Disjoint Set Union Find

{A}{B} 🕵️‍♂️U🕵️‍♀️

Disjoint Set Union Find

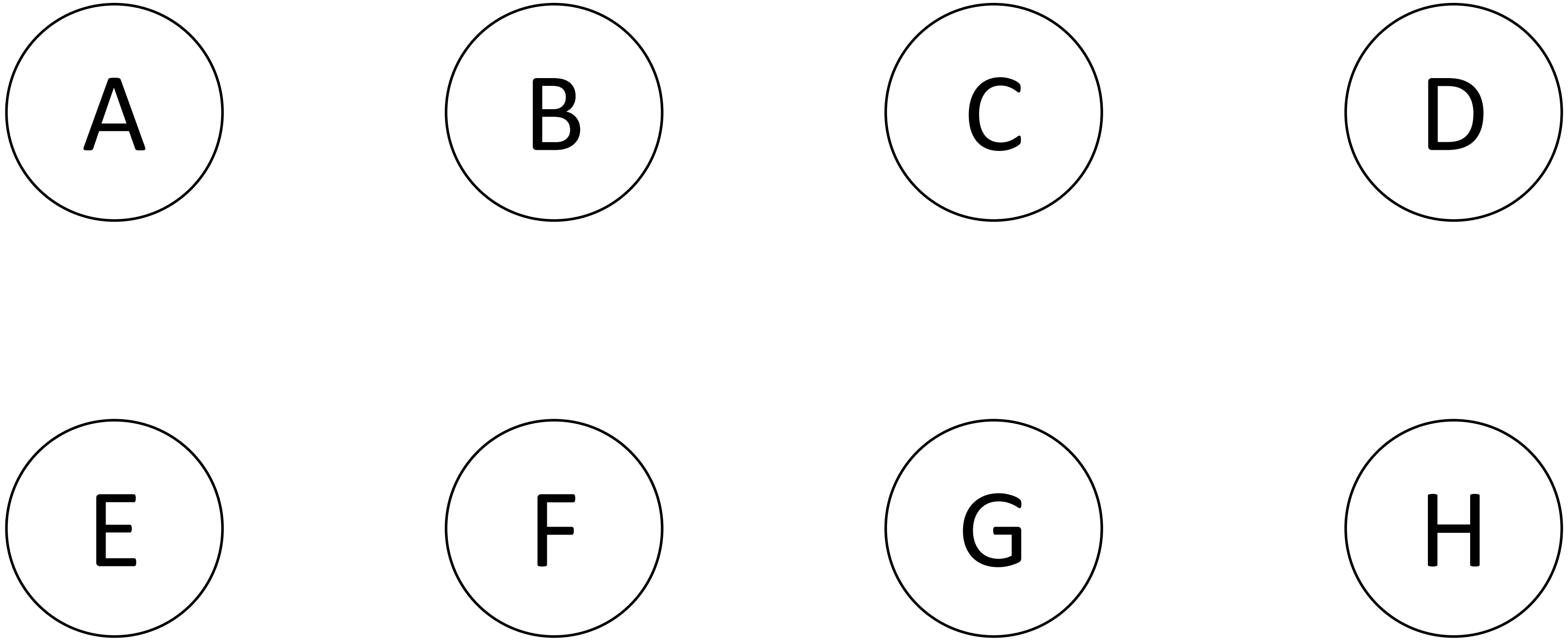
Make Set

Find

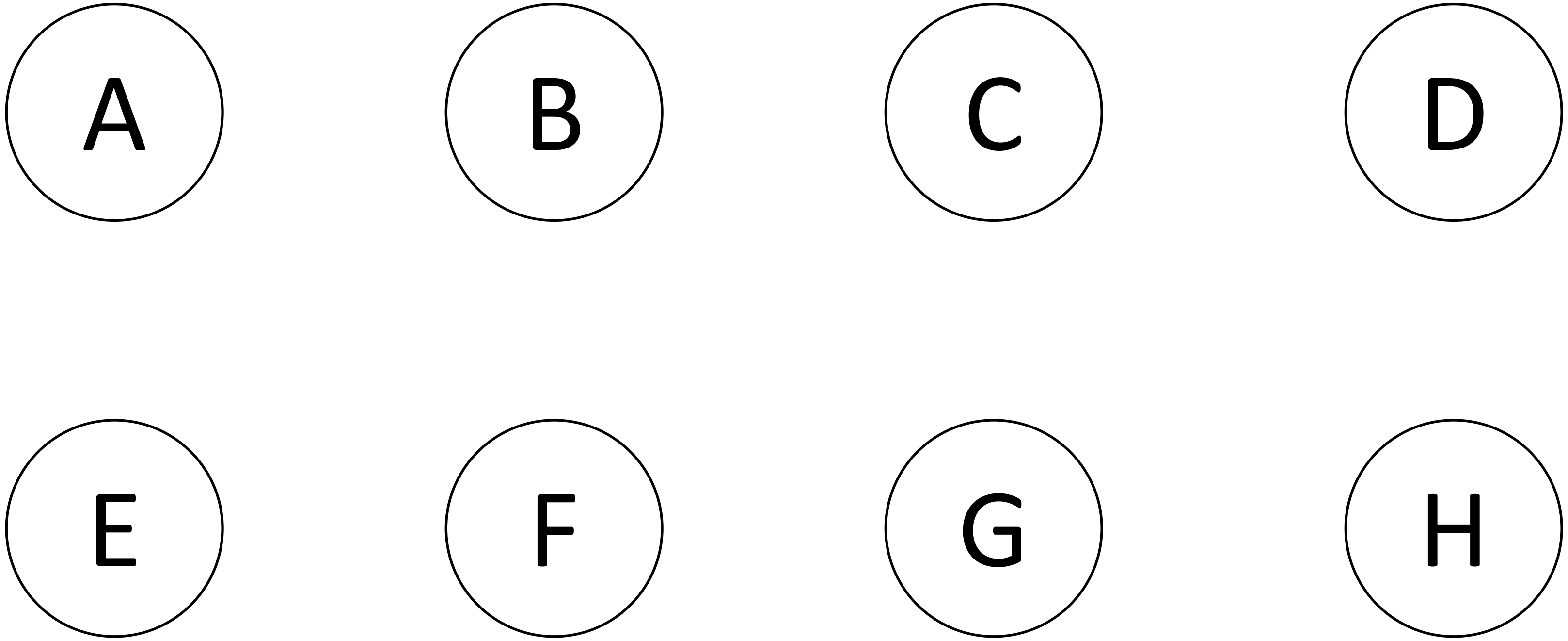
Union

Path Compression

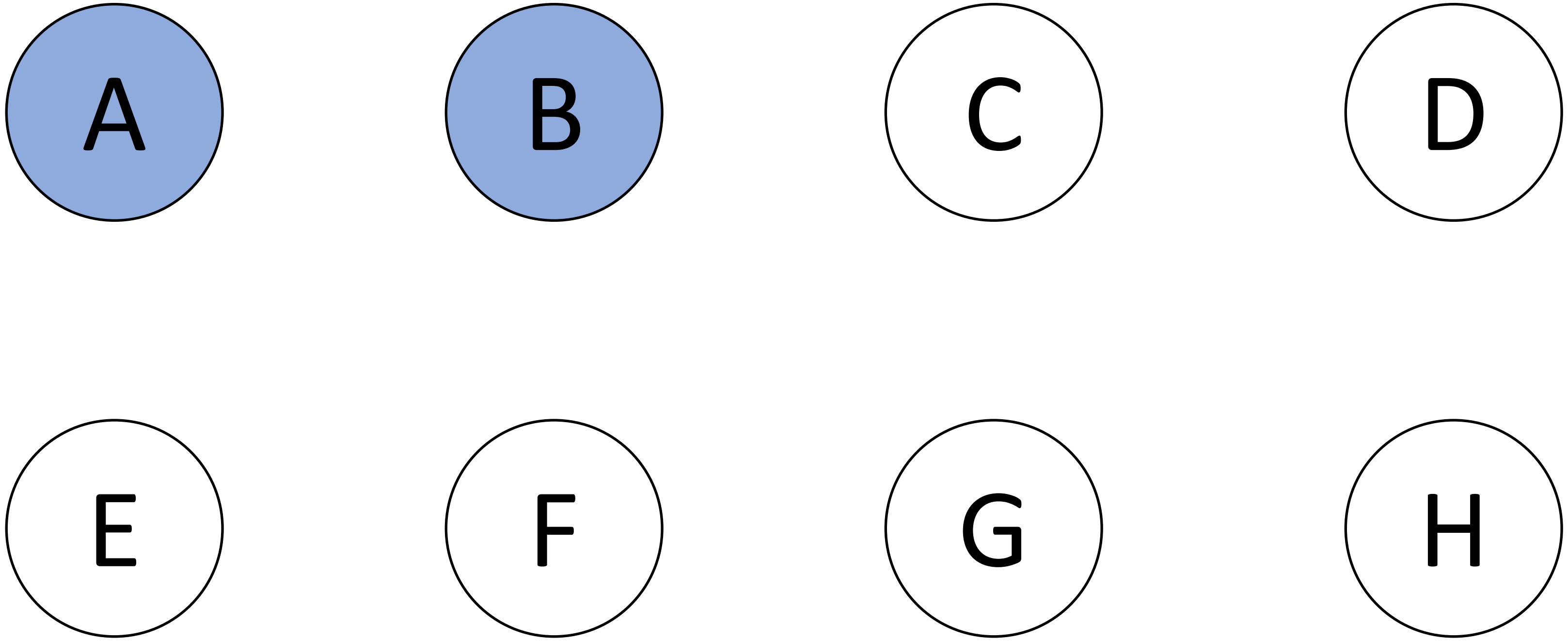
Make Set



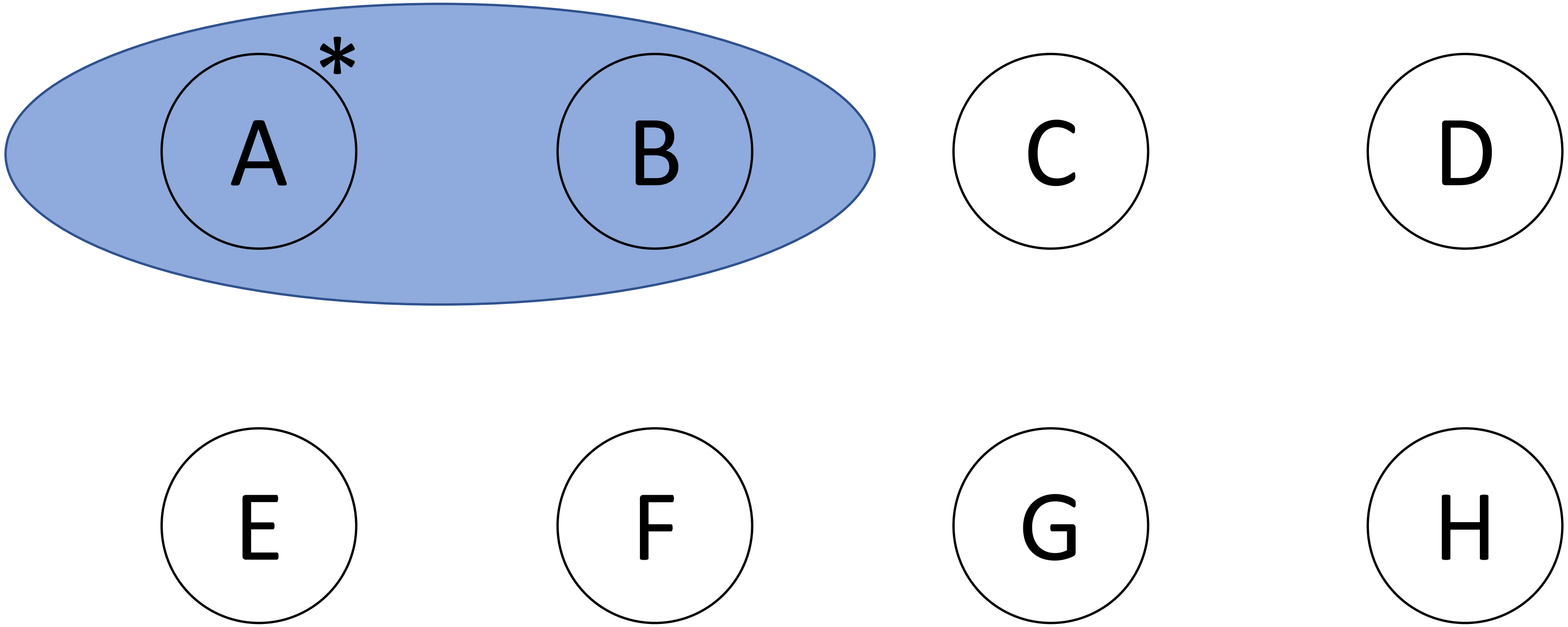
Union A B



Union A B

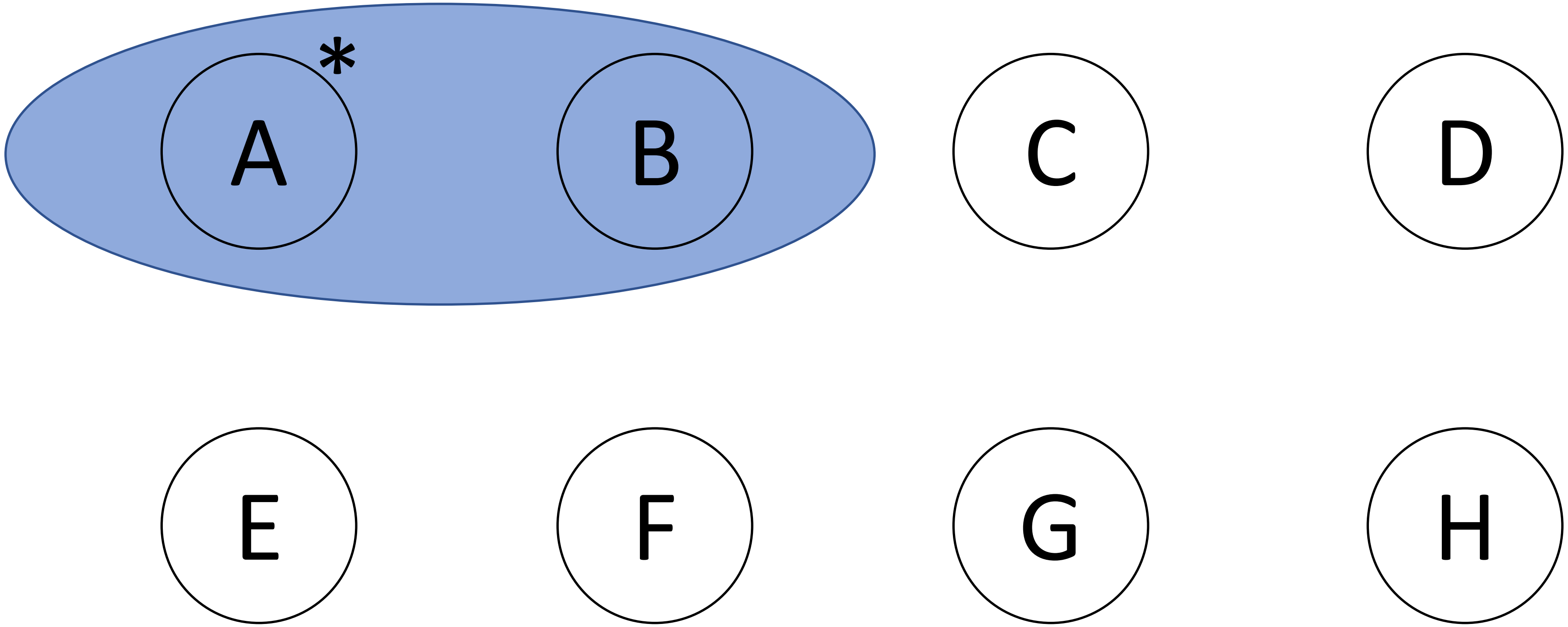


Union A B

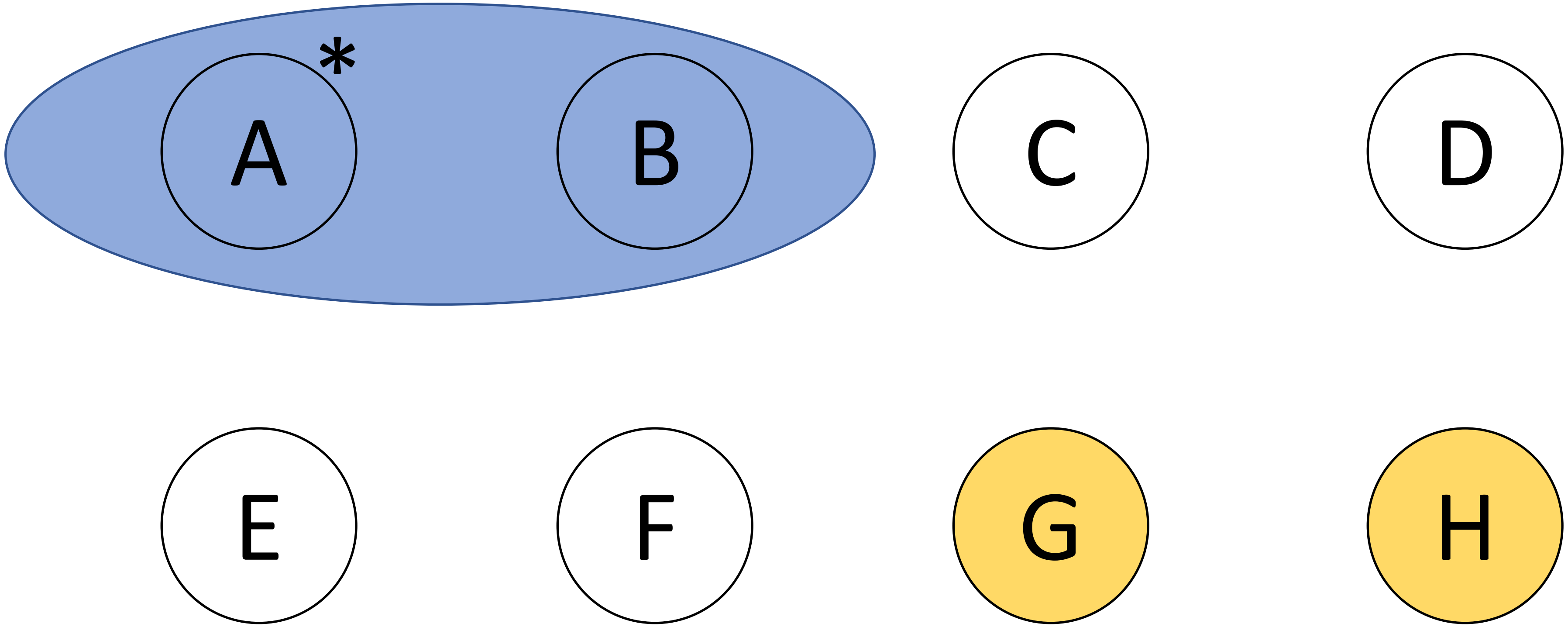


Find A
Find B

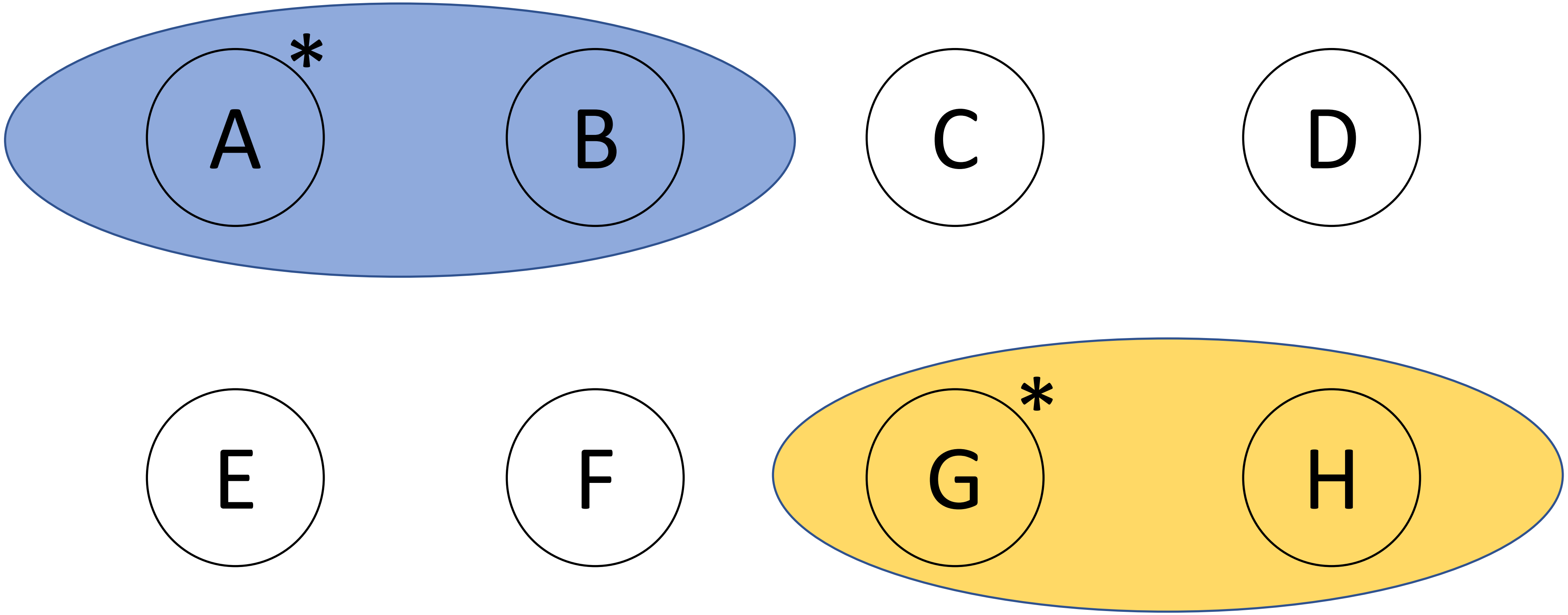
Union G H



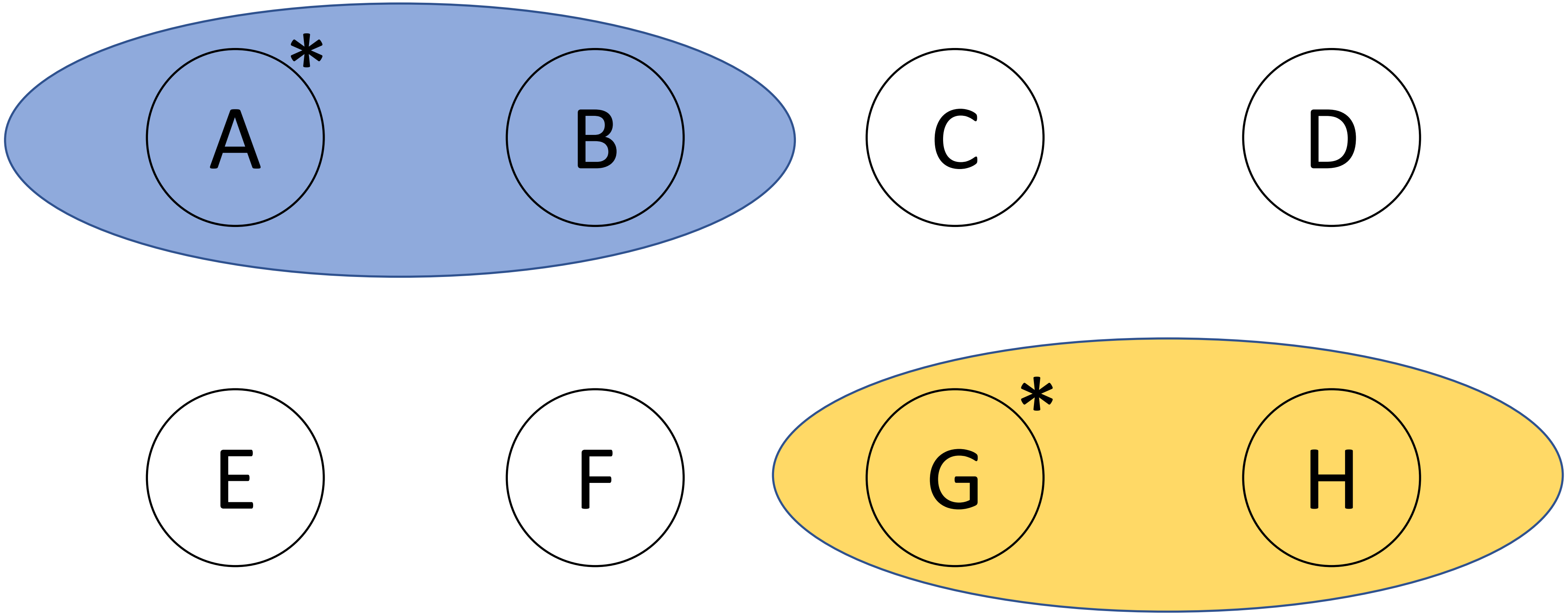
Union G H



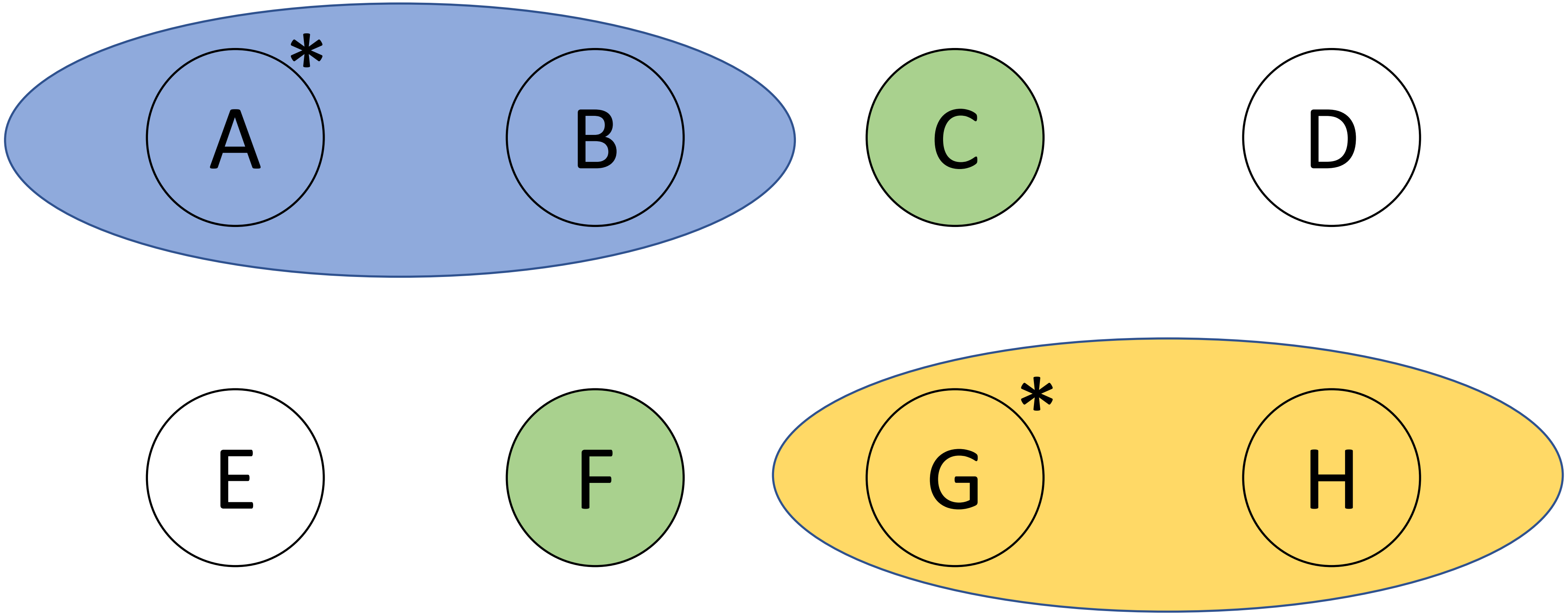
Union G H



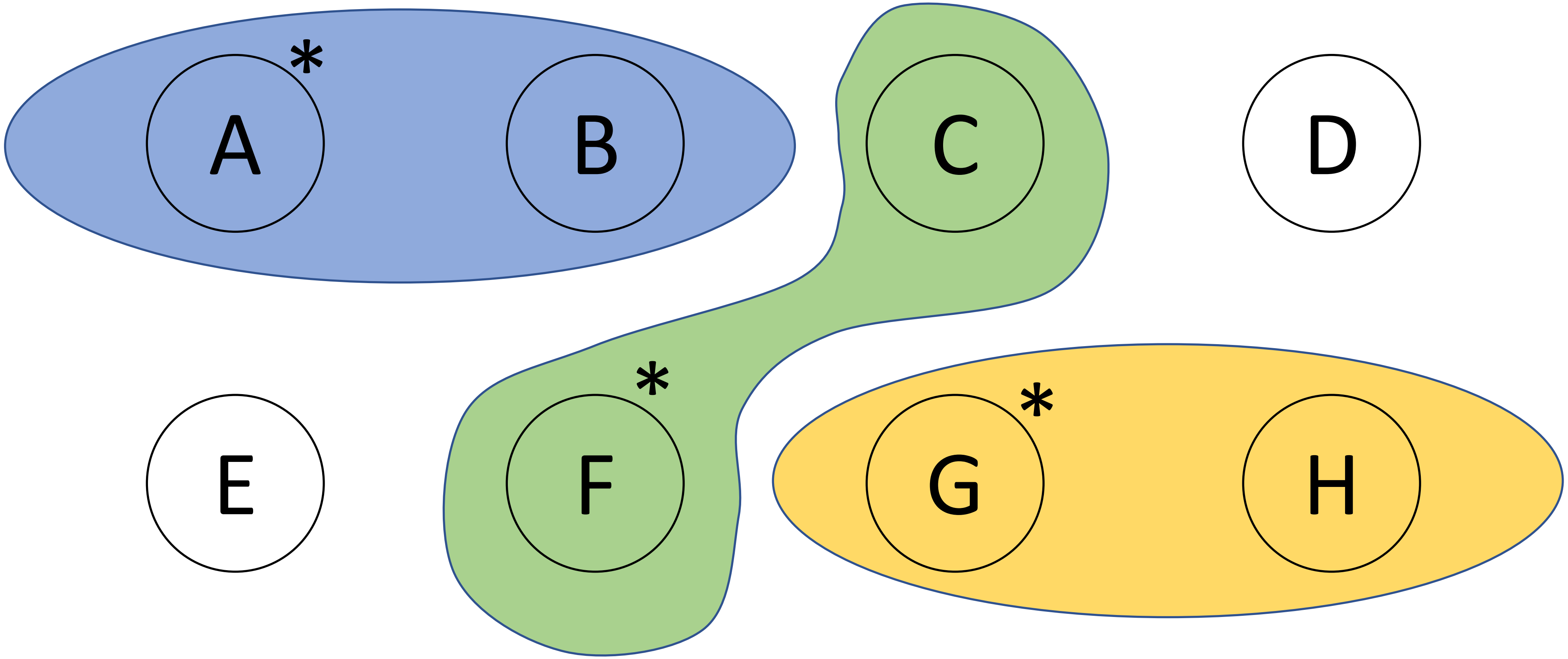
Union F C



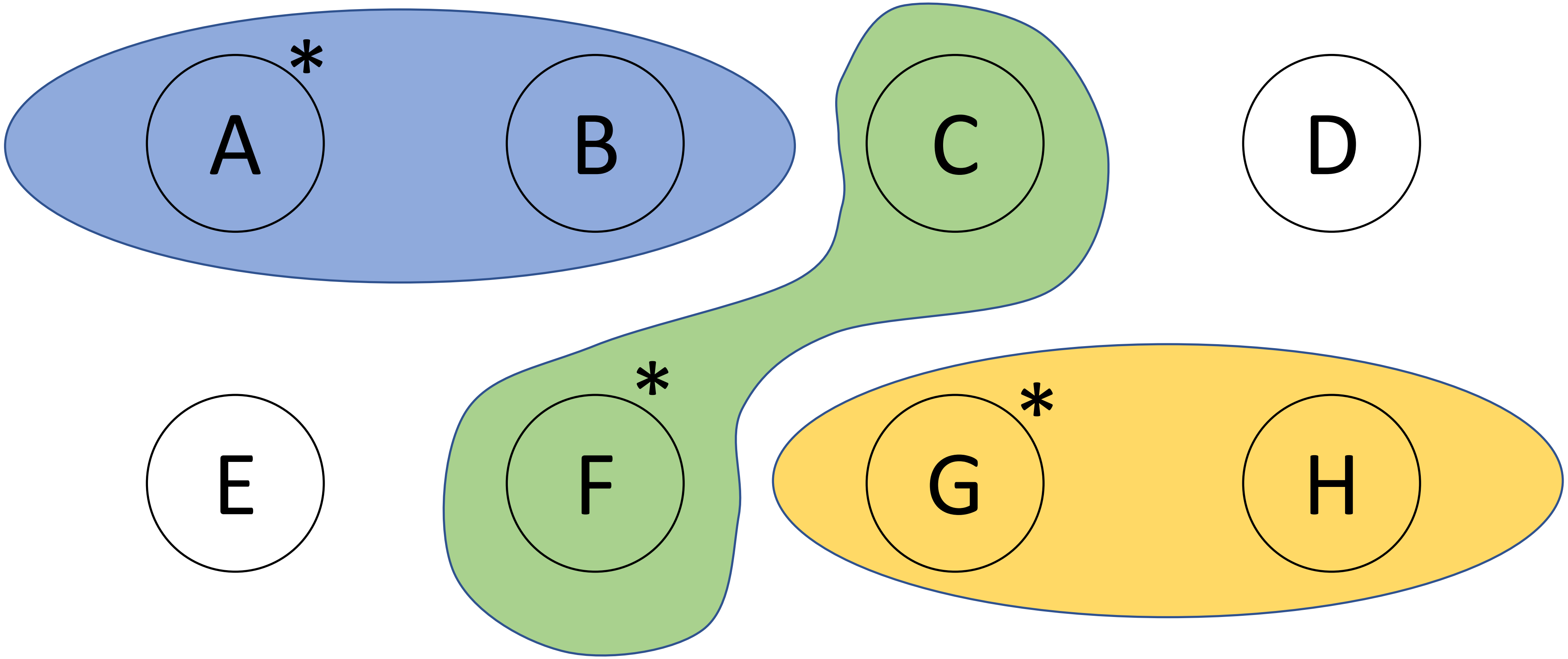
Union F C



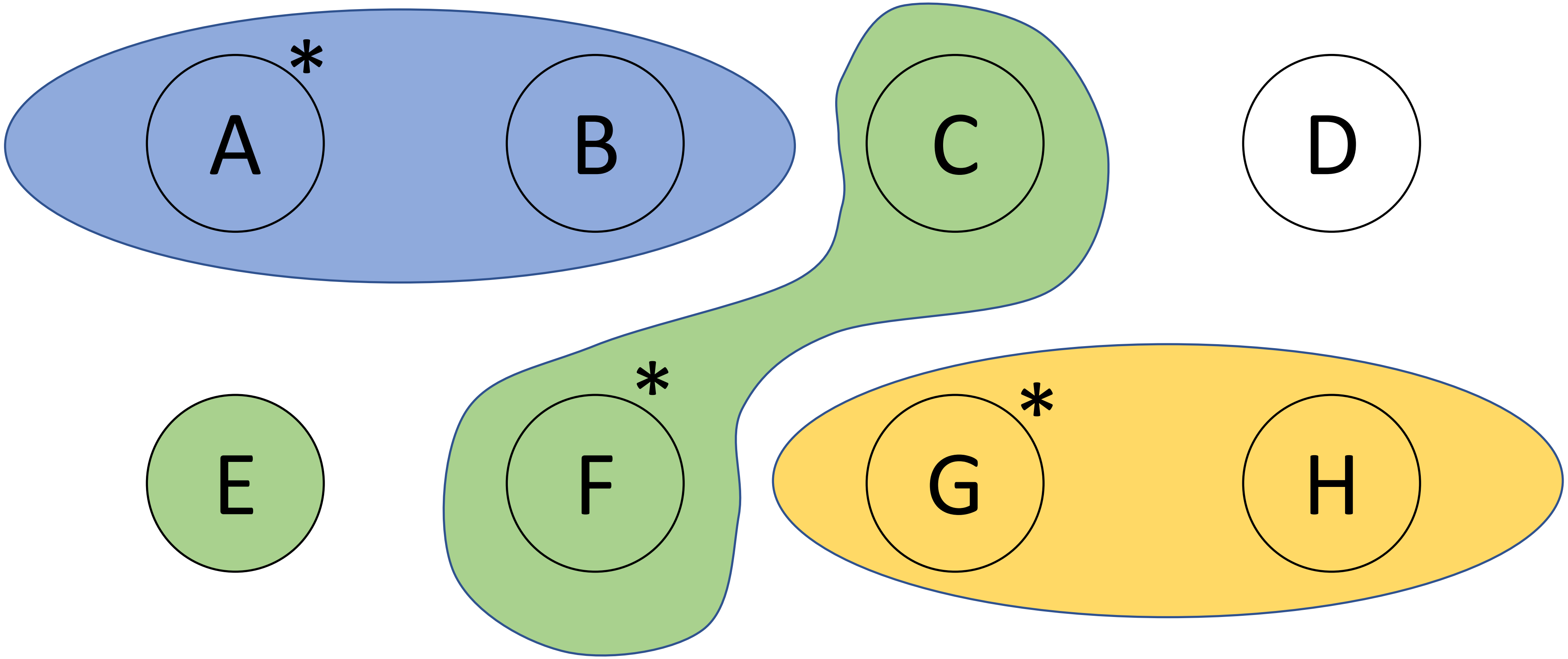
Union F C



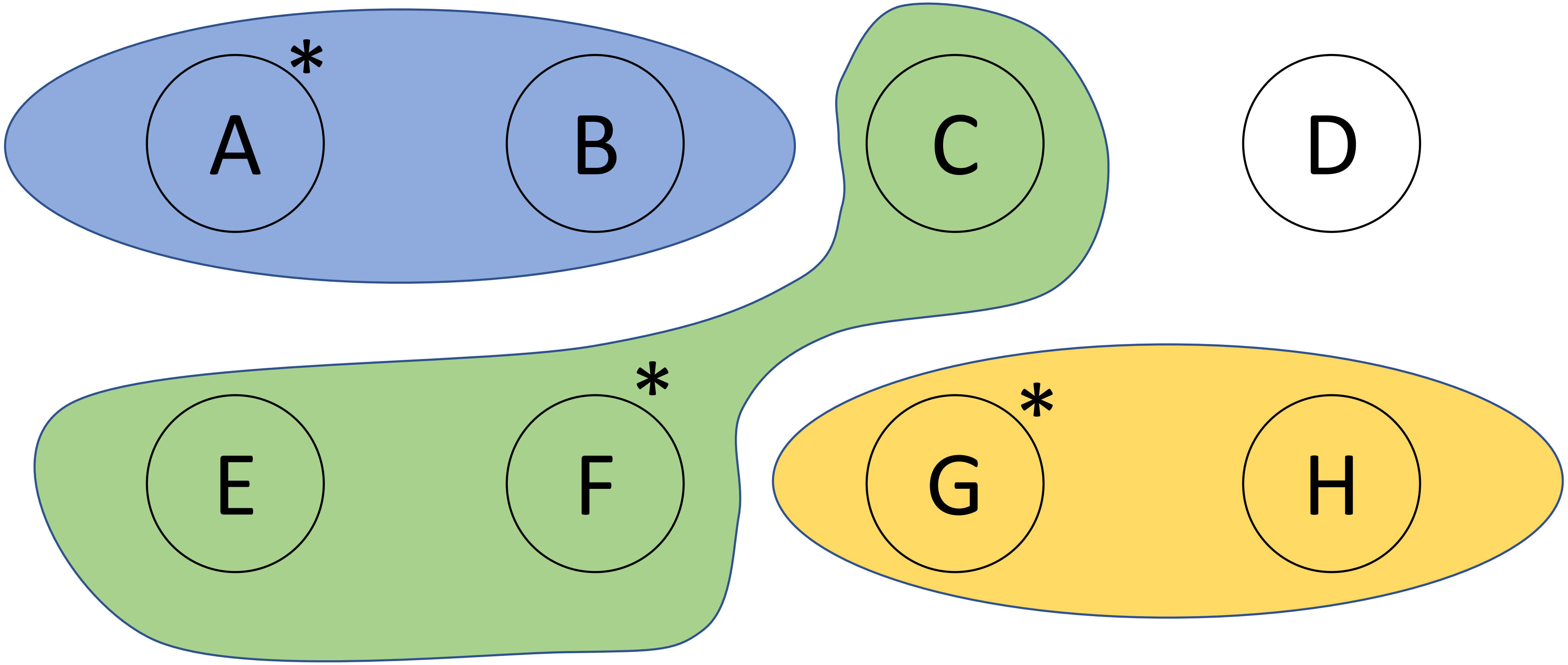
Union E F



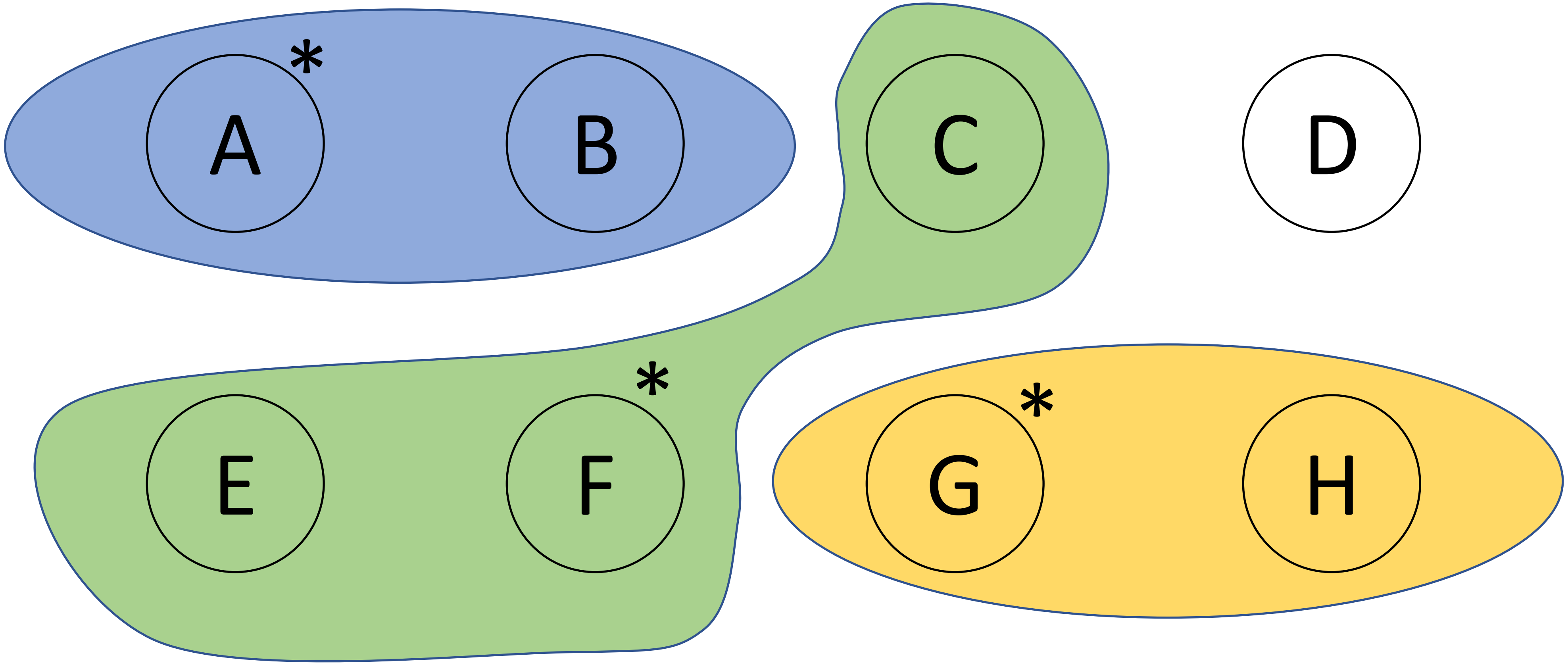
Union E F



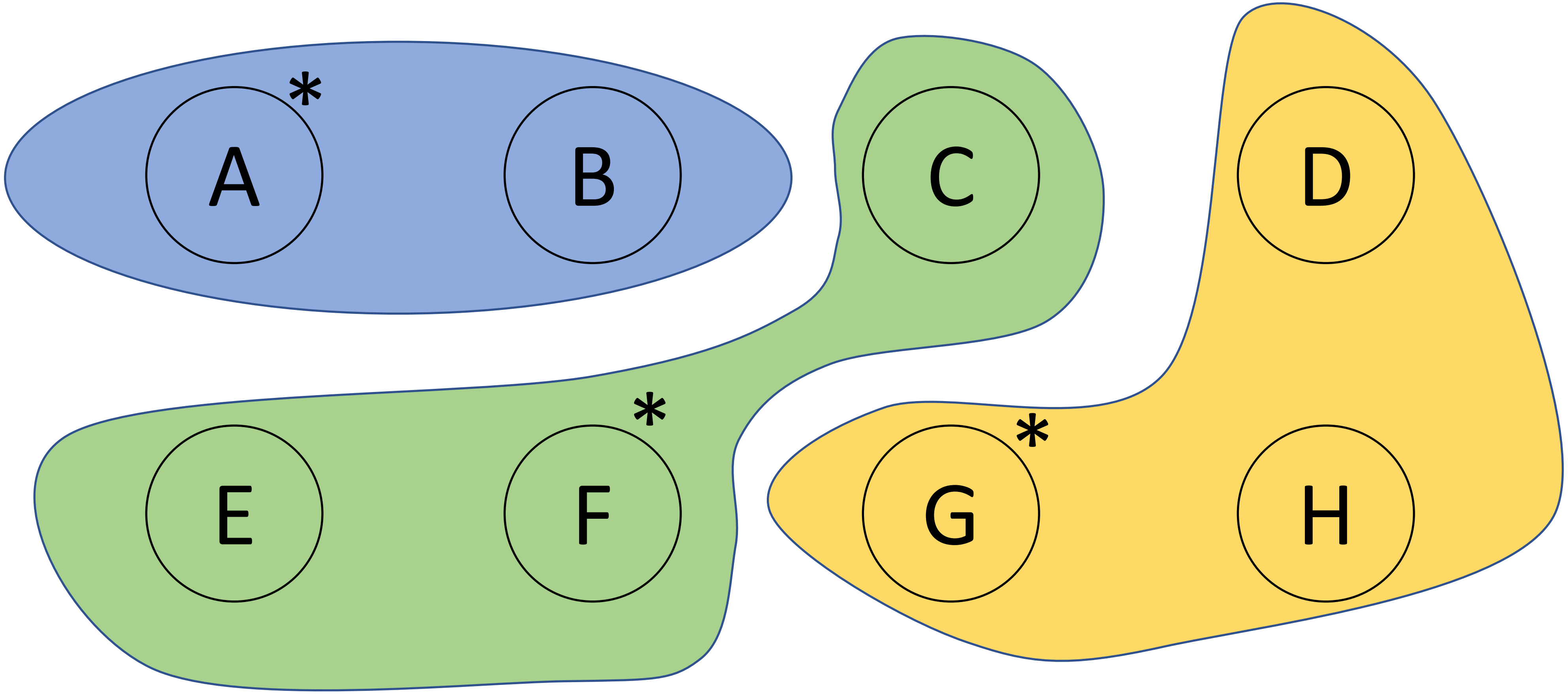
Union E F



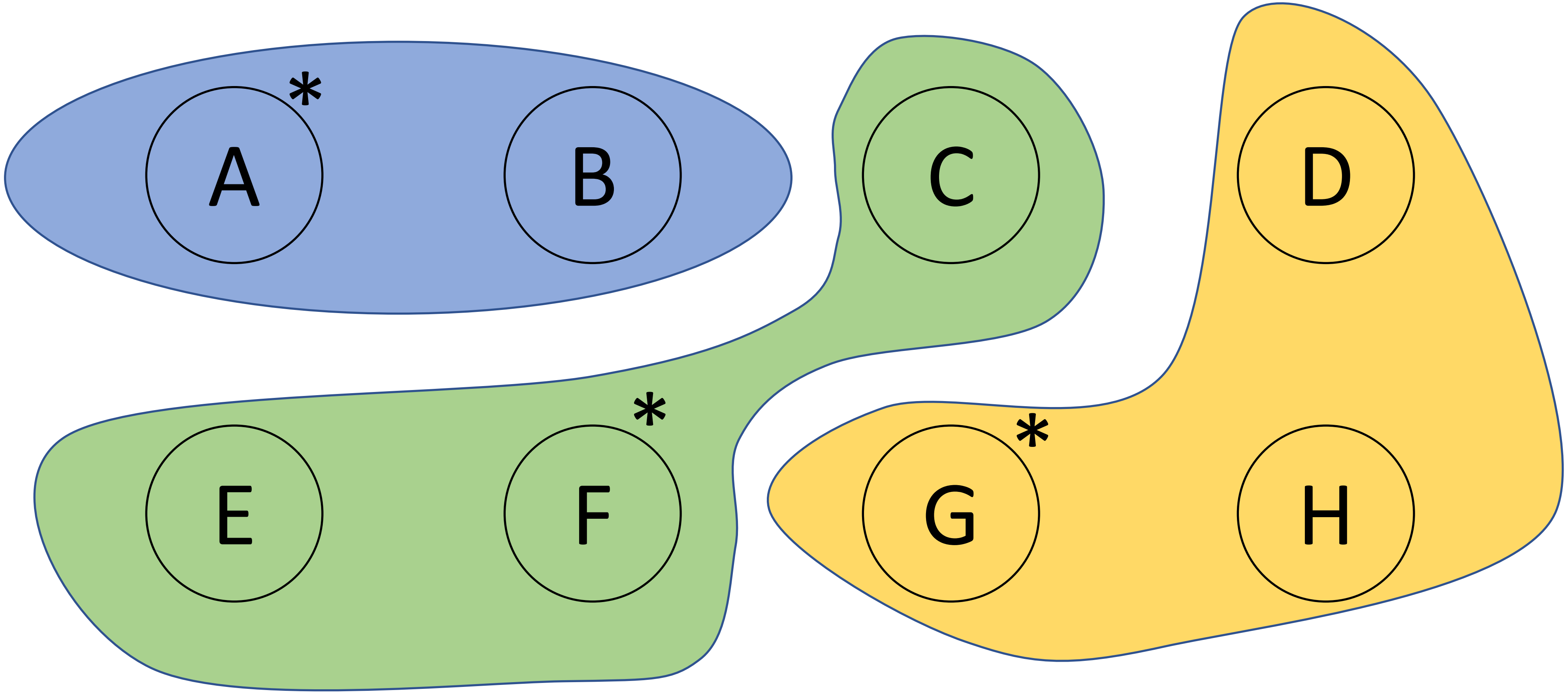
Union D G



Union D G

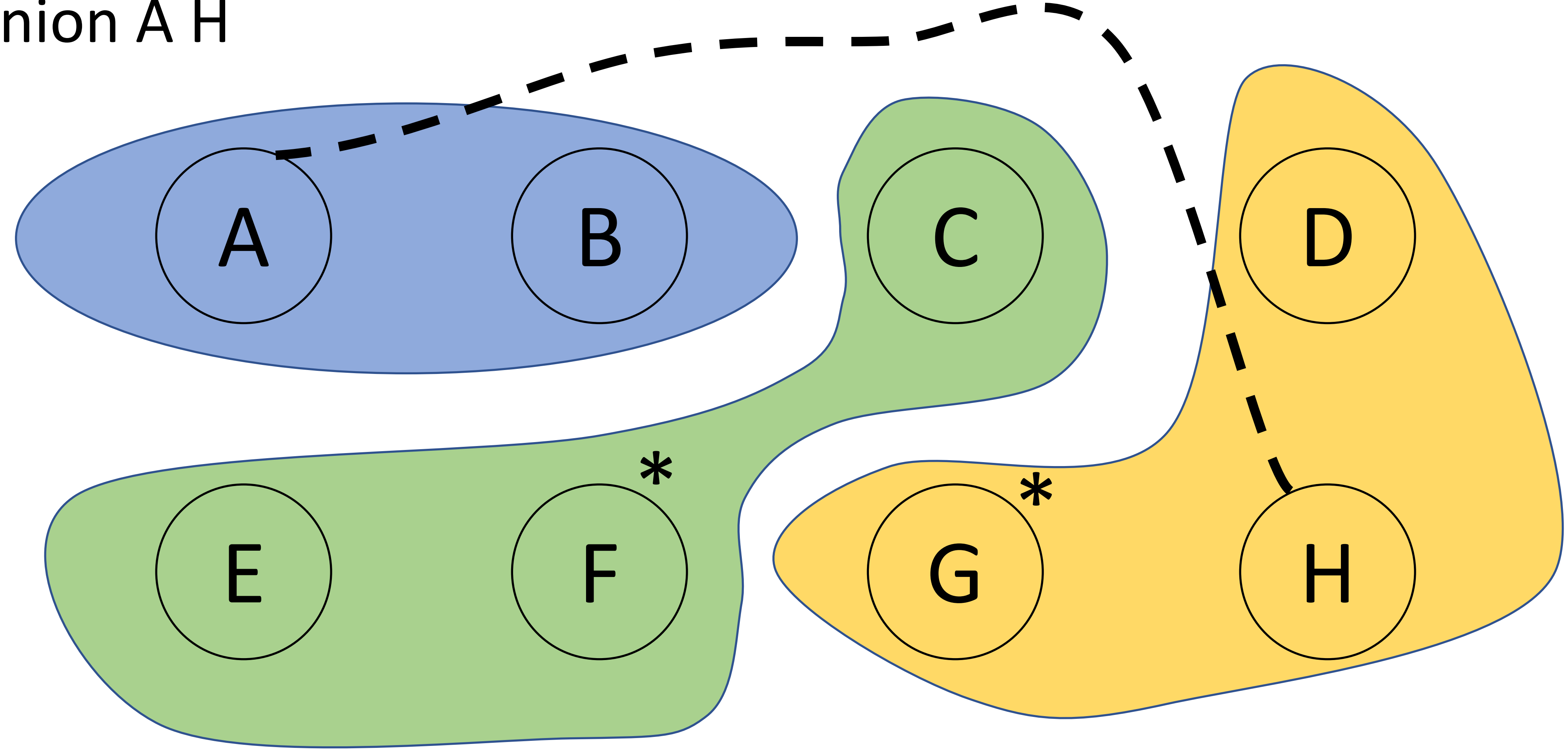


Union A H

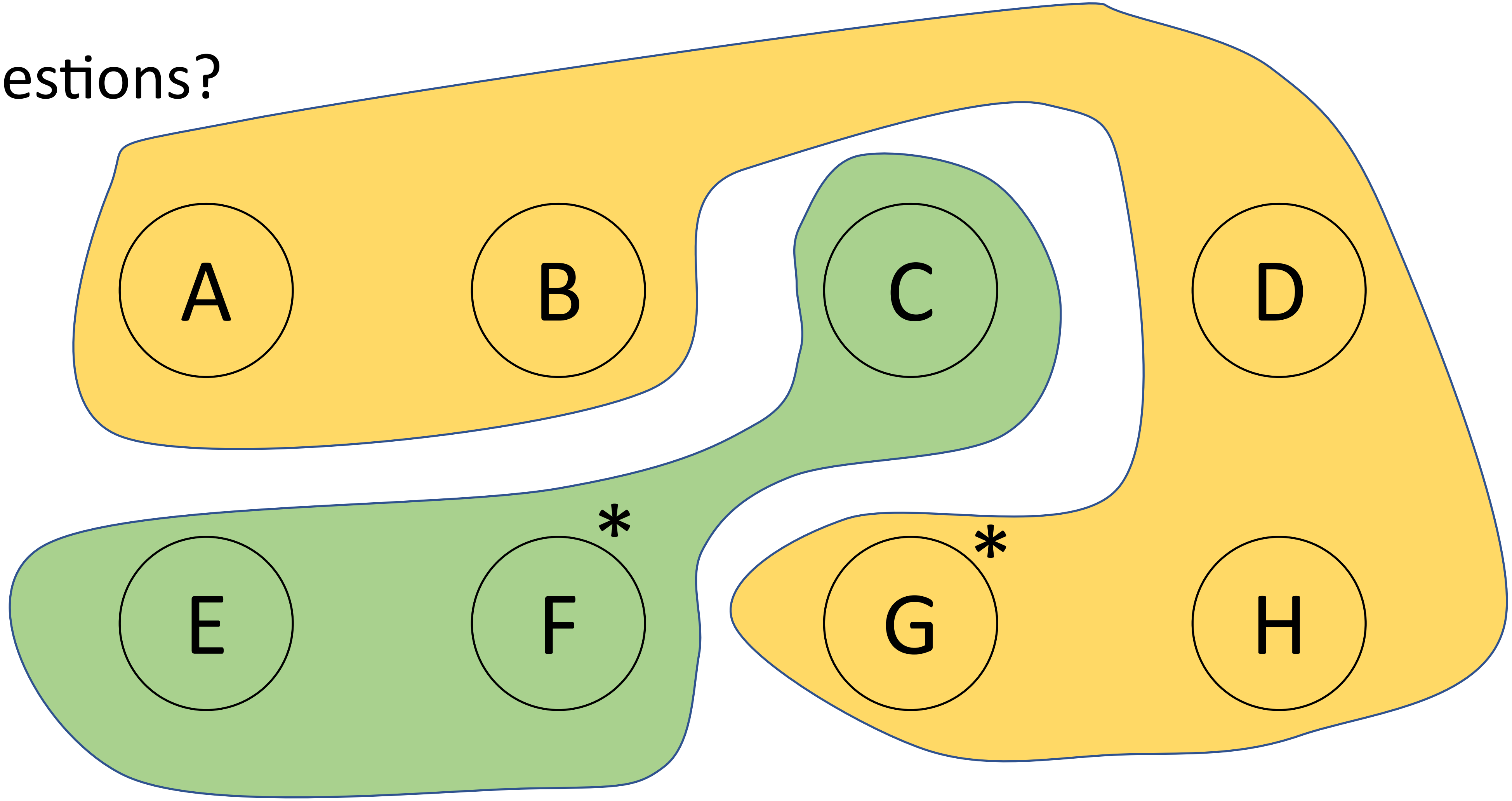


Union A H

Quiz: Color after union??



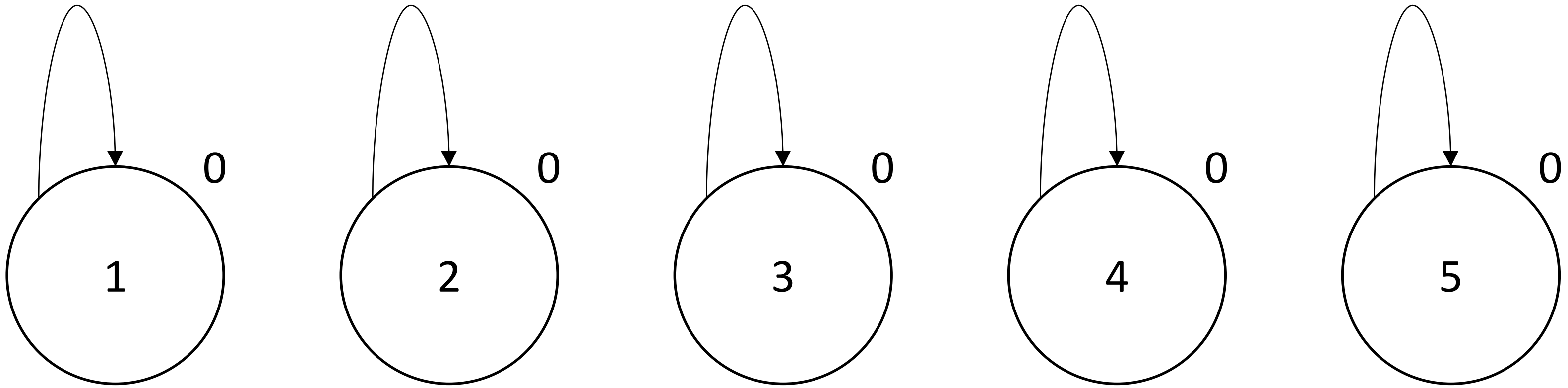
Questions?



Find: A, C, H

Make Set : Initialization Step

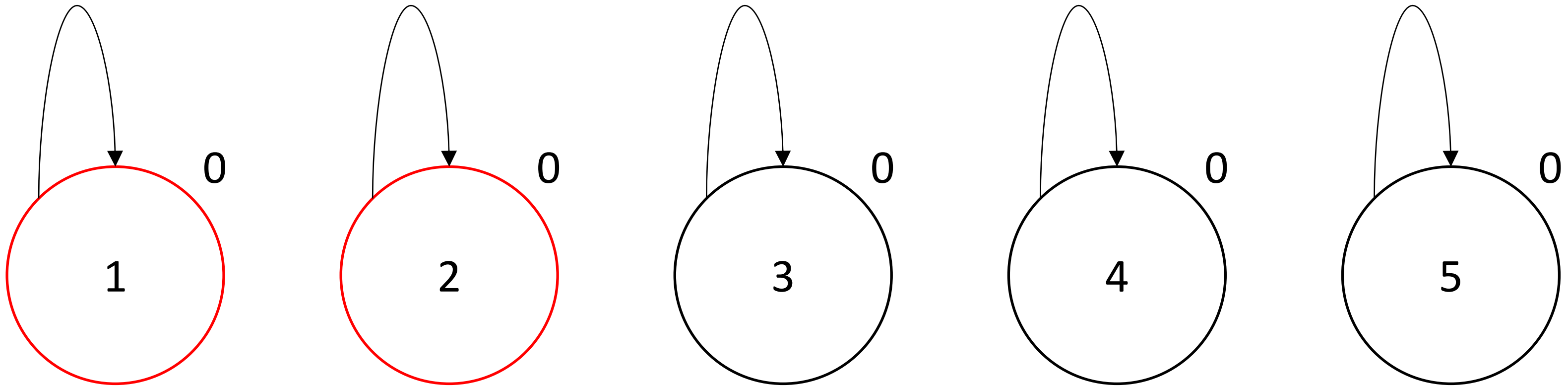
```
For all
    Set parent/representative to self
    Set rank to 0
```



Union 1 2

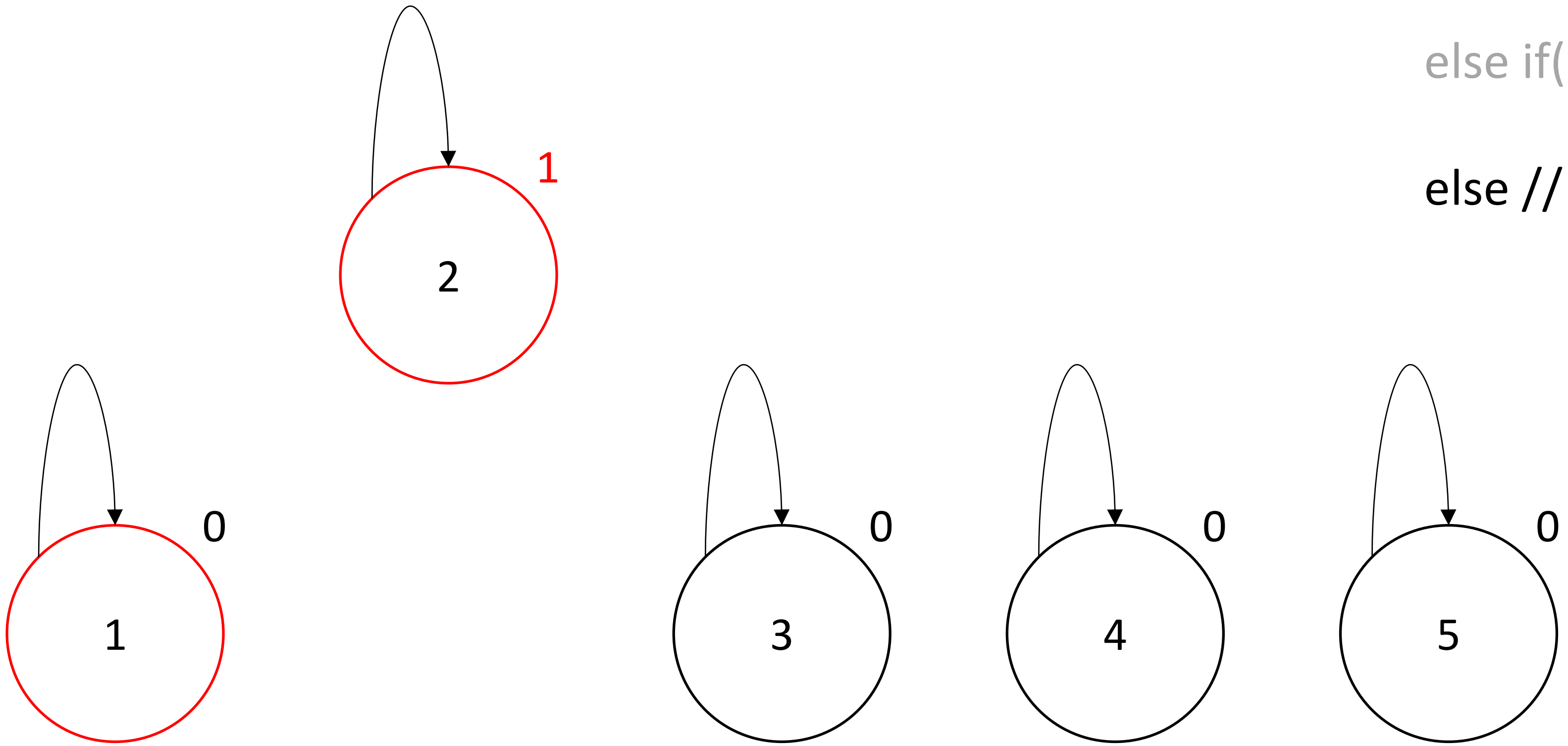
Find 1, Find 2

```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```



Union 1 2

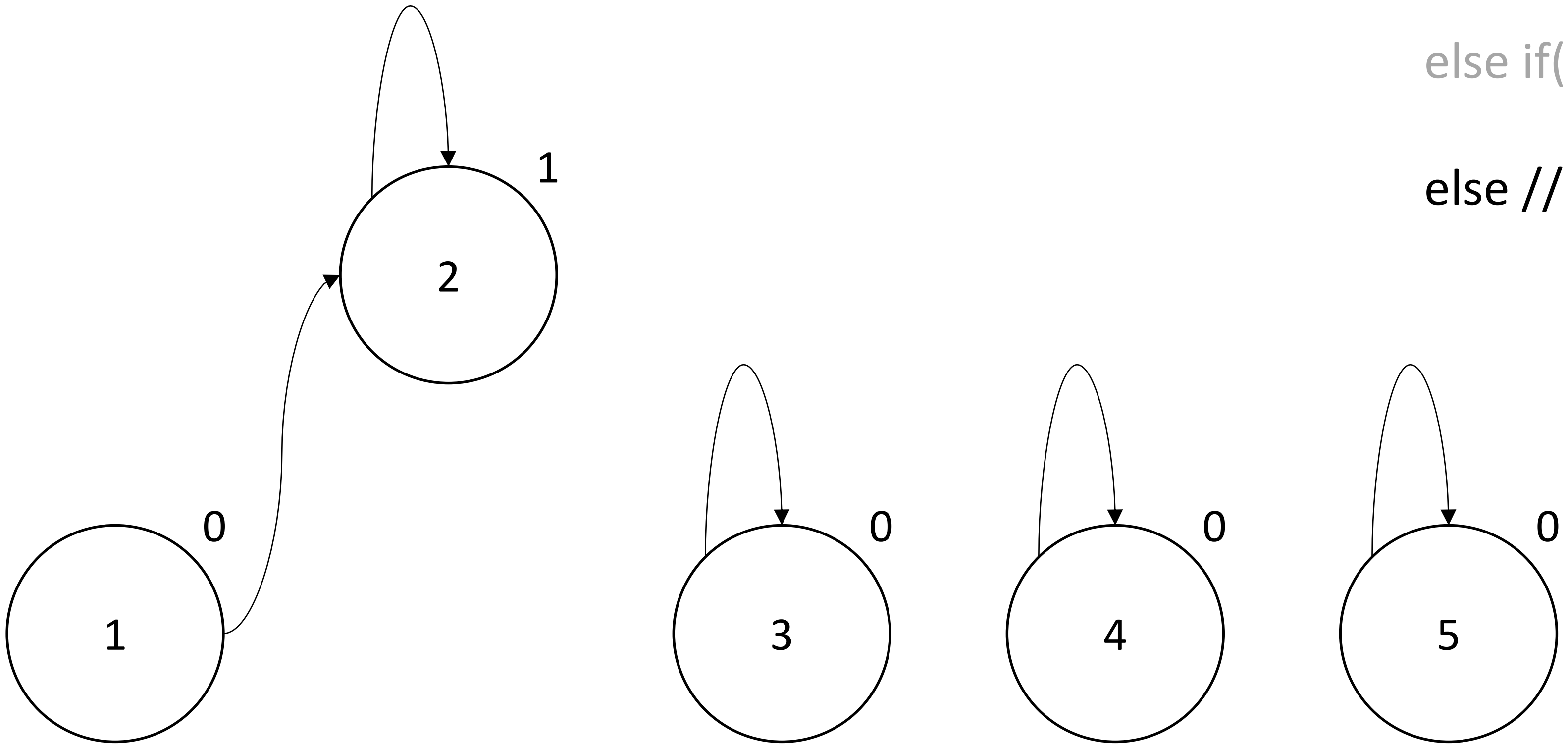
Promote if rank are same



```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Union 1 2

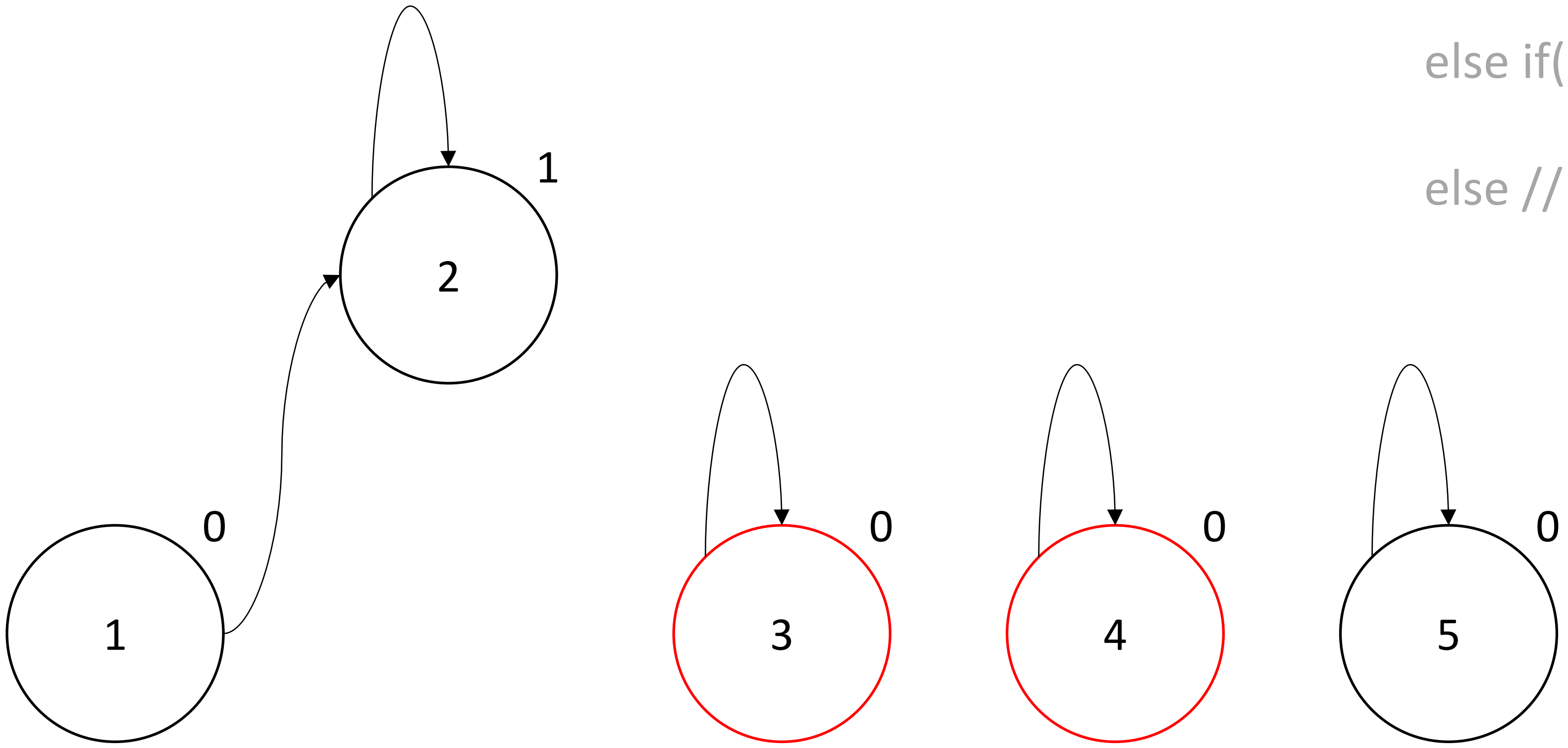
Set parent of 1 to 2



```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Union 4 3

Find 4, Find 3

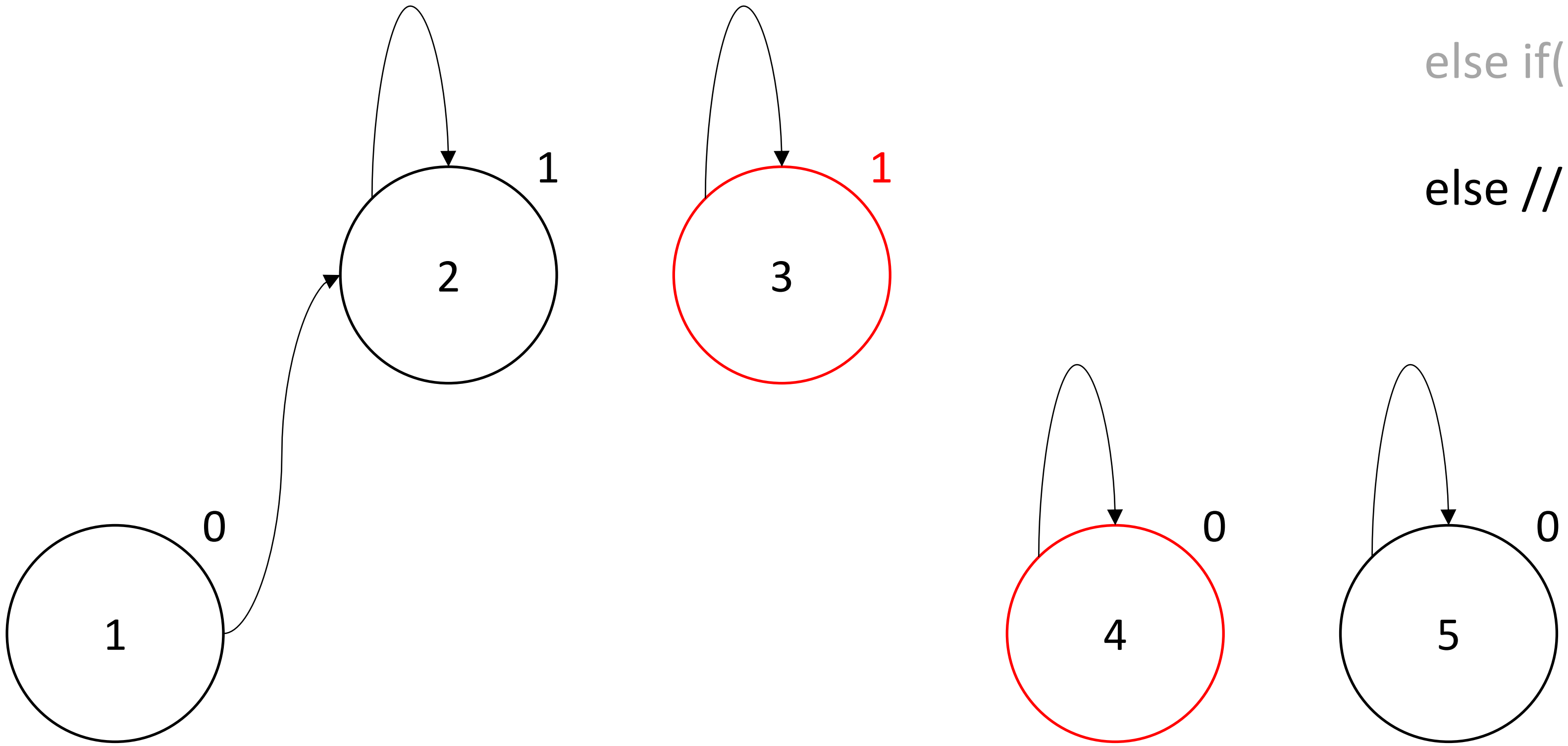


```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Union 4 3

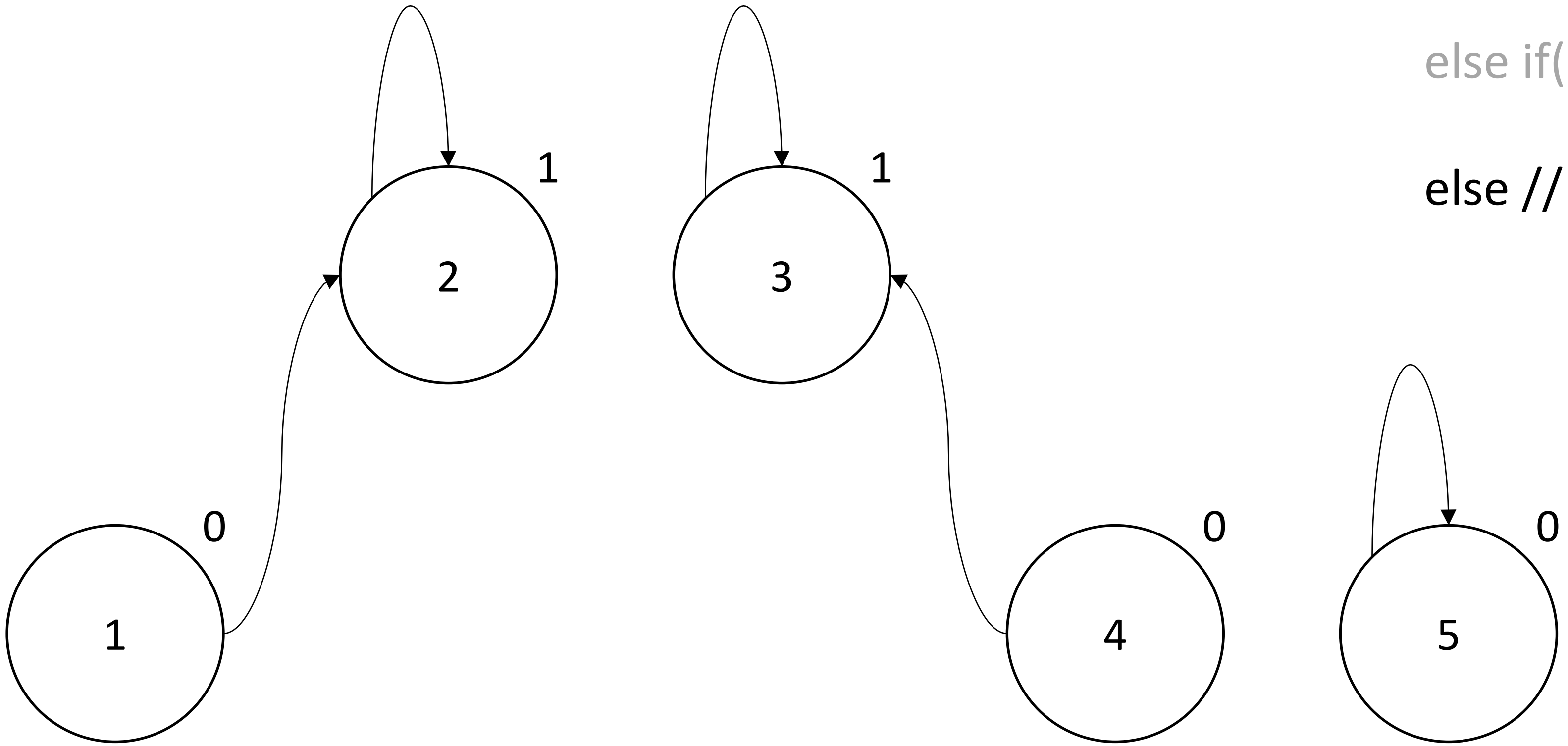
Promote if rank are same

```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```



Union 4 3

Set parent of 4 to 3

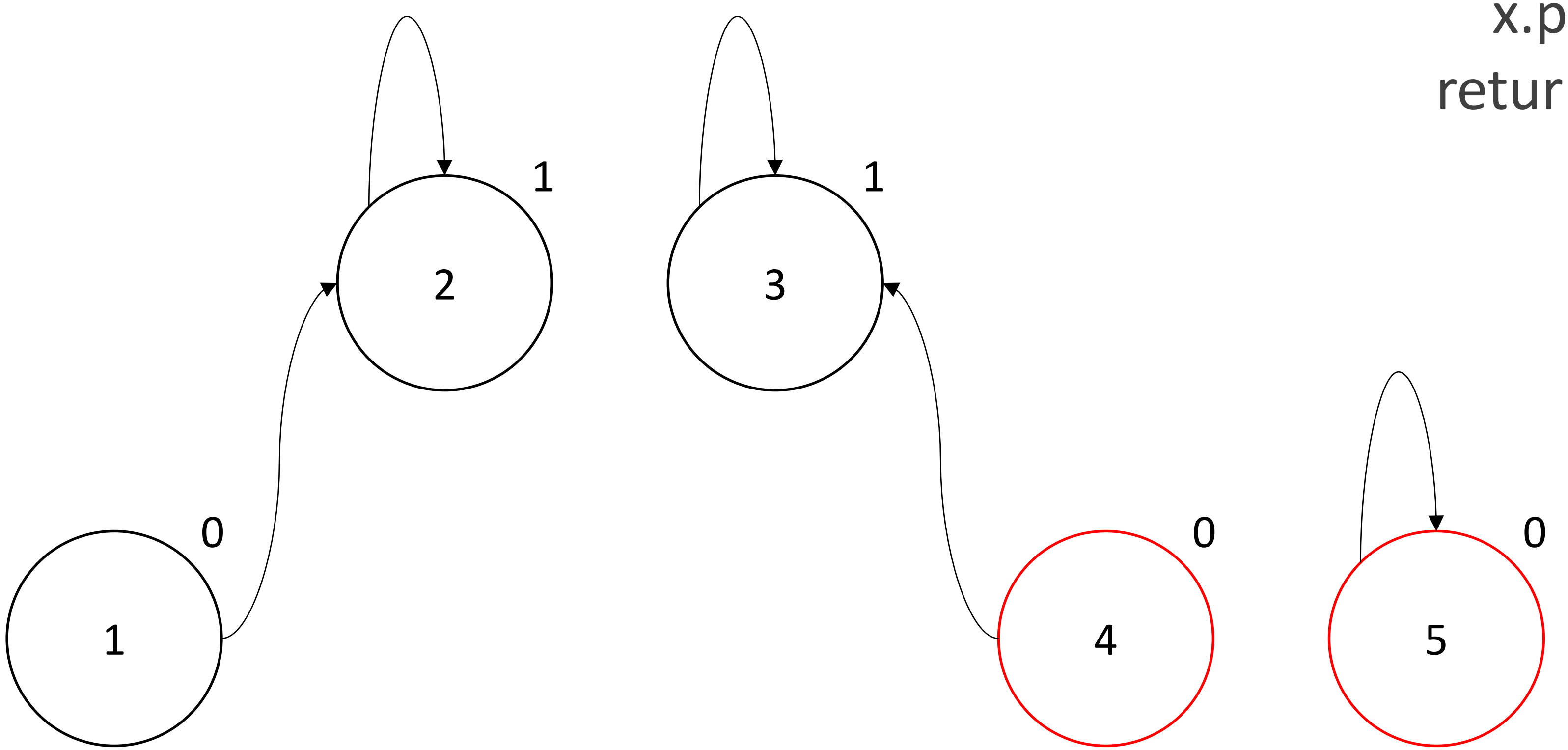


```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Union 4 5

Find 4, Find 5

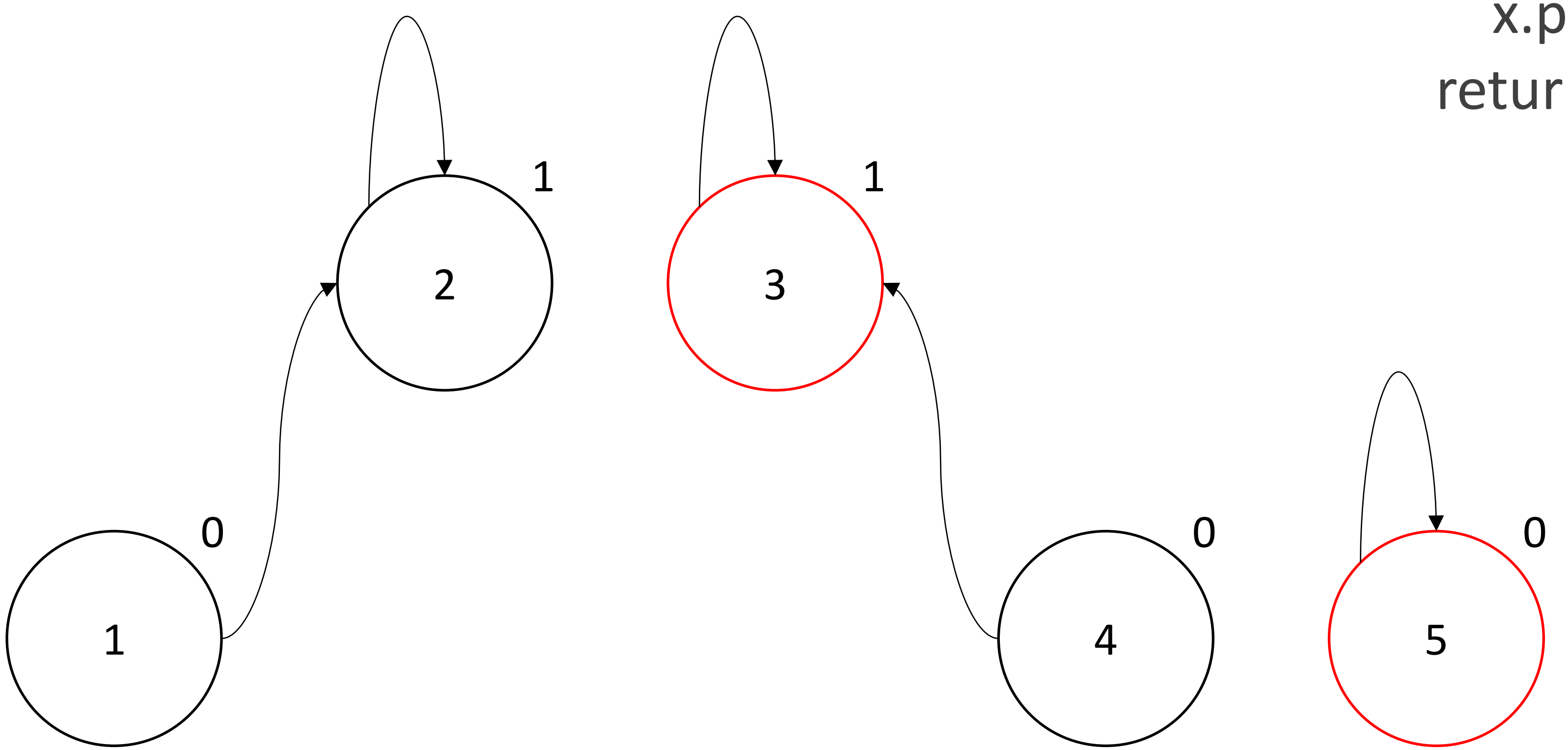
```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```



Union 4 5

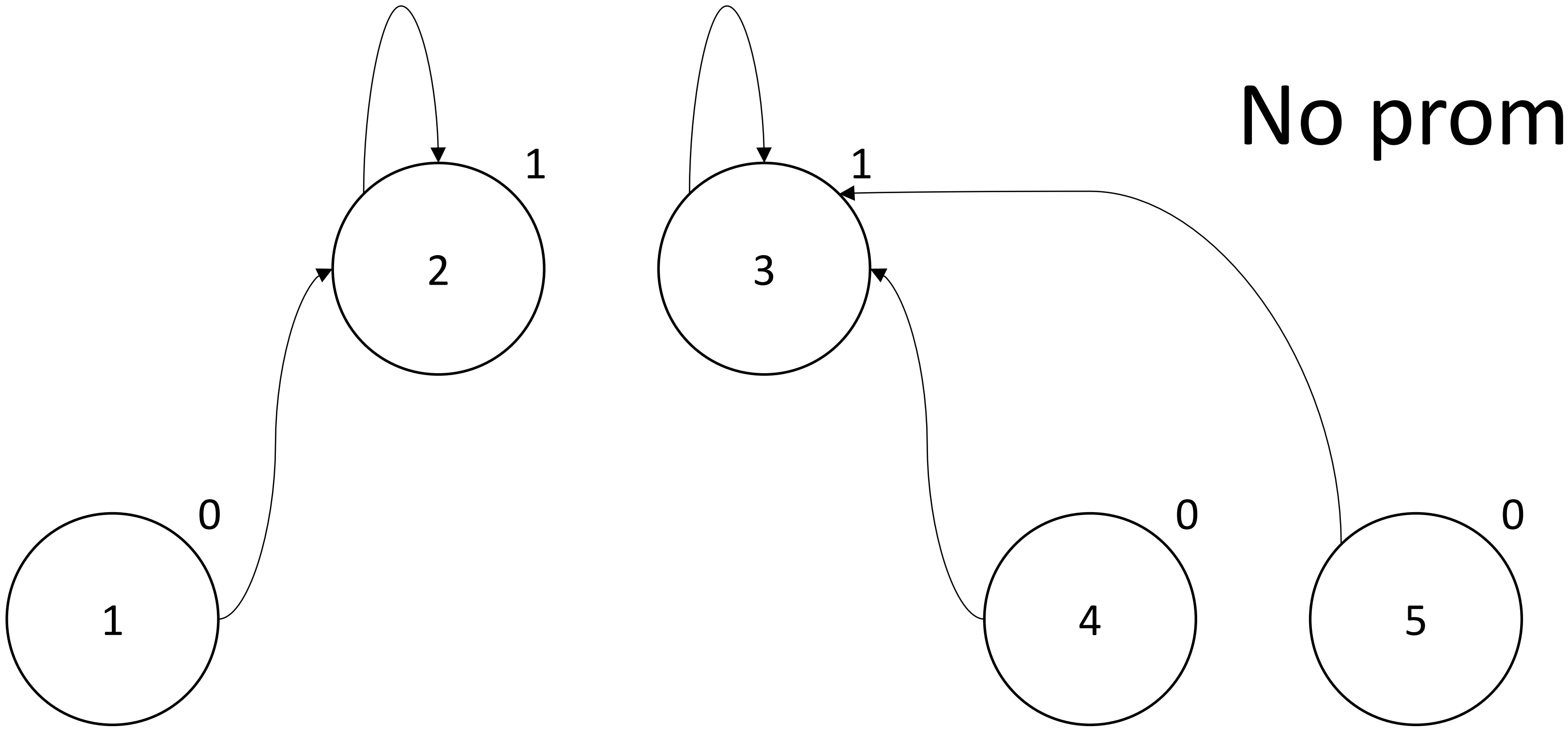
Find 4, Find 5

```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```



Union 4 5

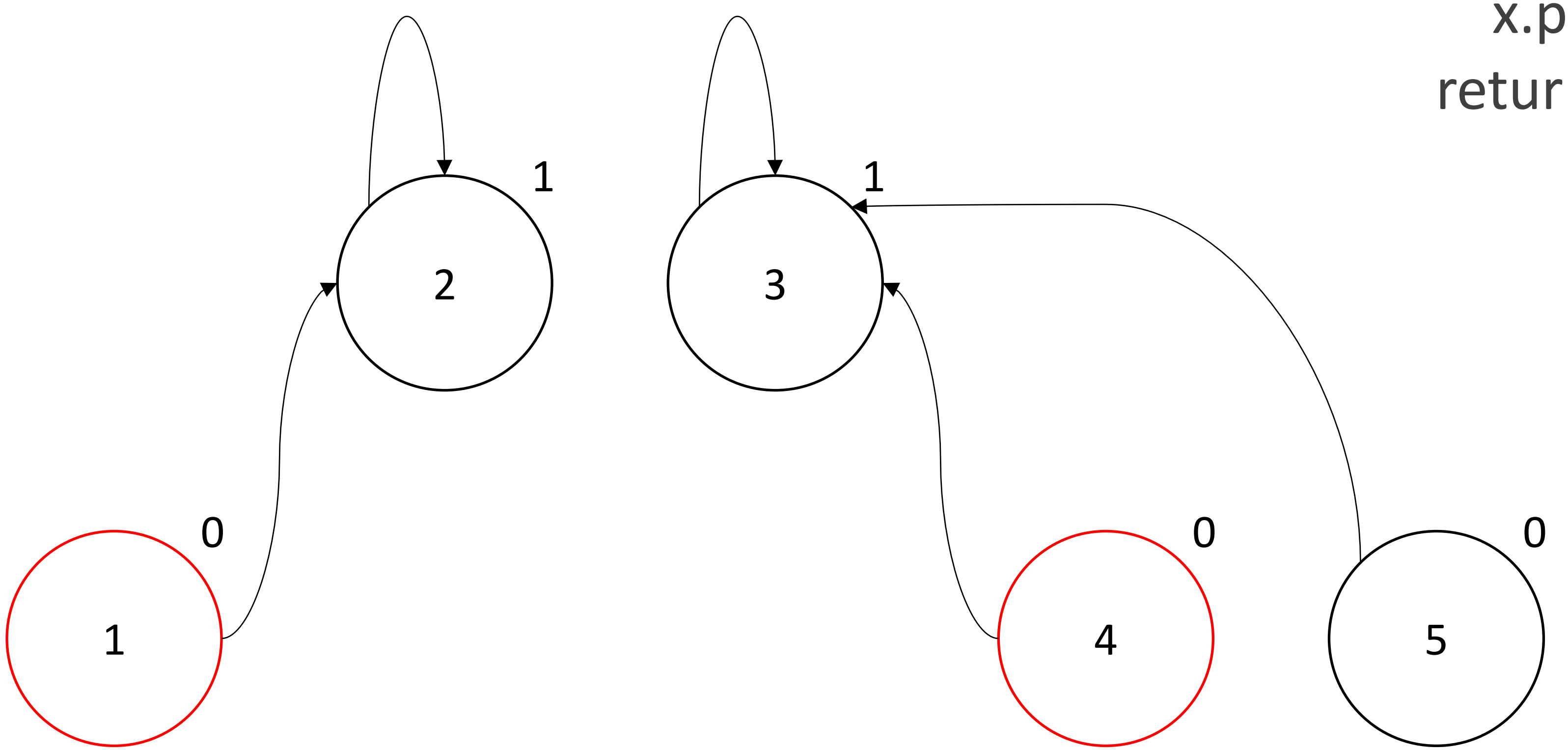
Set parent of 5 to 3



No promotion here

Union 4 1

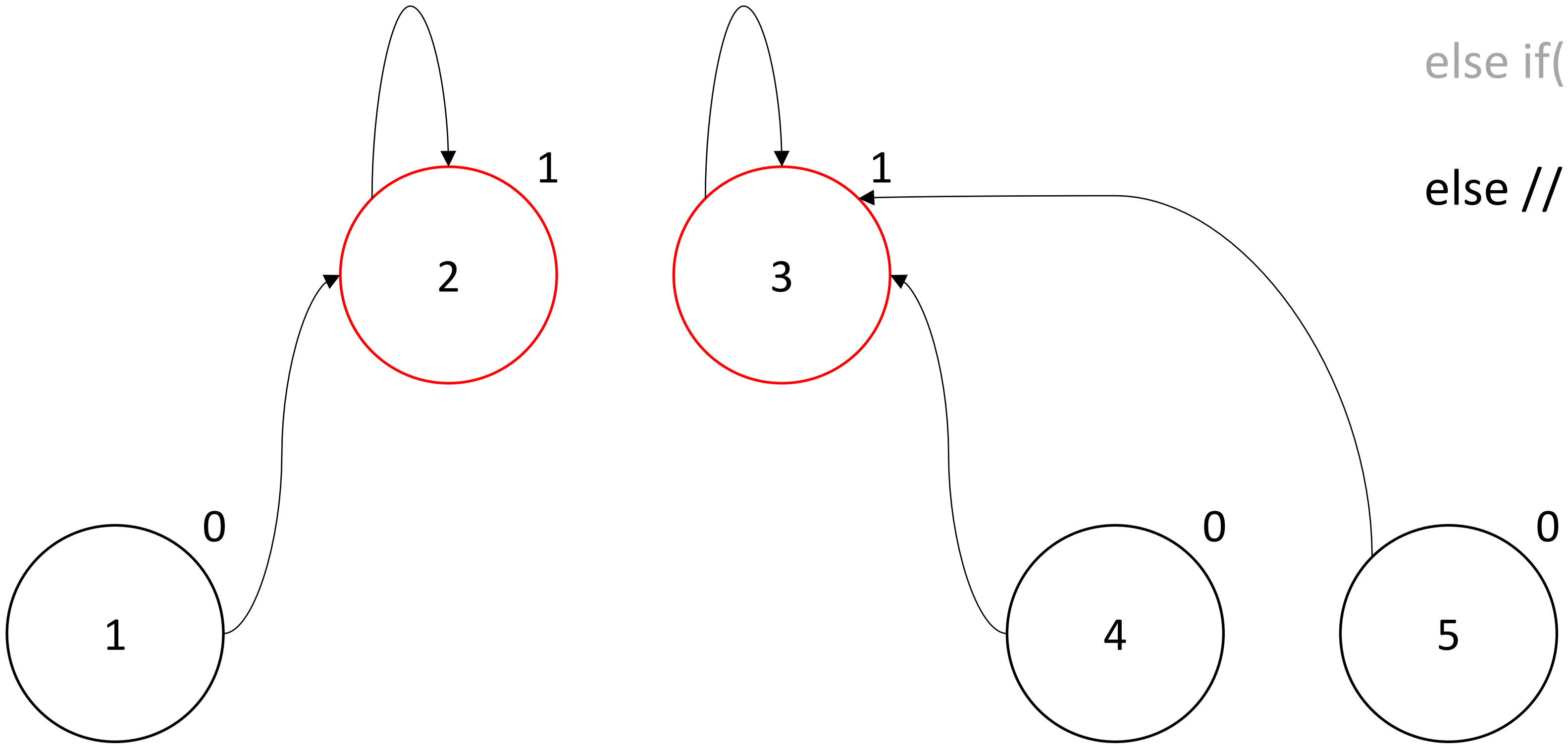
Find 4, Find 1



```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```

Union 4 1

Find 4, Find 1

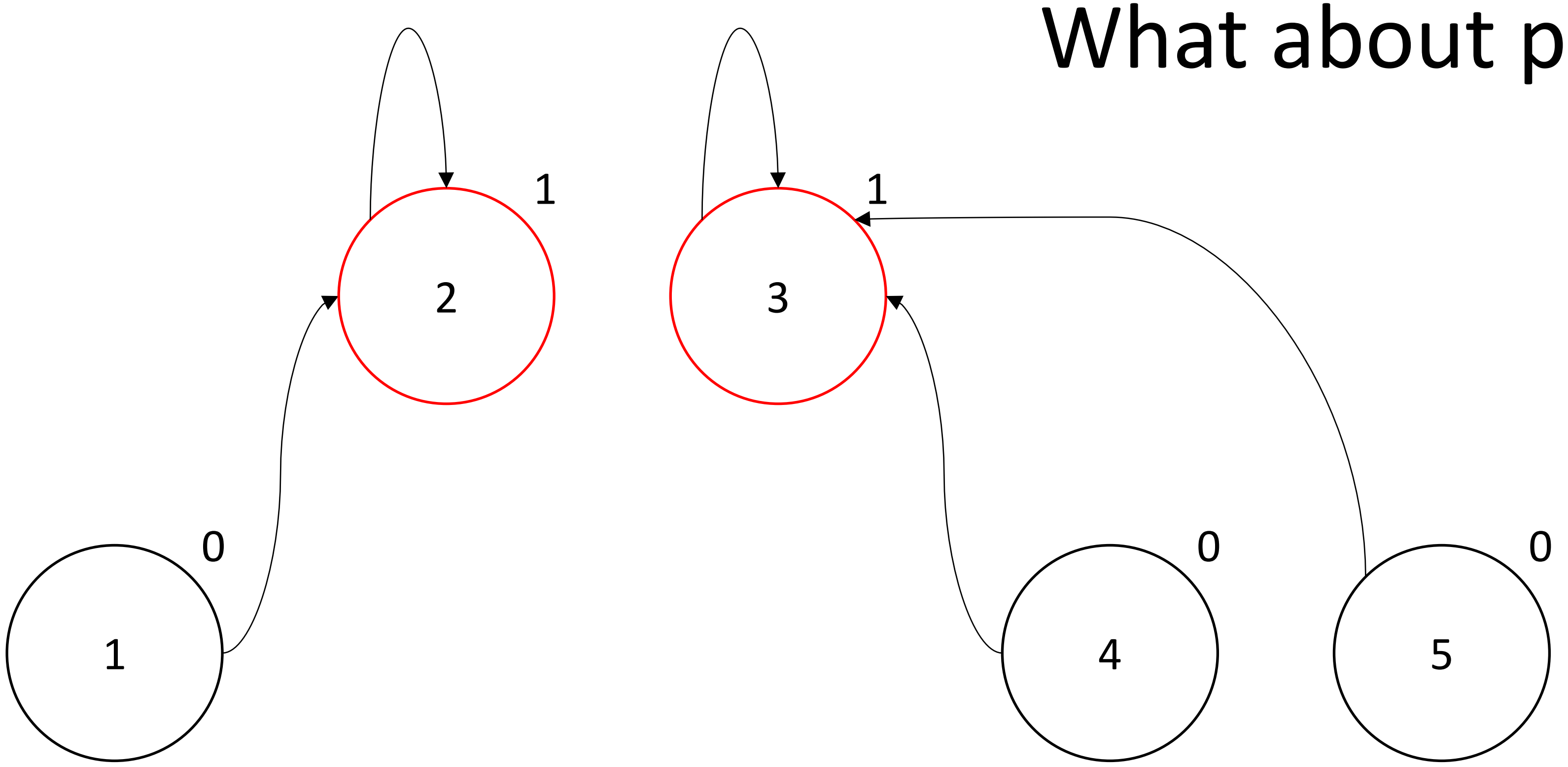


```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Union 4 1

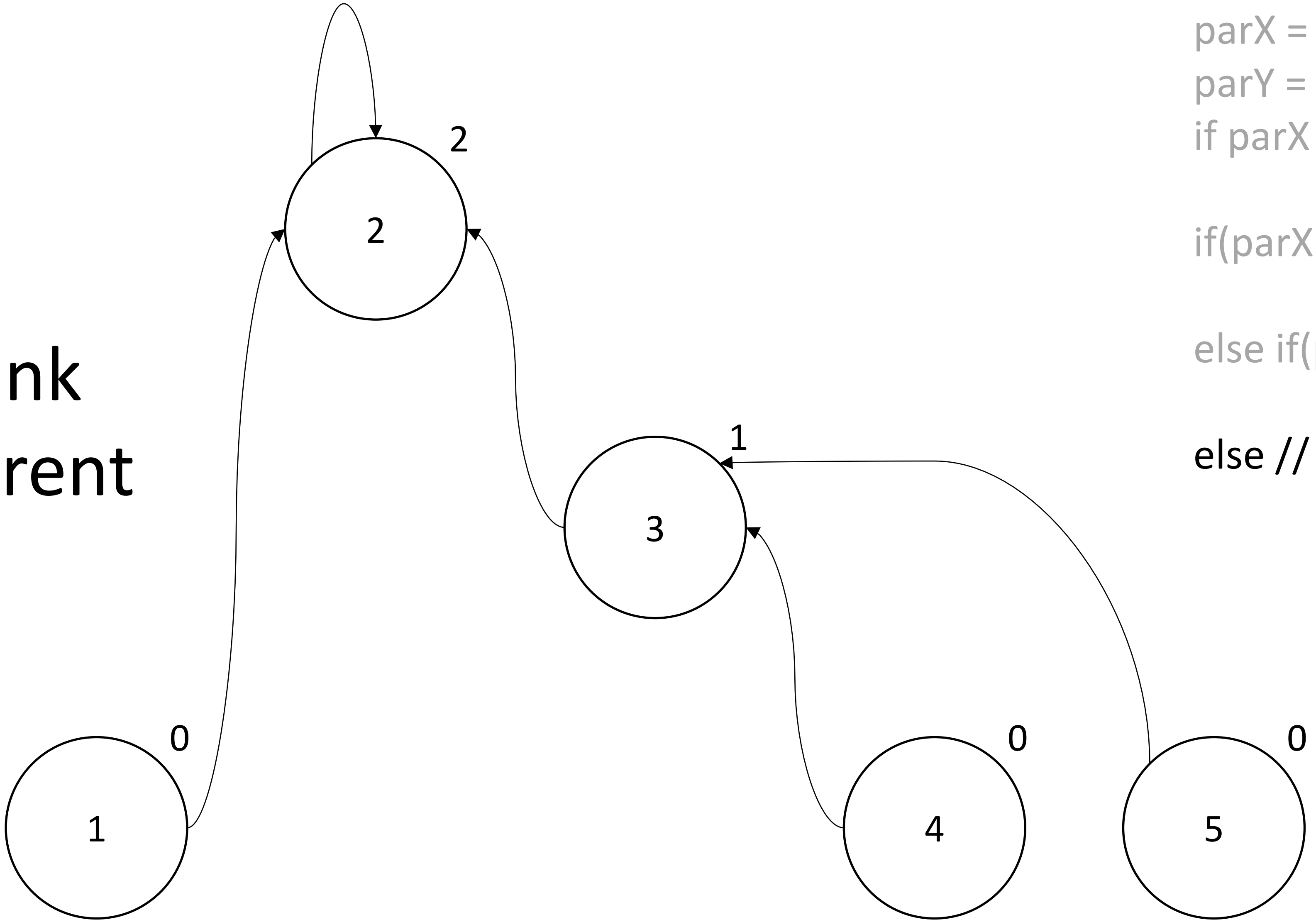
Find 4, Find 1

What about promotion?



Union 4 1

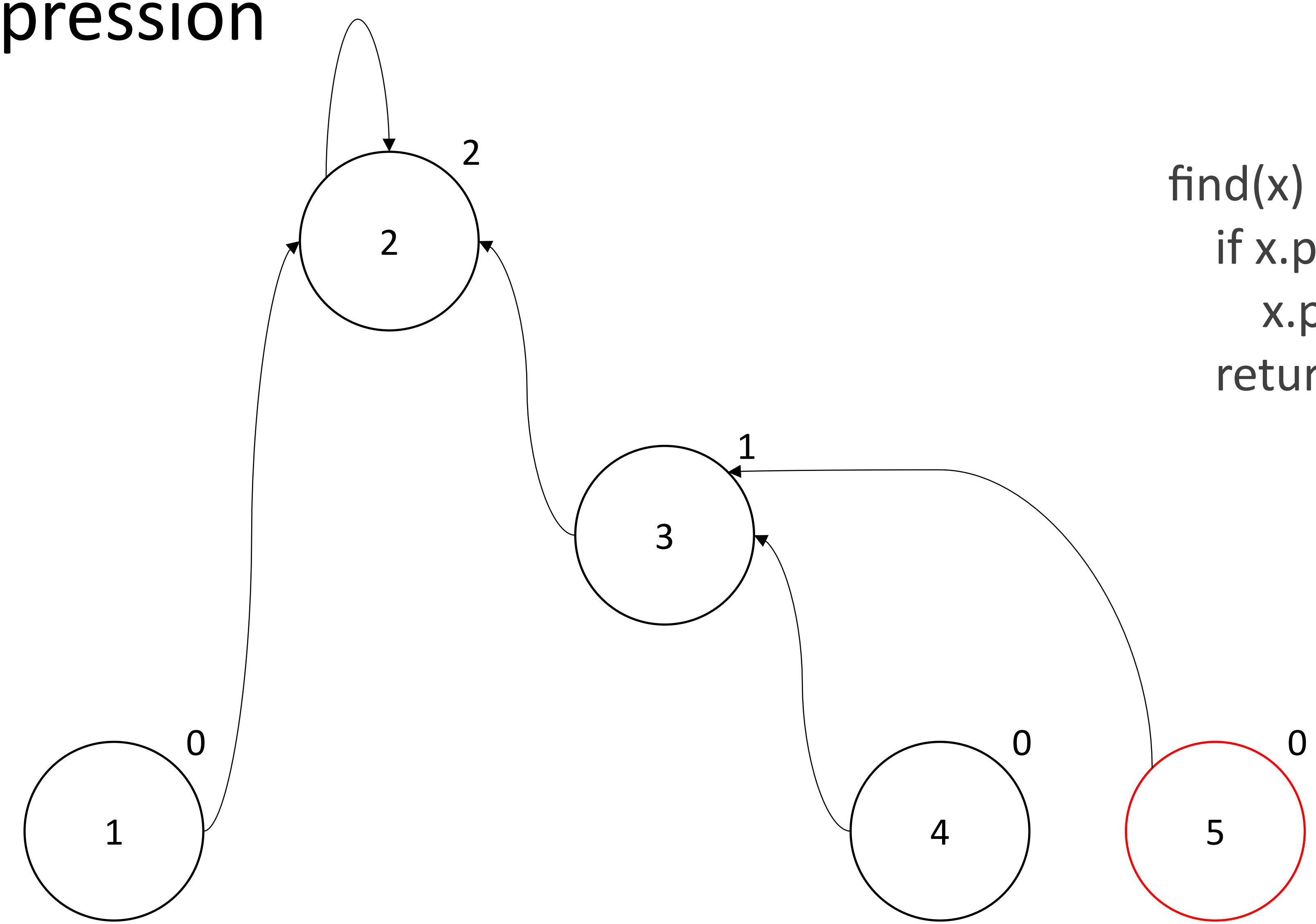
Update Rank
Update Parent



```
Union(x, y)
        parX = find(x)
        parY = find(y)
        if parX == parY
                //same set
        if(parX.rank > parY.rank)
                parY.par = parX
        else if(parY.rank > parX.rank)
                parX.par = parY
        else // both equal
                parY.rank++
                parX.par = parY
```

Path Compression

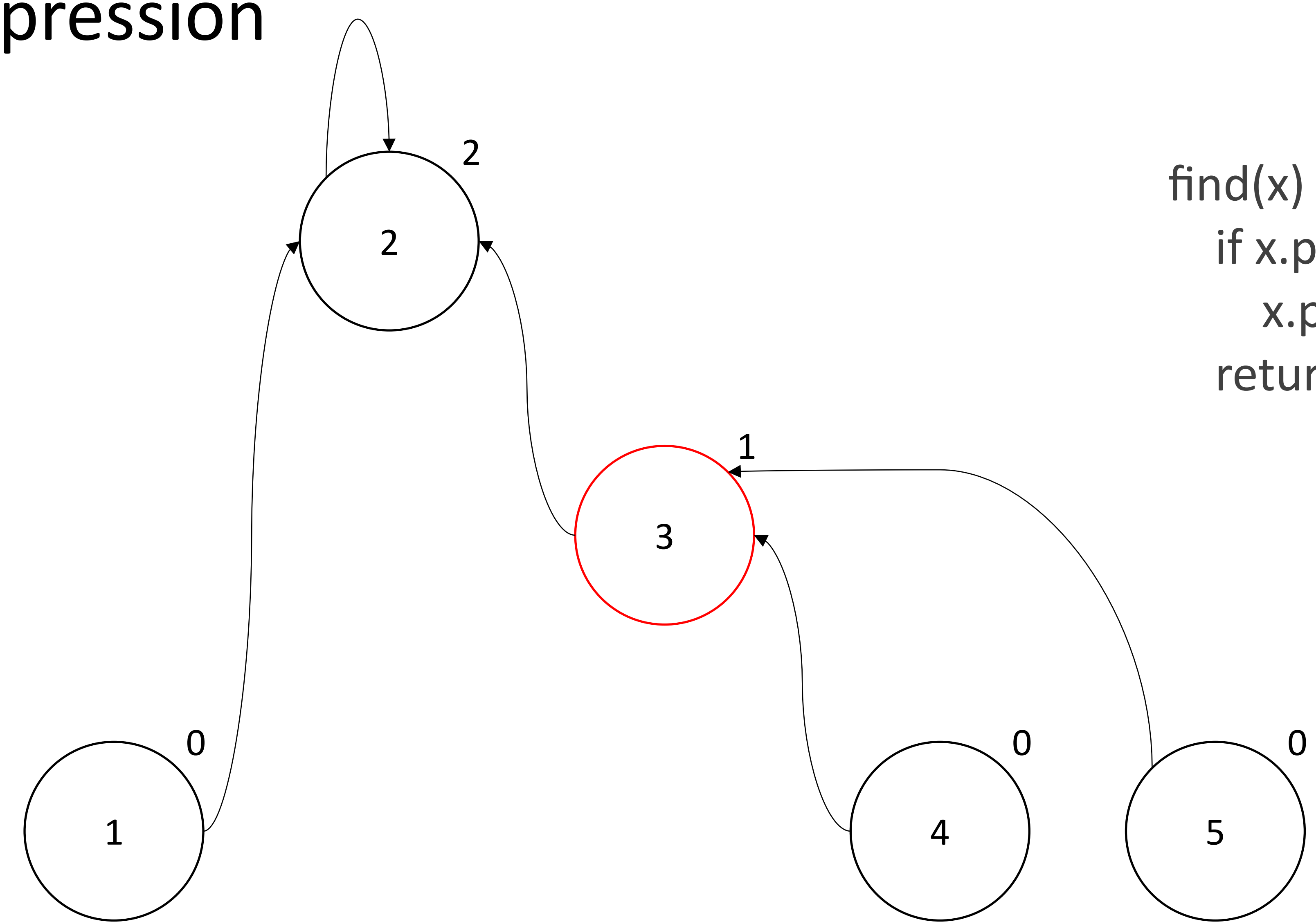
Find 5



```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```

Path Compression

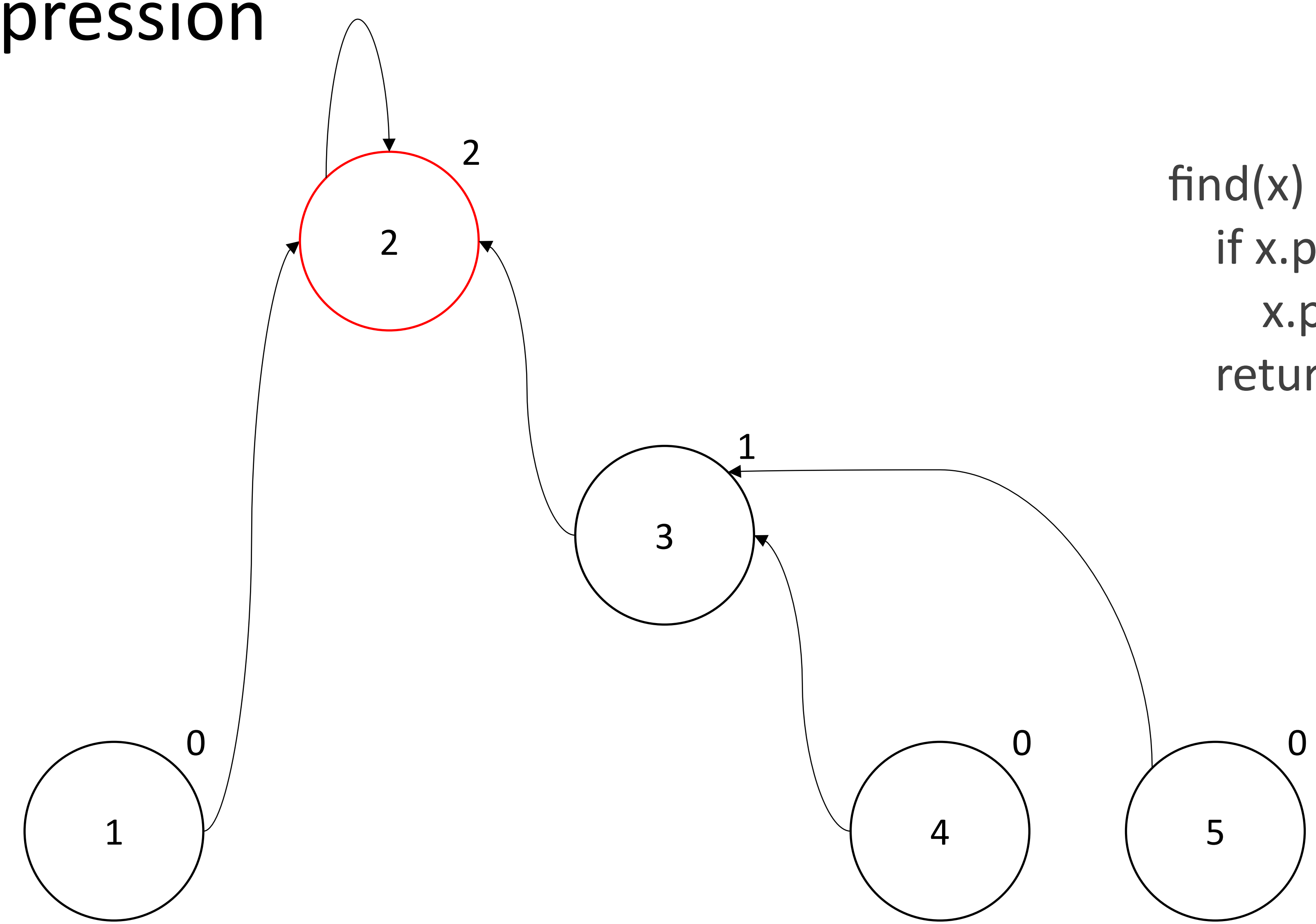
Find 5



```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```

Path Compression

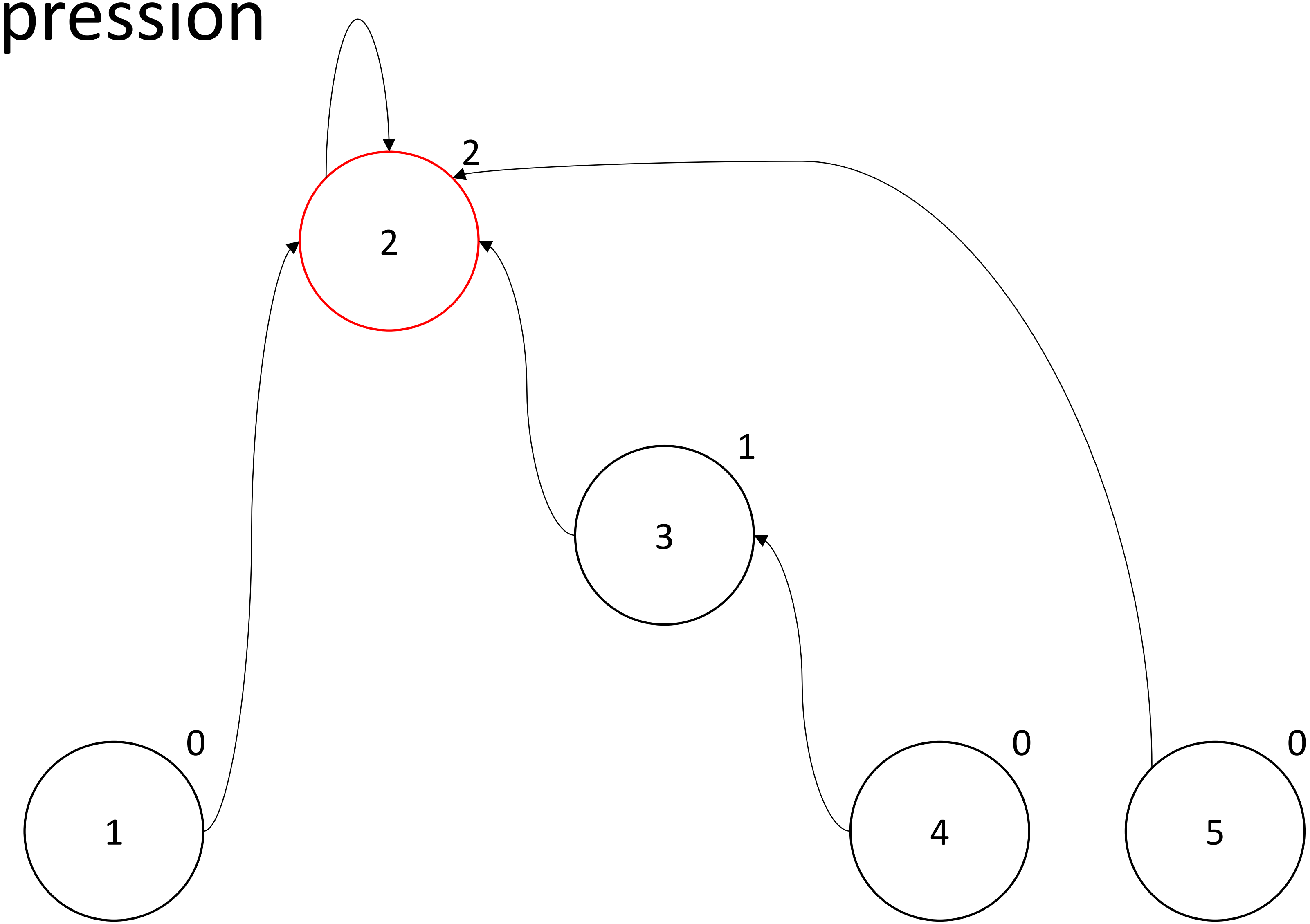
Find 5



```
find(x)
    if x.par != x
        x.par = find(x.par)
    return x.par
```

Path Compression

Find 5



Union Find Applications

Connectivity related problems

- Detect cycle in Graph
- Connected Components
- Puzzle

Minimum Spanning Tree

- Kruskal’s Algorithm

Union Find

{A}U{B}

Questions?