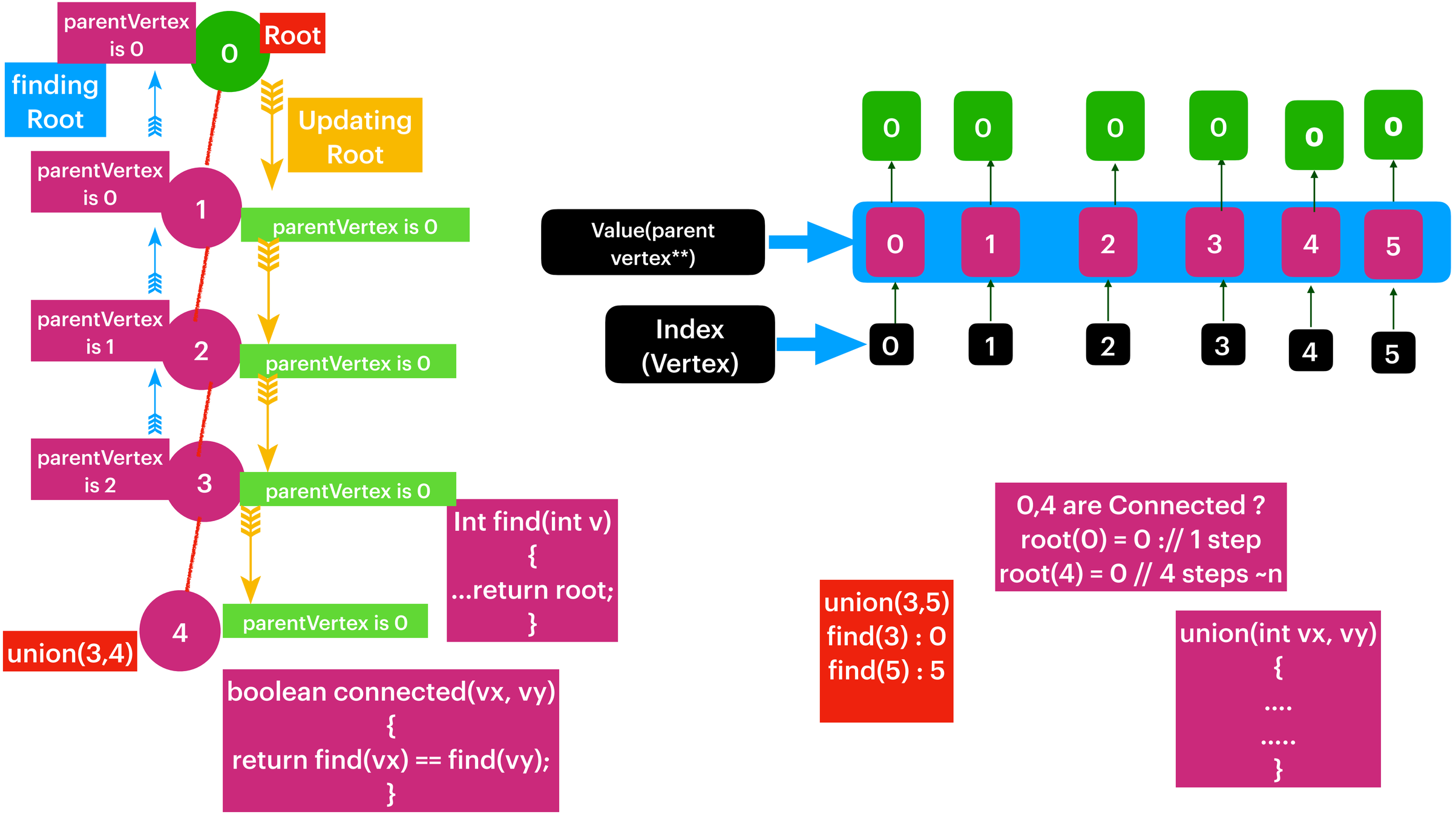
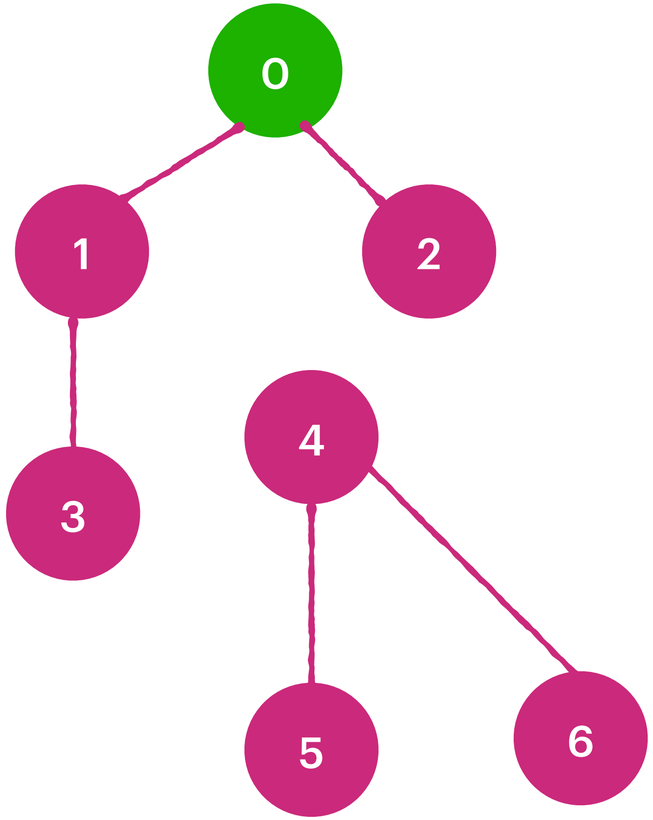
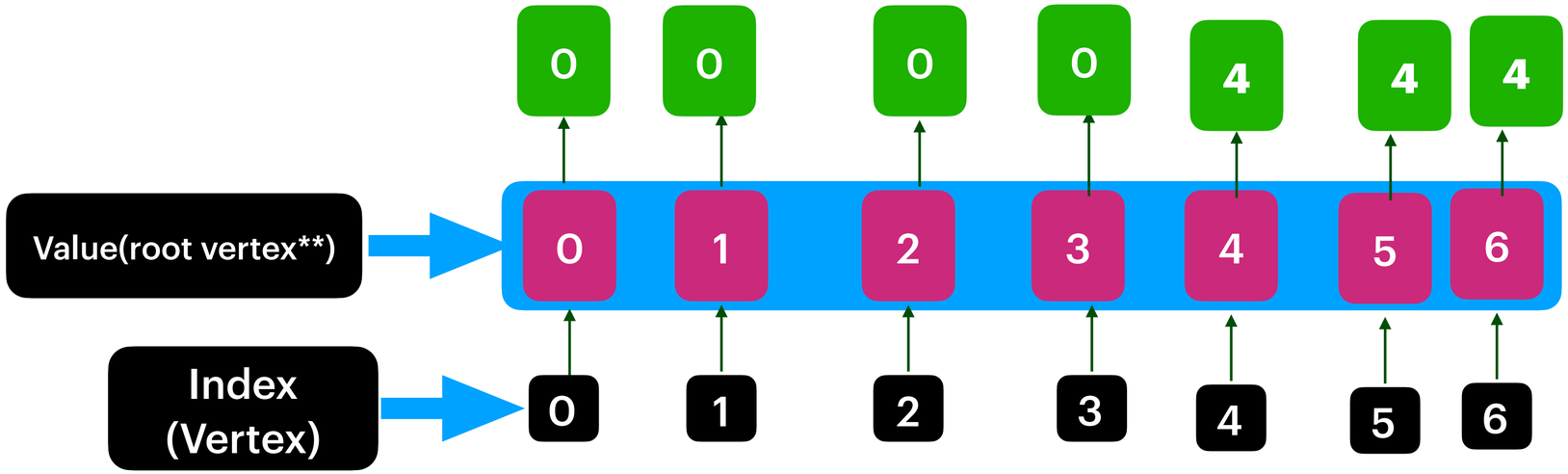

parentVertex is 0
0
Root
finding Root
Updating Root
parentVertex is 0
1
parentVertex is 0
parentVertex is 1
2
parentVertex is 0
parentVertex is 2
3
parentVertex is 0
4
parentVertex is 0
union(3,4)
Int find(int v)
{
...return root;
}
boolean connected(vx, vy)
{
return find(vx) == find(vy);
}
Value(parent vertex**)
Index (Vertex)
0 0 0 0 0 0
0 1 2 3 4 5
0 1 2 3 4 5
0,4 are Connected ?
root(0) = 0 :// 1 step
root(4) = 0 // 4 steps ~n
union(3,5)
find(3) : 0
find(5) : 5
union(int vx, vy)
{
....
.....
}



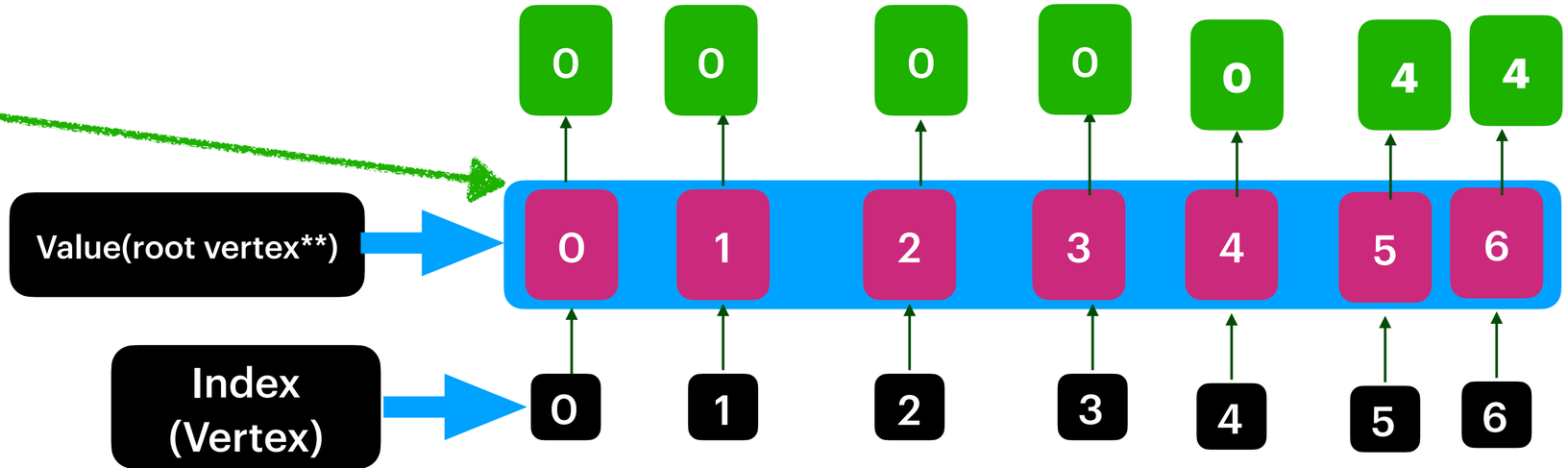
Quick Union :

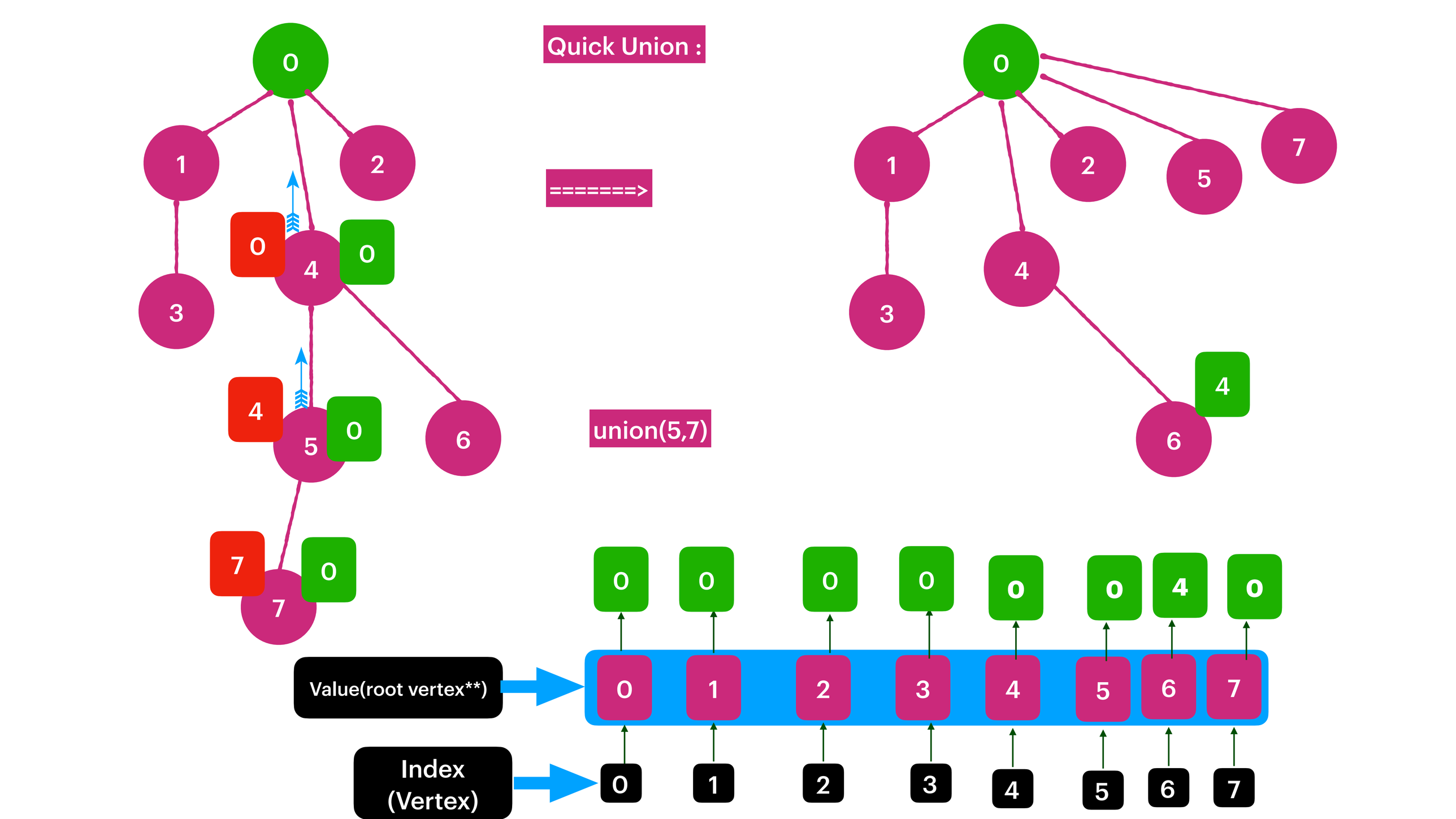


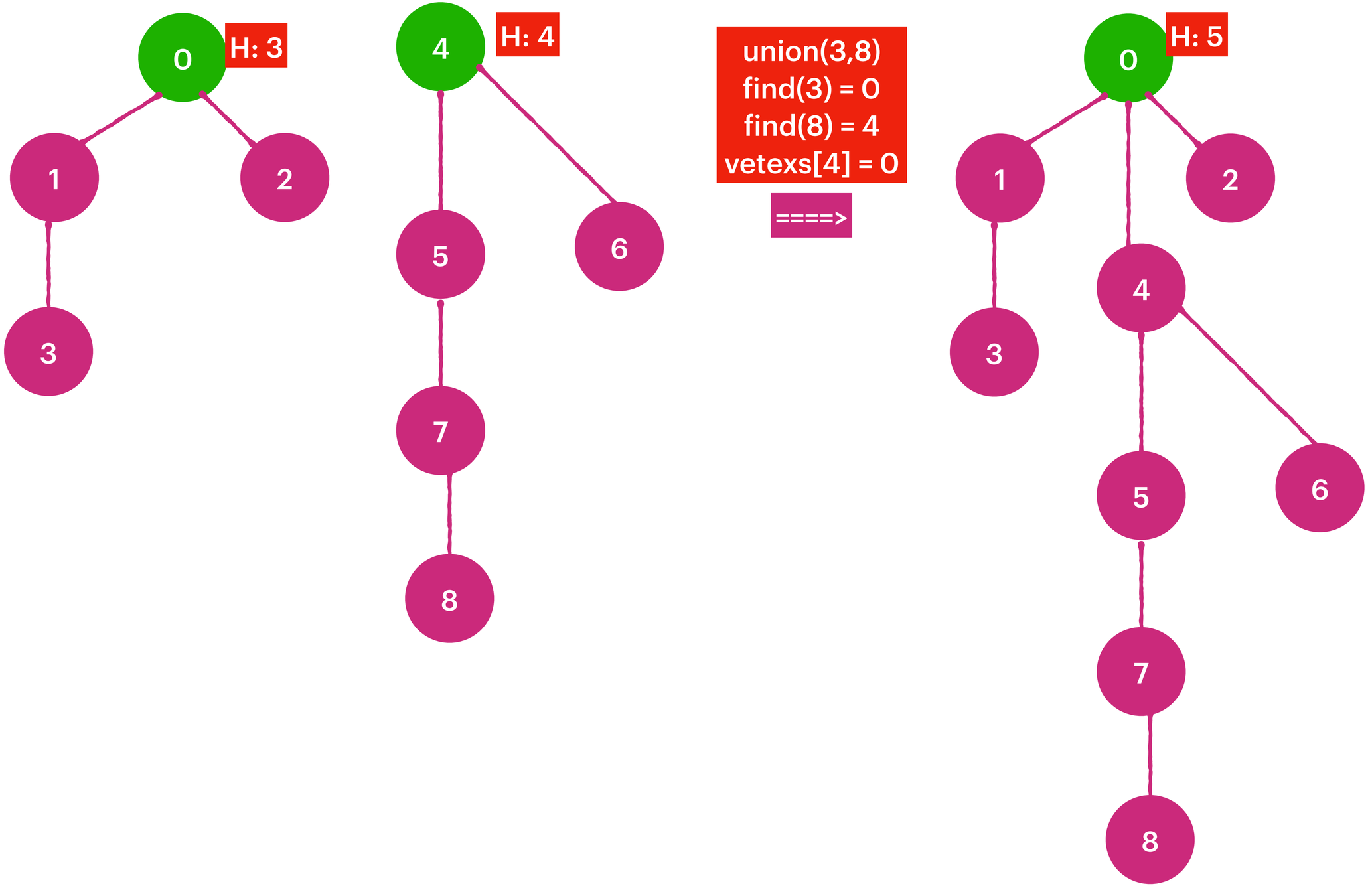
union(0,1)
union(0,2)
union(1,3)
union(4,5)
union(4,6)

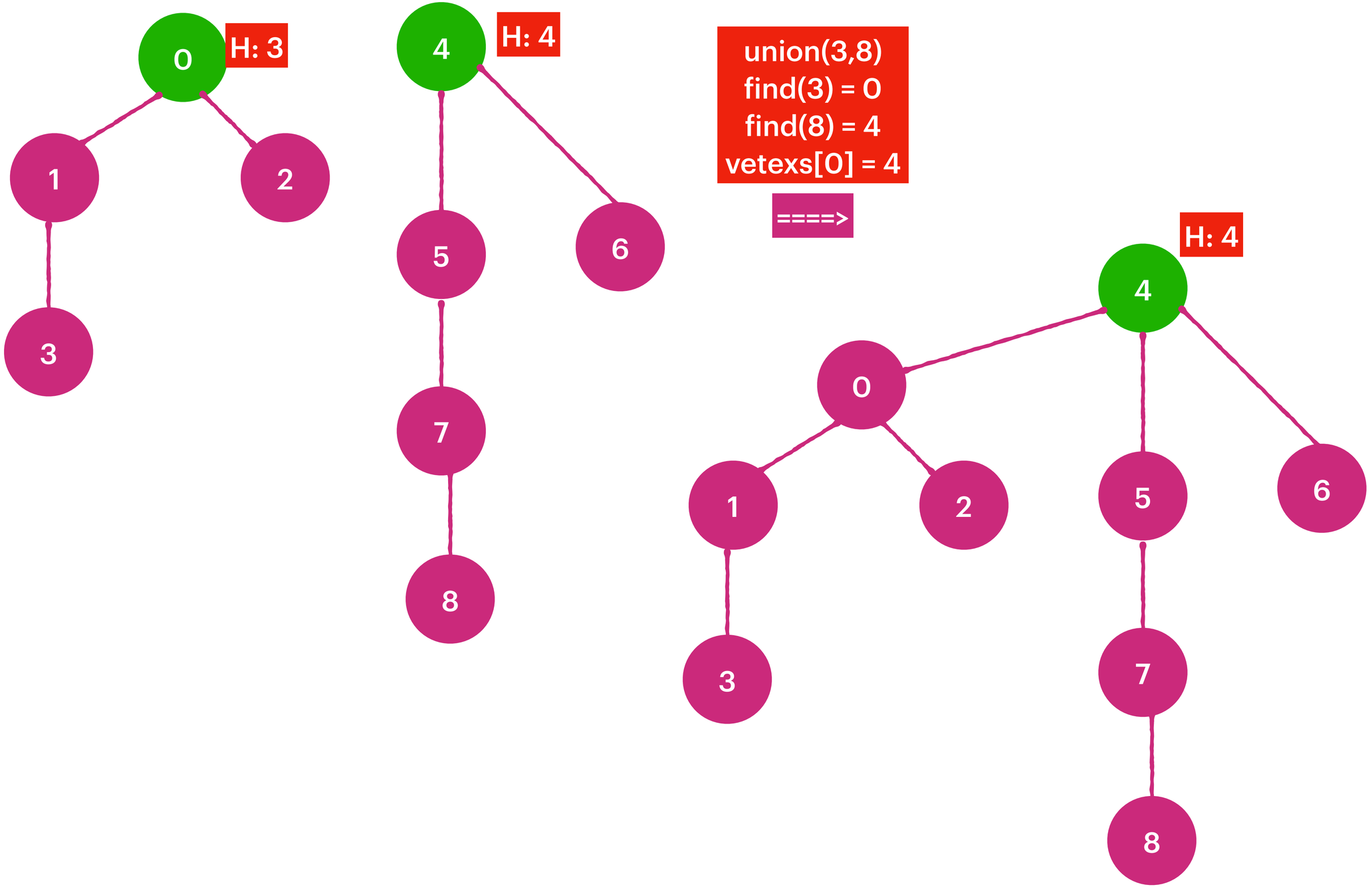
union(0, 1)
Int rootX = find(x); // 0
Int rootY = find(y); // 1
root[rootY] = rootX; //

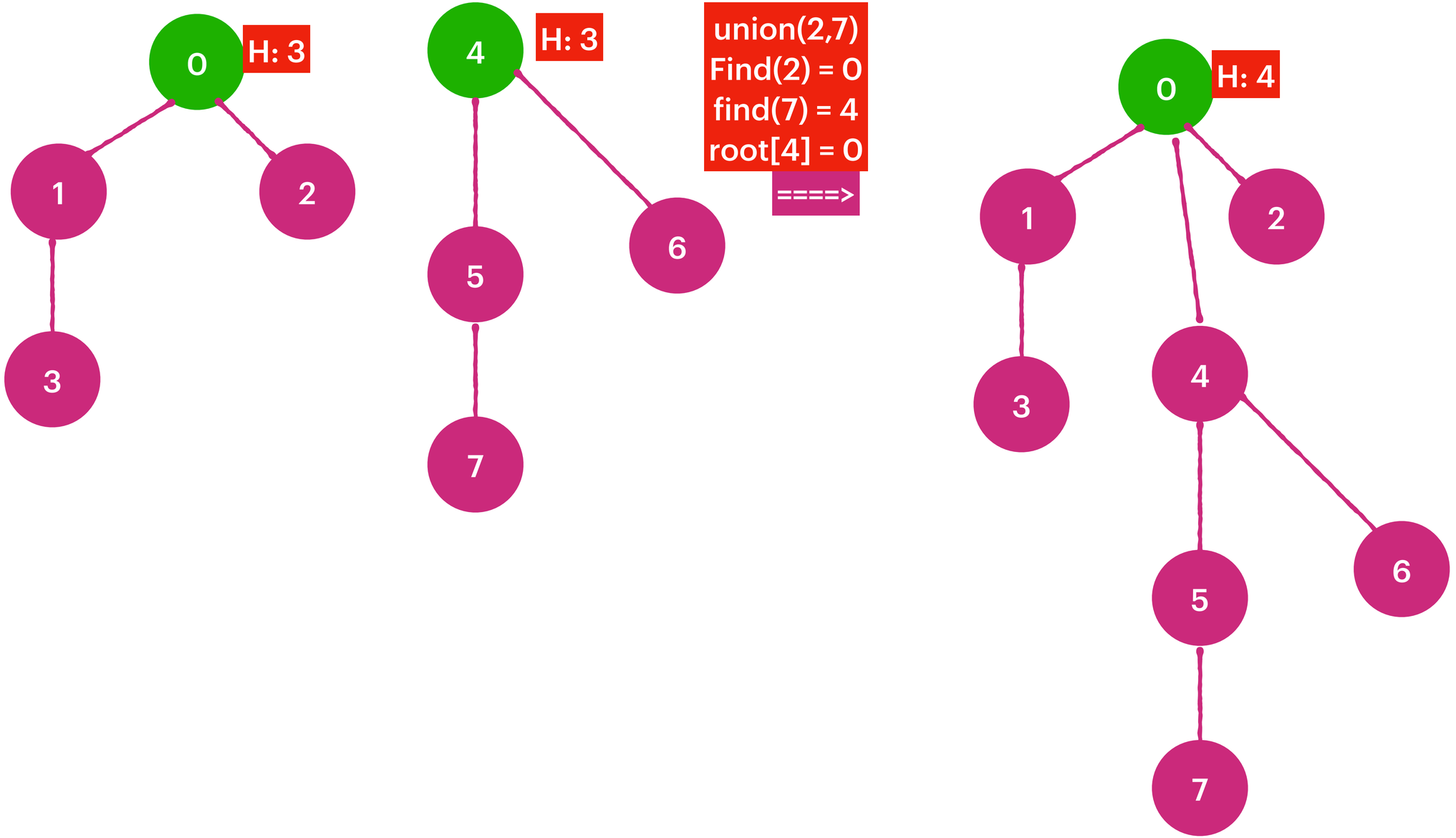
After union(3,5)
Int rootX = find(3) = 0
Int rootY = find(5) = 4
root[rootY] = rootX
root[4] = 0

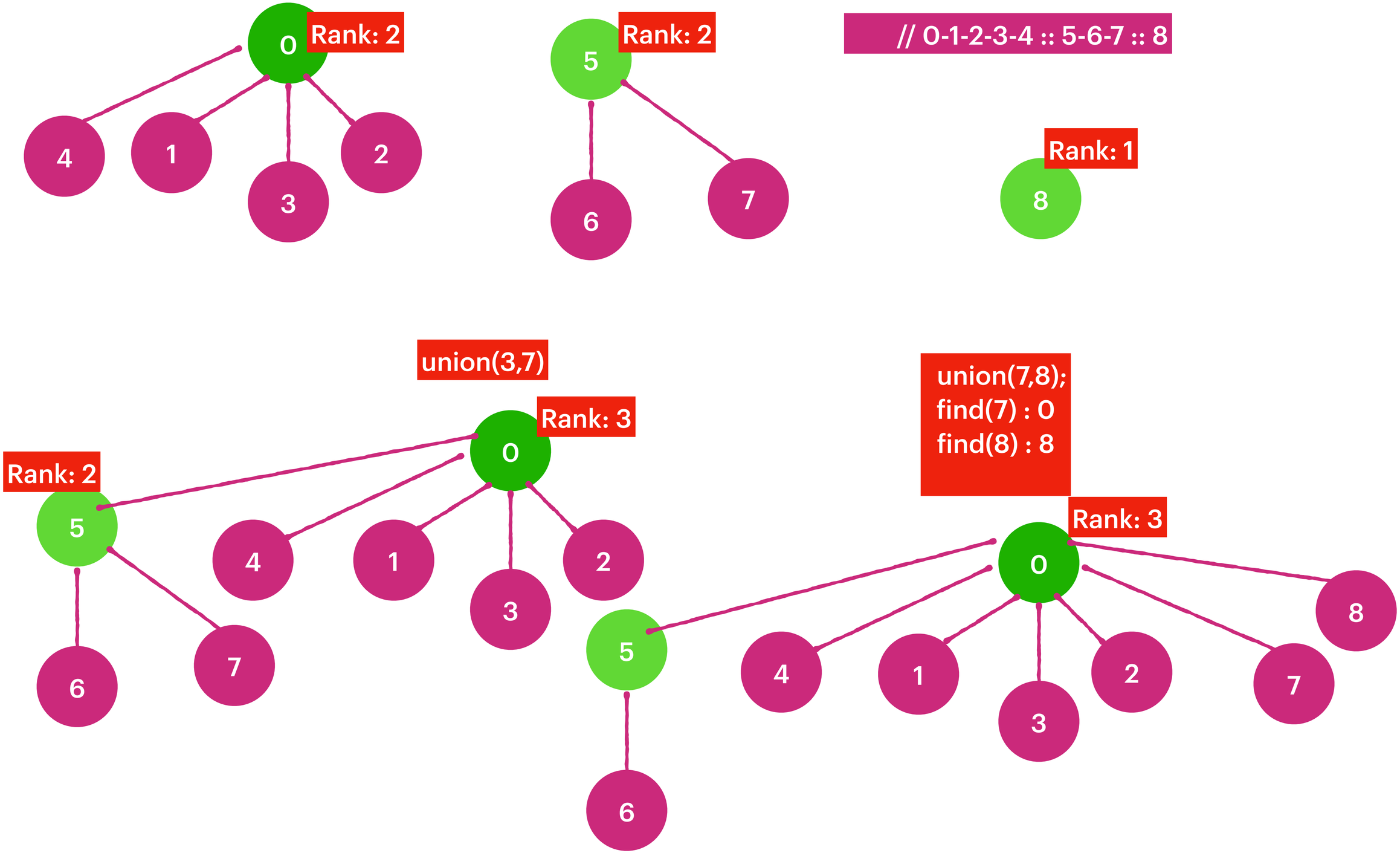


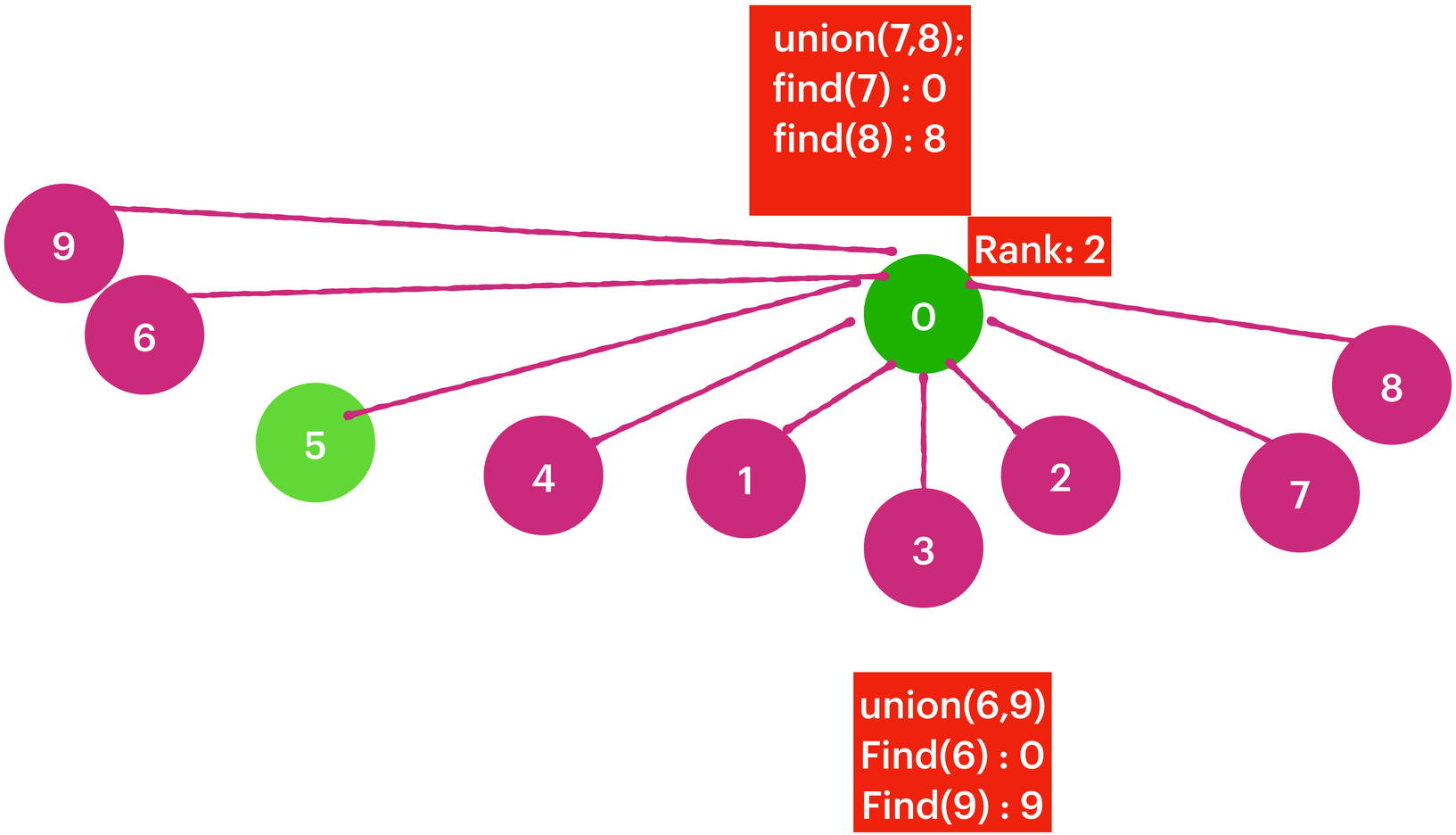

Quick Union :
=======>
union(5,7)
Value(root vertex**)
Index
(Vertex)


0
H: 3
1
2
3
4
H: 4
5
6
7
8
union(3,8)
find(3) = 0
find(8) = 4
vetexs[4] = 0
====>
0
H: 5
1
2
3
4
5
6
7
8


0
H: 3
1
2
3
4
H: 4
5
6
7
8
union(3,8)
find(3) = 0
find(8) = 4
vetexs[0] = 4
====>
4
H: 4
0
1
2
5
6
3
7
8


0
H: 3
1
2
3
4
H: 3
5
6
7
union(2,7)
Find(2) = 0
find(7) = 4
root[4] = 0
====>
0
H: 4
1
2
3
4
5
6
7


0
Rank: 2
4
1
3
2
5
Rank: 2
6
7
// 0-1-2-3-4 :: 5-6-7 :: 8
Rank: 1
8
union(3,7)
Rank: 3
0
Rank: 2
5
4
1
3
2
6
7
5
6
union(7,8);
find(7) : 0
find(8) : 8
Rank: 3
0
4
1
3
2
8
7


union(7,8);
find(7) : 0
find(8) : 8
Rank: 2
9
6
0
8
5
4
1
2
7
3
union(6,9)
Find(6) : 0
Find(9) : 9