NCKU Programming Contest Training Course
Disjoint set
2018/02/23

Chun-Chi, Fang

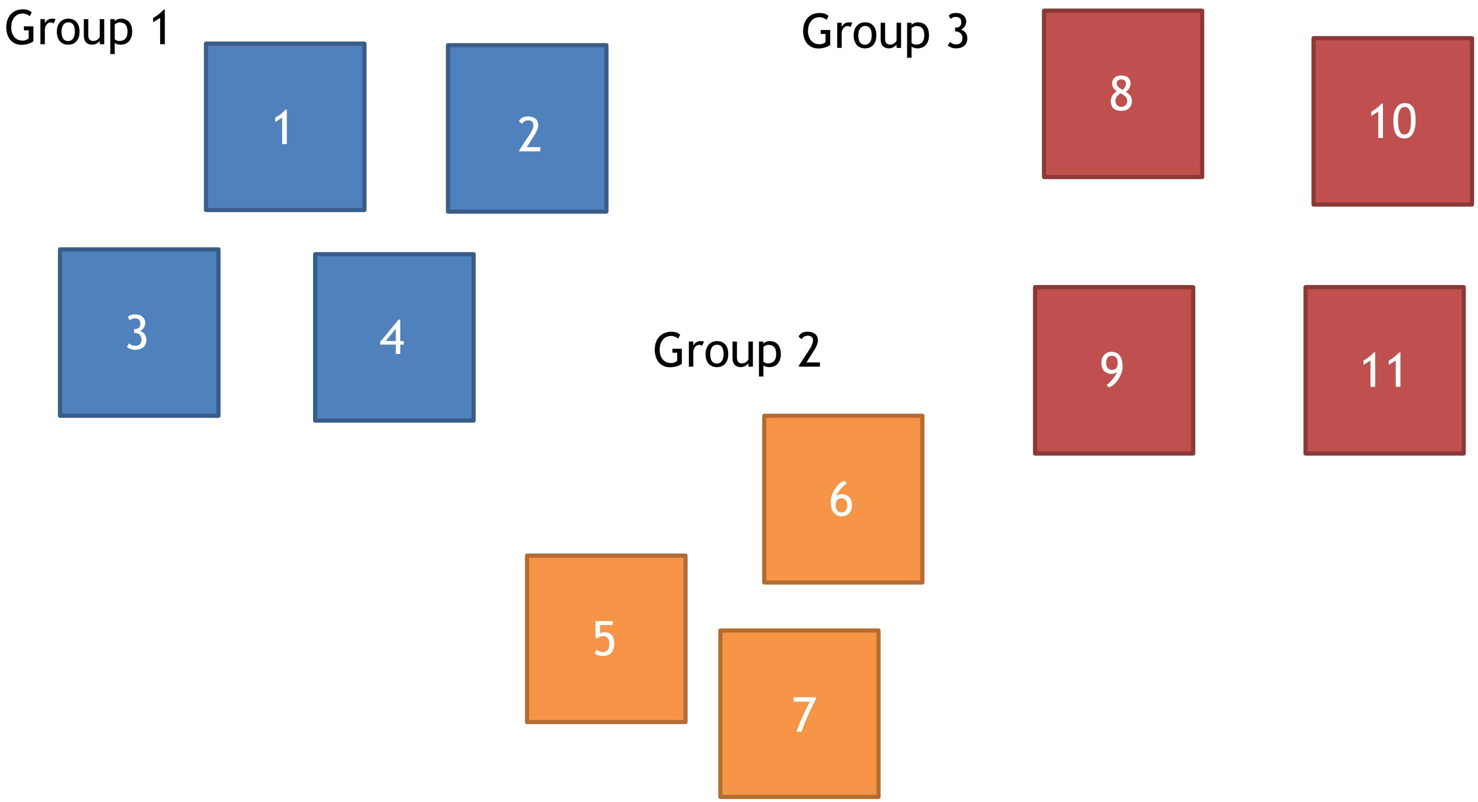
khtp91113@gmail.com

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by electron & free999

Disjoint Set

After classifying elements, we have several disjoint sets.



Group 1

1

2

3

4

Group 2

6

5

7

Group 3

8

10

9

11

Disjoint Set

We want to know which group elements belonged to

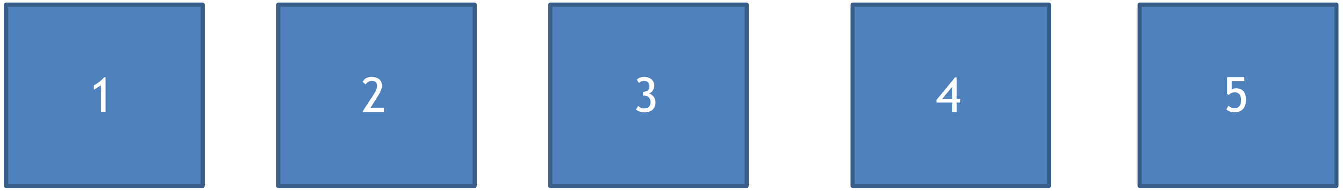
Element 1 in group 1
Element 2 in group 1
Element 5 in group 2
Element 10 in group 3
…

Disjoint Set

- Main Operation
  - Union
  - Find

Disjoint Set

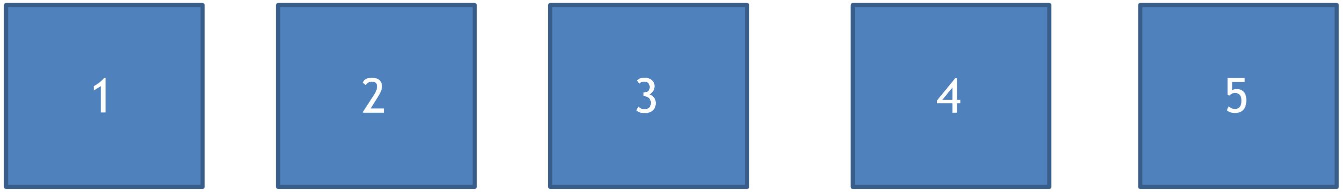
- Initial State



1 2 3 4 5

Disjoint Set

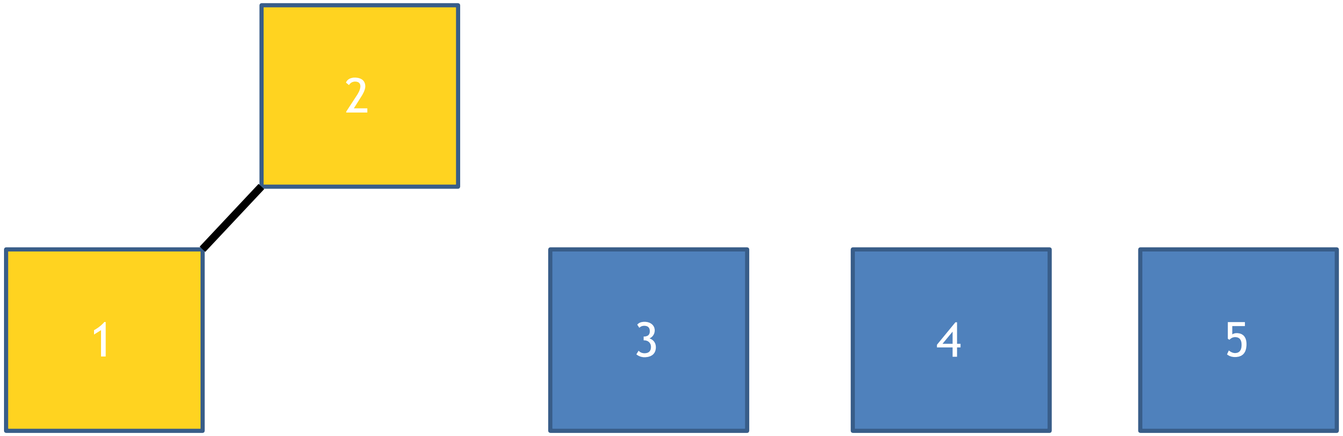
- Find(1) : return 1



1 2 3 4 5

Disjoint Set

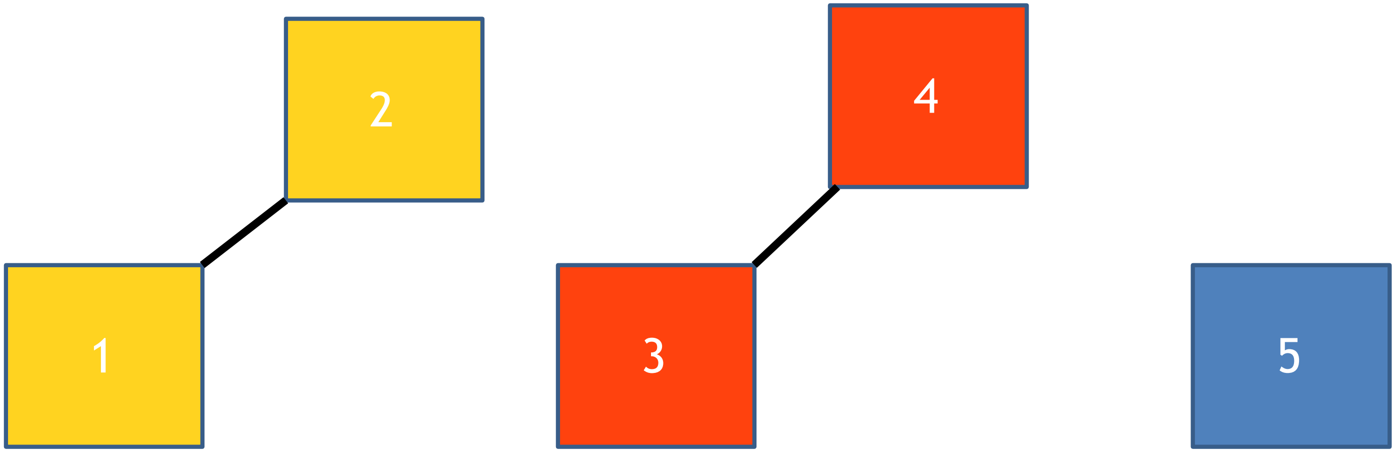
- Union(1, 2)



made by electron & free999

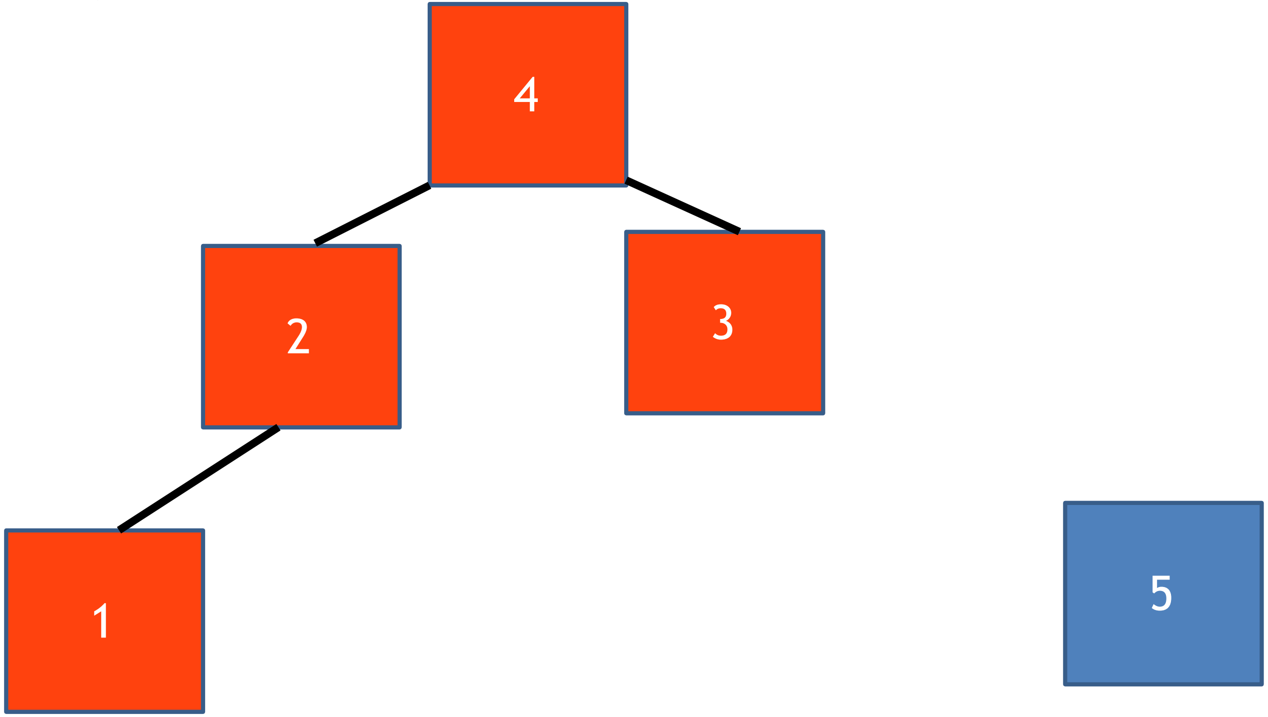
Disjoint Set

- Union(3, 4)



Disjoint Set

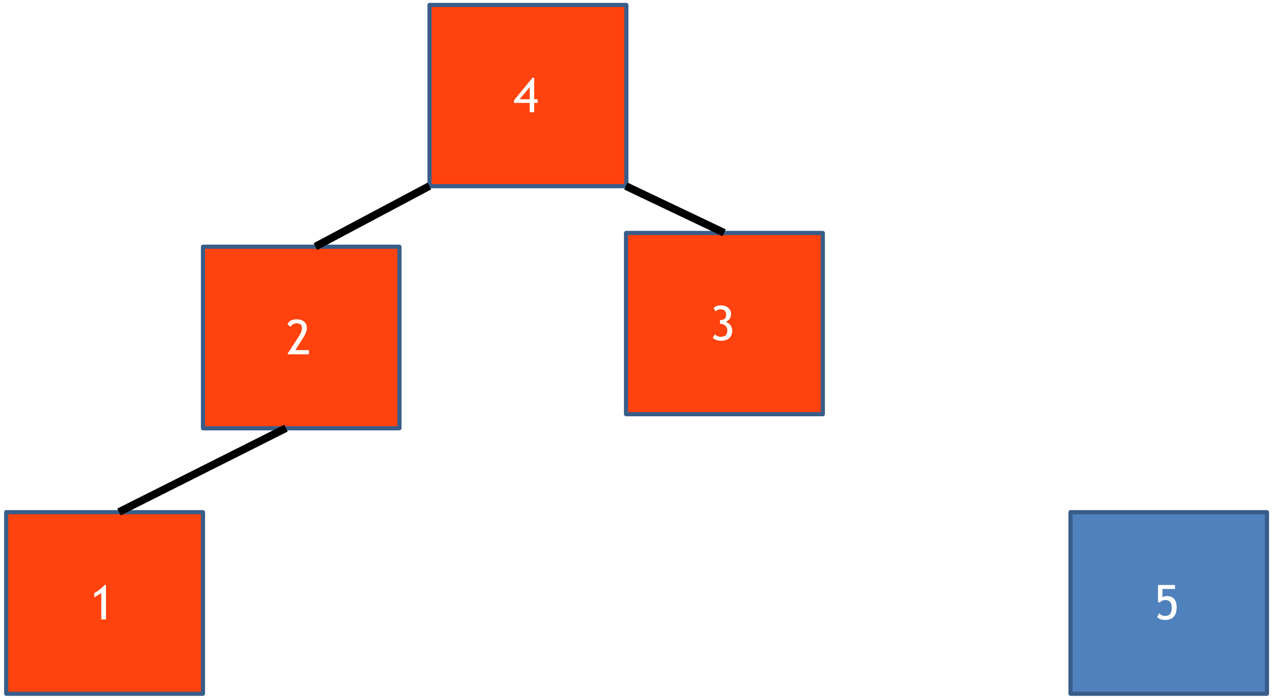
- Union(1, 3)



made by electron & free999

Disjoint Set

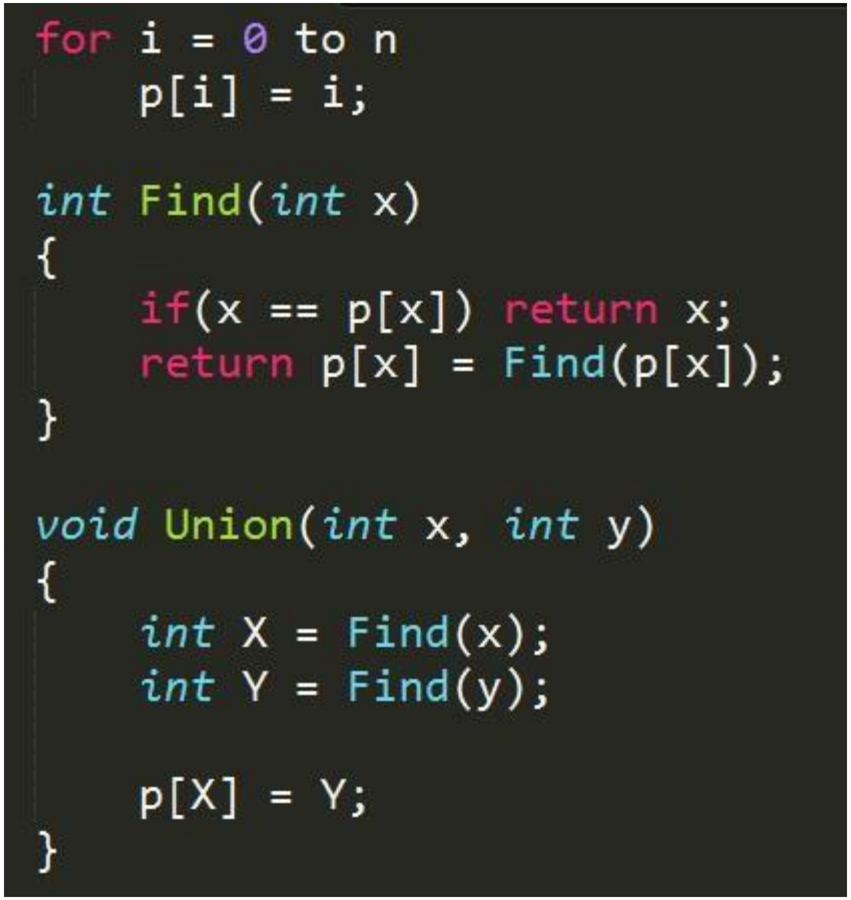
- Find(1) => 4
- Find(5) => 5



made by electron & free999

Disjoint Set

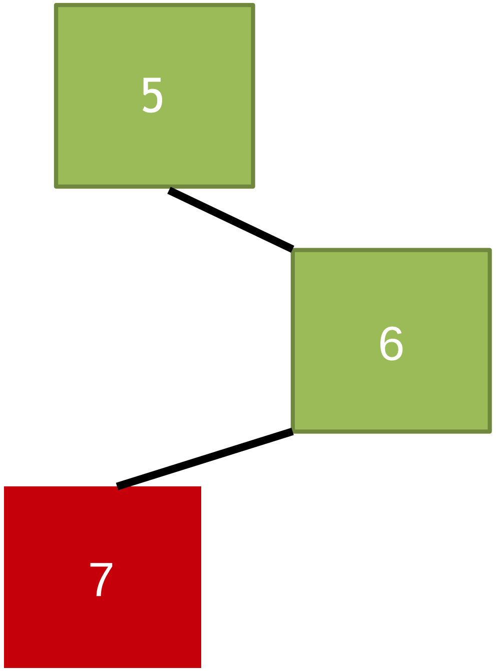
- Find
  - –Find the root of each group
- Union
  - –Combine two group



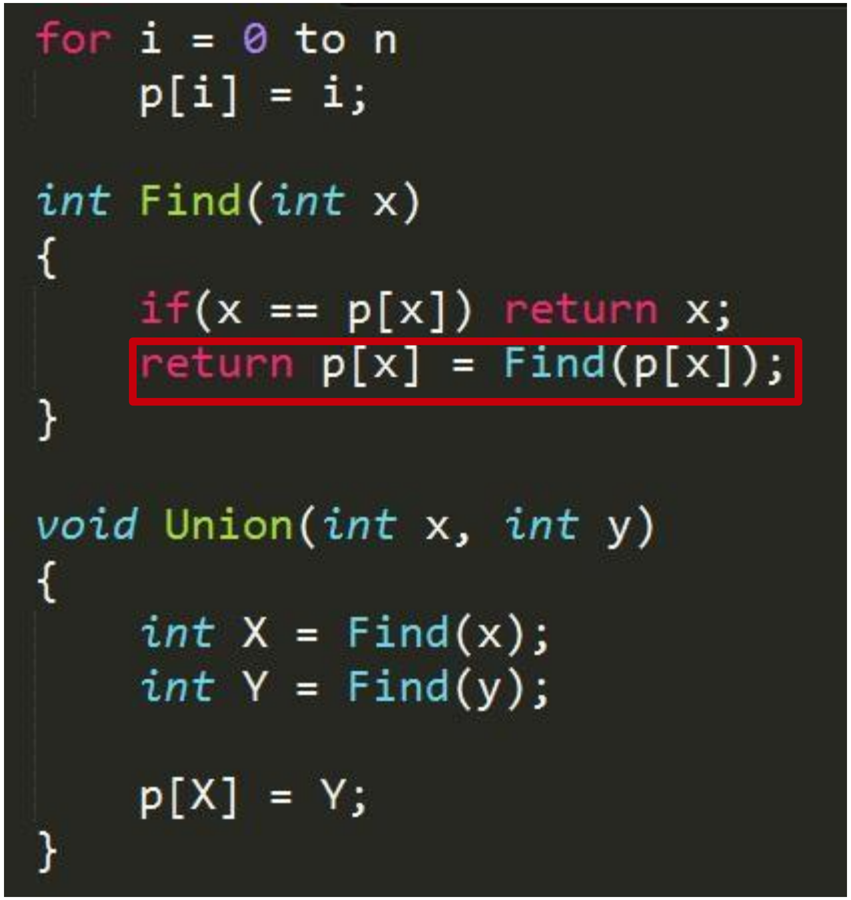
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

Disjoint Set

- Find(7) :



X = 7

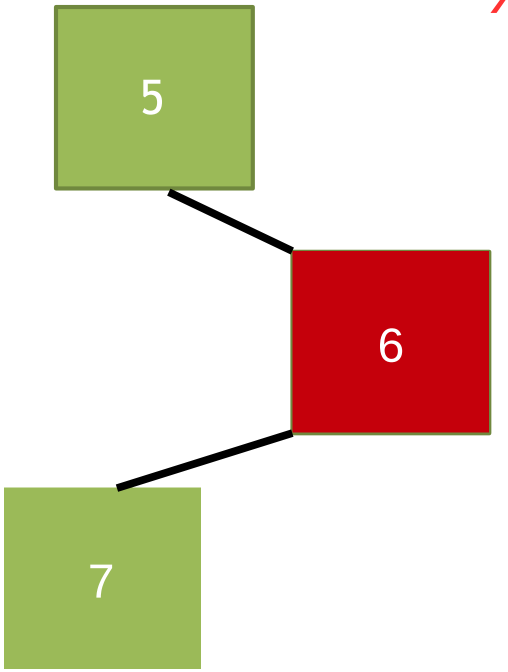


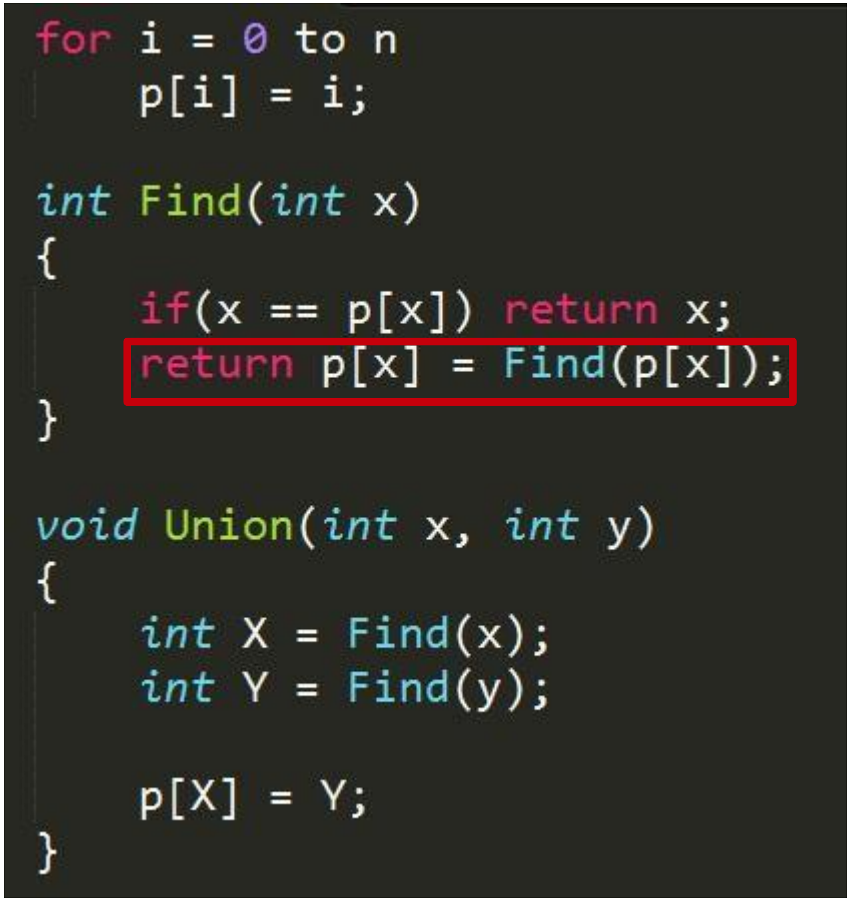
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

Disjoint Set

- Find(7) :

X = 6

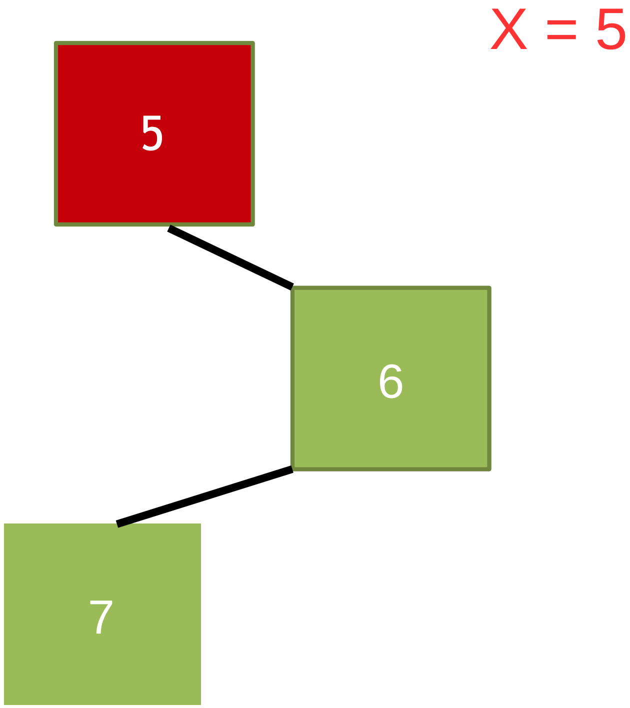


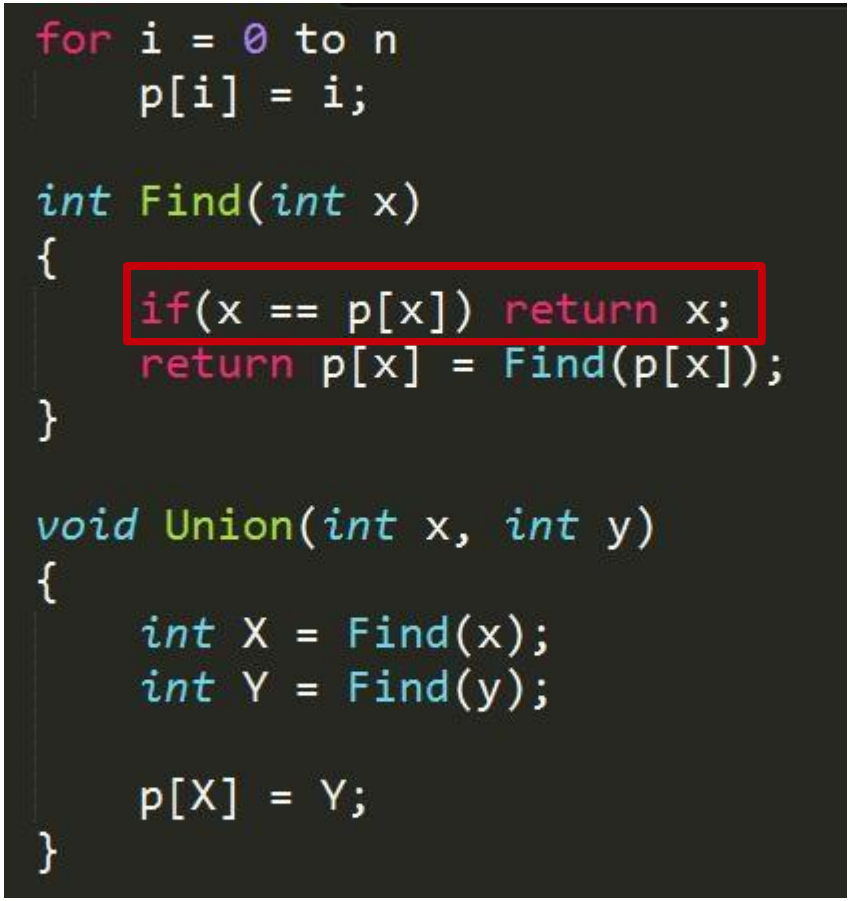


```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

Disjoint Set

- Find(7) : return 5

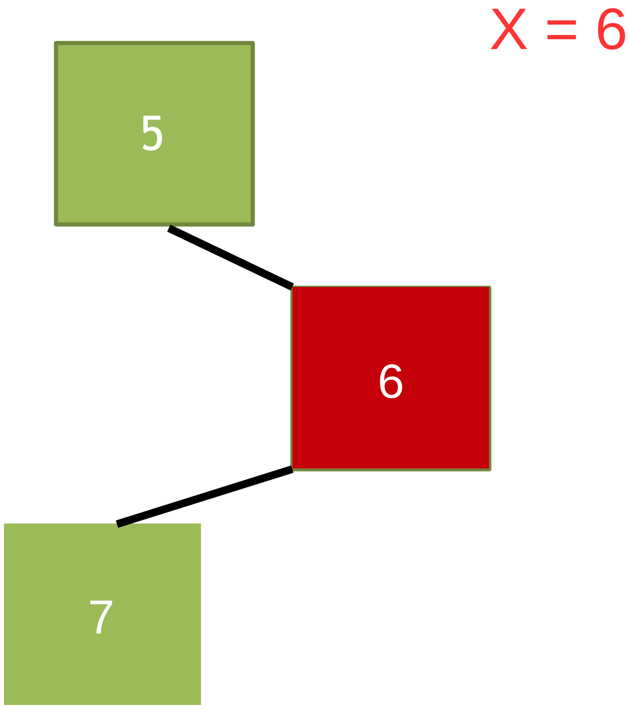


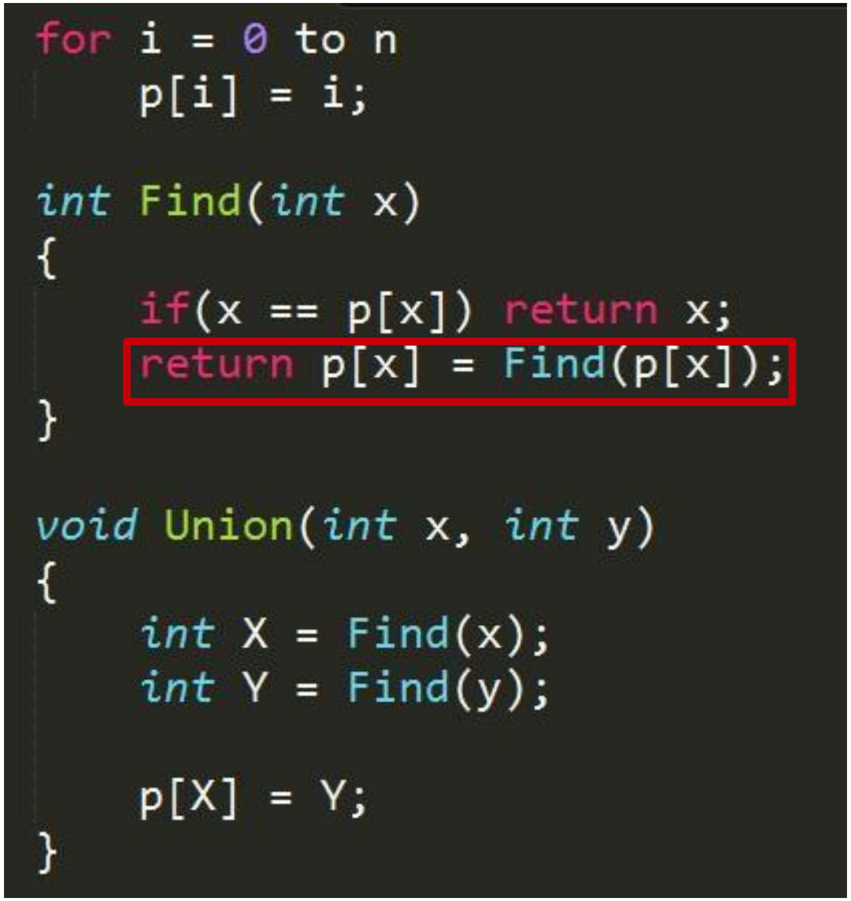


```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

Disjoint Set

- Find(7) : return 5



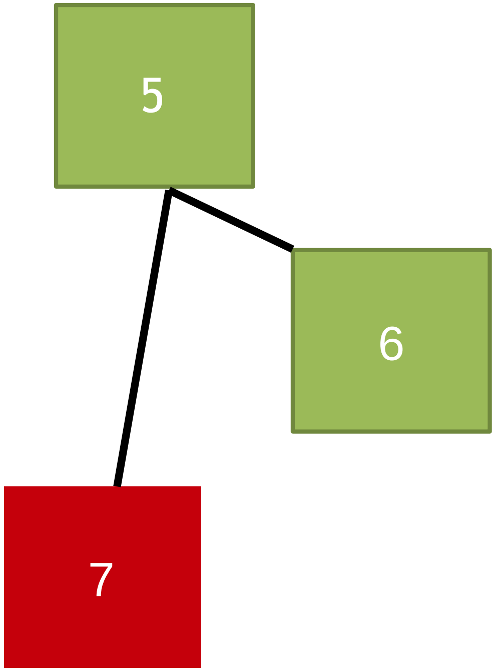


```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

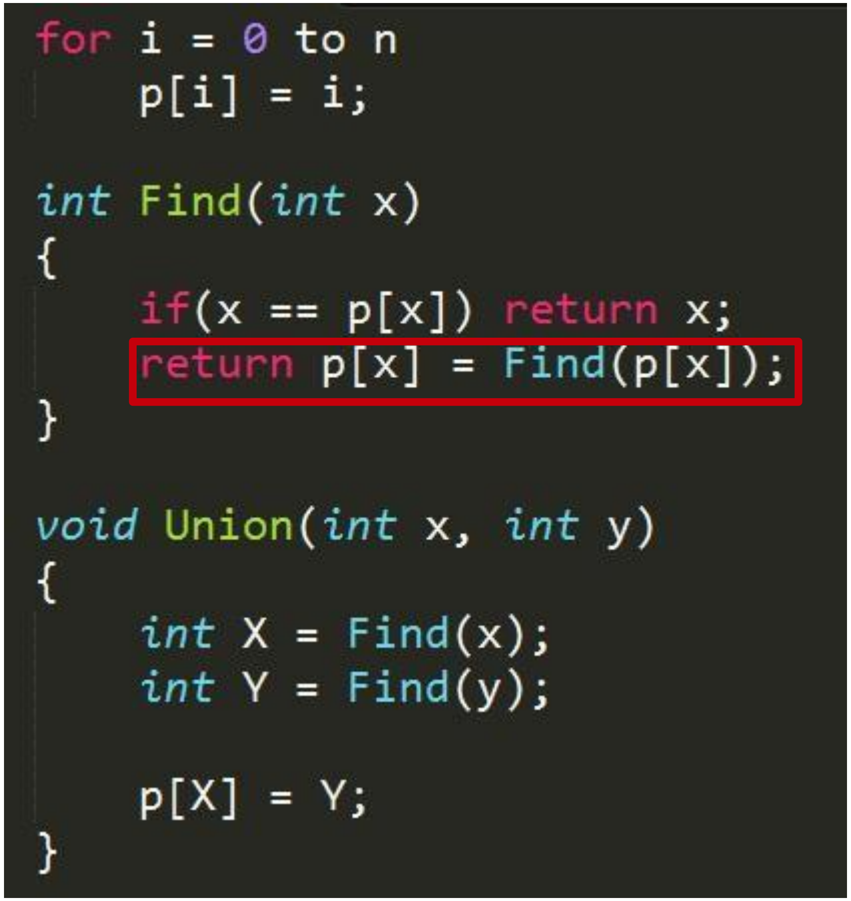
made by electron & free999

Disjoint Set

- Find(7) : return 5



X = 7

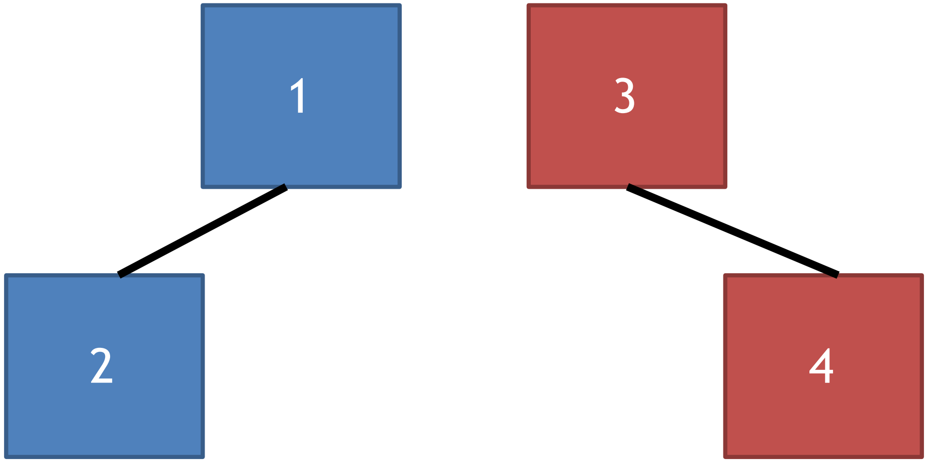


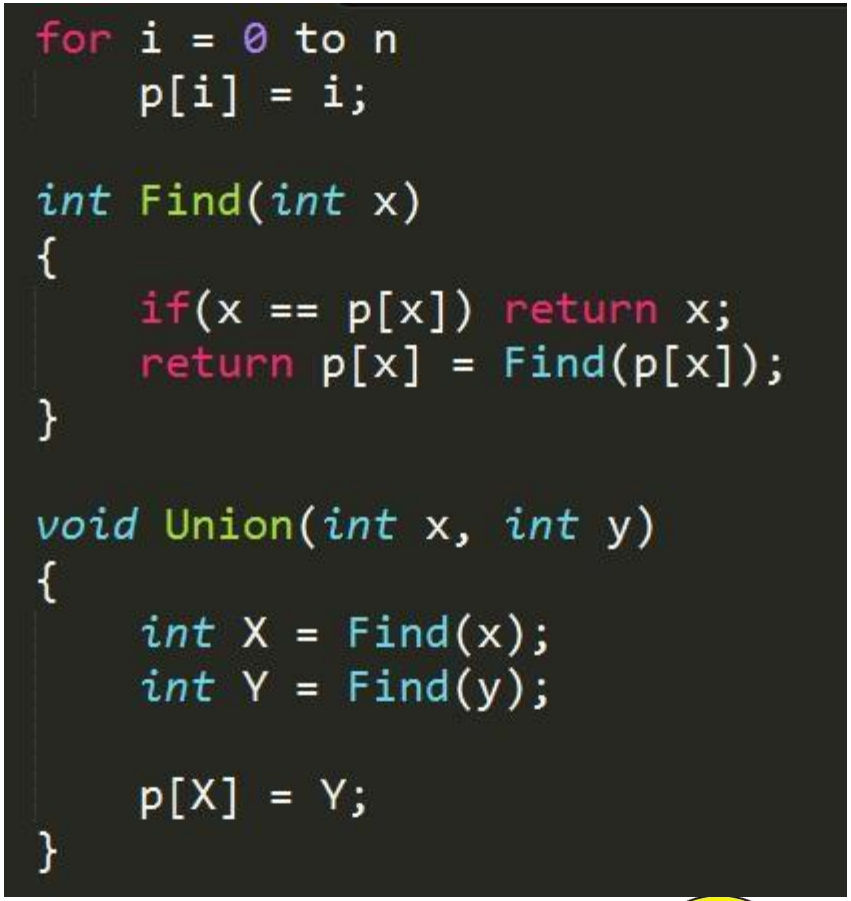
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)



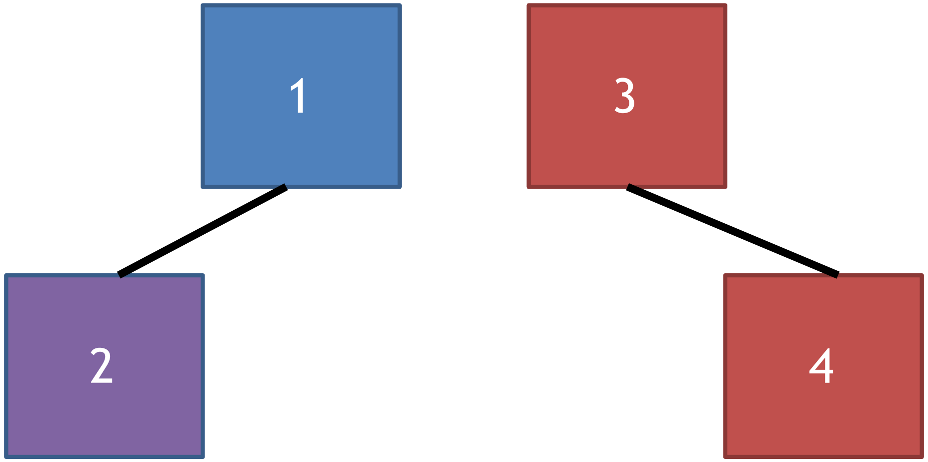


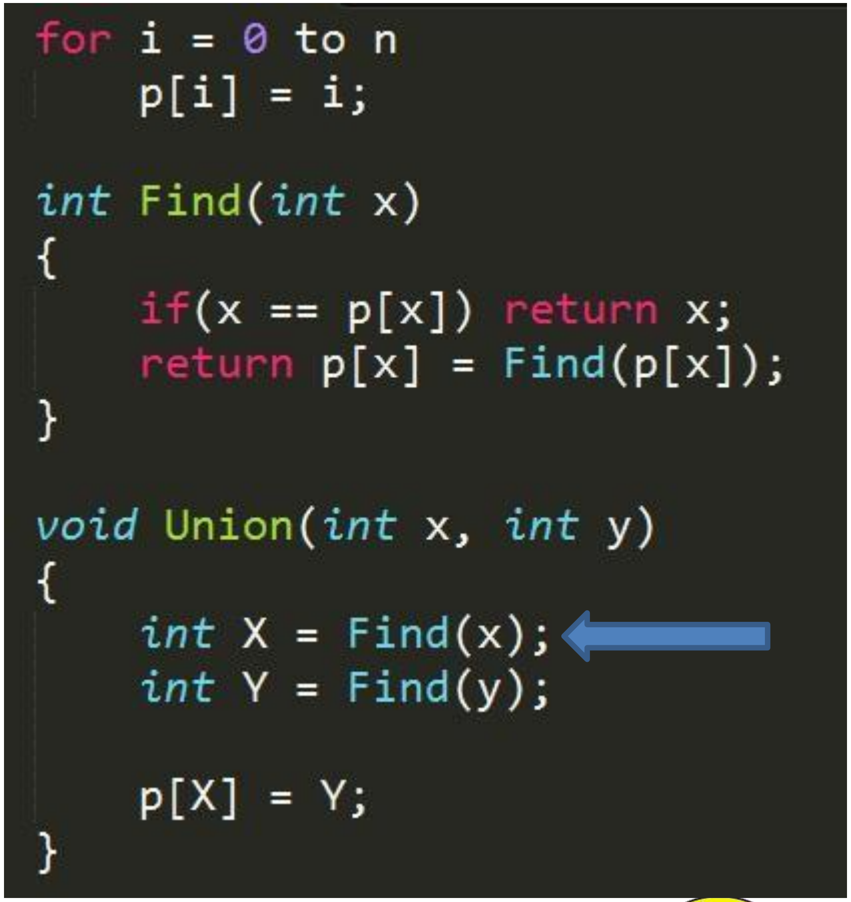
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)



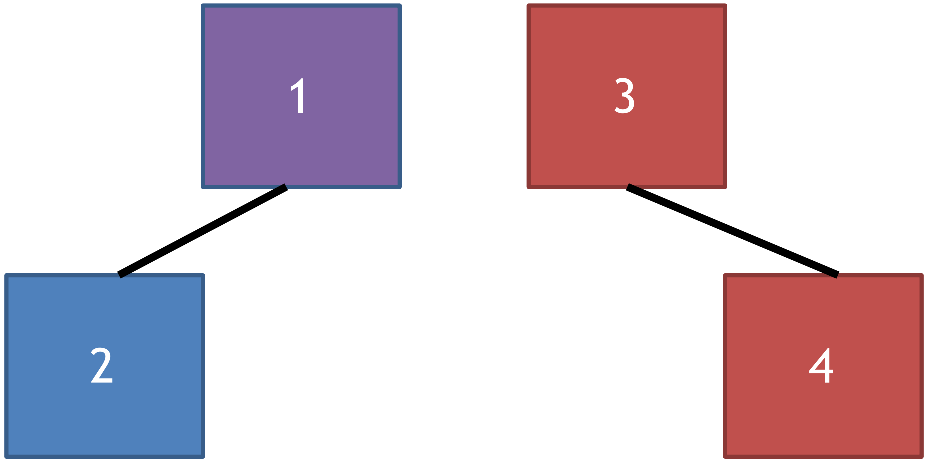


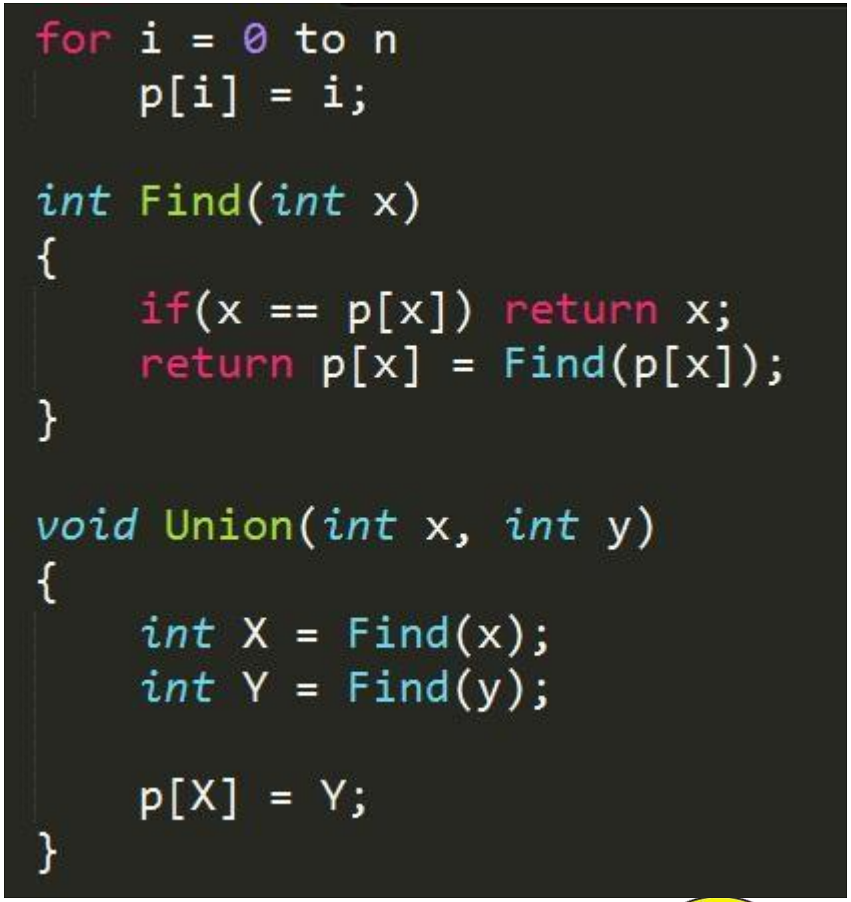
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)



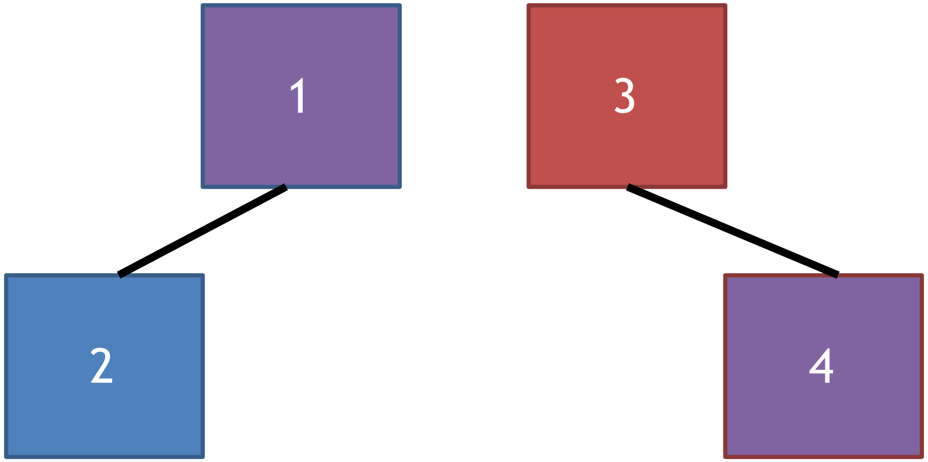


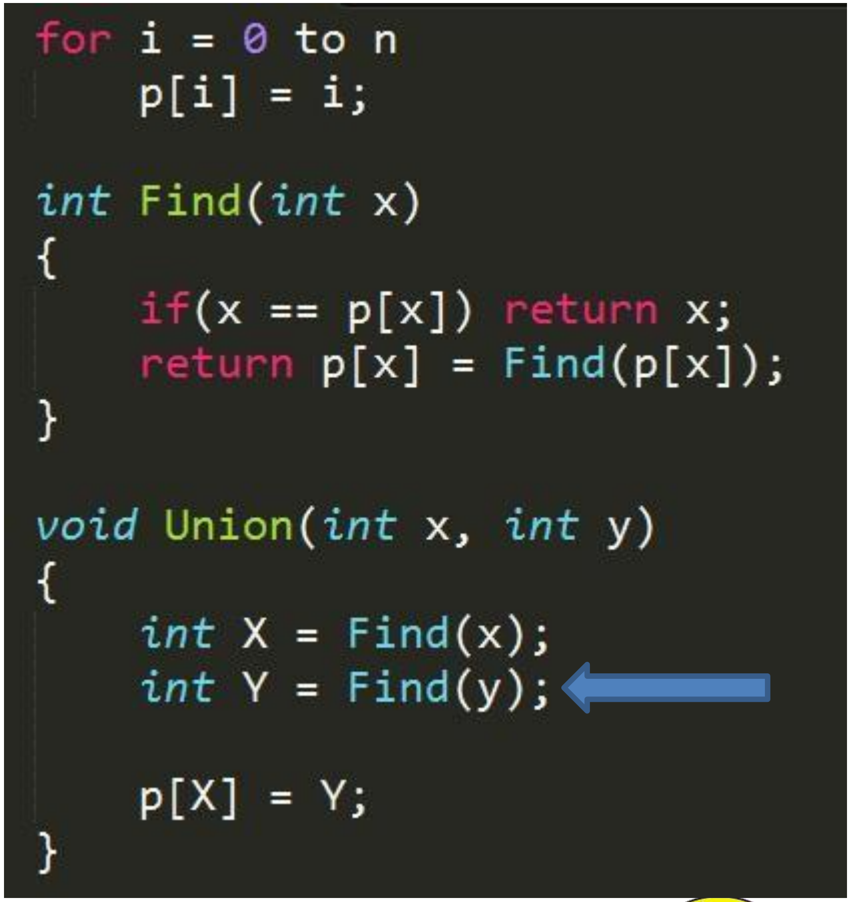
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)



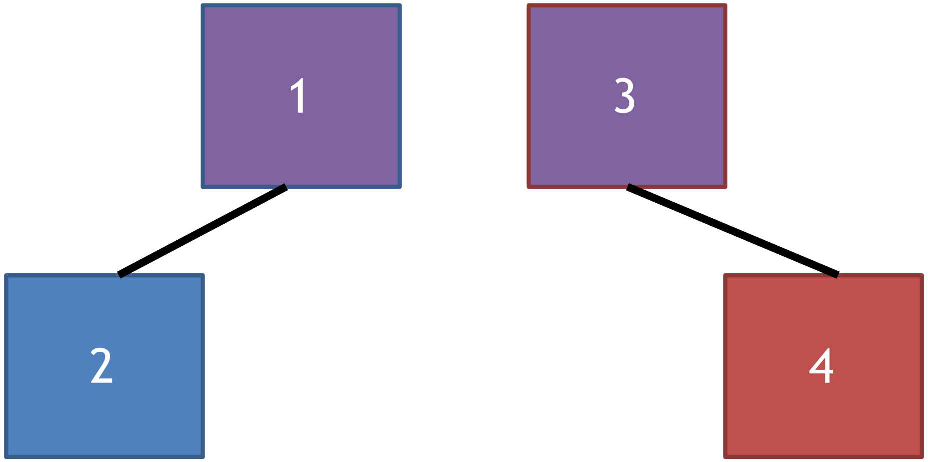


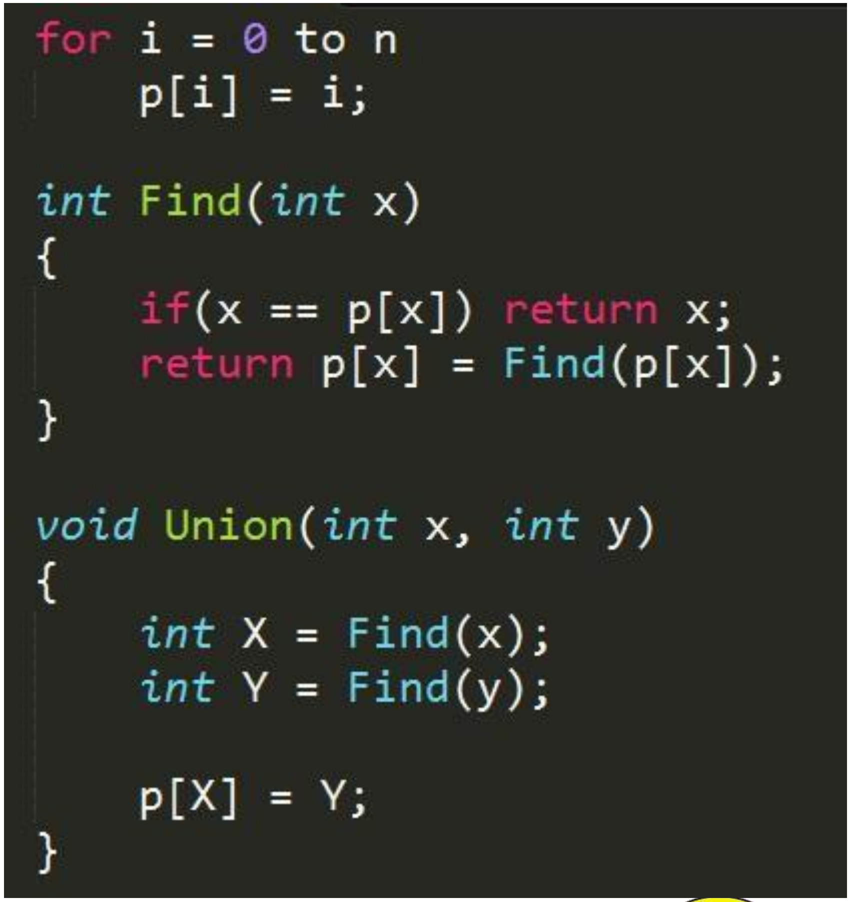
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)



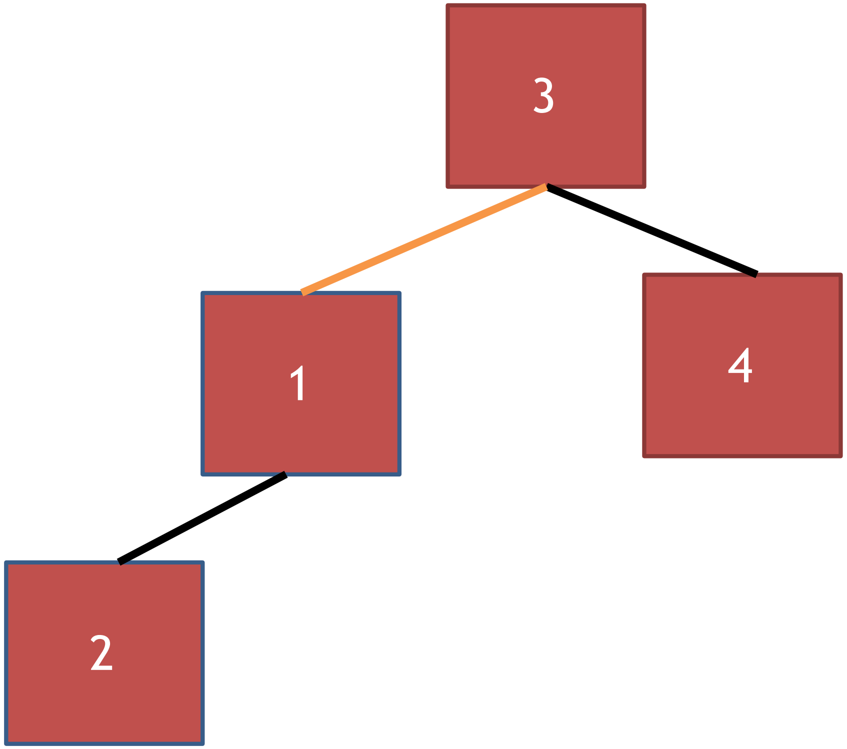


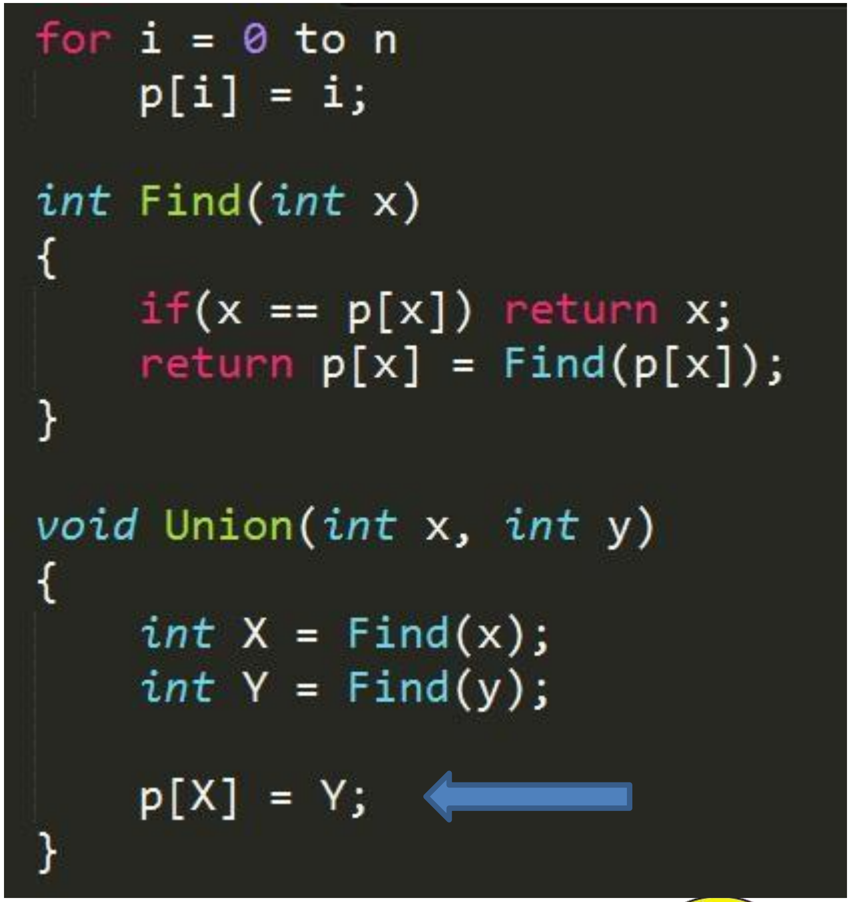
```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Disjoint Set

- Union(2, 4)





```
for i = 0 to n
    p[i] = i;

int Find(int x)
{
    if(x == p[x]) return x;
    return p[x] = Find(p[x]);
}

void Union(int x, int y)
{
    int X = Find(x);
    int Y = Find(y);

    p[X] = Y;
}
```

made by electron & free999

Example 1

UVa 10583 - Ubiquitous Religions

Problem Description

There are so many different religions in the world today that it is difficult to keep track of them all. You are interested in finding out how many different religions students in your university believe in. You know that there are n students in your university ($0 < n \leq 50000$). It is infeasible for you to ask every student their religious beliefs. Furthermore, many students are not comfortable expressing their beliefs. One way to avoid these problems is to ask m ($0 \leq m \leq n(n - 1)/2$) pairs of students and ask them whether they believe in the same religion (e.g. they may know if they both attend the same church). From this data, you may not know what each person believes in, but you can get an idea of the upper bound of how many different religions can be possibly represented on campus. You may assume that each student subscribes to at most one religion.

made by electron & free999

Example 1

Input

The input consists of a number of cases. Each case starts with a line specifying the integers n and m. The next m lines each consists of two integers i and j, specifying that students i and j believe in the same religion. The students are numbered 1 to n.The end of input is specified by a line in which n = m = 0.

Output

For each test case, print on a single line the case number (starting with 1) followed by the maximum number of different religions that the students in the university believe in.

made by electron & free999

Example 2

UVa 879 – Circuit Nets

Problem Description

An electronic circuit consists of components, pins, and wires. Figure 1 shows a circuit with the three components A, B, and C. Each wire connects a pair of pins. Two pins a and b are electrically equivalent if they are either connected by a wire or there is a sequence a1, a2, . . . , ak of pins such that a, a1; a1, a2; . . . ; ak−1, ak; and ak, b are all connected by wires. A net is a maximal set of electrically equivalent pins. Maximal means that no pin outside the net is electrically equivalent to a pin in the net.

Given a set of pins and a set of wires your task is to determine the number of nets defined within the circuit. The figure below illustrates the circuit with three nets described in the sample input section.

made by electron & free999

Example 2

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. The sequence of pins is always numbered from 1 to N. The first input value is the value of N. The sequence of wires is described by pairs of pin values. It is given at the input as sequence of numbers, separated by a space. The first two numbers define the first pair, the two numbers that follow define the second pair, and so on. The input ends with the end of file mark.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. The output is just the number of nets found.

Exercise

- Uva(4)
  - 793, 10583, 10685, 10158
- POJ(1)
  - 1703