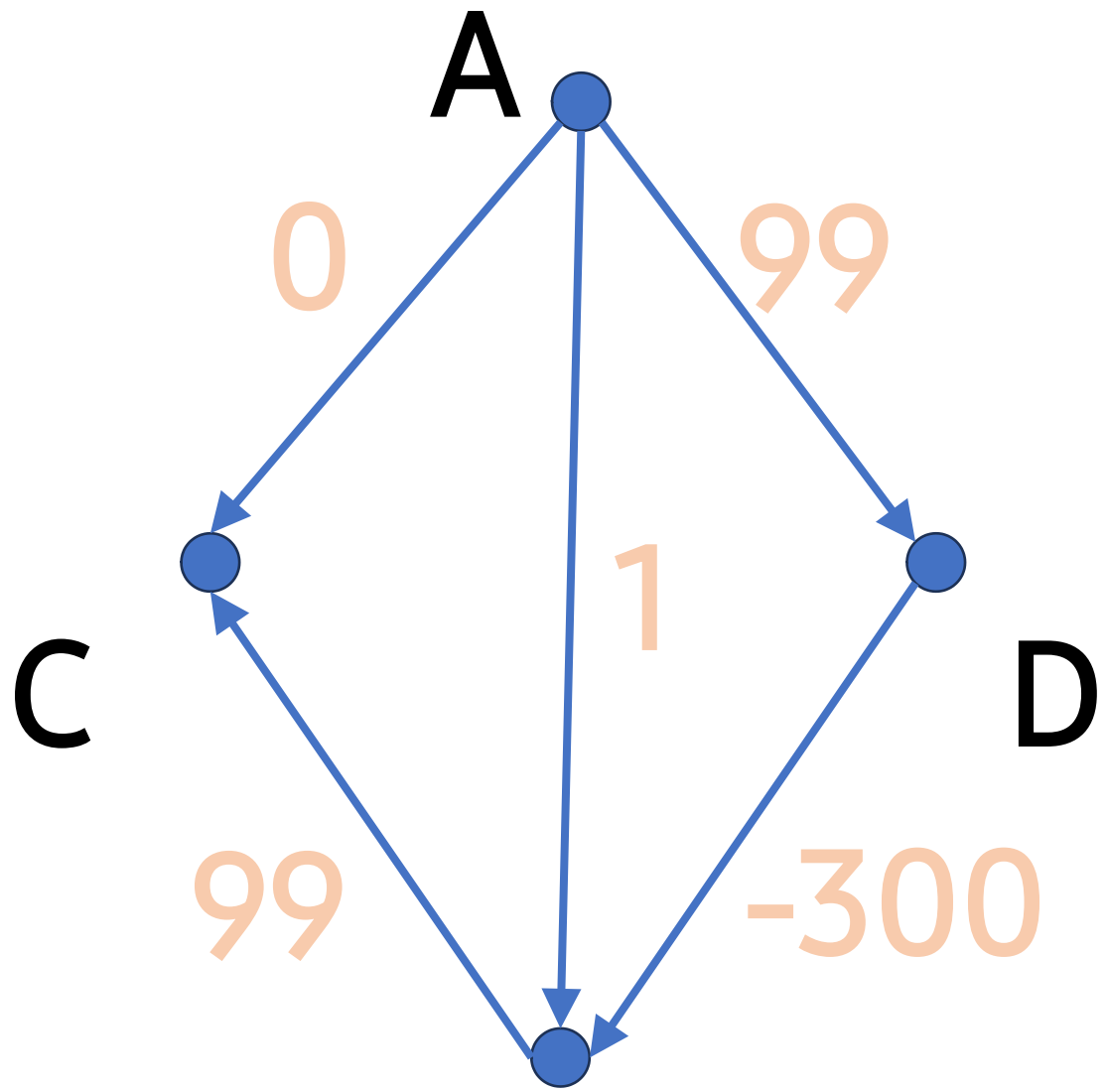
Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, ~~E~~				3, B
~~C~~, D		5, E	9, E	
~~D~~			6, C	
	2, A	5, E	6, C	3, B

Problem #4: Find the shortest path from A to D.

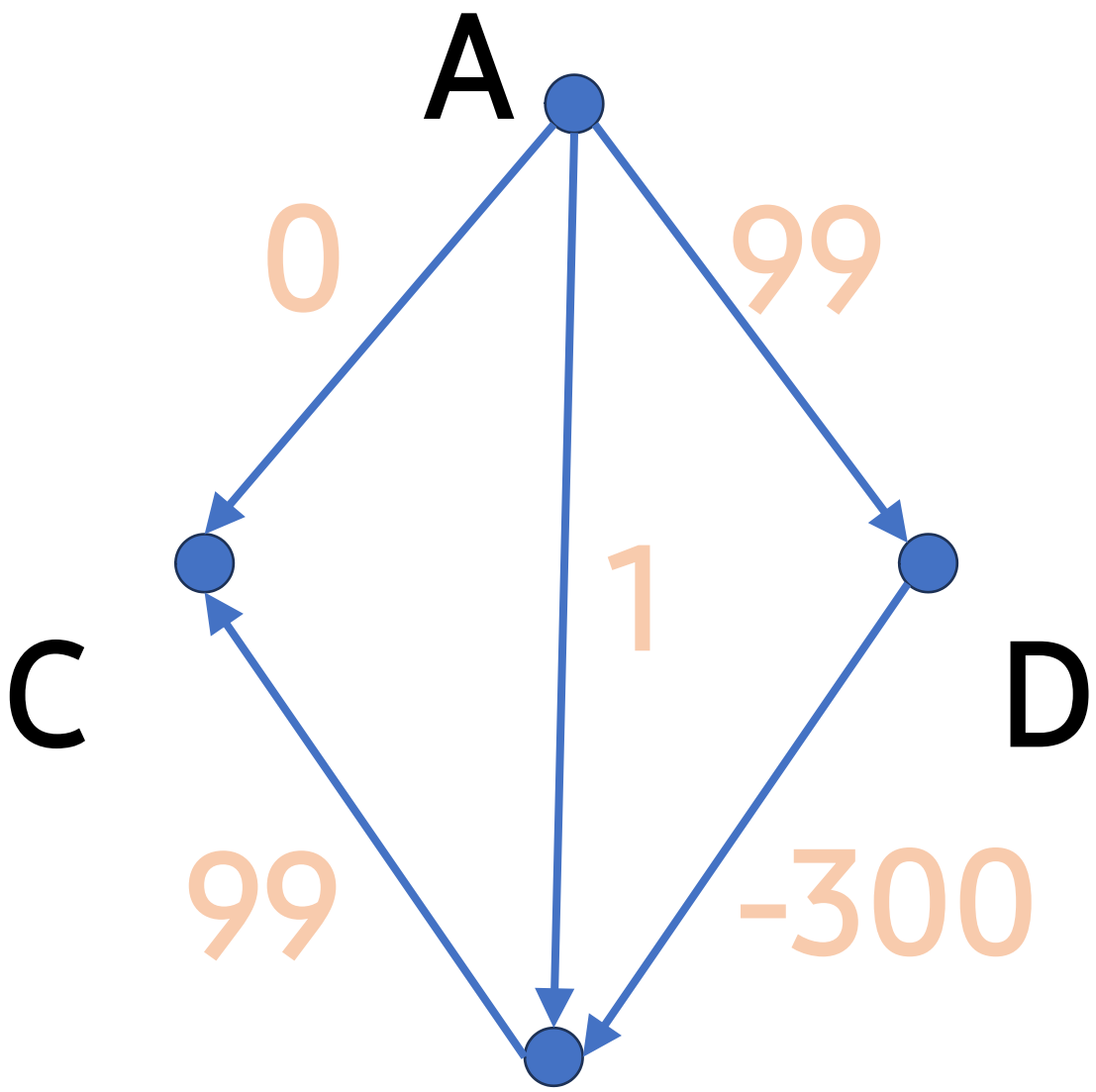


Set Q	B	C	D	E
~~B~~, C, D, E	2, A	6, A	Inf, A	Inf, A
C, D, ~~E~~				3, B
~~C~~, D		5, E	9, E	
~~D~~			6, C	
	2, A	5, E	6, C	3, B

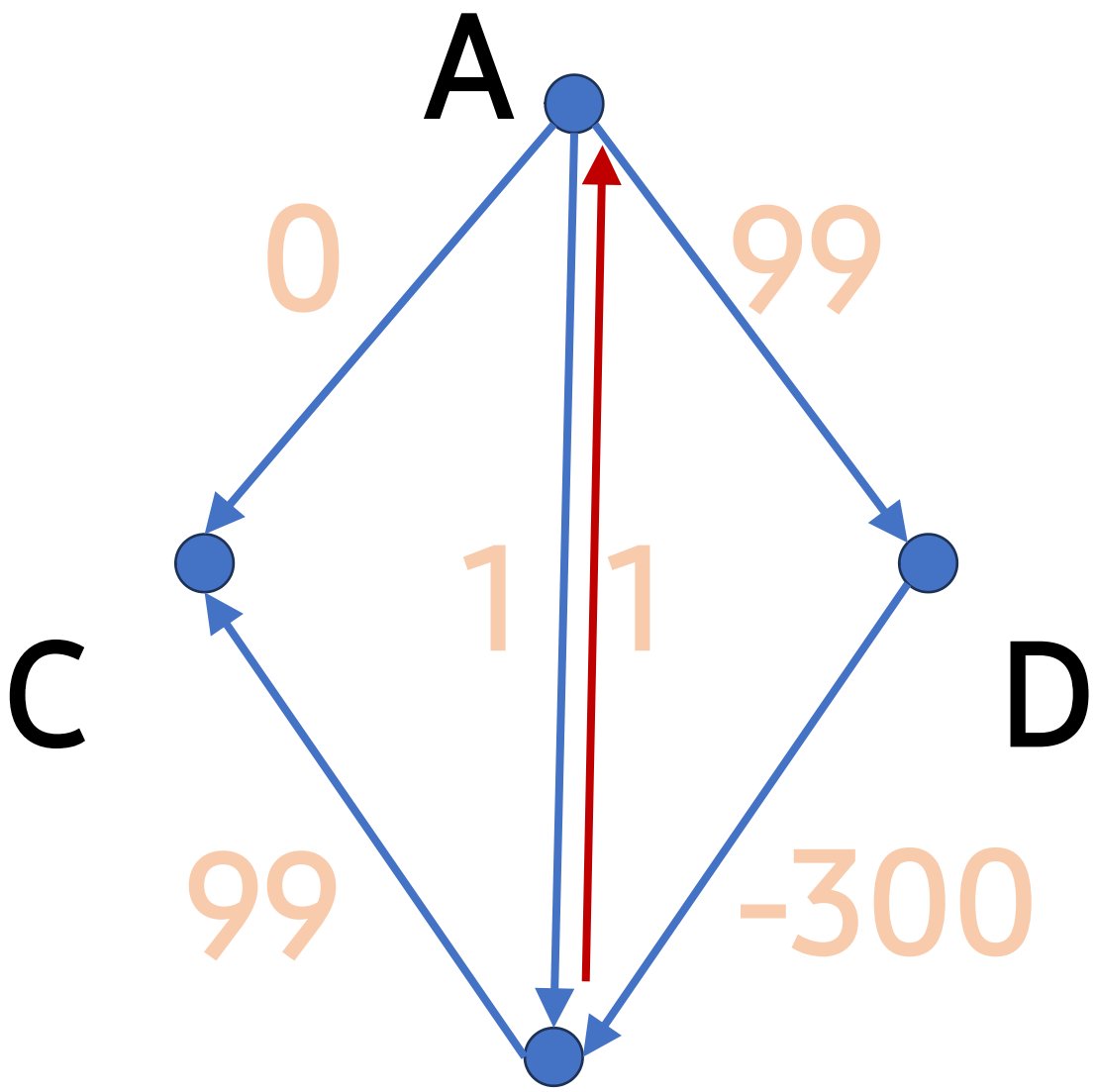
What do you think about Dijkstra algorithm?



Exercise #5: Find the shortest path from A to all using Dijkstra.



Set Q	B	C	D
B, C, D	1, A	0, A	99, A
B, D			
D			
	-201, D	0, A	99, A



Set Q	B	C	D
B, C, D	1, A	0, A	99, A
B, D			
D			
	-201, D	0, A	99, A

Assumptions

- Paths with same weight sum are all equal.
- All weights must be non-negative.
  - Non-negative cycle.