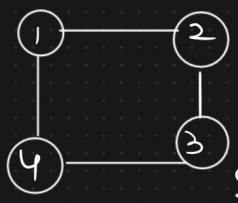
Graph

↳ Non Linear Based Data structure
cyclic or acyclic

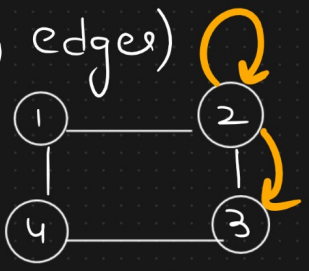
Tree → acyclic

↳ Every tree can be considered as Graph but vice-versa is not true.

Graph → (vertex, edges)



Simple Graph



Multi Graph

Simple Graph	Multi Graph
↳ Self loops are not allowed	Self loops are allowed
↳ Parallel edges are not allowed	Parallel edges are allowed

Min degree = 0

Simple Graph

- Null Graph
- complete Graph

Max degree = V-1

Null Graph

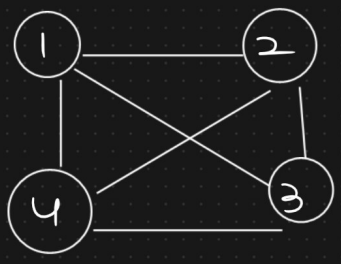
$|V| = 4$

$|E| = 0$

Degree of each vertex $= 0$

Complete Graph

$V = n$



$|V| = 4$

$|E| = 6$

Max Degree $= V - 1$

Min Degree $= 0$

Degree of each vertex $= 3$

$(V-1)$

$$\text{Sum of Degree} = |V| * |V-1| = 4 * 3 = 12$$

$$\text{Sum of Degree} = 2 * |E|$$

$$2 * 6 = 12$$

$$\text{Sum of Degree} = 2E \quad \text{— (1)}$$

$$\frac{\text{Sum of Degree}}{2} = E$$

(3) —

$$\frac{V * (V-1)}{2} = E$$

$V = 4$

$$E_4 = \frac{4 * 3}{2} = 6$$

Relationship b/w E & V in a Complete Graph

$$V * (V-1) = 2E$$

$$V^2 - V = 2E$$

$$|E| = \Theta(V^2)$$

log on both sides;

$$\log E = 2 \log V$$

$$\log E = \Theta(\log V) \quad \text{--- (4)}$$

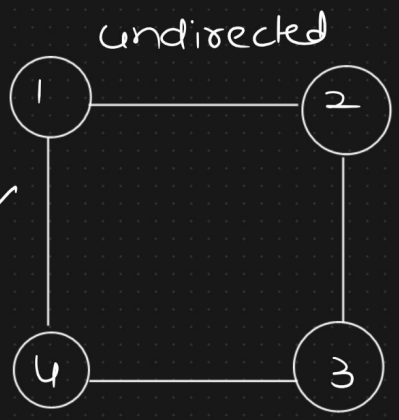
$|E|$ = # of edges

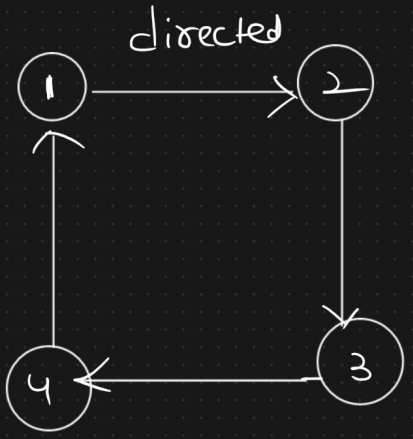
Graph → adjacency list (List)

Graph → adjacency matrix (Array)

8

1 → 2 ✓
2 → 2 ✓
3 → 2 ✓
4 → 2 ✓





'1' : [2, 4]
'2' : [1, 3]
'3' : [2, 4]
'4' : [1, 3]

Sum of Degree = 2E
= 4 * 2
= 8

$\Theta(2E)$

= 2 * 4
= 8

Adjacency matrix

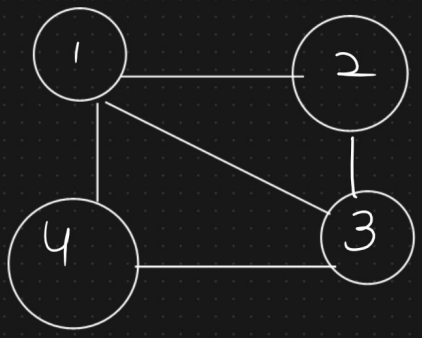
1 → Edge from one vertex to another

0 → No edge

	1	2	3	4
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

$V \times V$

$\Theta(V^2)$



1 → 3
2 → 2
3 → 3
4 → 2

10

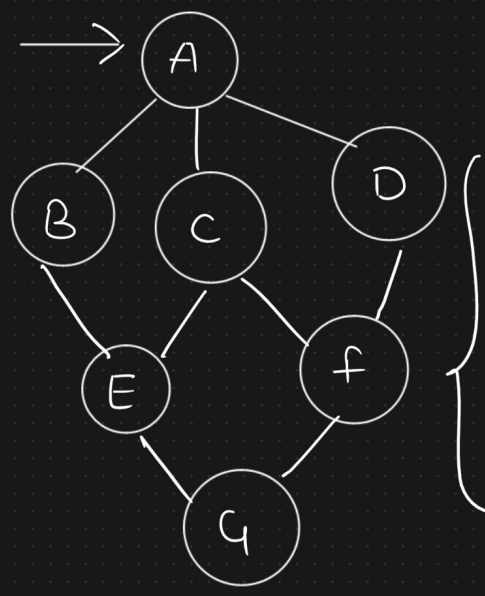
$2 * E = 2 * 5 = 10$

Undirected graph

Depth first Traversal

↳ Recursion

↳ Stack Data Structure



A → [B, C, D]
B → [A, E]
C → [A, E, f]
D → [A, f]
E → [B, C, G]
f → [C, D, G]
G → [E, f]

↳ take care of what vertex are already visited

visited = Boolean

Set → no duplicates

Visited

1	1	1	1	1	1	1
∅	∅	∅	∅	∅	∅	∅
A	B	C	D	E	f	G

~~DFT(A)~~

ω = B C D

↑

~~DFT(B)~~

ω = A E

↑ ↑

~~DFT(E)~~

ω = B, C, G

↑ ↑

~~DFT(C)~~

ω = A, E, f

↑ ↑ ↑

~~DFT(f)~~

A, B, E, C, f, D, G

DfT(v):

visited(v) = 1

print(v)

for all ω adj to v:

if ω is not visited:

DfT(ω)

↳ Recursion

A, B, E, C, F, D, G

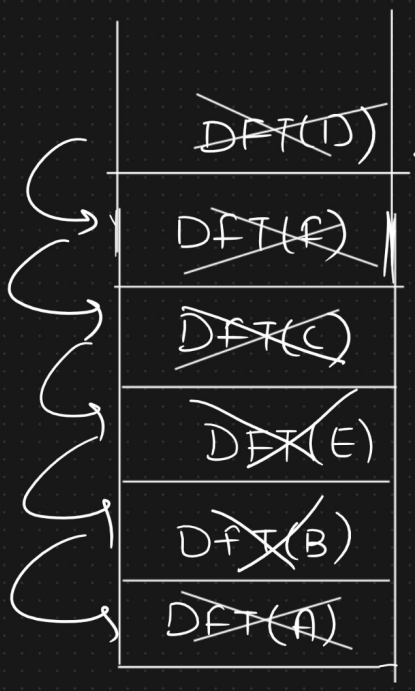
ω = E D G

DFT(D)

ω = A, F

DFT(G)

ω = E F



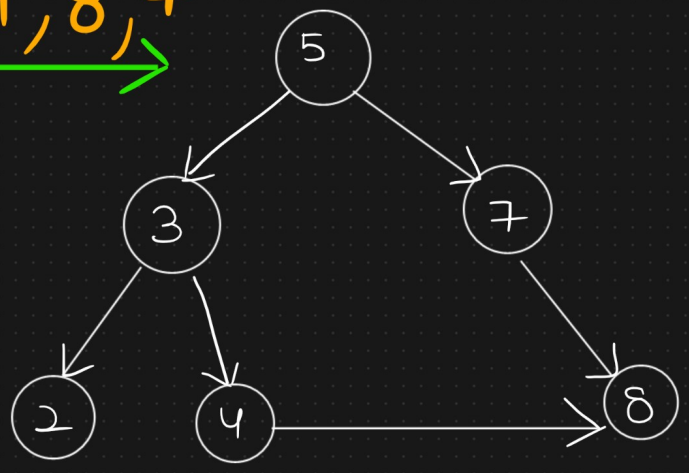
Stack

set

Visited = 5, 3, 2, 4, 8, 7

Directed graph

5, 3, 2, 4, 8, 7



5 → [3, 7]

3 → [2, 4]

7 → [8]

2 → [ ]

4 → [8]

8 → [ ]

DFT(5)

ω = 3, 7

DFT(3)

ω = 2, 4

DFT(2)

ω = [ ]

DFT(7)

ω = 8

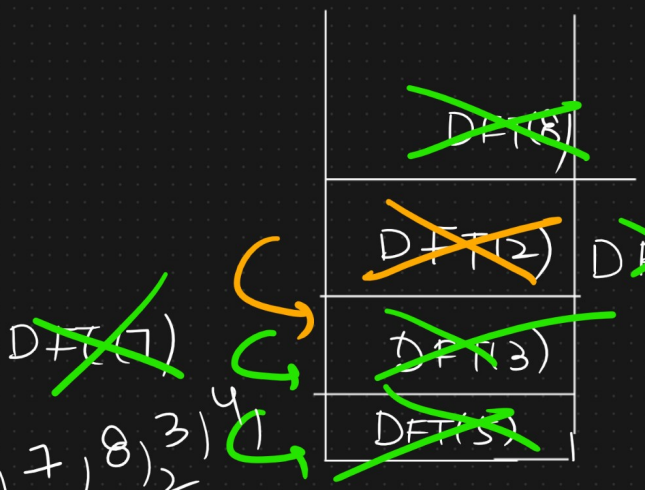
DFT(4)

ω = 8

DFT(8)

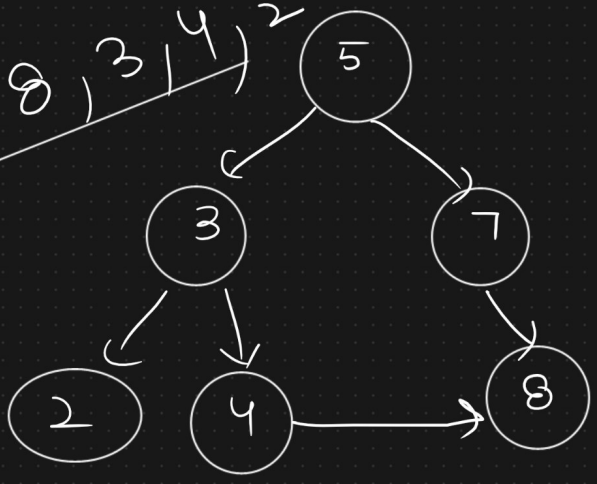
ω = [ ]

- 5, 7, 8, 3, 4, 2 — correct
- 5, 3, 2, 4, 8, 7 — correct
- 5, 3, 2, 4, 7, 8 — wrong



visited = 5, 7, 8, 3, 4, 2

5, 7, 8, 3, 4, 2



DFT(5)

ω = 7, 3

DFT(7)

ω = 8

DFT(8)

ω = []

DFT(3)

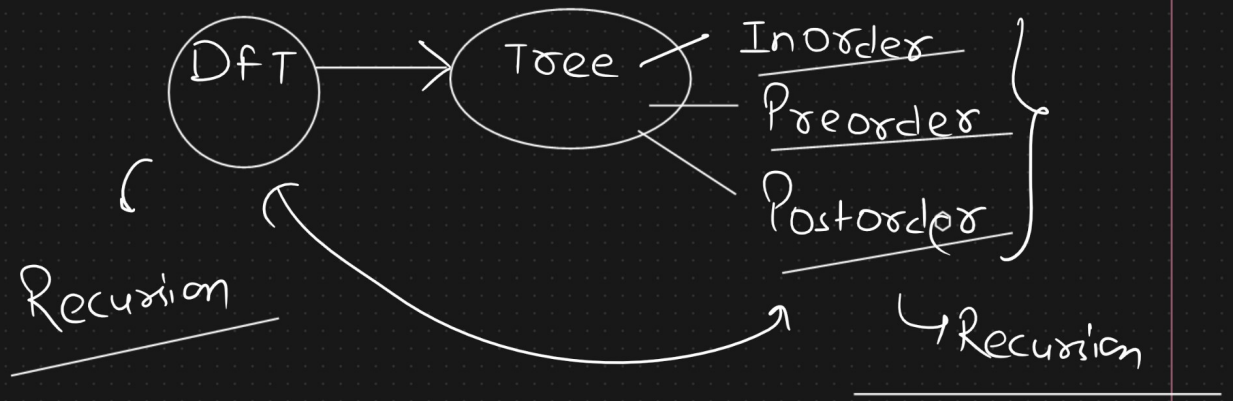
ω = 4, 2

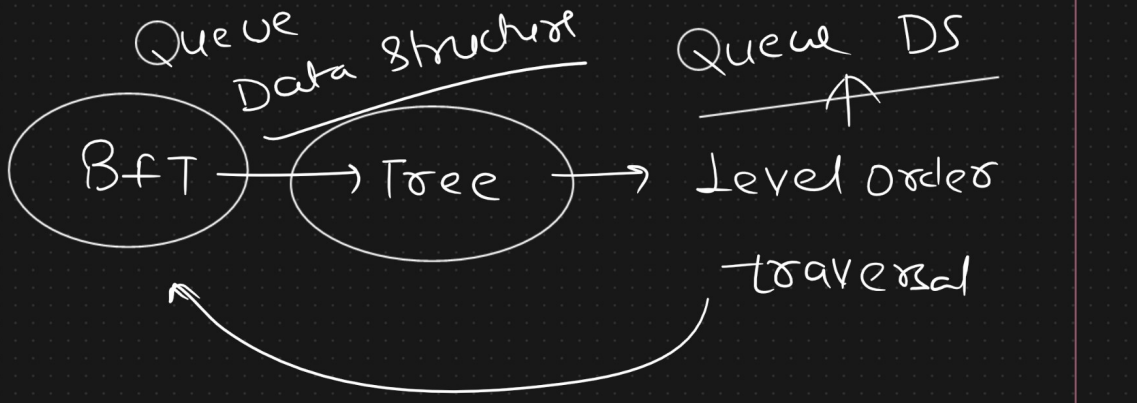
DFT(4)

ω = 8

DFT(2)

ω = ()


DfT
Tree
Inorder
Preorder
Postorder
Recursion
Recursion


Queue
Data structure
Queue DS
BfT
Tree
level order
traversal