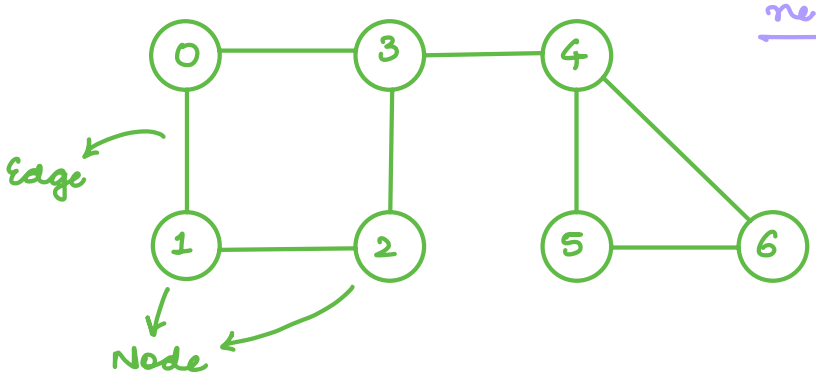
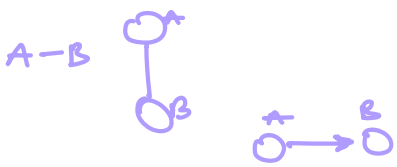
Graph:

↳ Data Structure that consist of vertices/nodes and edges.



- Array
- Stack
- LL
- Trees
  - BT
  - BST
  - AVL
- Graph
- Heap

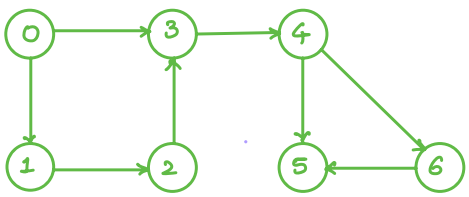


Real life application:

- Paths in a city
- Social network (LinkedIn/Facebook)
  In facebook, each person is represented with a node
  If 2 people are connected then there is an edge b/w them.

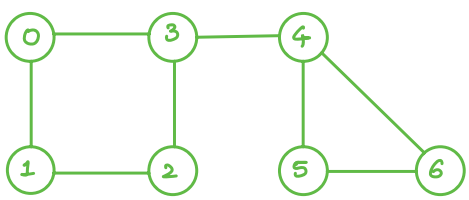
Types of Graph

- Directed Graph



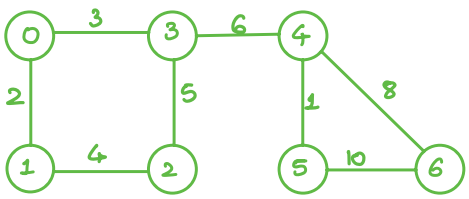
Edge Direction

- Undirected Graph



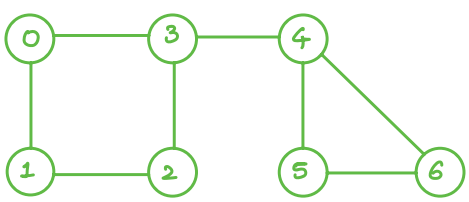
0 → 1
1 → 0

- Weighted Graph

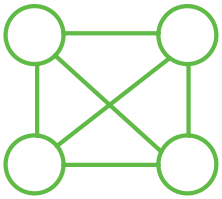


Weight

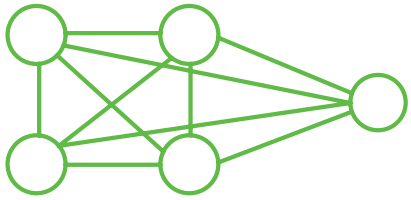
- Un-weighted Graph



Complete Graph



4 nodes, 6 edges → $\frac{4 \times 3}{2} = 6$



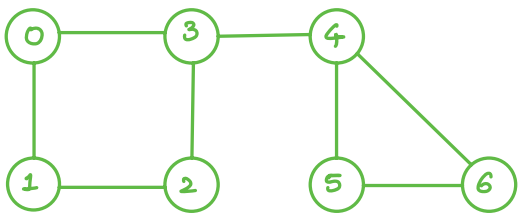
5 nodes, 10 edges → $\frac{5 \times 4}{2} = 10$

n nodes/vertices

no of edges: $nC_2 = \frac{n(n-1)}{2} = O(n^2)$

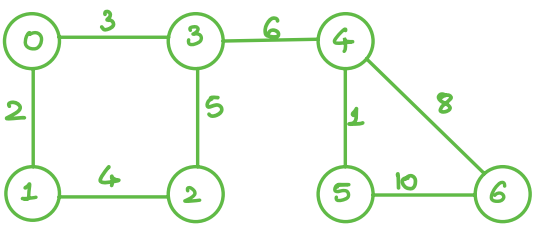
Graph Representation

- Adjacency Matrix



	0	1	2	3	4	5	6
0	0	1	0	1	0	0	0
1	1	0	1	0	0	0	0
2	0	1	0	1	0	0	0
3	1	0	1	0	1	0	0
4	0	0	0	1	0	1	1
5	0	0	0	0	1	0	1
6	0	0	0	0	1	1	0

→ Symmetric Matrix



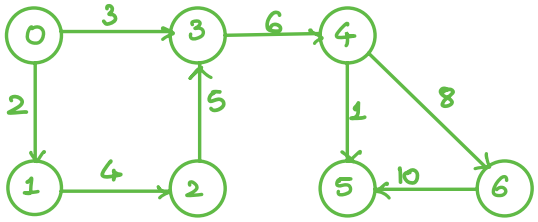
$\frac{16}{2} = 8$

	0	1	2	3	4	5	6	
0	0	2	0	3	0	0	0	: 2
1	2	0	4	0	0	0	0	: 2
2	0	4	0	5	0	0	0	: 2
3	3	0	5	0	6	0	0	: 3
4	0	0	0	6	0	1	8	: 3
5	0	0	0	0	1	0	10	: 2
6	0	0	0	0	8	10	0	: 2

1 → weight

16

→ Symmetric Matrix



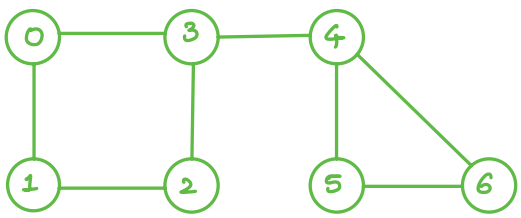
	0	1	2	3	4	5	6	
0	0	2	0	3	0	0	0	: 2
1	0	0	4	0	0	0	0	: 1
2	0	0	0	5	0	0	0	: 1
3	0	0	0	0	6	0	0	: 1
4	0	0	0	0	0	1	8	: 2
5	0	0	0	0	0	0	0	: 0
6	0	0	0	0	0	10	0	: 1

sum

Non Symmetric Matrix

- Adjacency List ← better

graph sparse: less no. of ≠0 entries : saves space

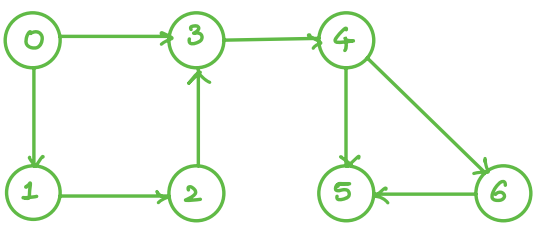


no of edges? $\frac{16}{2} : 8$

0	→	1	3	: 2
1	→	0	2	: 2
2	→	1	3	: 2
3	→	0	2	4 : 3
4	→	3	5	6 : 3
5	→	4	6	: 2
6	→	4	5	: 2

nbrs store

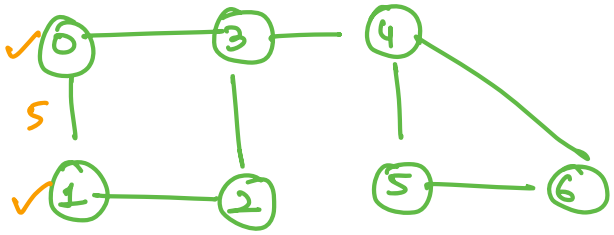
16

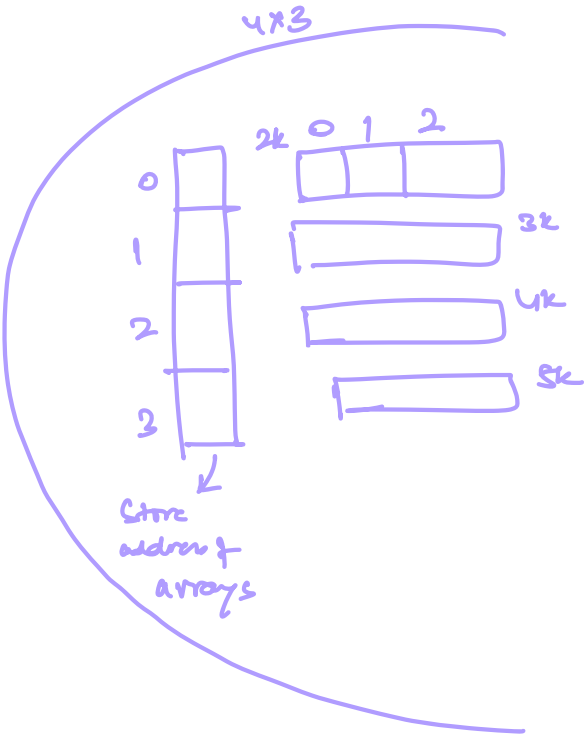


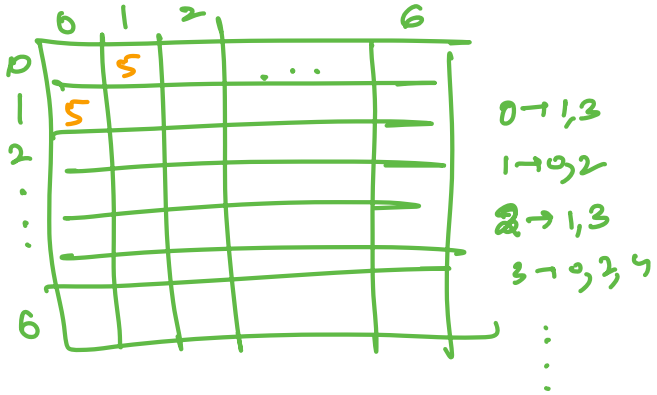
0	→	1	3	: 2
1	→	2		: 1
2	→	3		: 1
3	→	4		: 1
4	→	5	6	: 2
5	→			
6	→	5		: 1

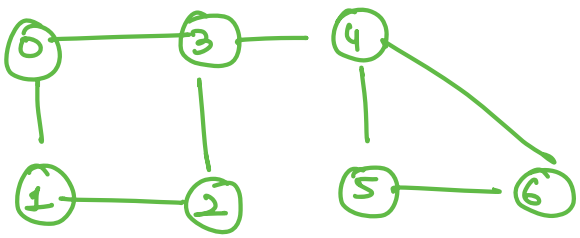
8

SEARCHING









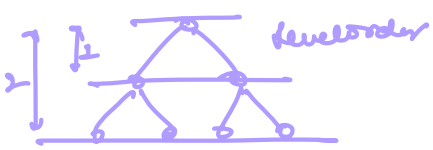
src dst
0 5

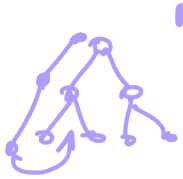
src → dst path ?

Queue — Breadth first search (BFS)

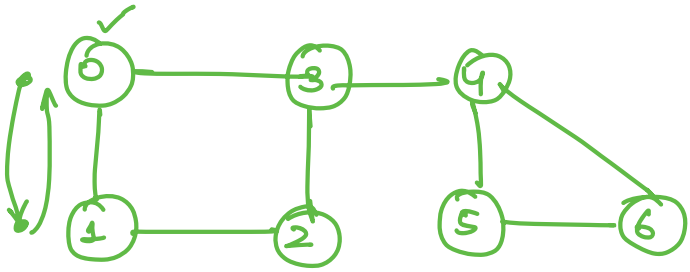
Stack — Depth first search (DFS)

LR
Pre/Post/In

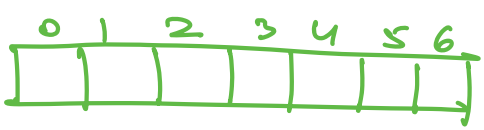




BFS (Breadth first search)

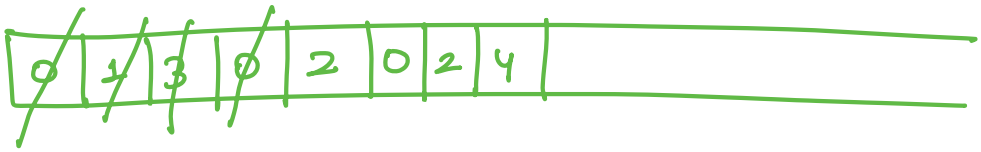


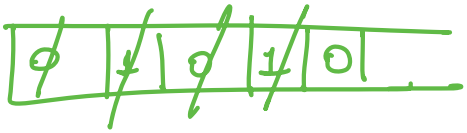
visited:



src 0 dst 5

- src node insert initiate

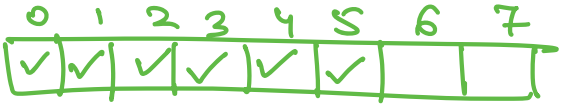




- remove
- dst?
- nbrs

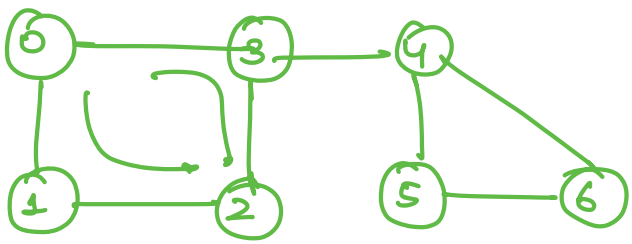
you insert only those nbrs which are unvisited

visited:



src / dst

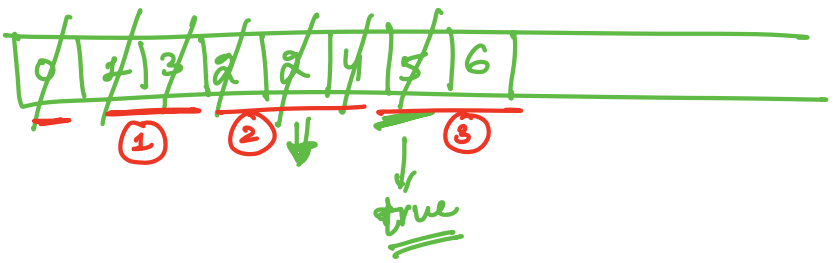
0 / 5

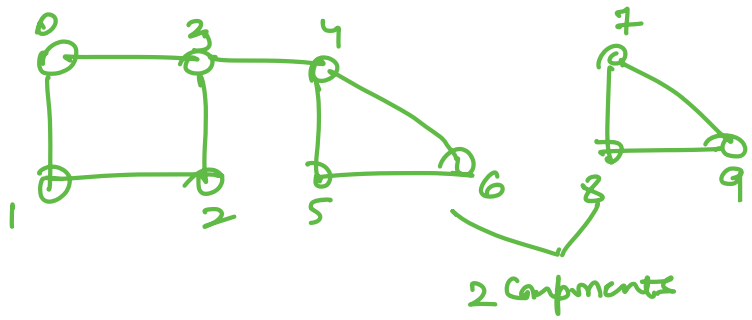


① src node
insert work
initiate

if already visited ignore ← - Remove - visited

- dst?
- nbrs unvisited



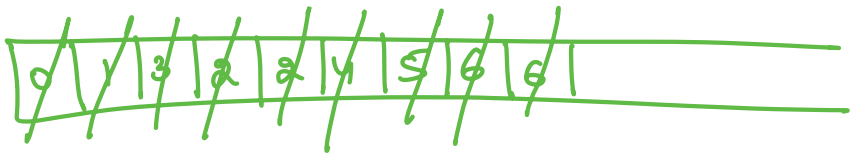


Graphs

0 —— 8 ?

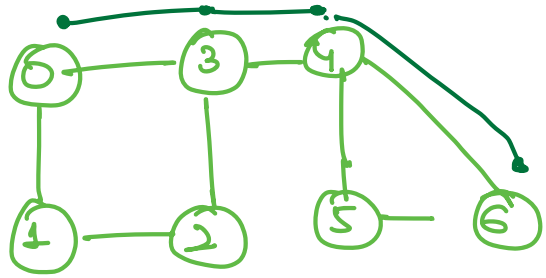
2 Components

Graphs >, 2 : forest



- R
- V
- D
- N

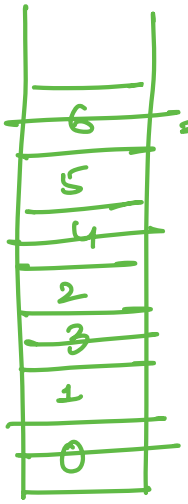
Depth first search (DFS)



src = 0
dst = 6

0 ✓ 1 2 ✓ 3 ✓ 4 5 ✓ 6

- Remove
- Visited
- Dst?
- Nghbrs



⇒ dst = true

Searching

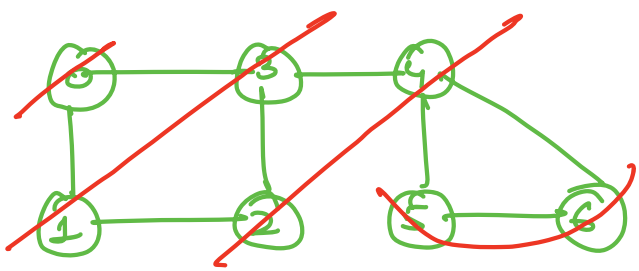
- BFS — Queue
- DFS — Stack

dst → src path?

Traversal (Printing)

- Breadth First Traversal (BfT)
- Depth First Traversal (DfT)

BfT:



- Traverse entire graph
- nodes print
- Breadth wise

pick any node

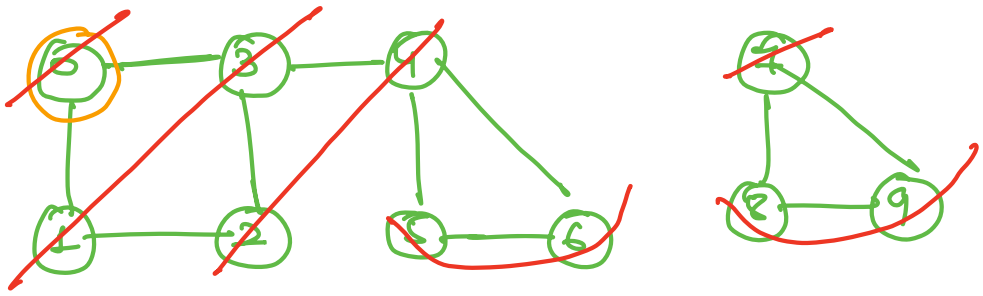
0 | 1 3 (1 distance away) | 2 4 | 5 6 } output

0 3 1 4 2 — 5 6 / 6 5 } or } output

0 1 2 3 4 5 6

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

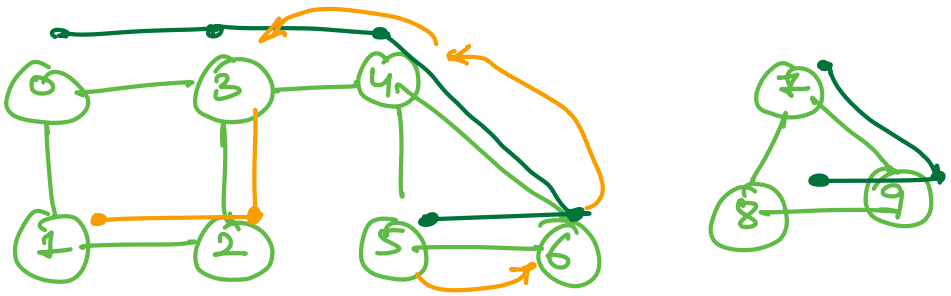


- Queue
- Visited
- Print
- nbrs unvisited

0 1 3 2 4 5 6 7 8 9

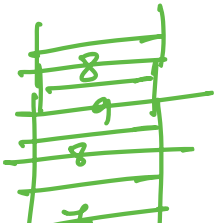
DFT:

0 1 2 3 4 5 6 7 8 9



0 1 2 3 4 5 6 7 8 9

Ignore.

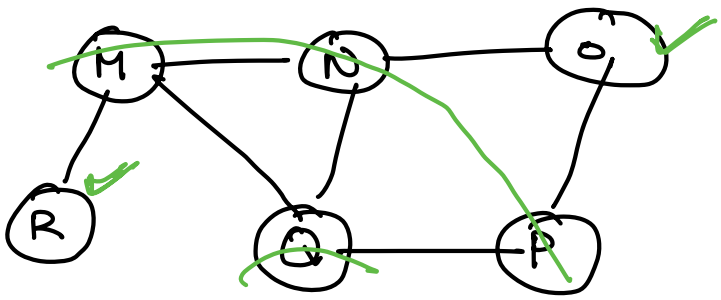


- Remove
- Visited

- Print
- nbrs unvisited

0 3 4 6 5 2 1 7 9 8

Q: BFT?



A. MNOPQR ✗

B. NQMPOR ✗

✓ C. QMNPRO

D. QMNPOR ✗

M NQR — NMbrs — QMbrs — RMbrs.

QRN

NRQ

QNP

+

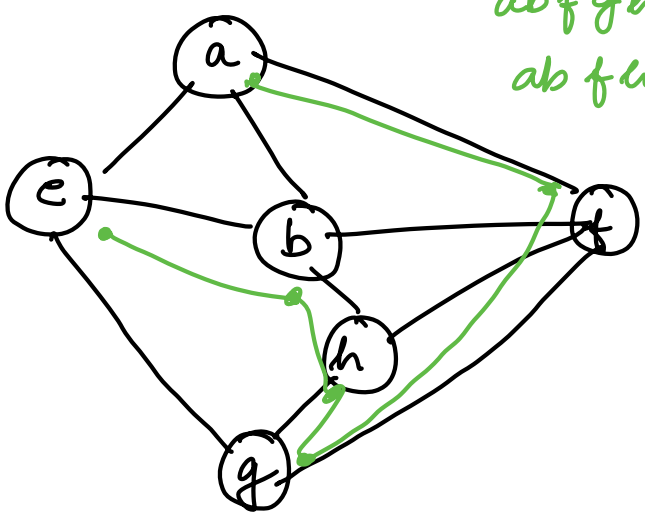
—

QMNP - - -

Q: DFT

abfghe ✓

abfhge ✓



a) abcghf

✗ b) abfehg

c) abfhge

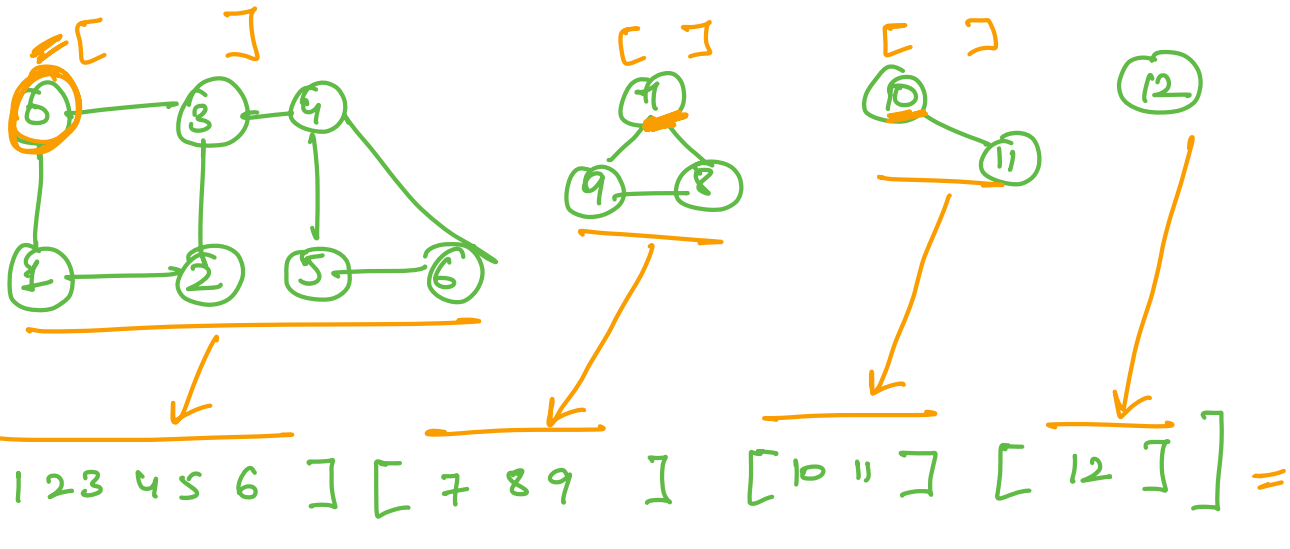
d) afghbe

- isConnected : no of components = 1

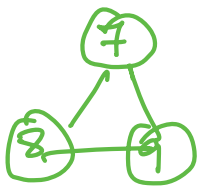
- isCyclic

- isTree
  no of components = 1 && no cycle

- getCC



big [[0 1 2 3 4 5 6] [7 8 9] [10 11] [12]] =



[[7 8 9]]