Graph Terminology

Graph

It's a way of connecting dot's.

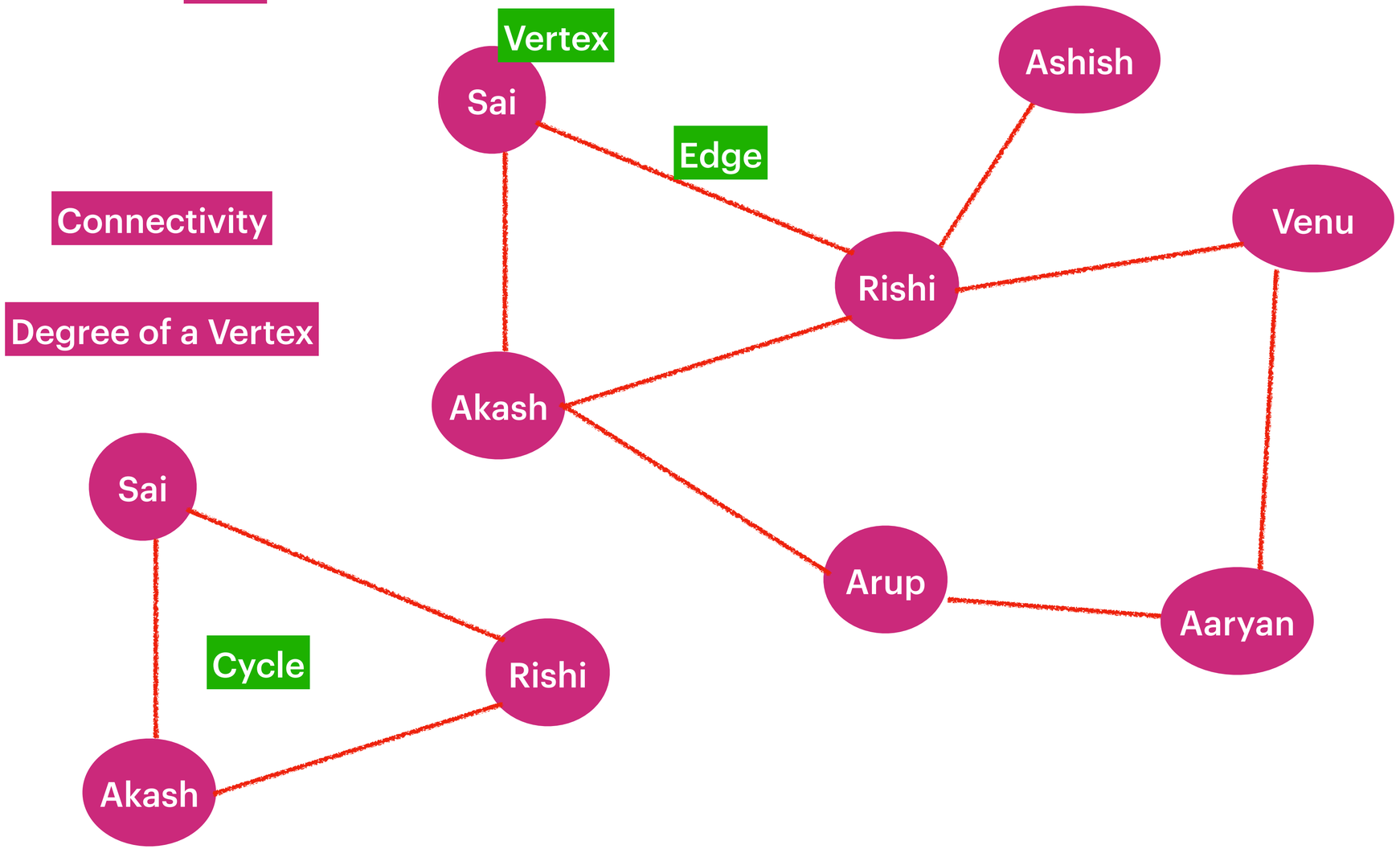
Path

Vertex

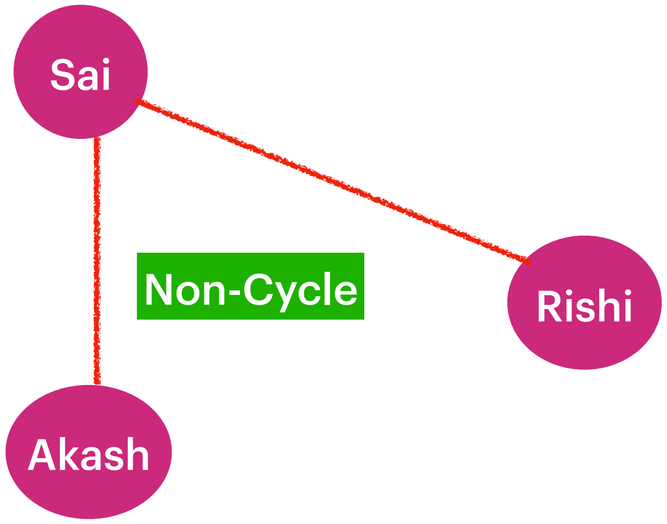
Edge

Connectivity

Degree of a Vertex



Cycle



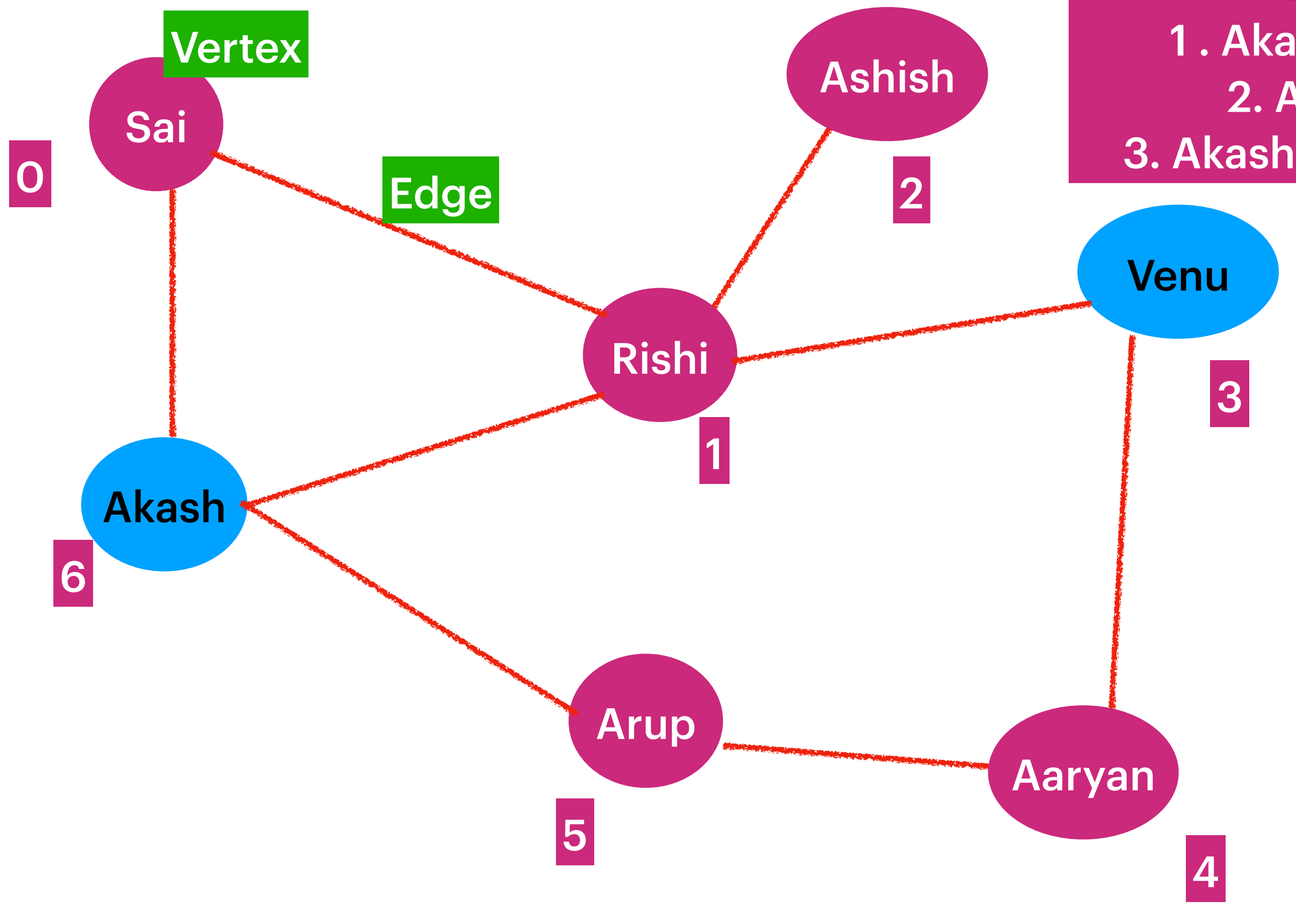
Non-Cycle

Path :
If there is connection exits,
Path just talks about no.of between Source & Destination vertices.

Graph

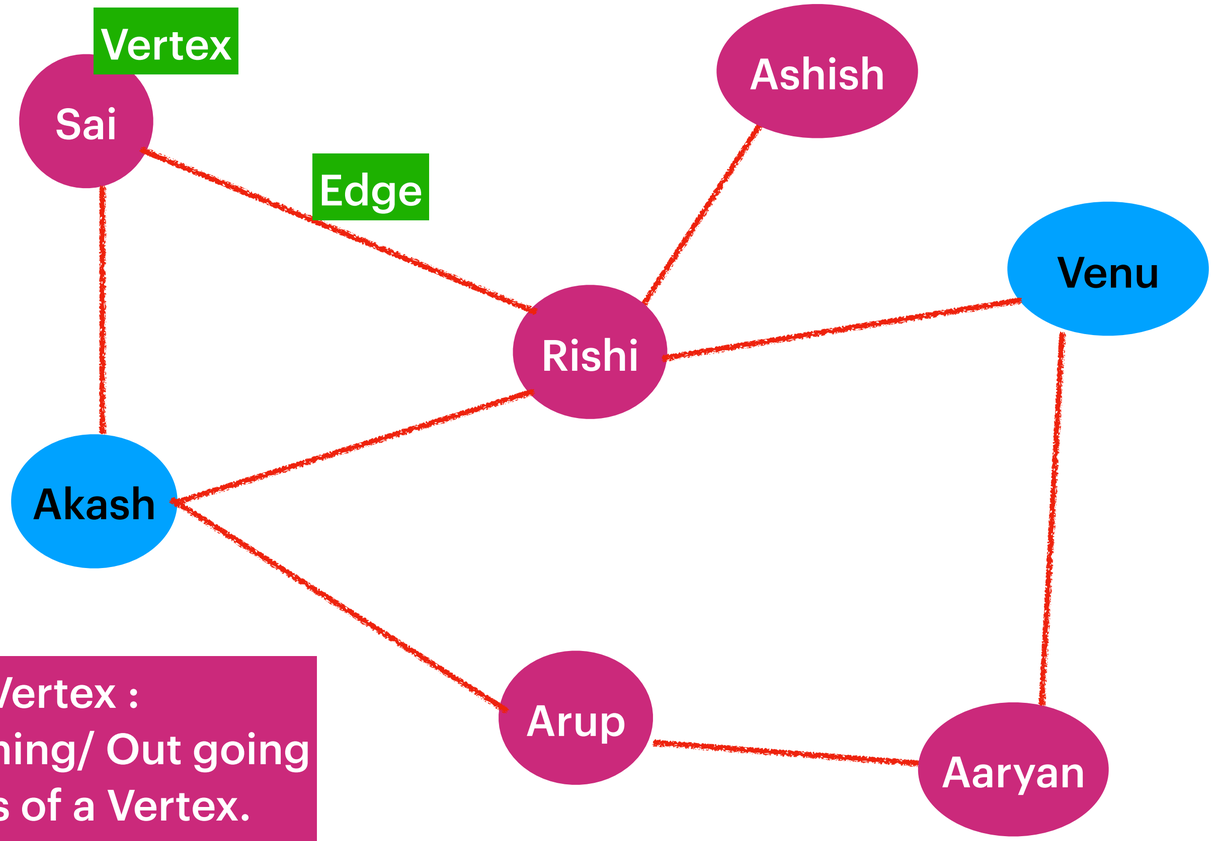
It's a way of connecting dot's.

Path between Akash <-> Venu
Does path exists ? Yes
1 . Akash — Sai — Rishi —Venu (3)
2. Akash — Rishi — Venu (2)
3. Akash — Arup — Aaryan — Venu (3)



Bidirectional Graph

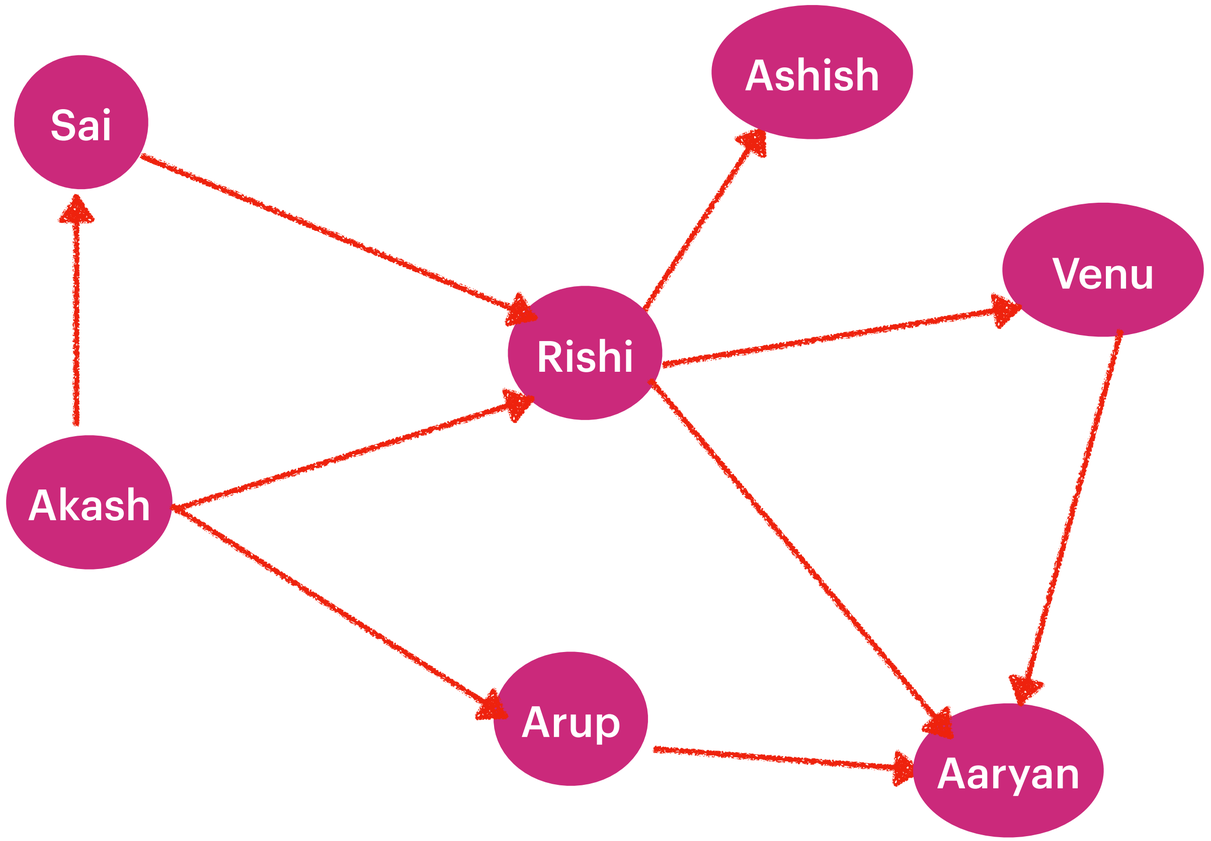
Here the connection exists in Both the ways.
Source <—> Destination



Degree Of Vertex :
Talks about Incoming/ Out going Connections of a Vertex.

DegreeOf(Akash) :
In-Degree =>3
Out-Degree =>3

Directed Graph



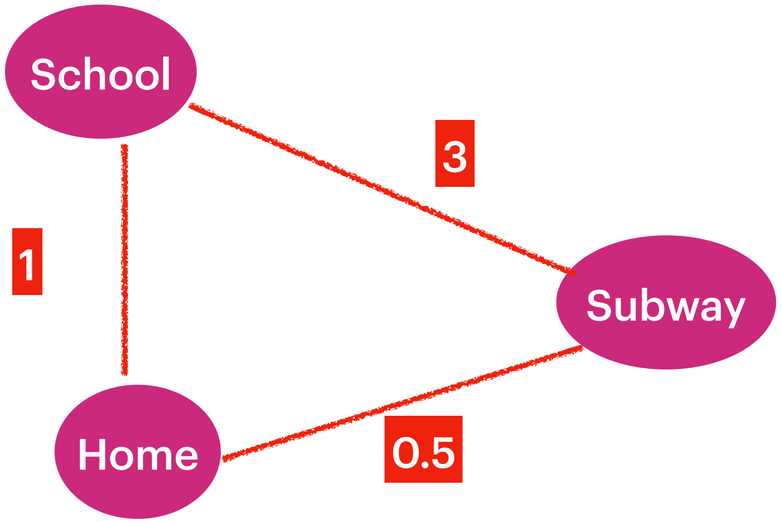
In-Degree (Sai) : 1

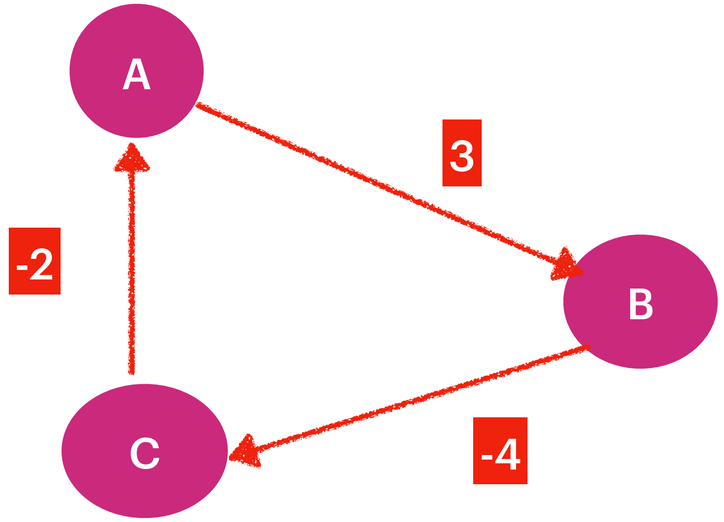
Out-Degree (Sai) : 1

In-Degree (Rishi) : 2

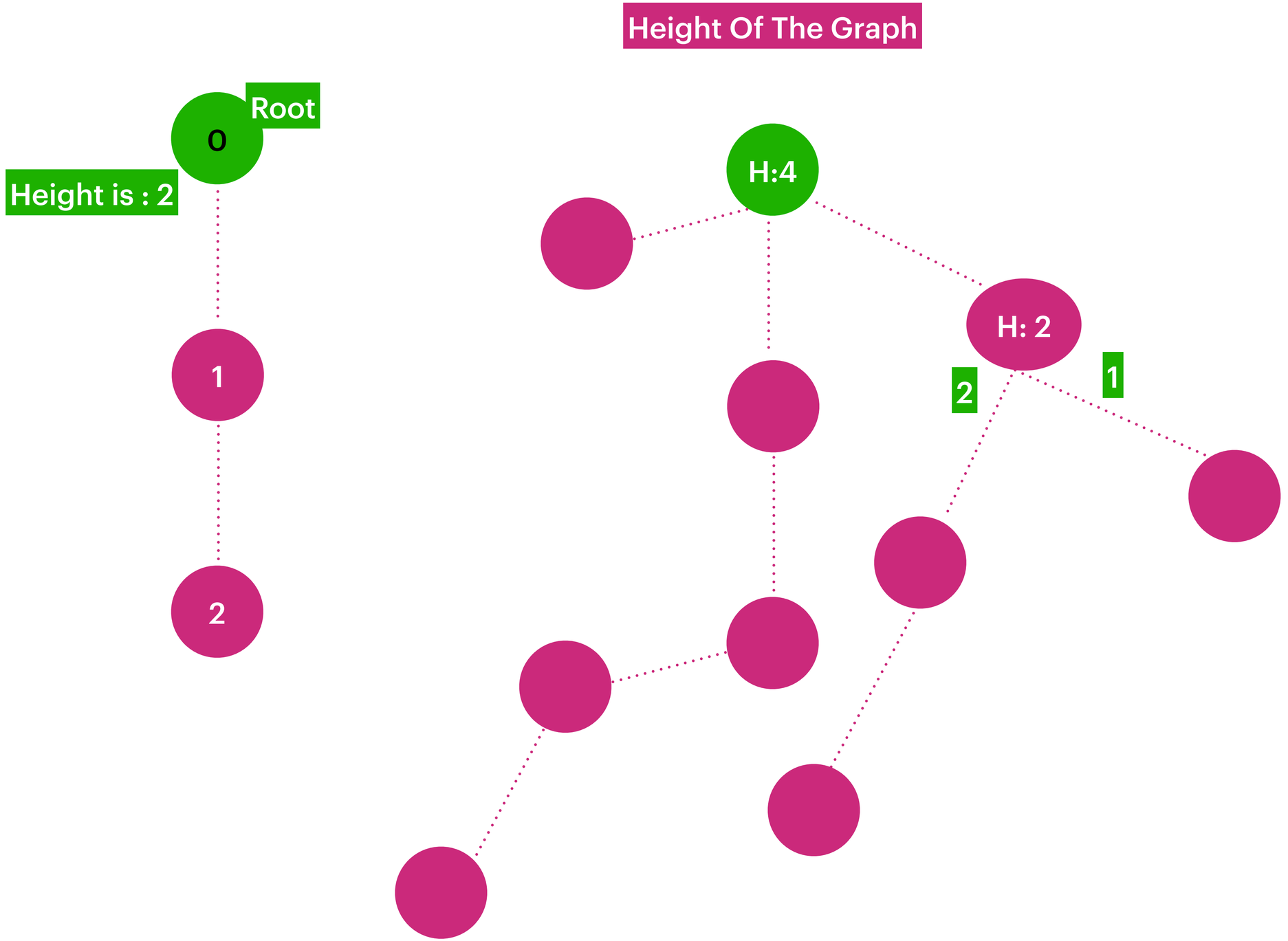
Out-Degree (Rishi) : 3

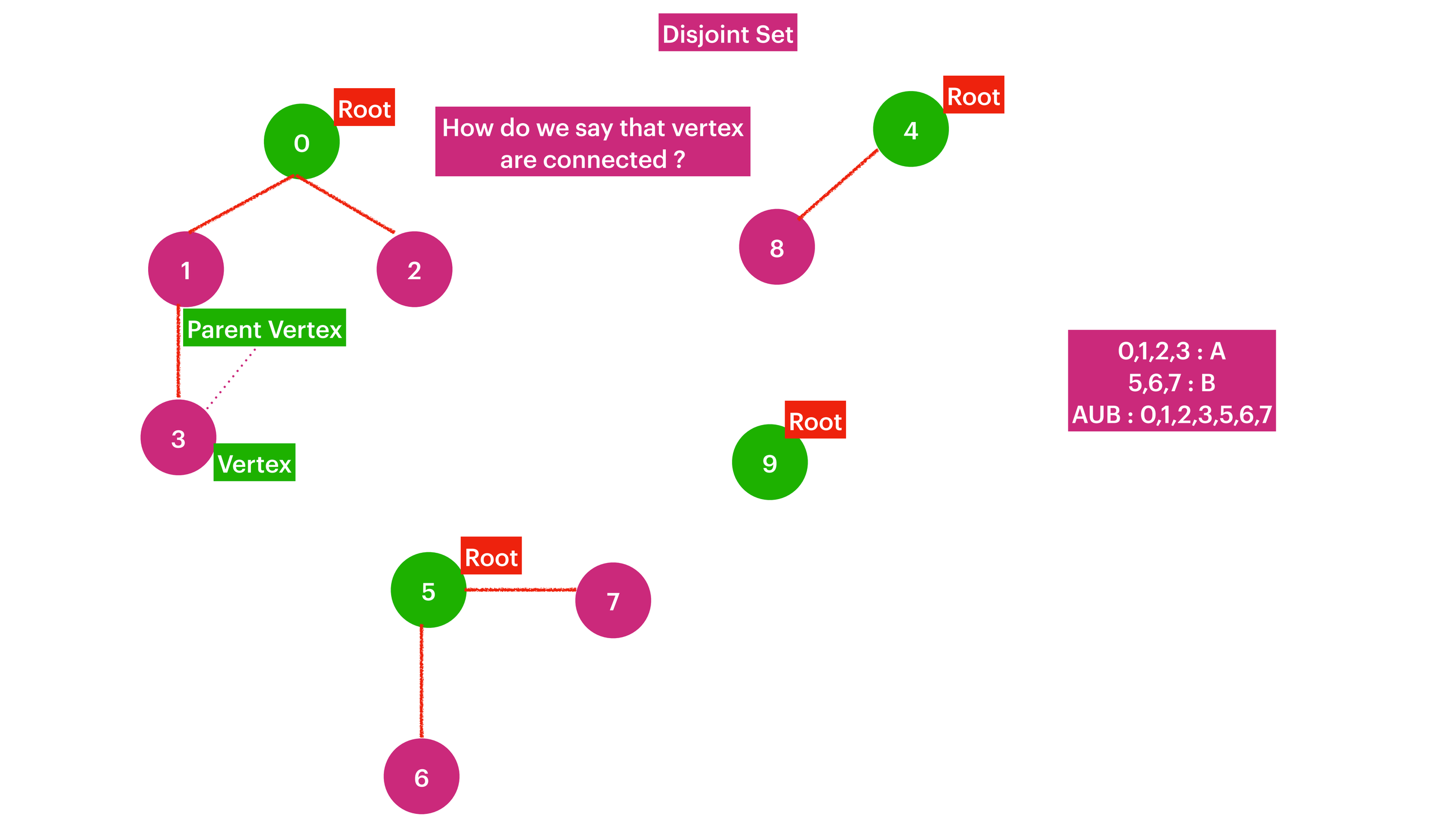
Weighted Graph

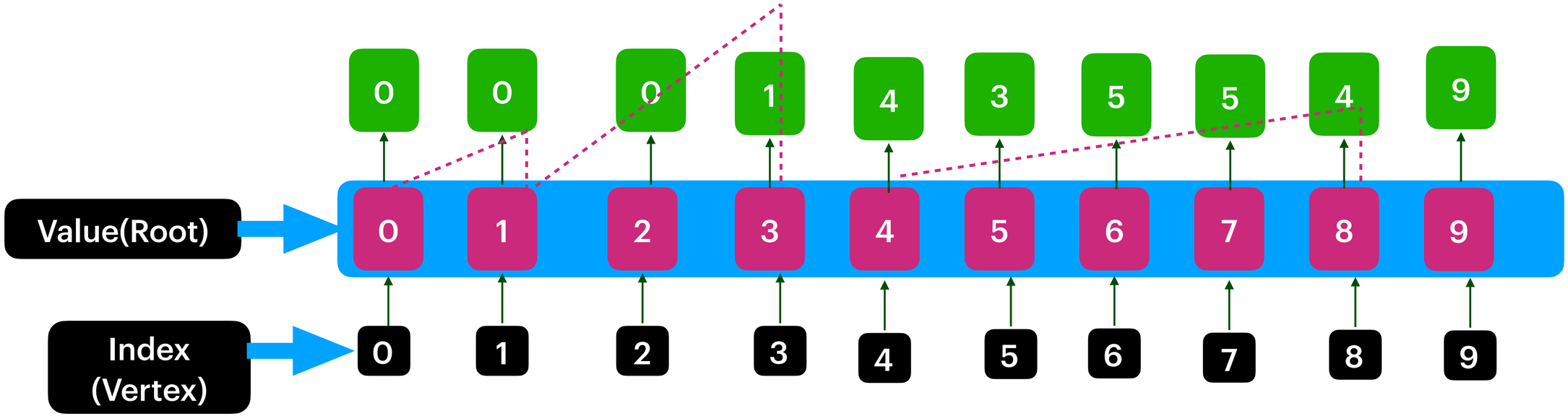




Negative Weight Cycle =
A -> B -> C =3 + (-4) + (-2) = -3


Height Of The Graph
Root
0
Height is : 2
1
2
H:4
H: 2
2
1


Disjoint Set
Root
0
How do we say that vertex are connected ?
Root
4
1
2
8
Parent Vertex
3
Vertex
0,1,2,3 : A
5,6,7 : B
AUB : 0,1,2,3,5,6,7
Root
9
Root
5
7
6



(0,1) , (0,2) , (1,3) , (4,8), (5,6), (5,7)

Connected(1,3) =
root(1) == root(3)
0 == 0 = true

connected(1,8)
root(1) == root(8)
0 == 4 (false)

connected(5,7)
5 == 5 : true

Quick Union:

union(vx, vy) => TimeComplexity : O(1)

connection(vx, vy) => TimeComplexity : O(n) + O(n) = O(2n) = O(n)

find(vertex) => Time Complexity : O(n)