		TC	SC
Fibonacci →	Recursion	$O(n^2)$	$O(n)$
Fibonacci →	Memoization	$O(n)$	$O(n)$
Fibonacci →	Tabulation	$O(n)$	$O(n)$
* Fibonacci →	Space Optimization	$O(n)$	$O(1)$

~~arr [n] = O(n)~~

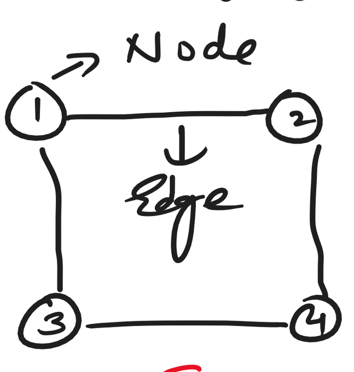
{ only using variables }
No extra array.

```
{ p2 = 0
  p1 = 1
( i = 2 to n ) → {
int curr = p1 + p2 ;
  p2 = p1 ;
  p1 = curr ;
}
ret p1 ;
```

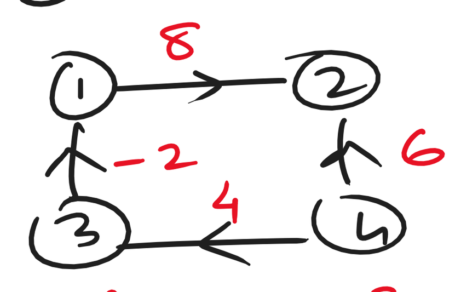
0+1 = 1
p_2 = 1
p_1 = 1   (2)
1, 2 = 3

Introduction to Graphs :

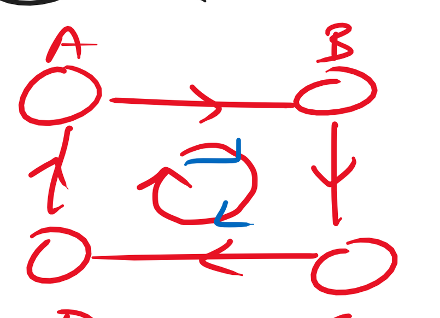
* A collection of entities called nodes.
* A node has some data.
* The nodes are connected to each other via edges.
* The edges can have values / weights.



→ Unweighted Undirected graph

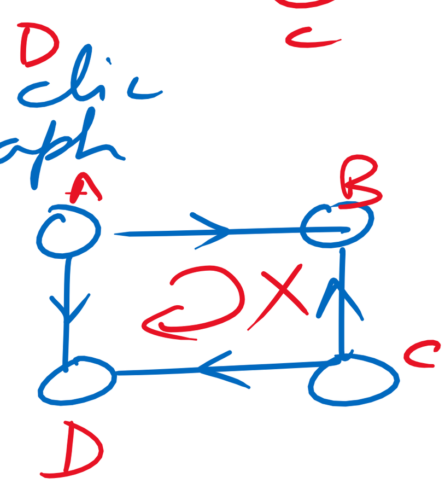


→ Directed Weighted graph

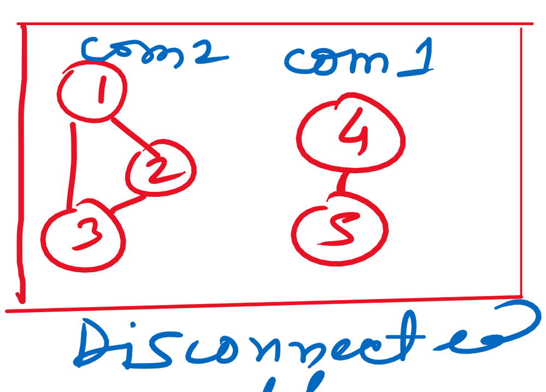


Cyclic graph

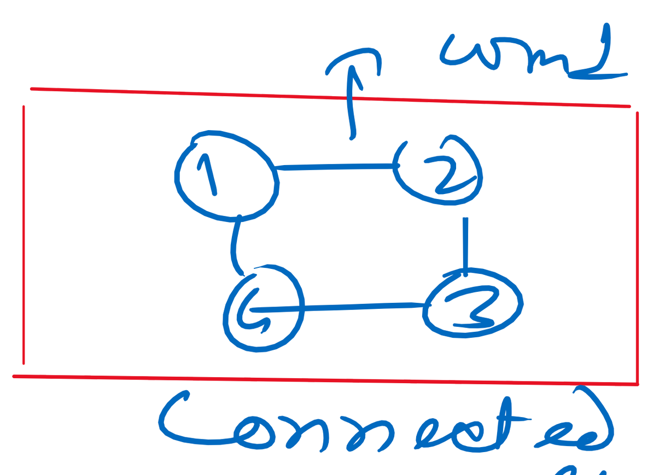
From node A I can reach back by traversing all nodes to the source node A.



→ DAG
Directed Acyclic graph

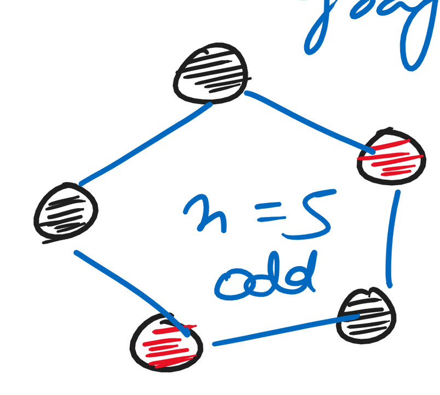


Disconnected graph



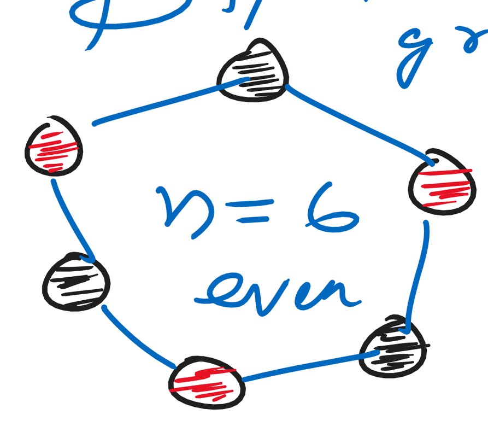
Connected graph

Un-bipartite graph



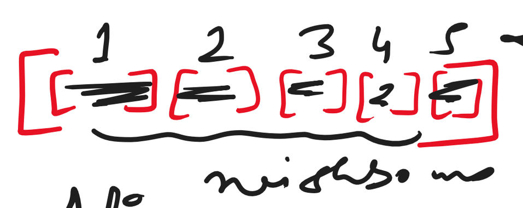
Adjacent nodes have same color or property.

Bipartite graph

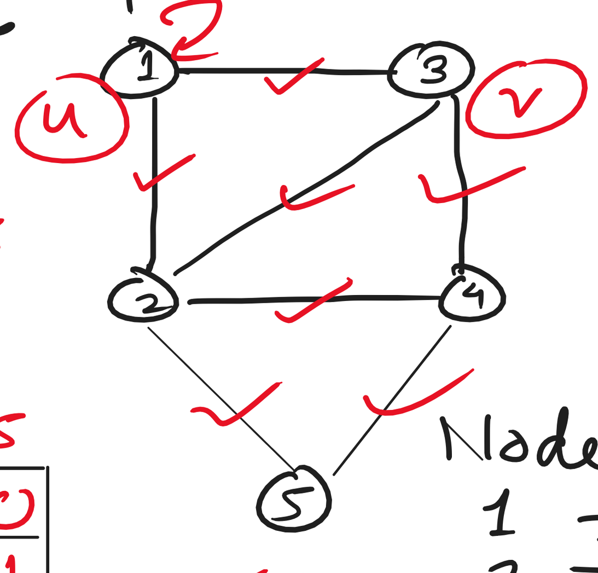


Adjacent nodes don't have same color or property

Representation : →



Adjacency Matrix xxx



u → u ✗
v → v
u → v
v → u
undirected

Adjacency List

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

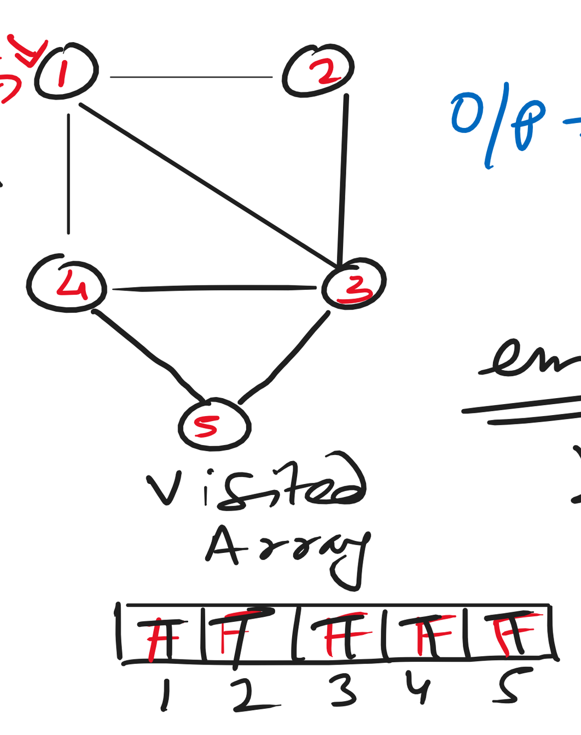
List<List<int>>

Node : List of Neighbors
1 → 2, 3
2 → 1, 3, 4, 5
3 → 1, 2, 4
4 → 2, 3, 5
5 → 2, 4

* Important: Graph Traversals

Breadth First Traversal :→ (LOT)

Who are your neighbours?

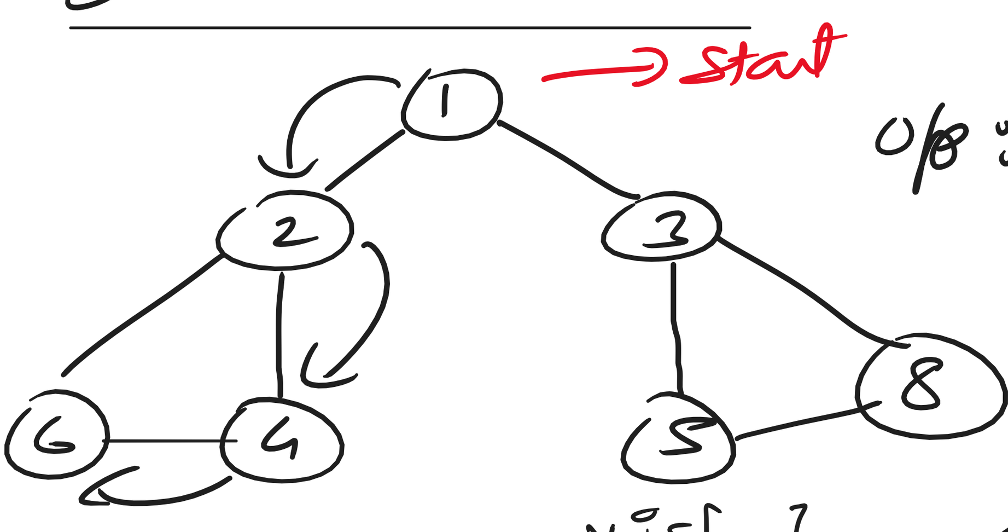


O/P → 1, 2, 3, 4, 5
0 1 2

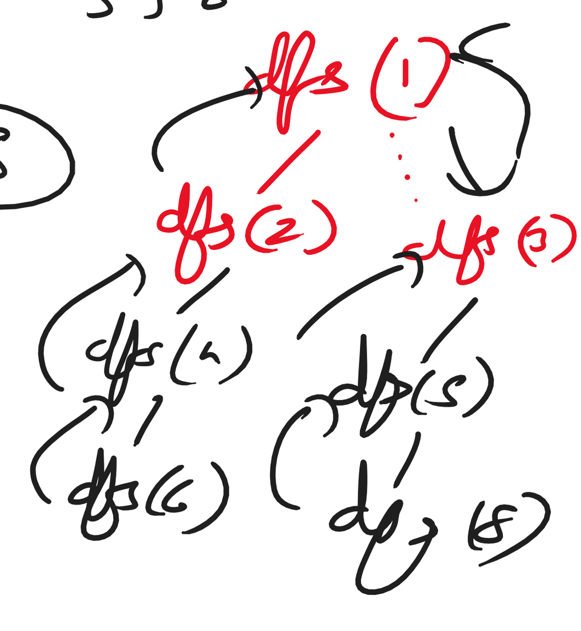
empty
Done

Adj List
→ 1 → ~~X~~, ~~X~~, ~~X~~
→ 2 → ~~X~~, ~~X~~
→ 3 → ~~X~~, ~~X~~, ~~X~~, ~~X~~
→ 4 → ~~X~~, ~~X~~, ~~X~~
→ 5 → ~~X~~, ~~X~~

DFS Traversal : → (Recursion)



O/P : 1, 2, 4, 6, 3, 5, 8

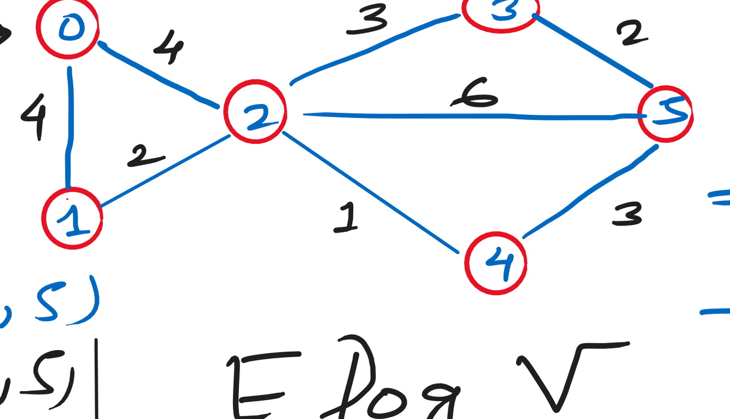


Adj-List
1 → 2, 3
2 → 1, 4, 6
3 → 1, 5, 8
4 → 2, 6
5 → 3, 8
6 → 2, 4
8 → 3, 5

vis[ ]
1 ~~F~~ T
2 ~~F~~ T
3 F
4 ~~F~~ T
5 F
6 ~~F~~ T
8 F

⇒ Shortest Distance Algorithms : →

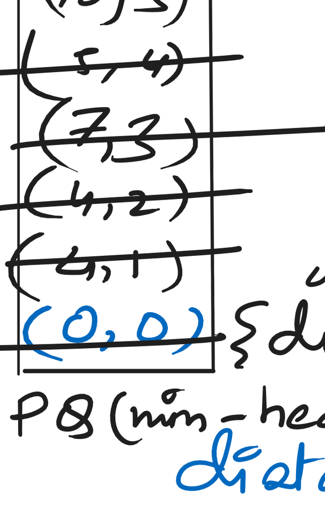
Dijkstra's Algorithm :⇒ Greedy Algo



* Set Data Structure
* Priority Queue (Min Heap)

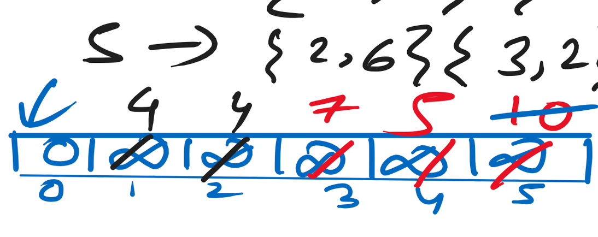
⇒ Adj List (Node, Weight)

0 → {1, 4} {2, 4} Pair
1 → {0, 4} {2, 2}
2 → {0, 4} {1, 2} {3, 3} {4, 1} {5, 6}
3 → {2, 3} {5, 2}
4 → {2, 1} {5, 3}
5 → {2, 6} {3, 2} {4, 3}



{dist, node}
PQ (min-heap)

E log V



distance array