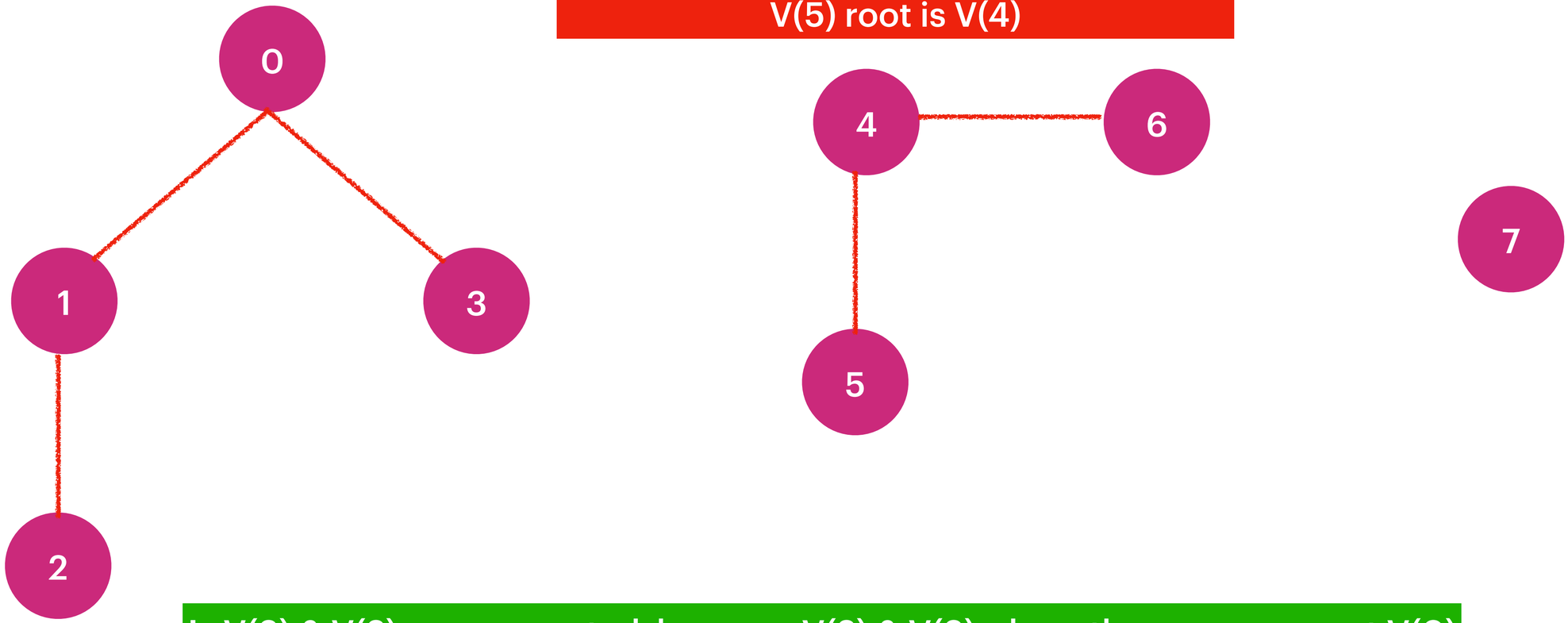
Graphs

Is V(2) & V(5) are not connected because
V(2) root is V(0)
&&
V(5) root is V(4)



Is V(3) & V(2) are connected because V(2) & V(3) share the common root V(0)

How do we check, does the vertices are connected or not ?

Answer is make use of DisJointSet

Returns the root of a given vertex.

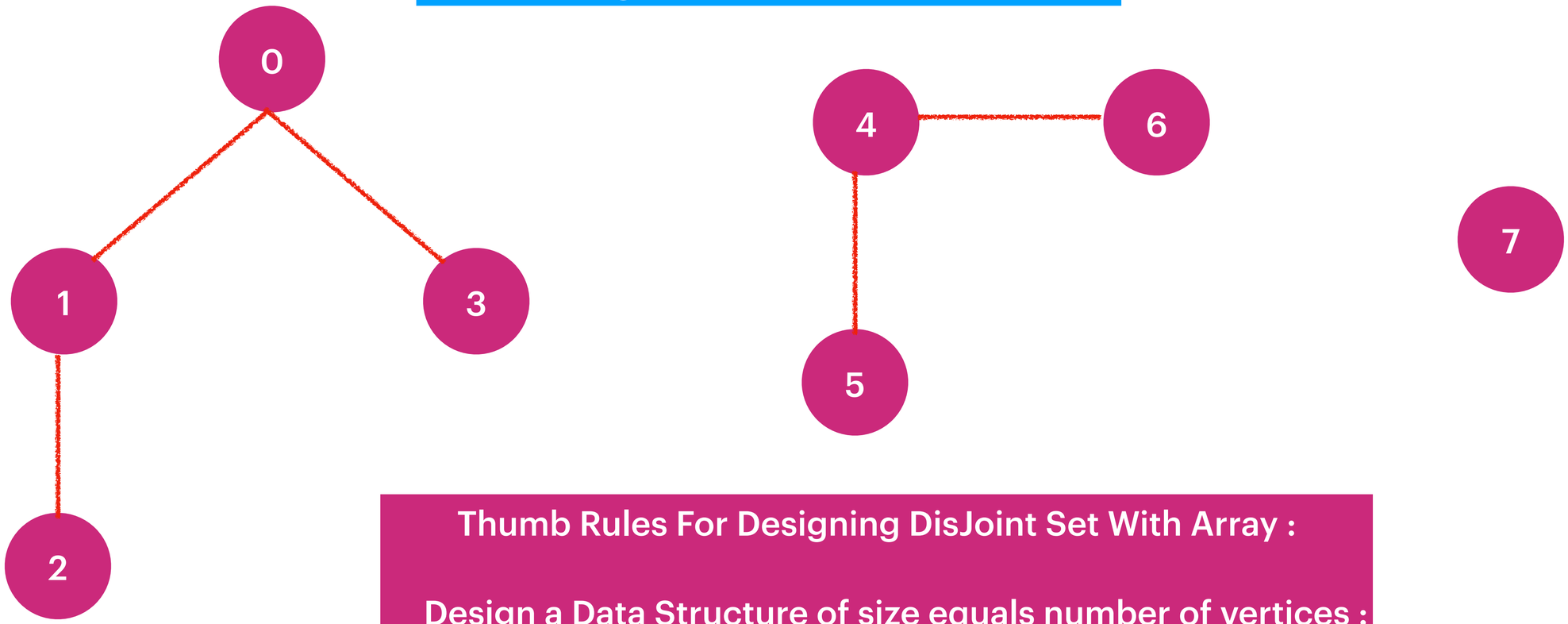
find(v)

Makes a connection between vertices

union(vx, vy) :

Construct DisJoint Set for the below Graphs

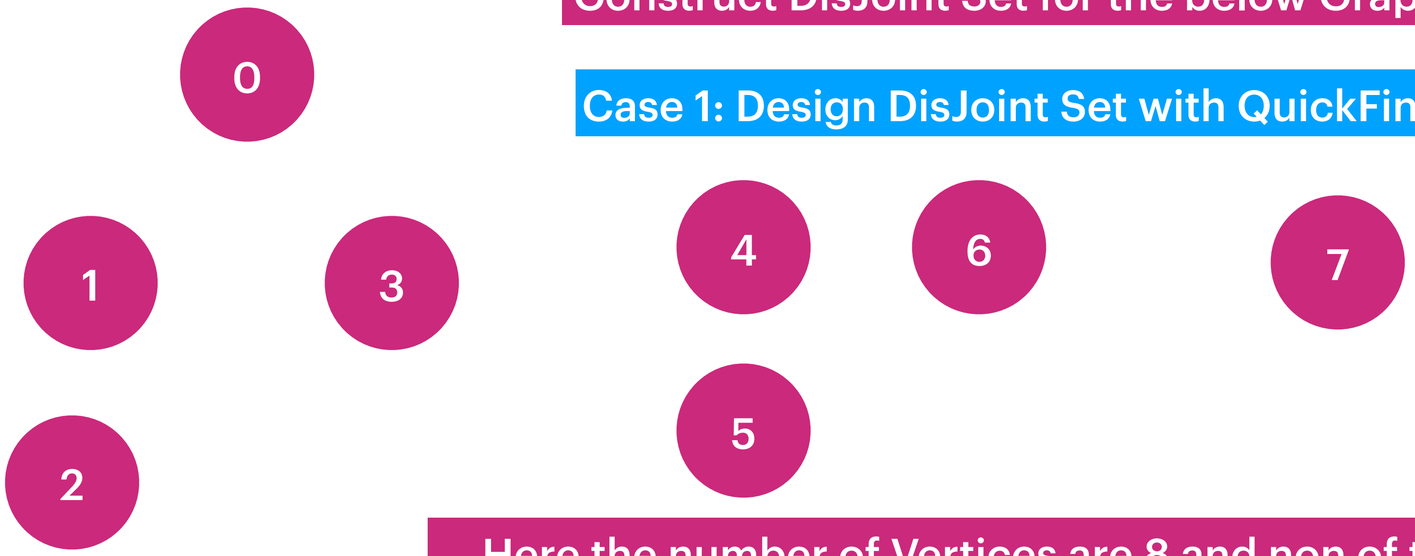
Case 1: Design DisJoint Set with QuickFind.



Thumb Rules For Designing DisJoint Set With Array :

Design a Data Structure of size equals number of vertices :
Form Independent connections.
Ex: (0,1) (0,3) (1,2) (4,5) (4,6) , (7,7)

Construct DisJoint Set for the below Graphs



Case 1: Design DisJoint Set with QuickFind.

Here the number of Vertices are 8 and non of them are connected.
We know that every individual vertex can act as root to ItSelf.

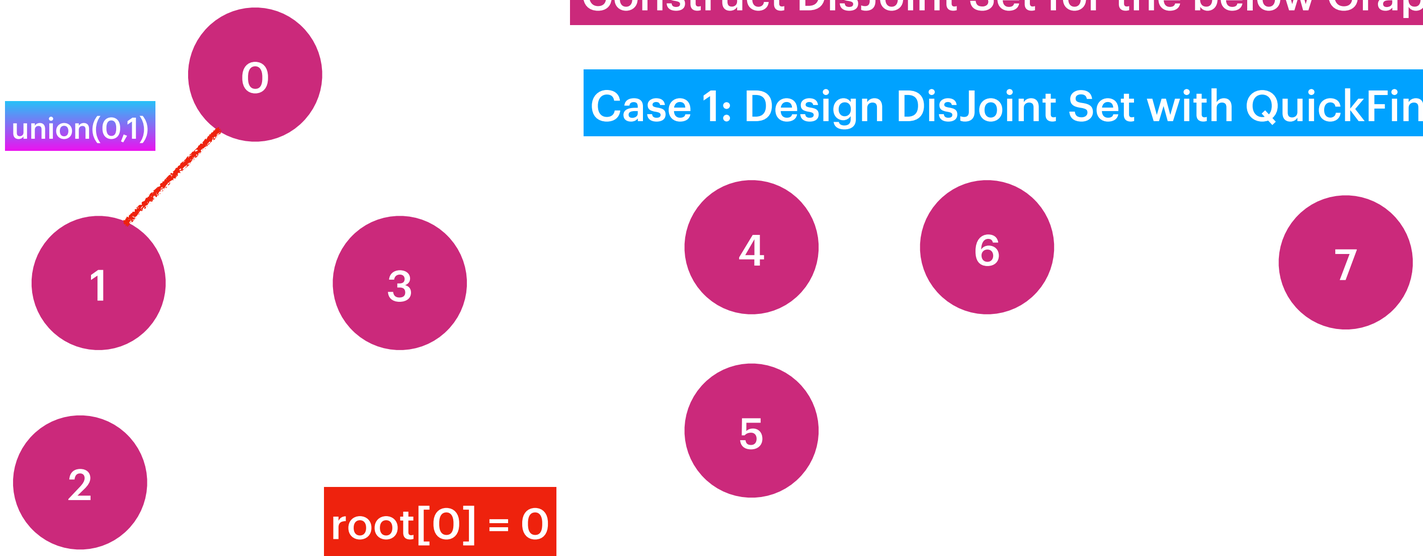
root	0	1	2	3	4	5	6	7
Index's/ Vertices	0	1	2	3	4	5	6	7

Array Index can be represented as Vertex.
Index value can be represented by root.

Construct DisJoint Set for the below Graphs

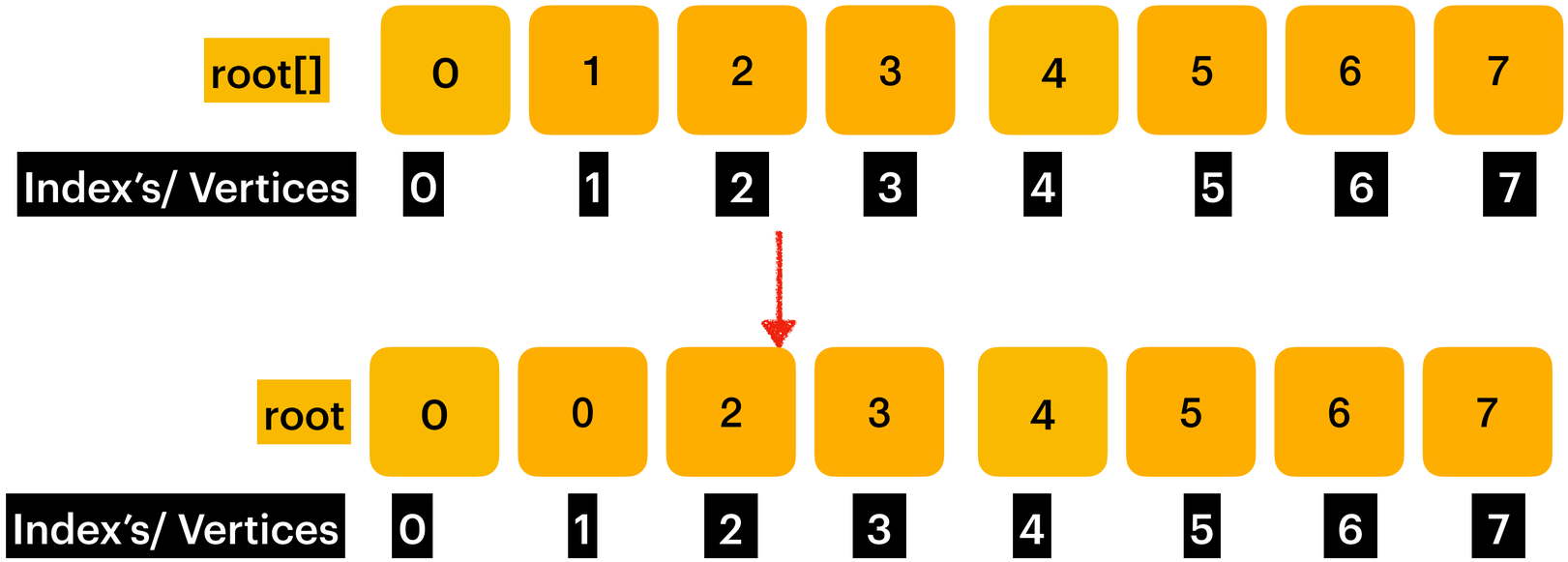
Case 1: Design DisJoint Set with QuickFind.

union(0,1)



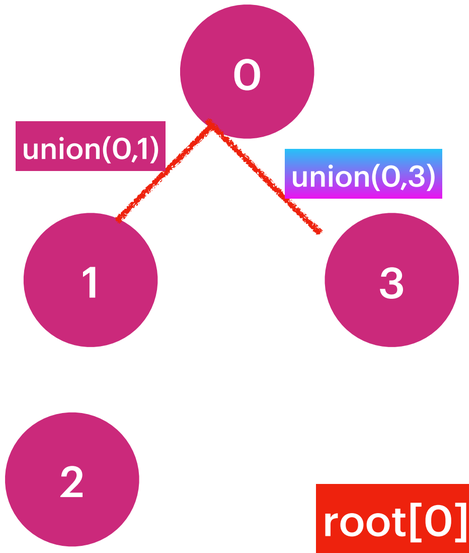

```
public void find(int v)
{
return root[v];
}
```

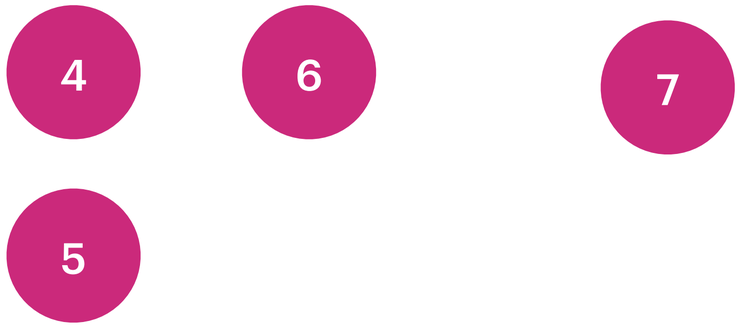
root[0] = 0
root[1] = 1



Construct DisJoint Set for the below Graphs

Case 1: Design DisJoint Set with QuickFind.

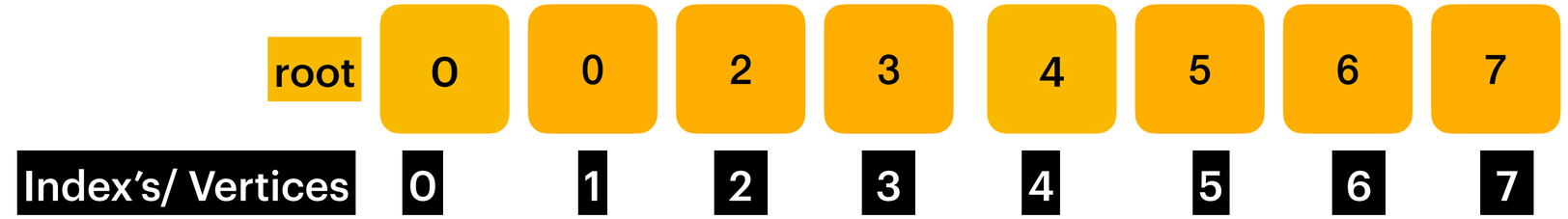


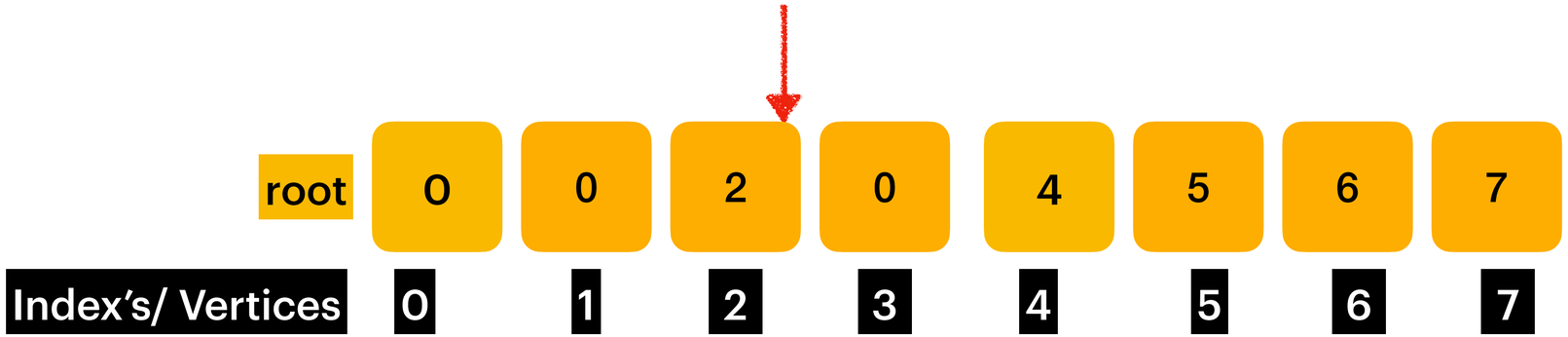



```
public void find(int v)
{
return root[v];
}
```

Time Complexity : O(1)

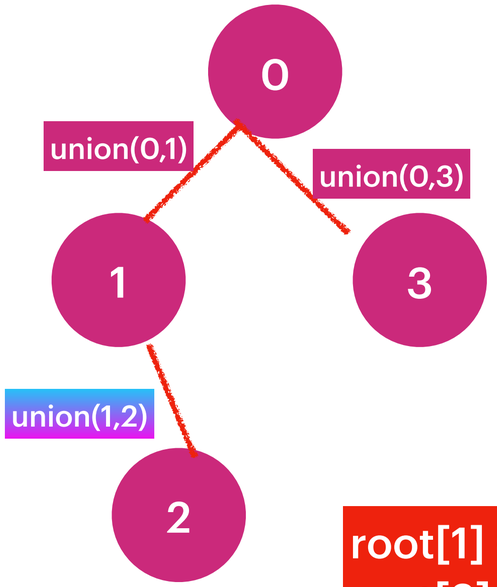
root[0] = 0
root[3] = 3

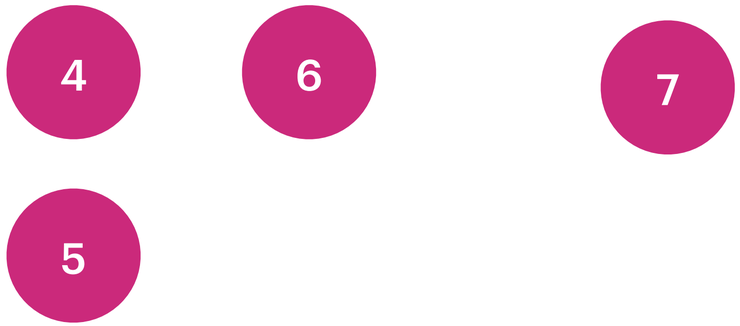




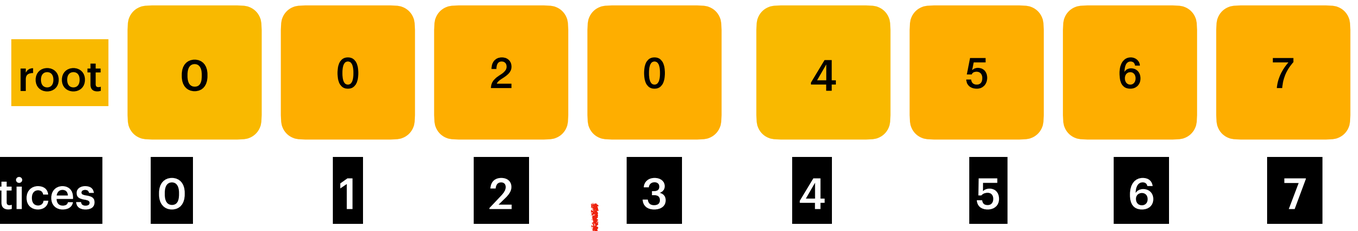
Construct DisJoint Set for the below Graphs

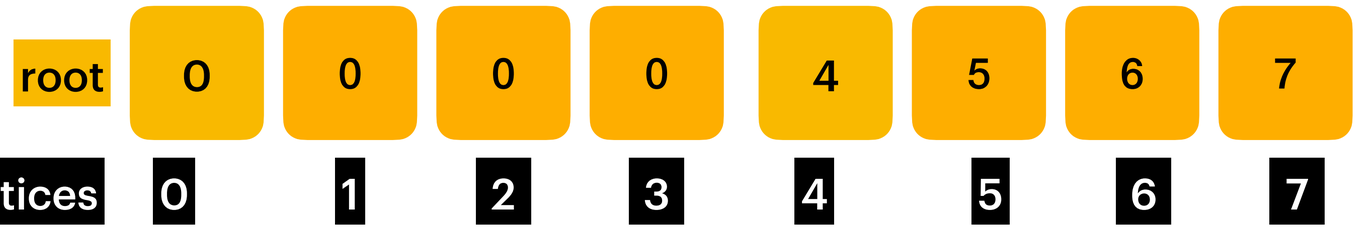
Case 1: Design DisJoint Set with QuickFind.





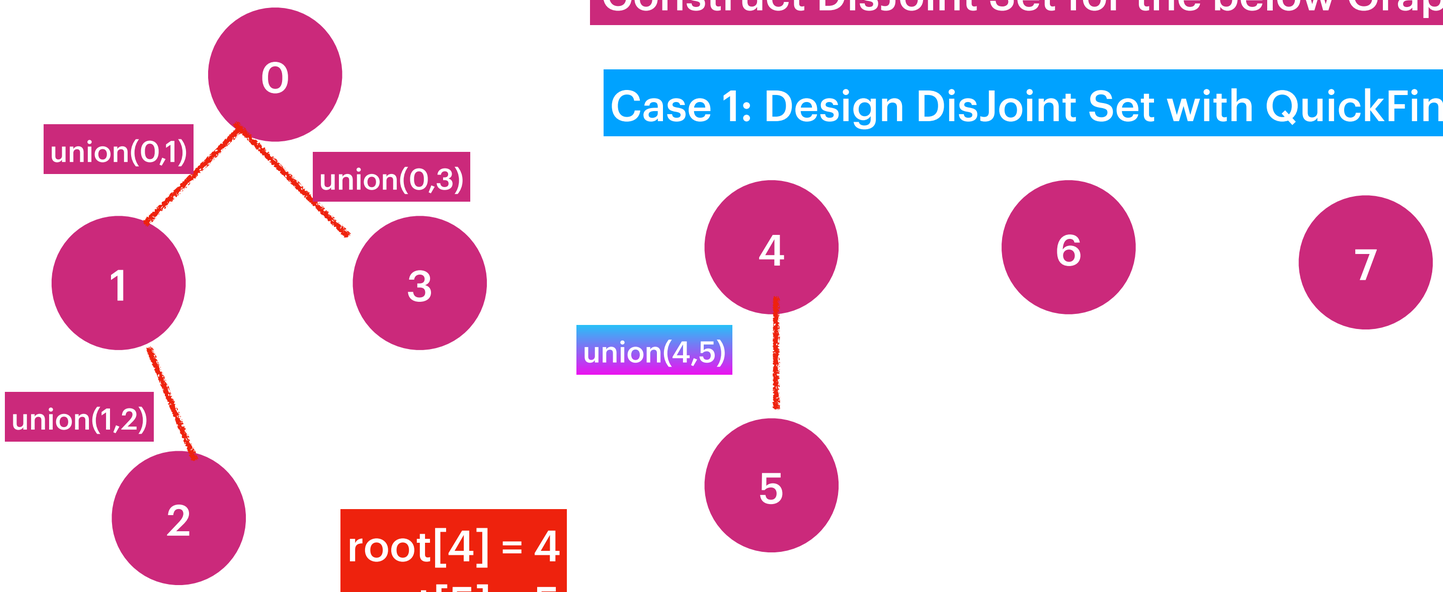
root[1] = 0
root[2] = 2





Construct DisJoint Set for the below Graphs

Case 1: Design DisJoint Set with QuickFind.



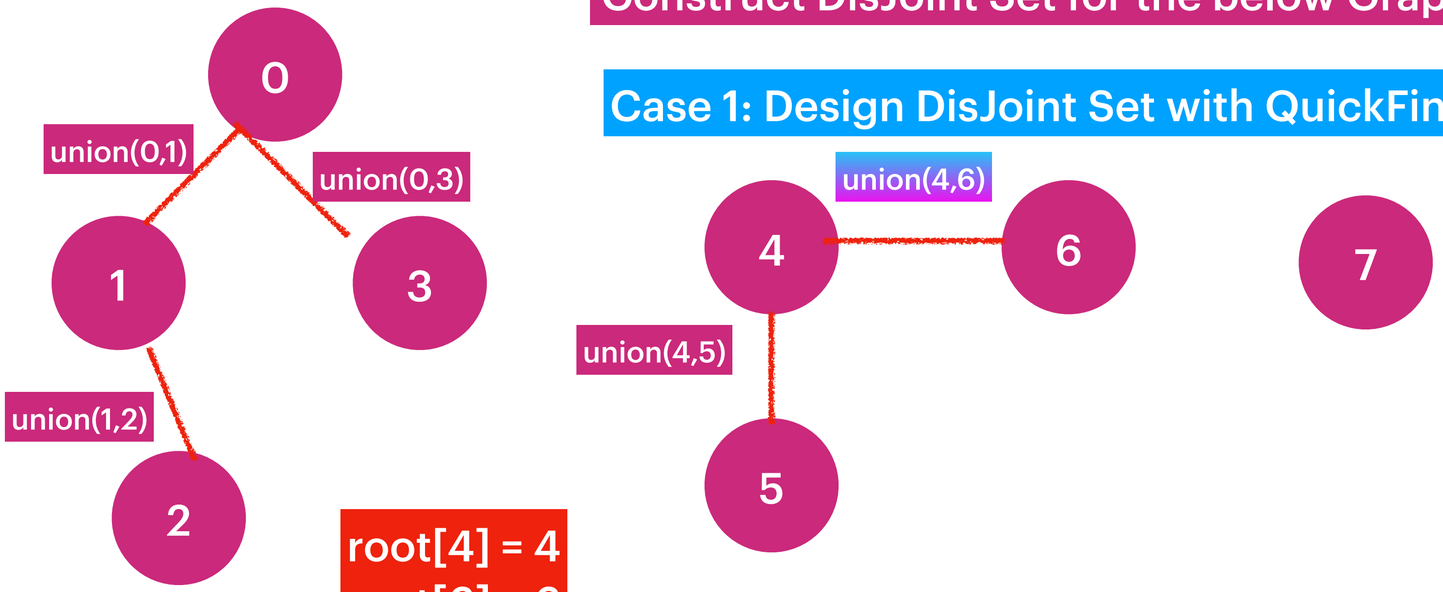
root[4] = 4
root[5] = 5

root	0	0	2	0	4	5	6	7
Index's/ Vertices	0	1	2	3	4	5	6	7

root	0	0	0	0	4	4	6	7
Index's/ Vertices	0	1	2	3	4	5	6	7

Construct DisJoint Set for the below Graphs

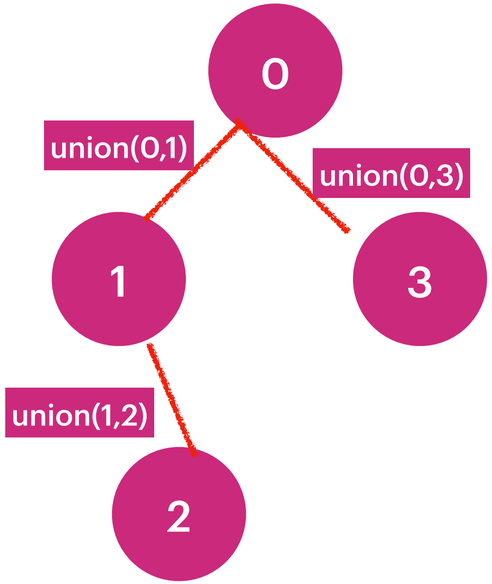
Case 1: Design DisJoint Set with QuickFind.



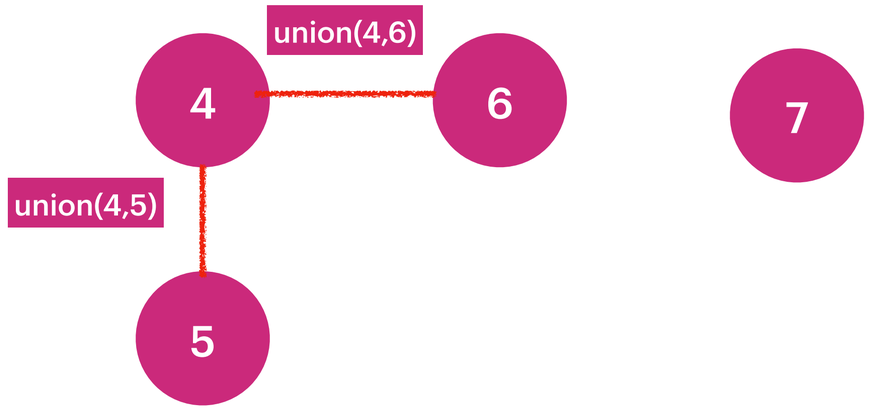
root[4] = 4
root[6] = 6

root	0	0	2	0	4	4	6	7
Index's/ Vertices	0	1	2	3	4	5	6	7

root	0	0	0	0	4	4	4	7
Index's/ Vertices	0	1	2	3	4	5	6	7



Case 1: Design DisJoint Set with QuickFind.

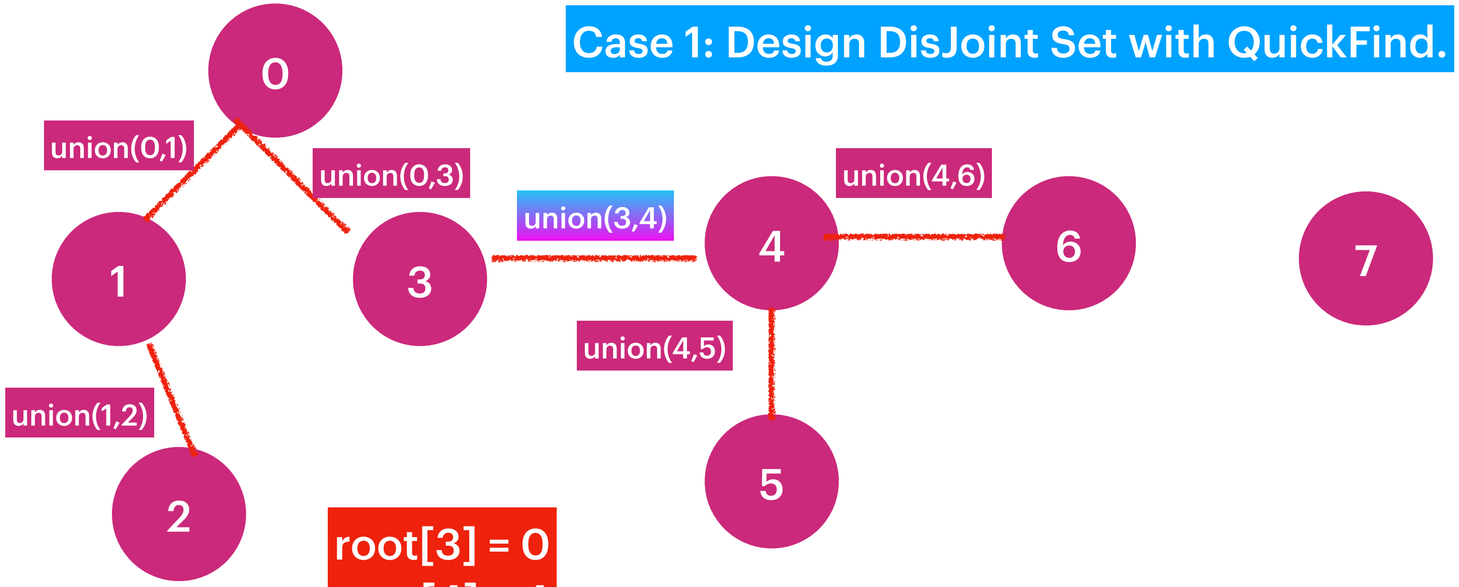


root	0	0	0	0	4	4	4	7
Index's/ Vertices	0	1	2	3	4	5	6	7

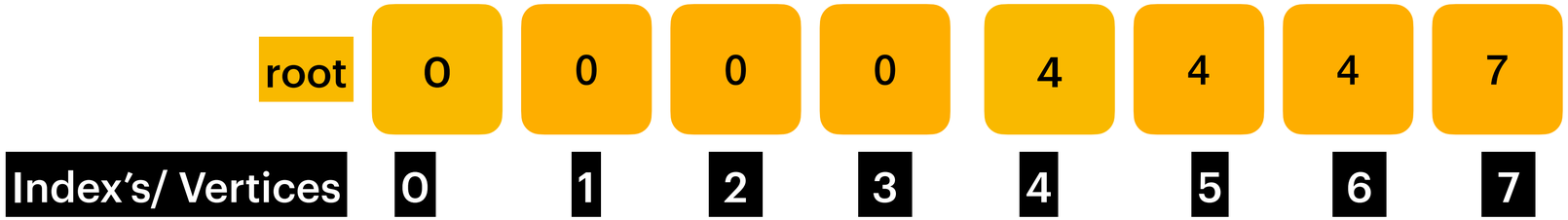
V(2) & V(3) connected. —> True (root[2] == root[3])

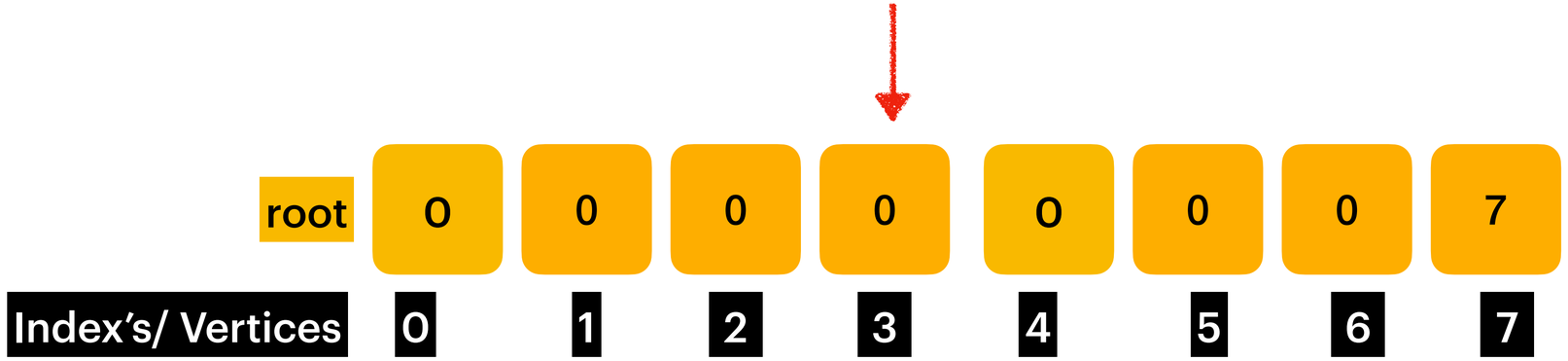
V(2) & V(5) are connected. —> False (root[2] != root[5])

Case 1: Design DisJoint Set with QuickFind.



root[3] = 0
root[4] = 4






```
public void union(int vx, int vy)
{
int rootX = find(vx);
int rootY = find(vy);
If(rootX != rootY)
{
root[vy] = rootX;
for(int v = 0 ; v < n ; v++)
{
if(root[v] == rootY)
{
root[v]  = rootX;
}
}
}
}
```

Time Complexity : O(n)

V(2) & V(6) connected. —> True (root[2] == root[6])