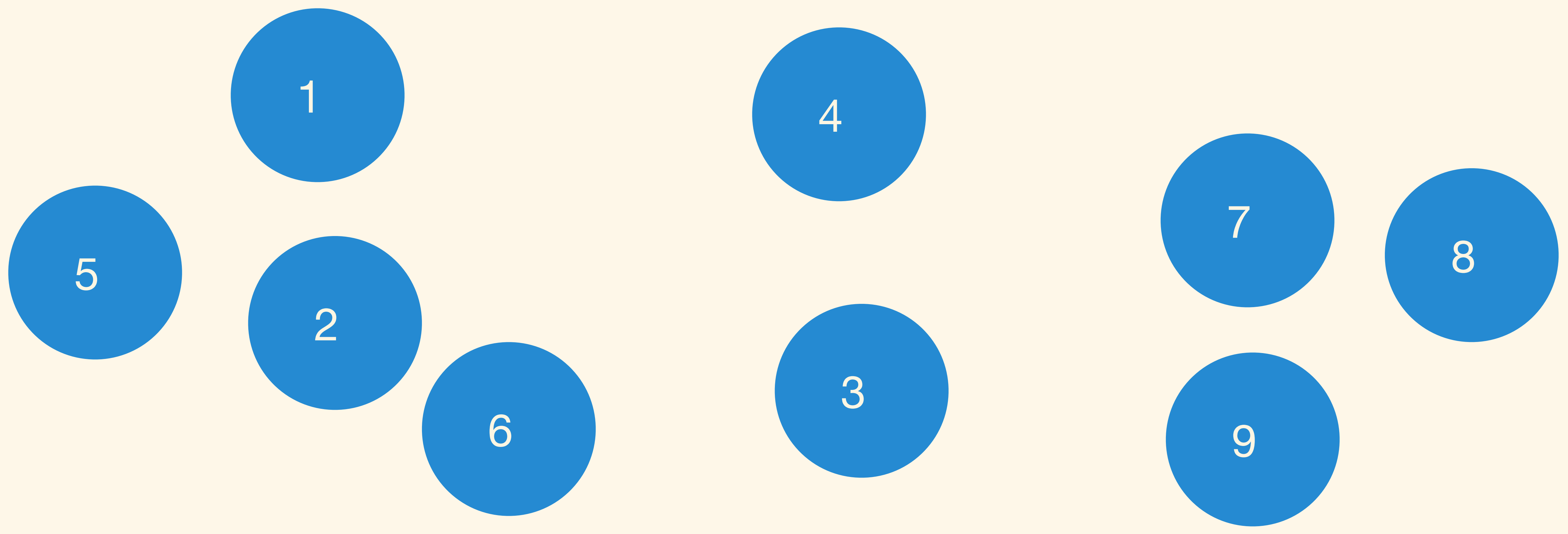
Disjoint Set

郭至軒（KuoE0）

KuoE0.tw@gmail.com
KuoE0.ch

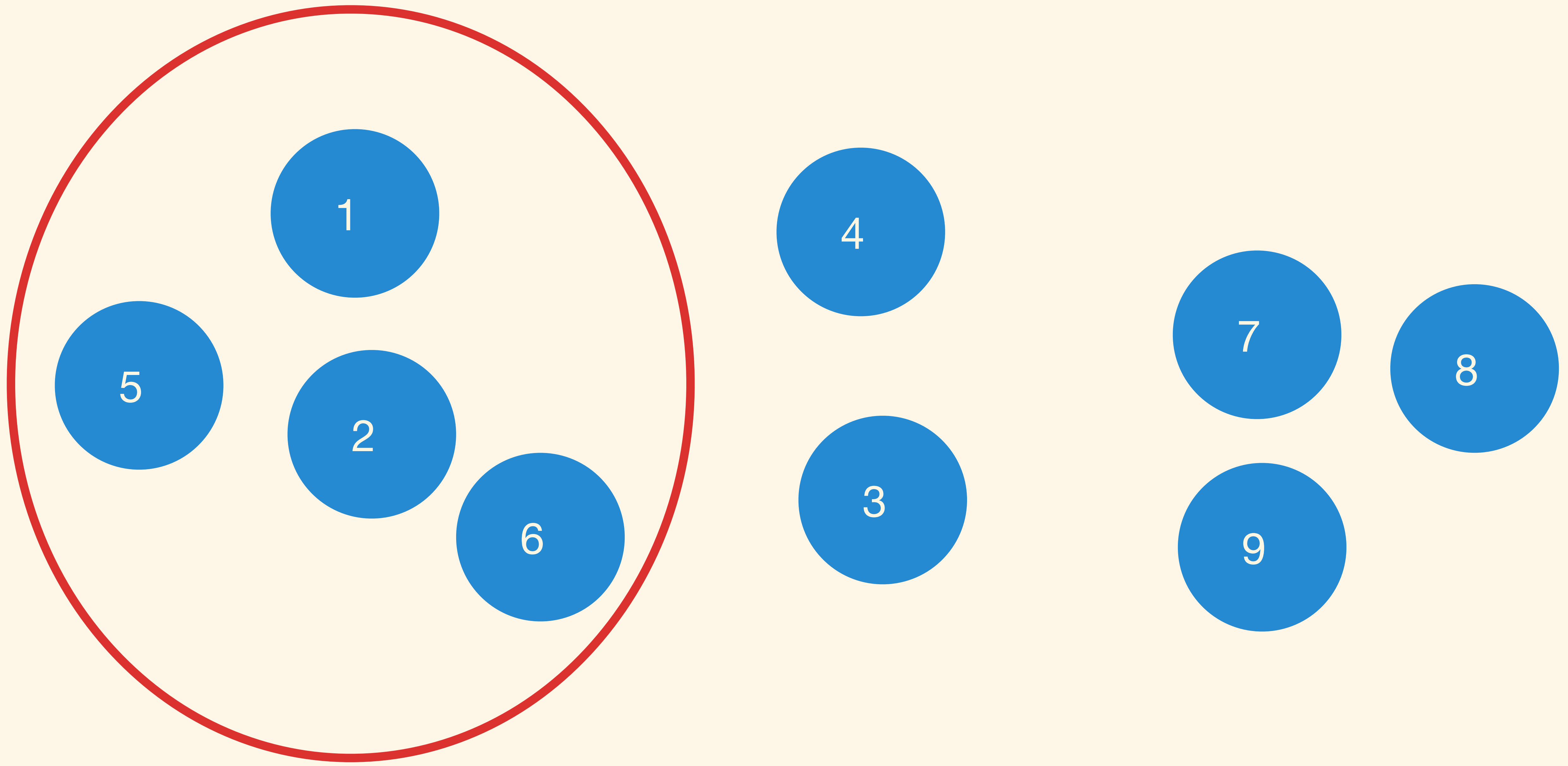
Disjoint Set

中國譯：並查集



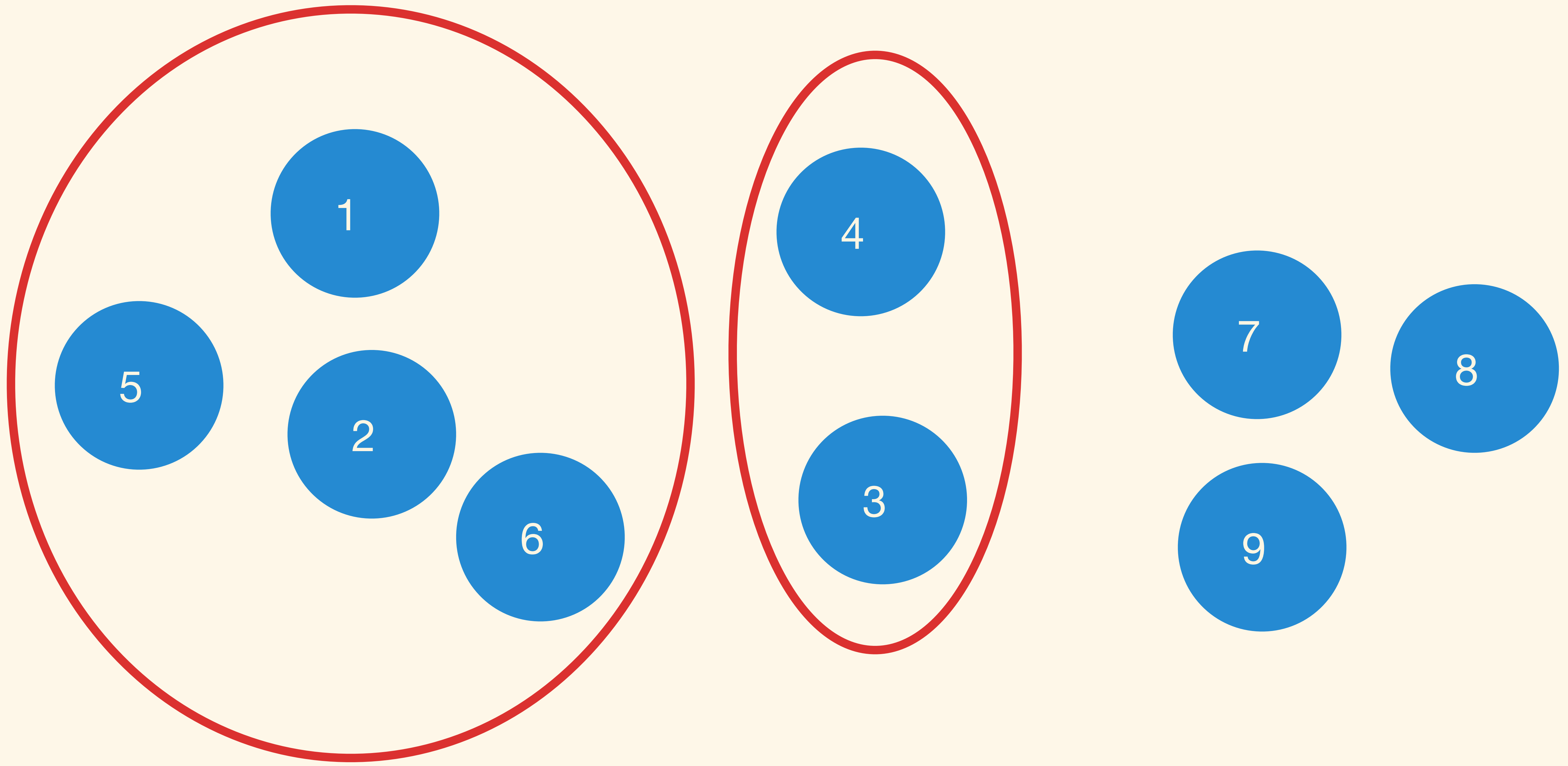
Disjoint Set

中國譯：並查集



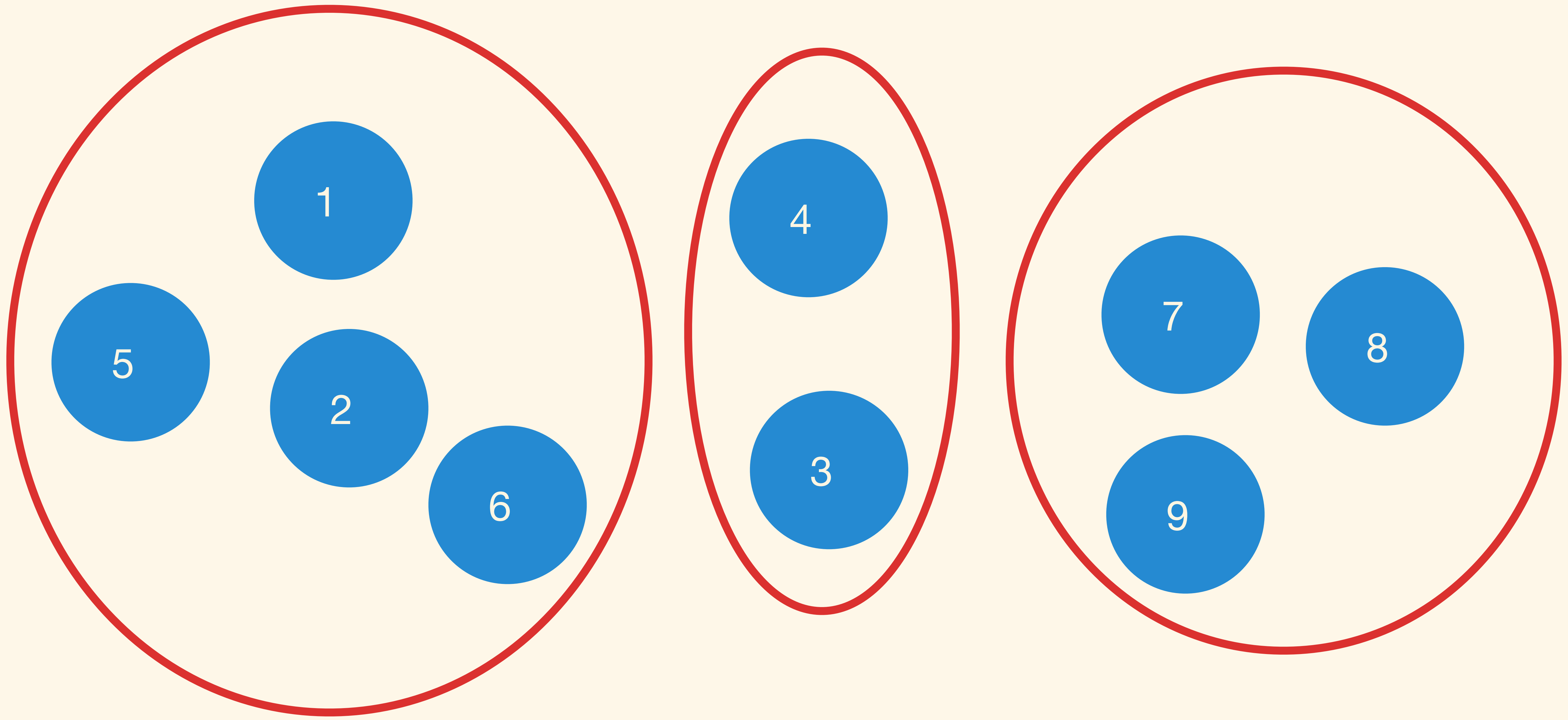
Disjoint Set

中國譯：並查集

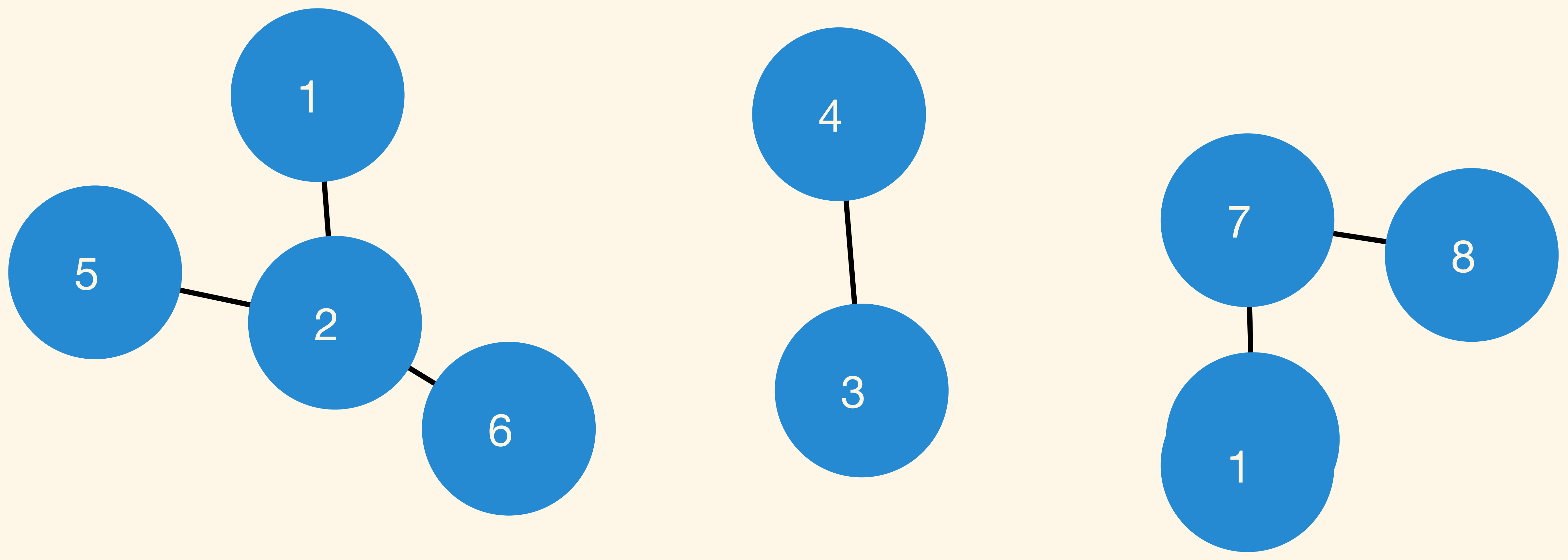


Disjoint Set

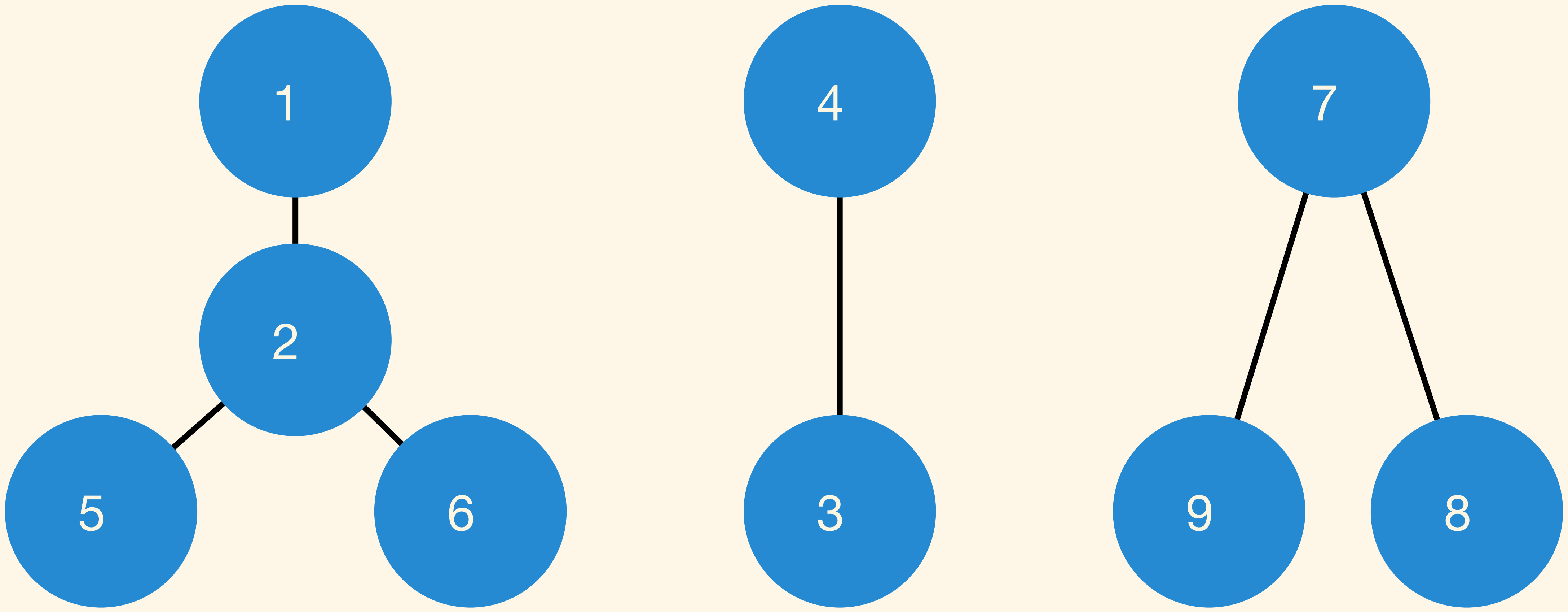
中國譯：並查集



Tree Structure



Tree Structure

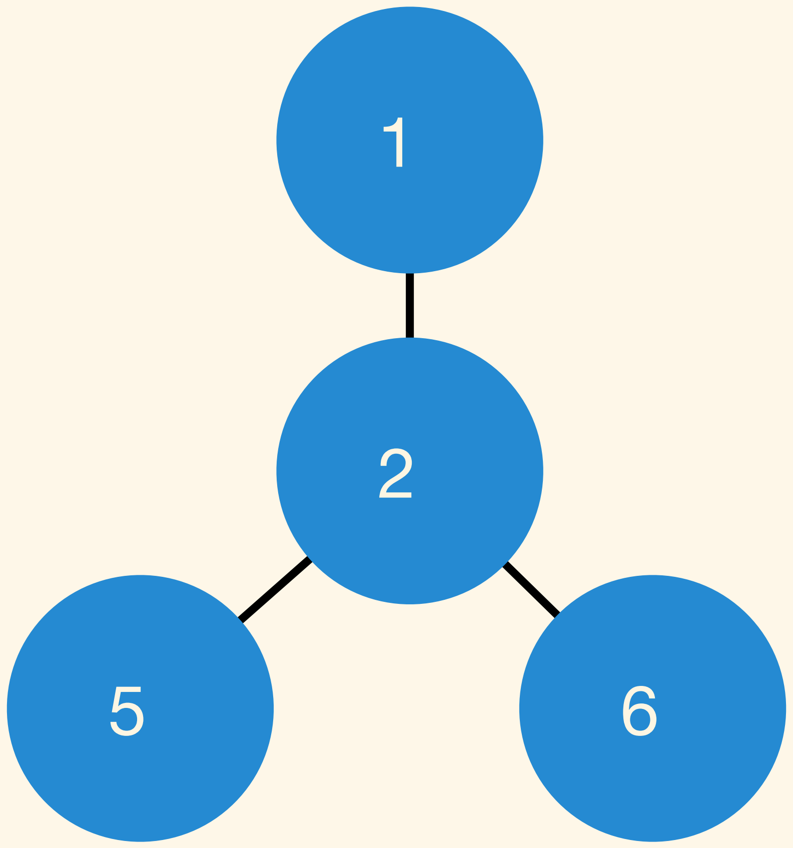


Data Structure

index	0	1	2	3	4
content	parent node	parent node	parent node	parent node	parent node

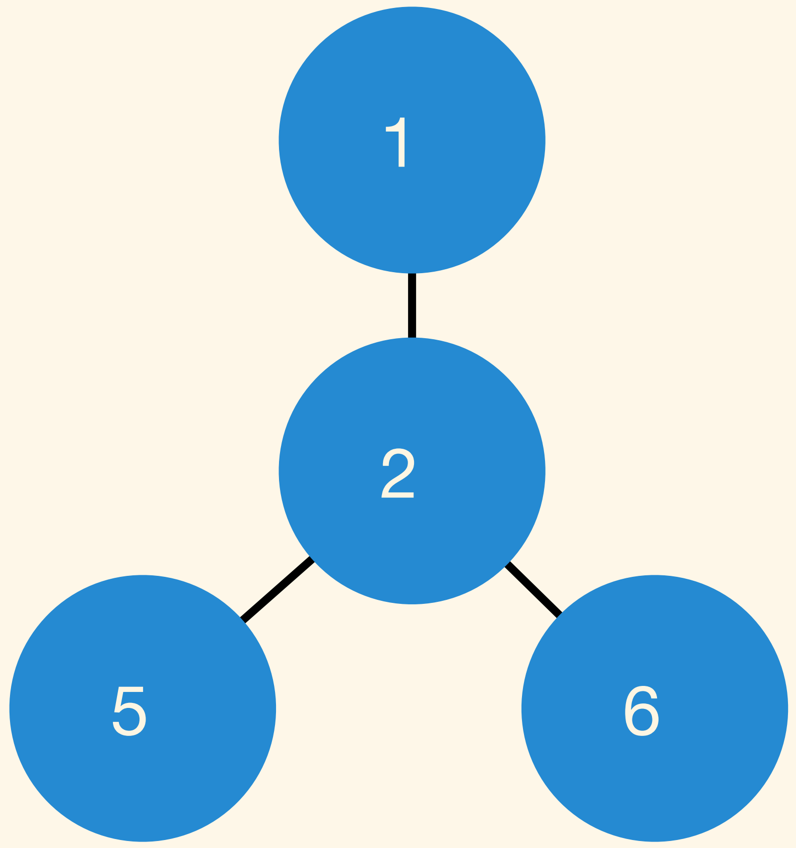
The parent of root is itself!

Find Operation

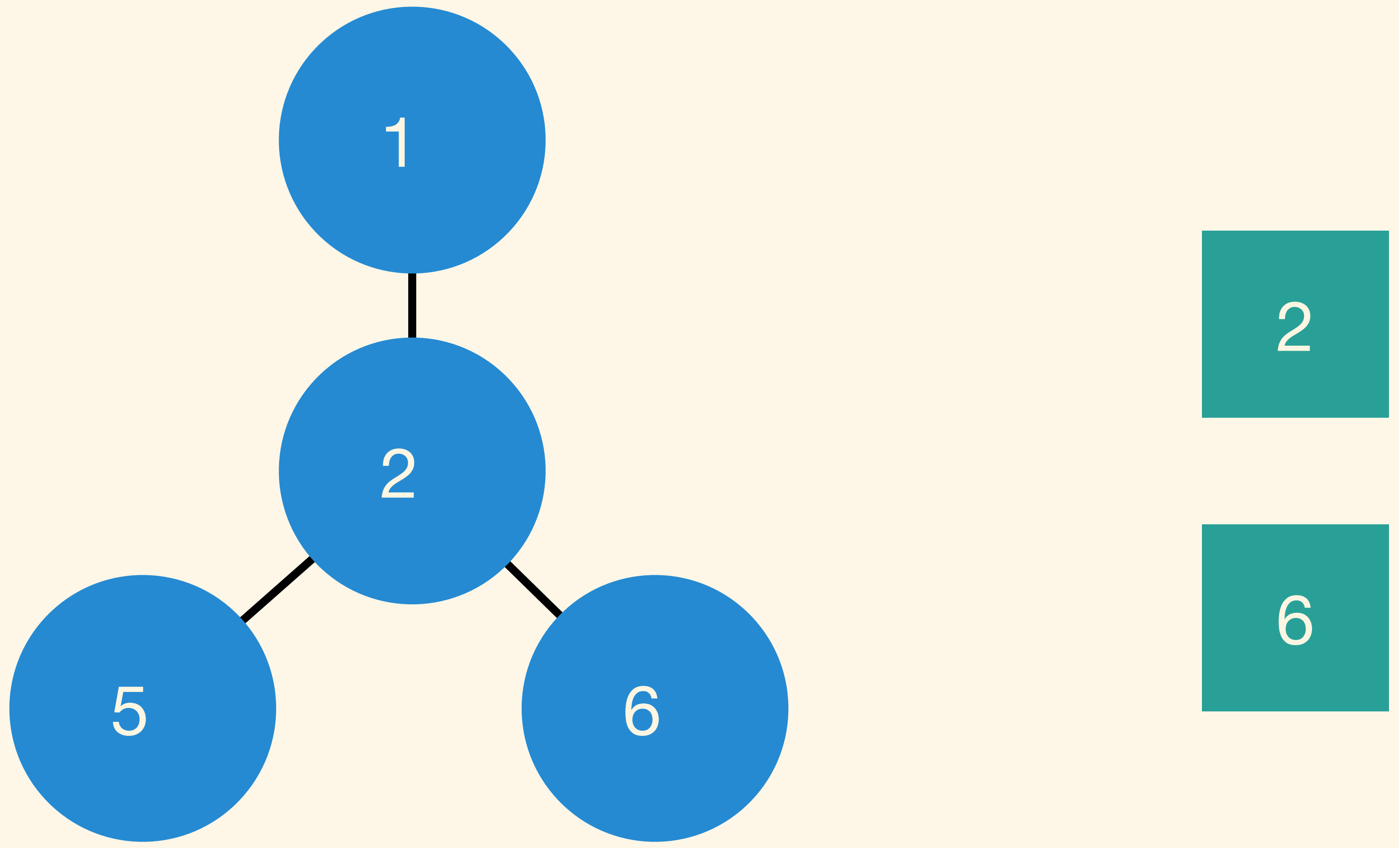


node	parent
1	**1**
2	**1**
5	**2**
6	**2**

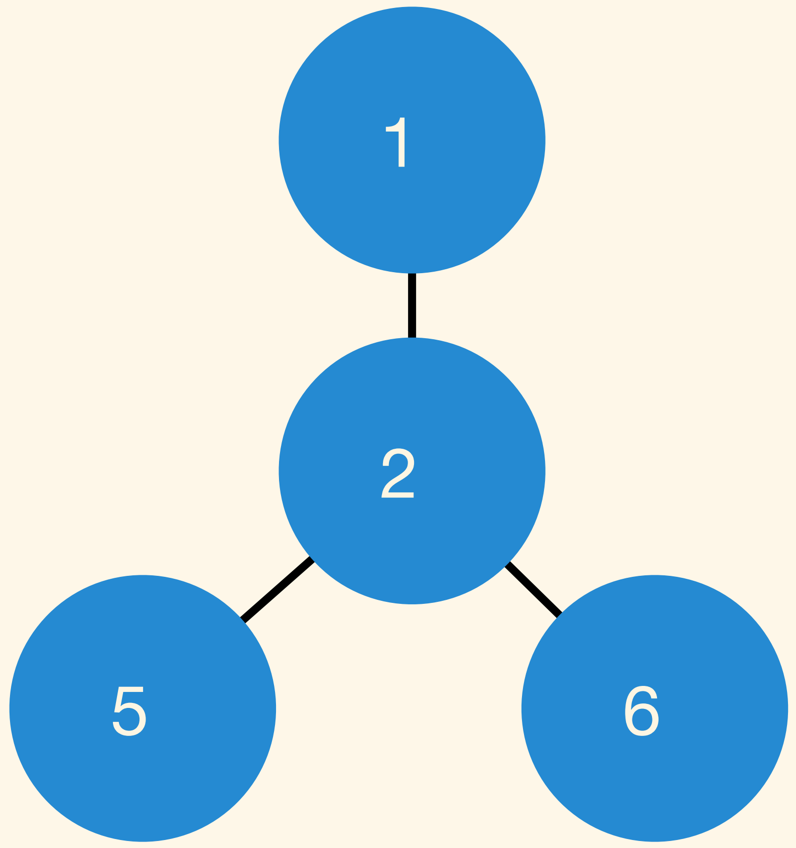
How to determine that node 2 and node 6 in the same set?

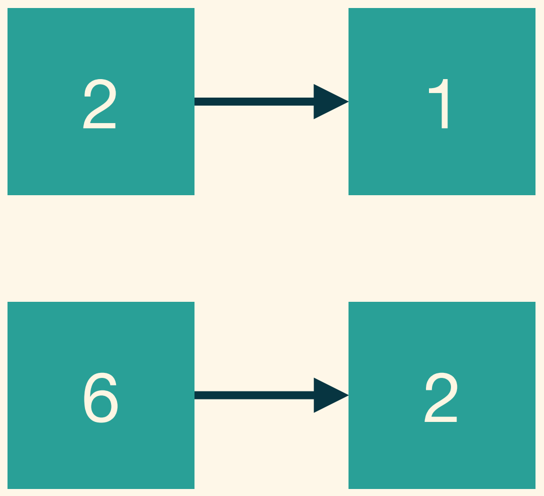


How to determine that node 2 and node 6 in the same set?

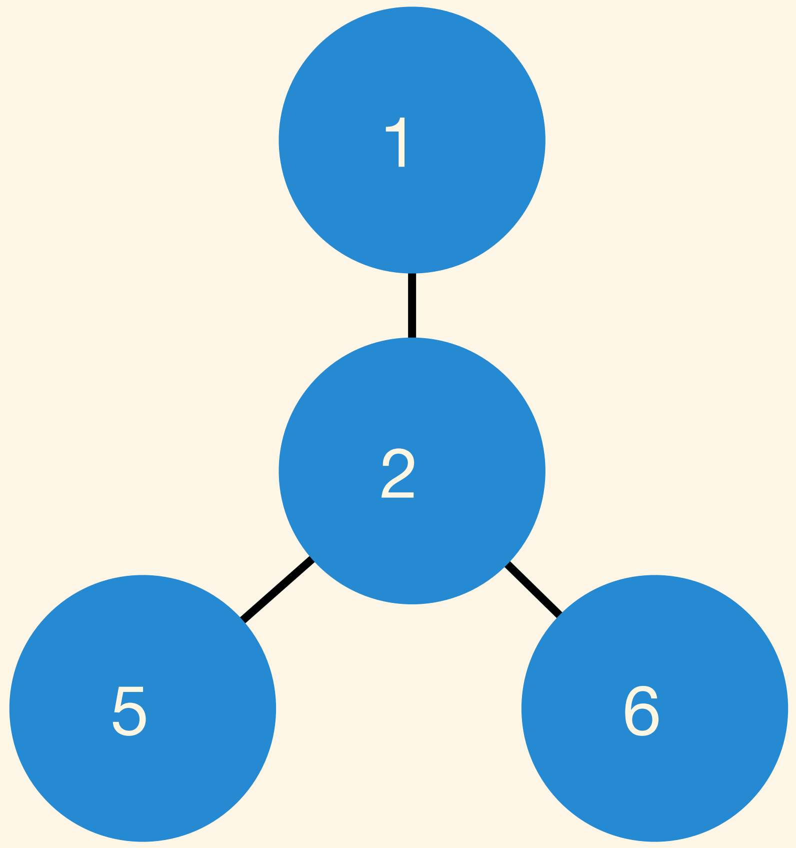


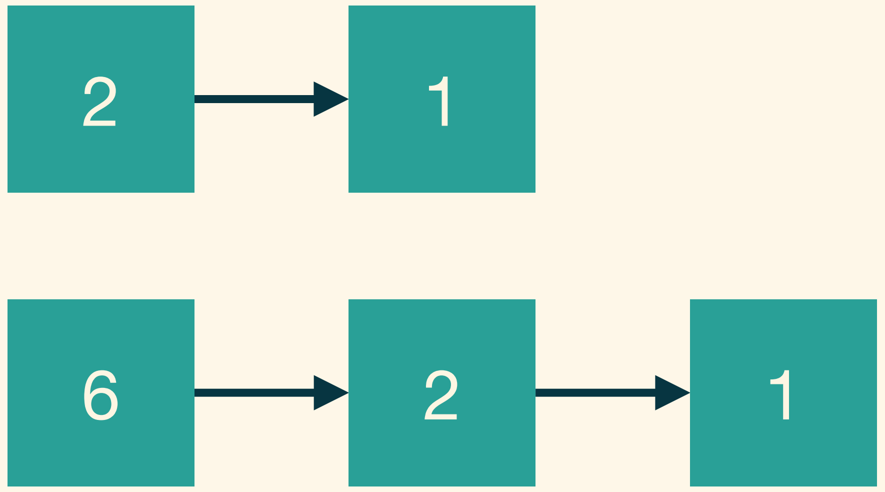
How to determine that node 2 and node 6 in the same set?




$$\begin{tikzcd}\boxed{2} \arrow[r] & \boxed{1} \\ \boxed{6} \arrow[r] & \boxed{2}\end{tikzcd}$$

How to determine that node 2 and node 6 in the same set?



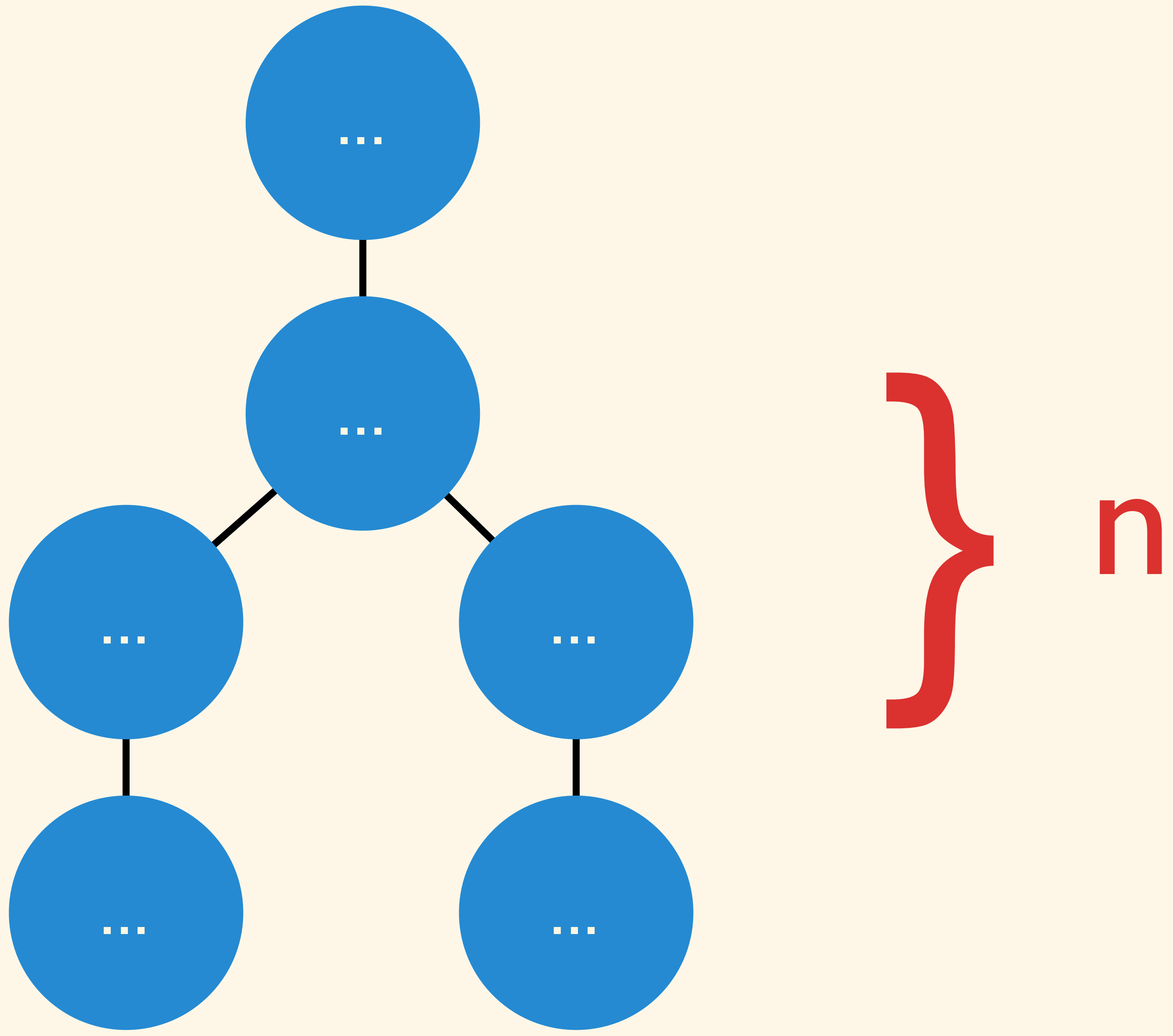

$$\begin{tikzcd}
\boxed{2} \arrow[r] & \boxed{1} & \\
\boxed{6} \arrow[r] & \boxed{2} \arrow[r] & \boxed{1}
\end{tikzcd}$$

Source Code

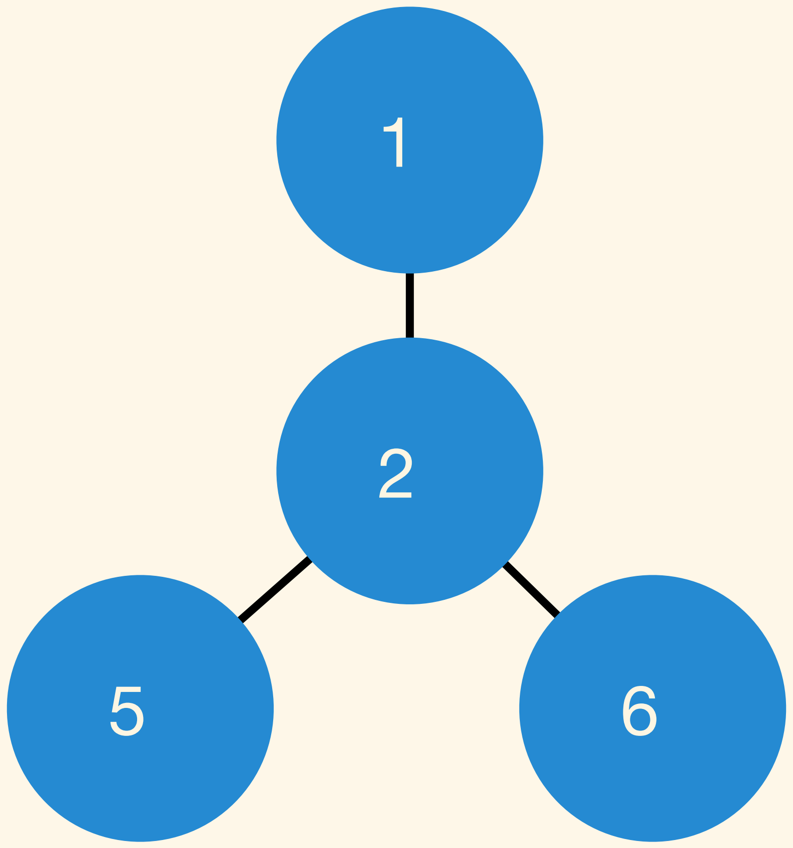
```
// parent is an array stored the parent
of nodes.

int findSet( int x ) {
    if ( parent[ x ] == x )
      return x;
  return findSet( parent[ x ] );
}
```

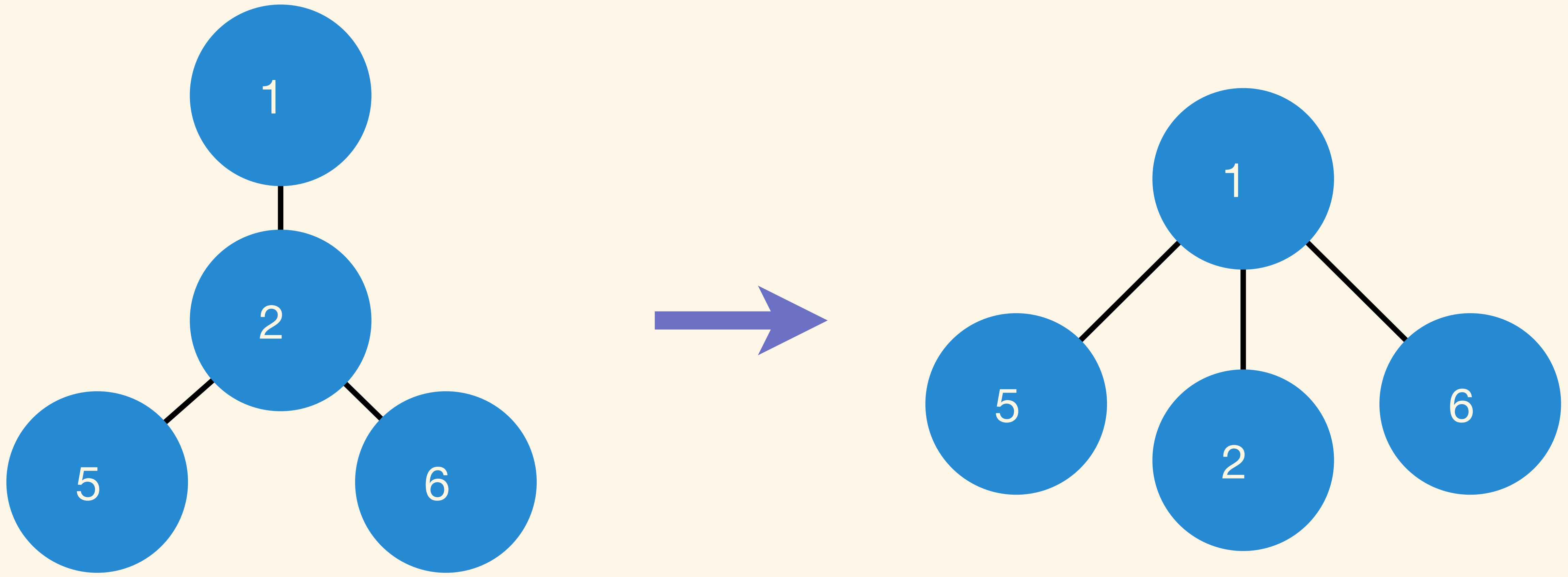
If the height of tree is very large...



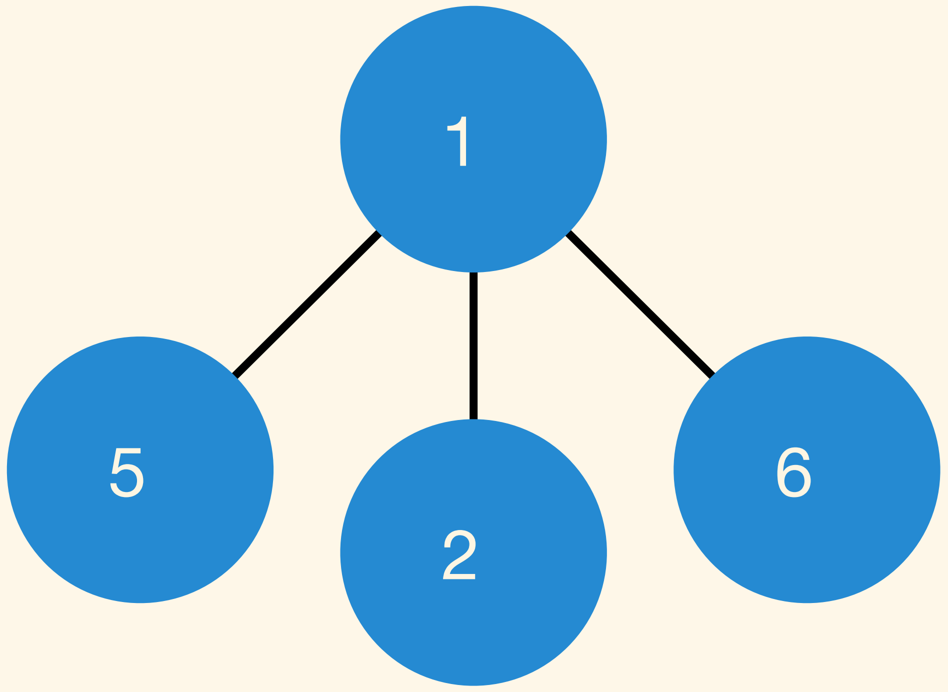
Reduce Height



Reduce Height



Reduce Height



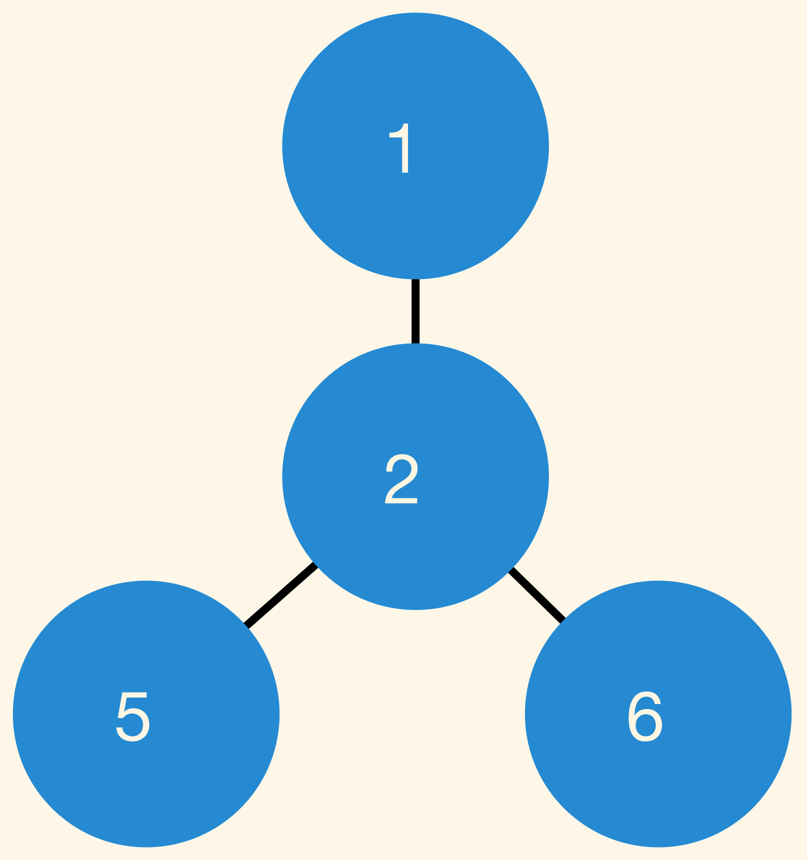
node	parent
1	1
2	1
5	1
6	1

Source Code

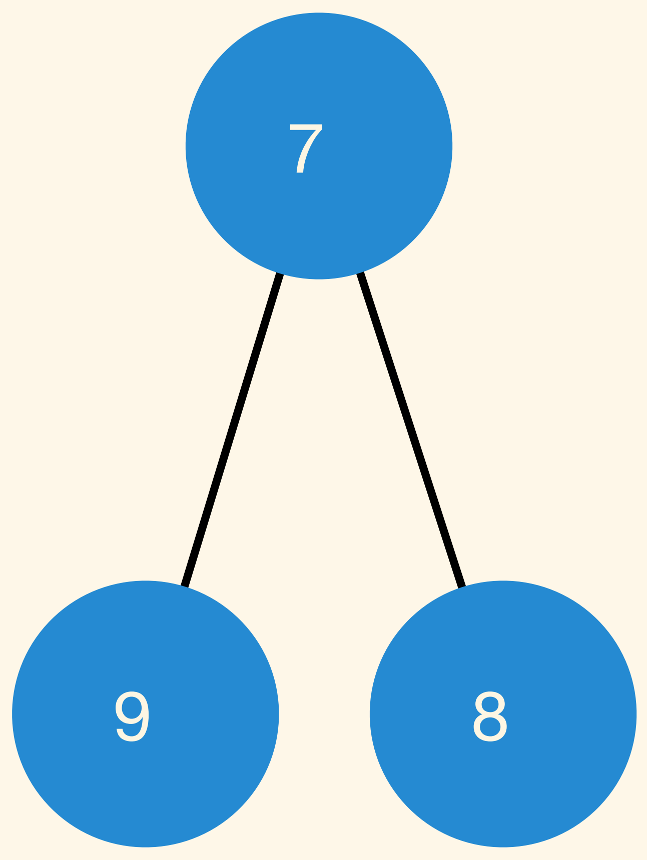
```
// parent is an array stored the parent of
nodes.

int findSet( int x ) {
    if ( parent[ x ] == x )
      return x;
  return parent[ x ] = findSet( parent[ x ] );
}
```

Union Operation

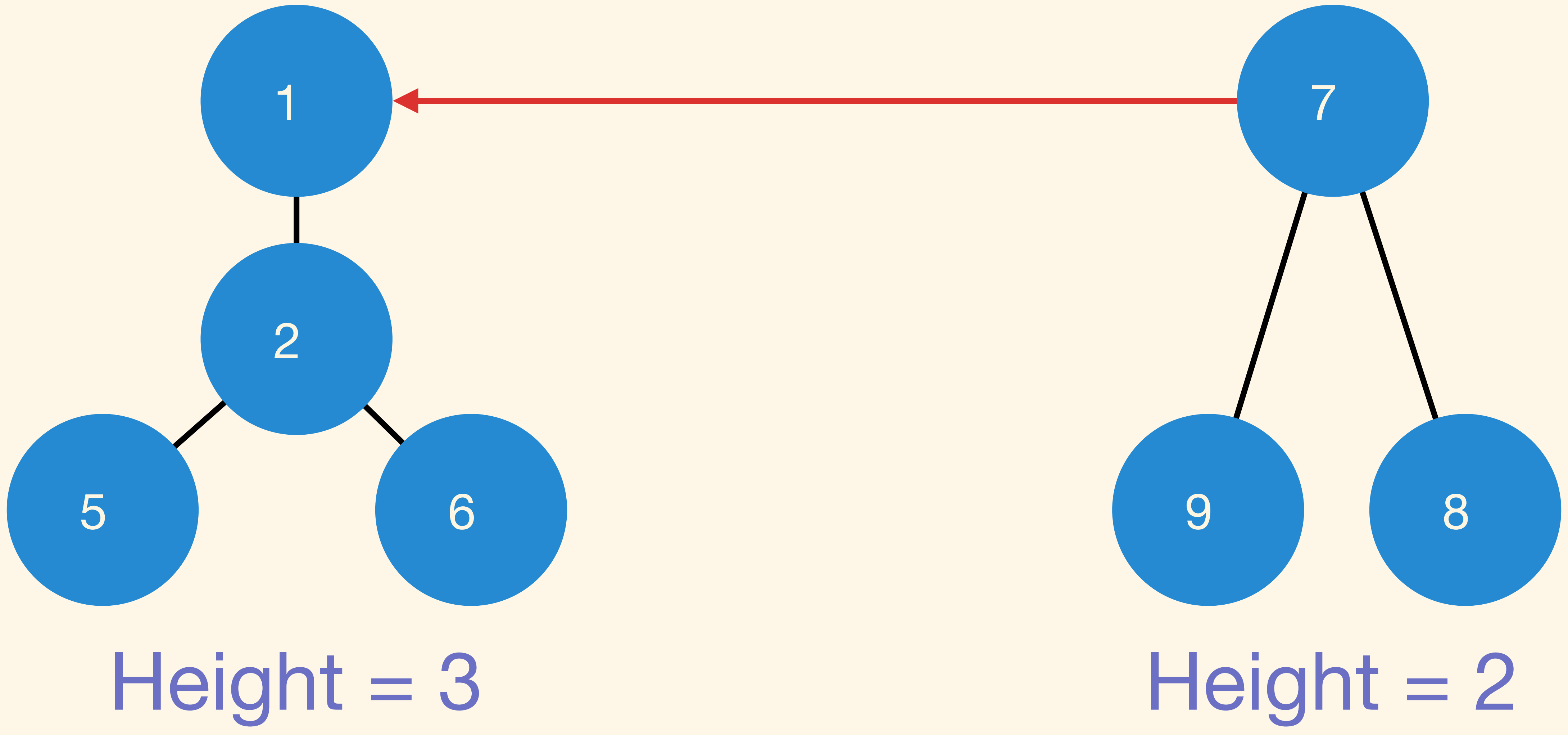


Height = 3

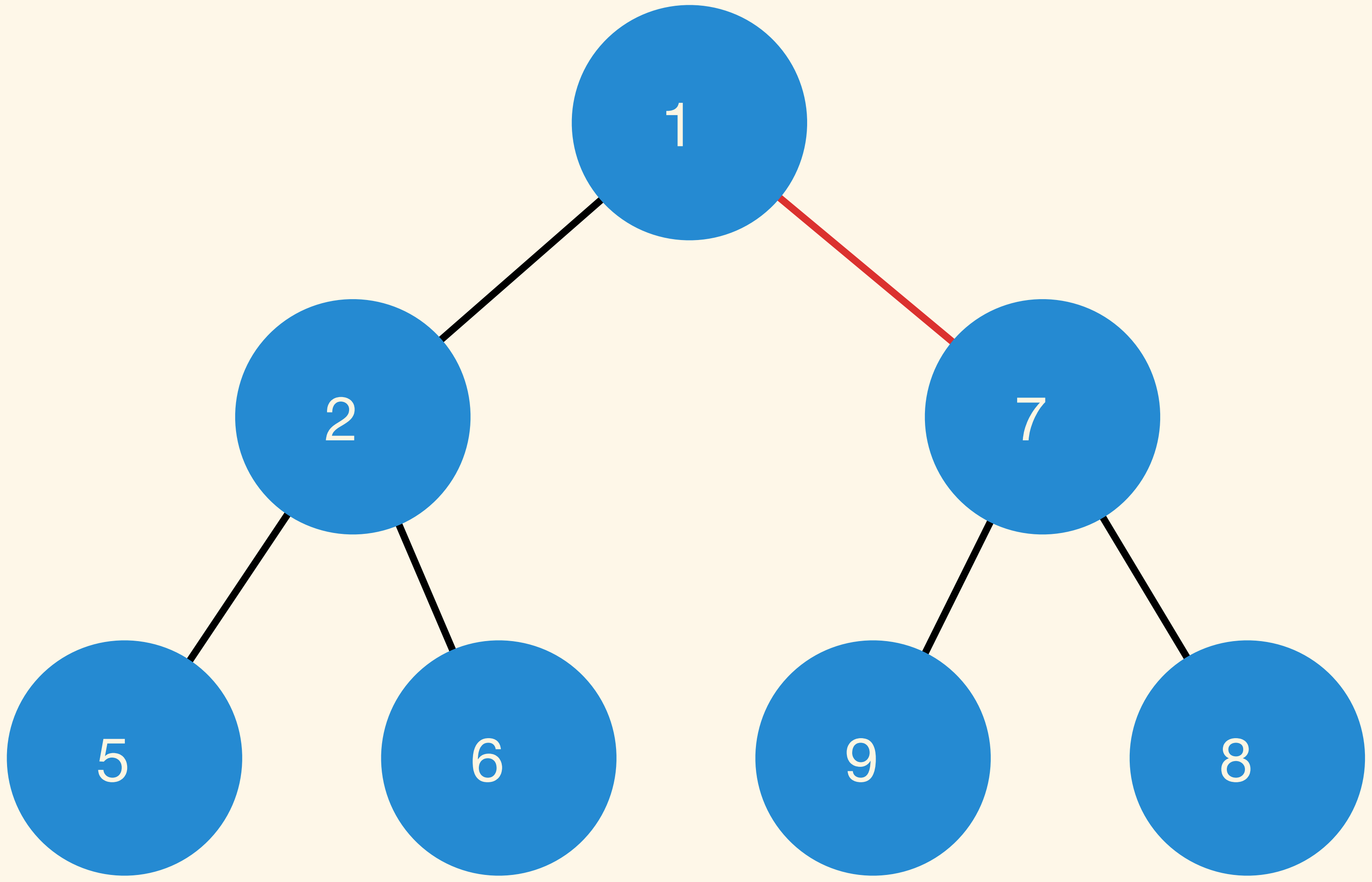


Height = 2

Union Operation



Union Operation



Height = 3

Source Code

```
// parent is an array stored the parent of
nodes.
// height is an array stored the height of tree

void unionSet( int a, int b ) {
  if ( a == b )
    return;
  if ( height[ a ] > height[ b ] )
    parent[ b ] = a;
  else {
    parent[ a ] = b;
    if ( height[ a ] == height[ b ] )
      ++height[ b ];
  }
}
```

Time Complexity

Find Operation	**O(1)**
Union Operation	**O(1)**

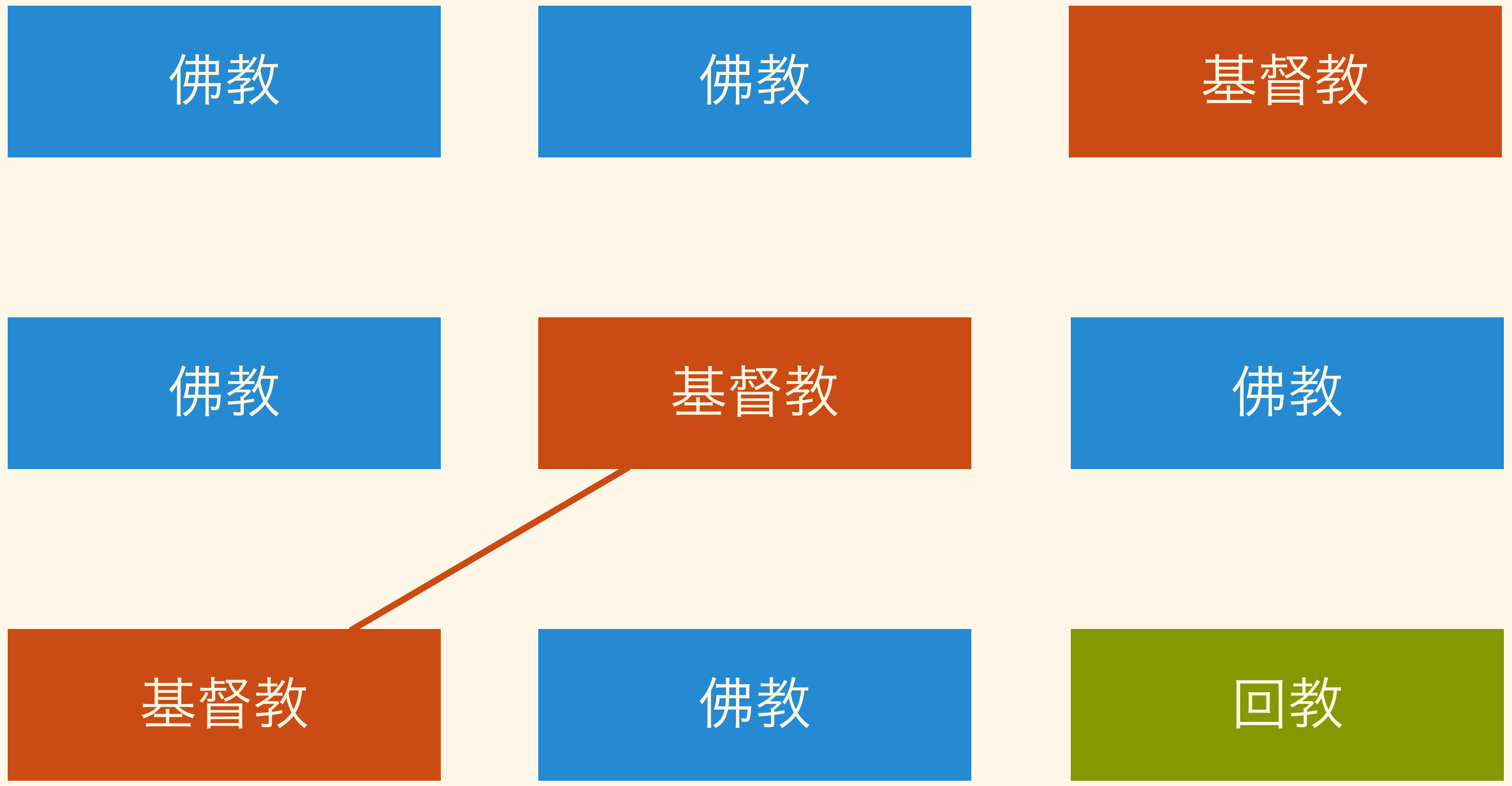
Practice Now

[UVa] 10583 - Ubiquitous Religions

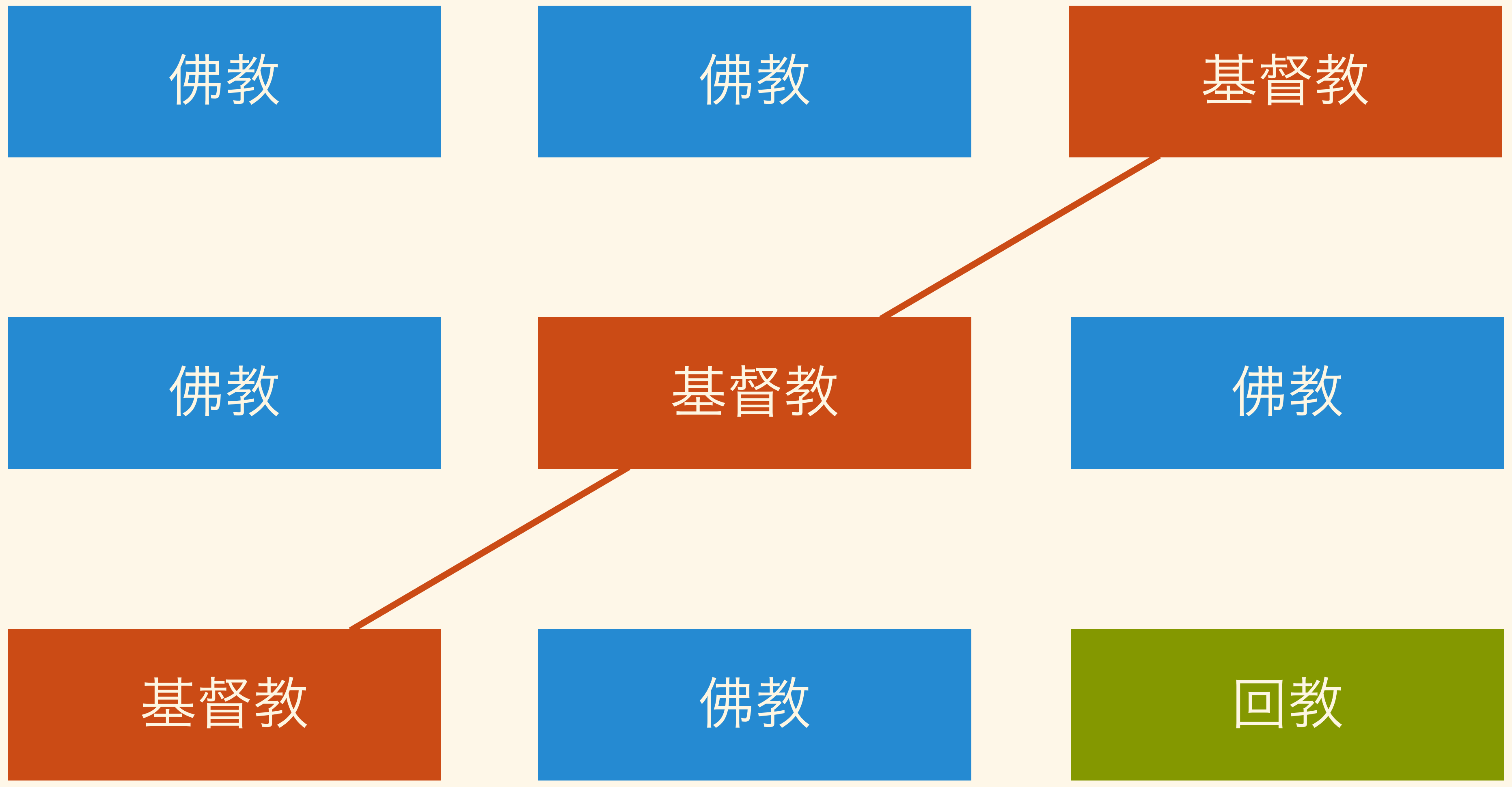
10583 - Ubiquitous Religions

佛教	佛教	基督教
佛教	基督教	佛教
基督教	佛教	回教

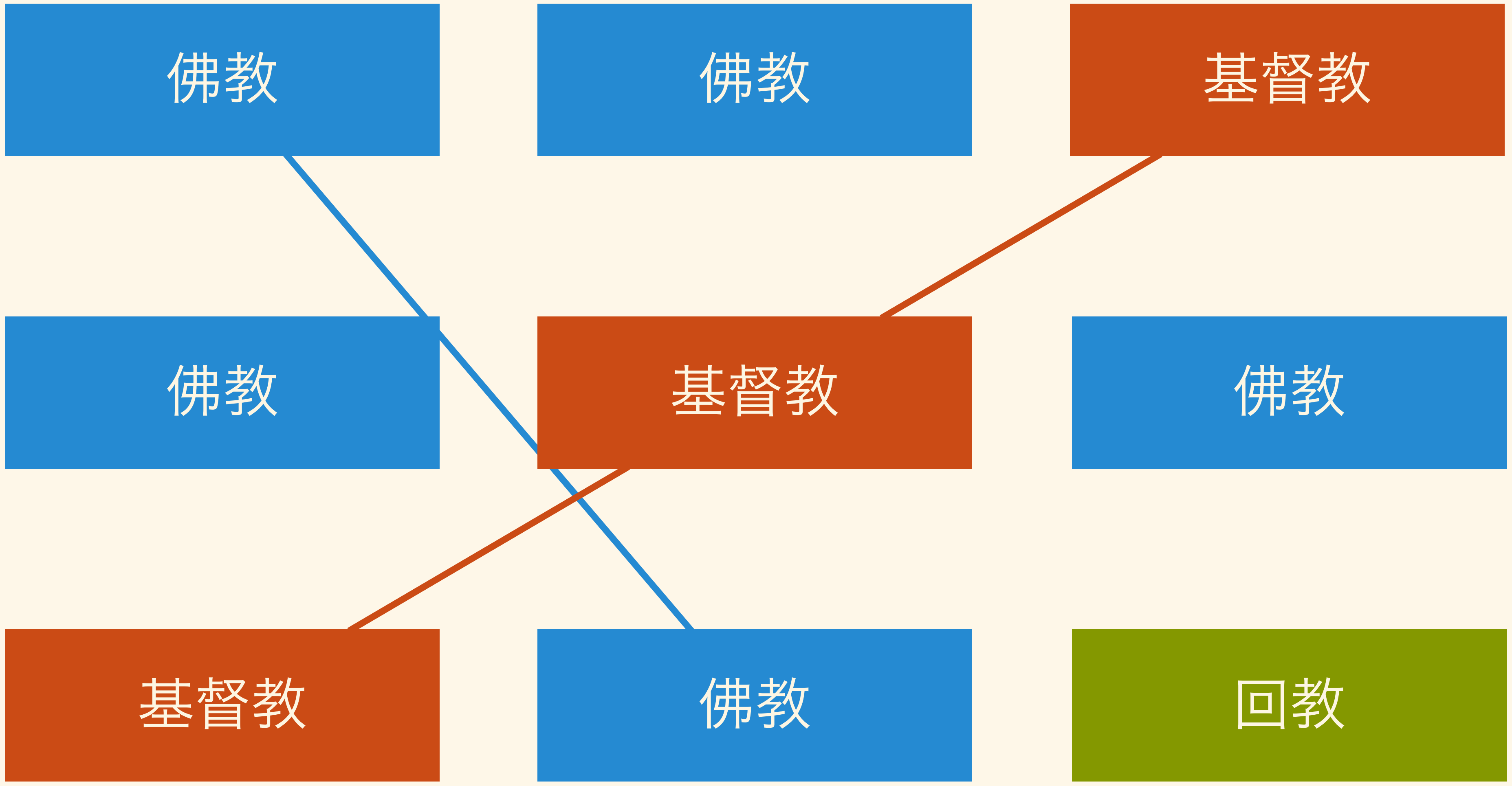
10583 - Ubiquitous Religions



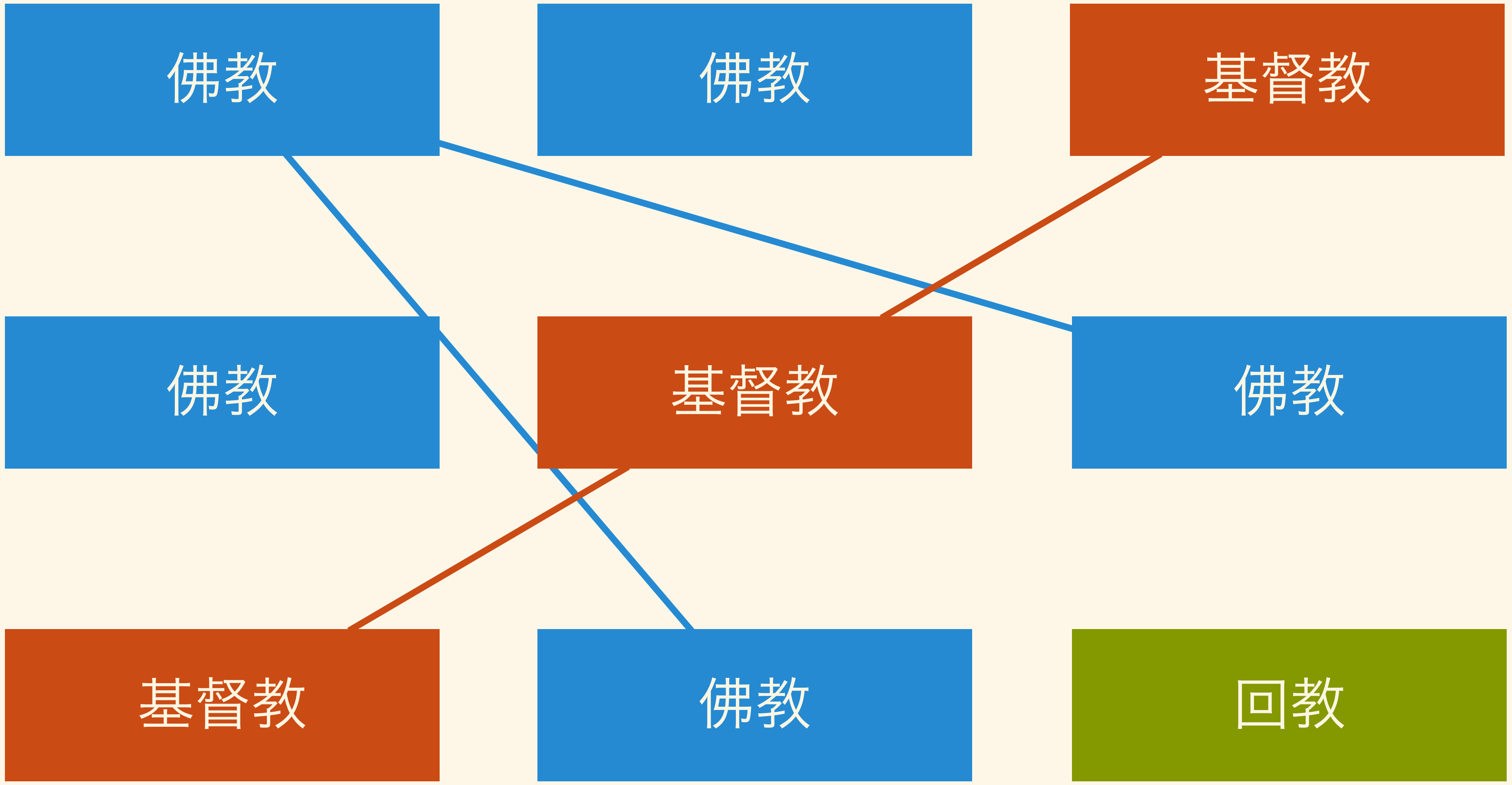
10583 - Ubiquitous Religions



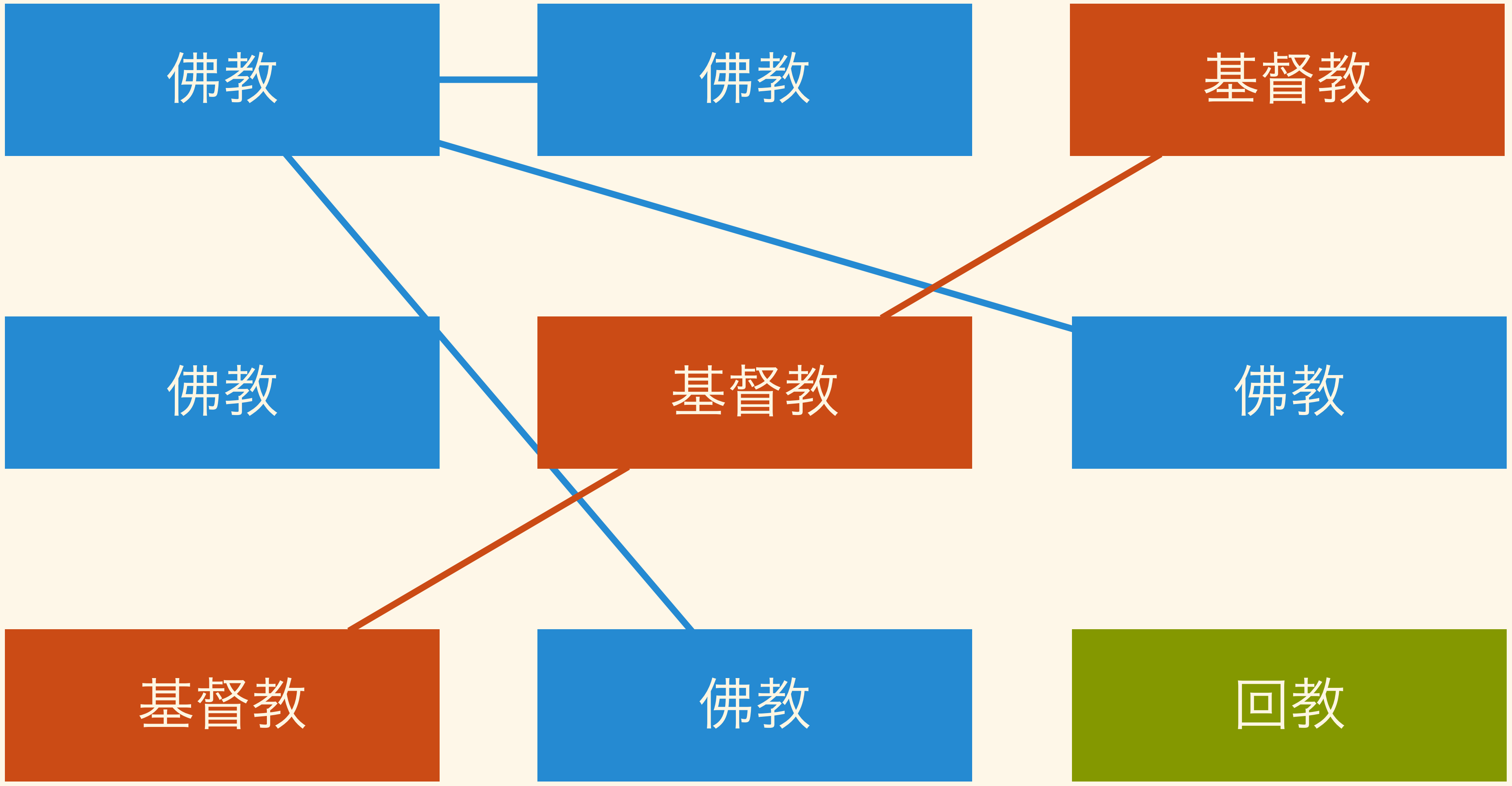
10583 - Ubiquitous Religions



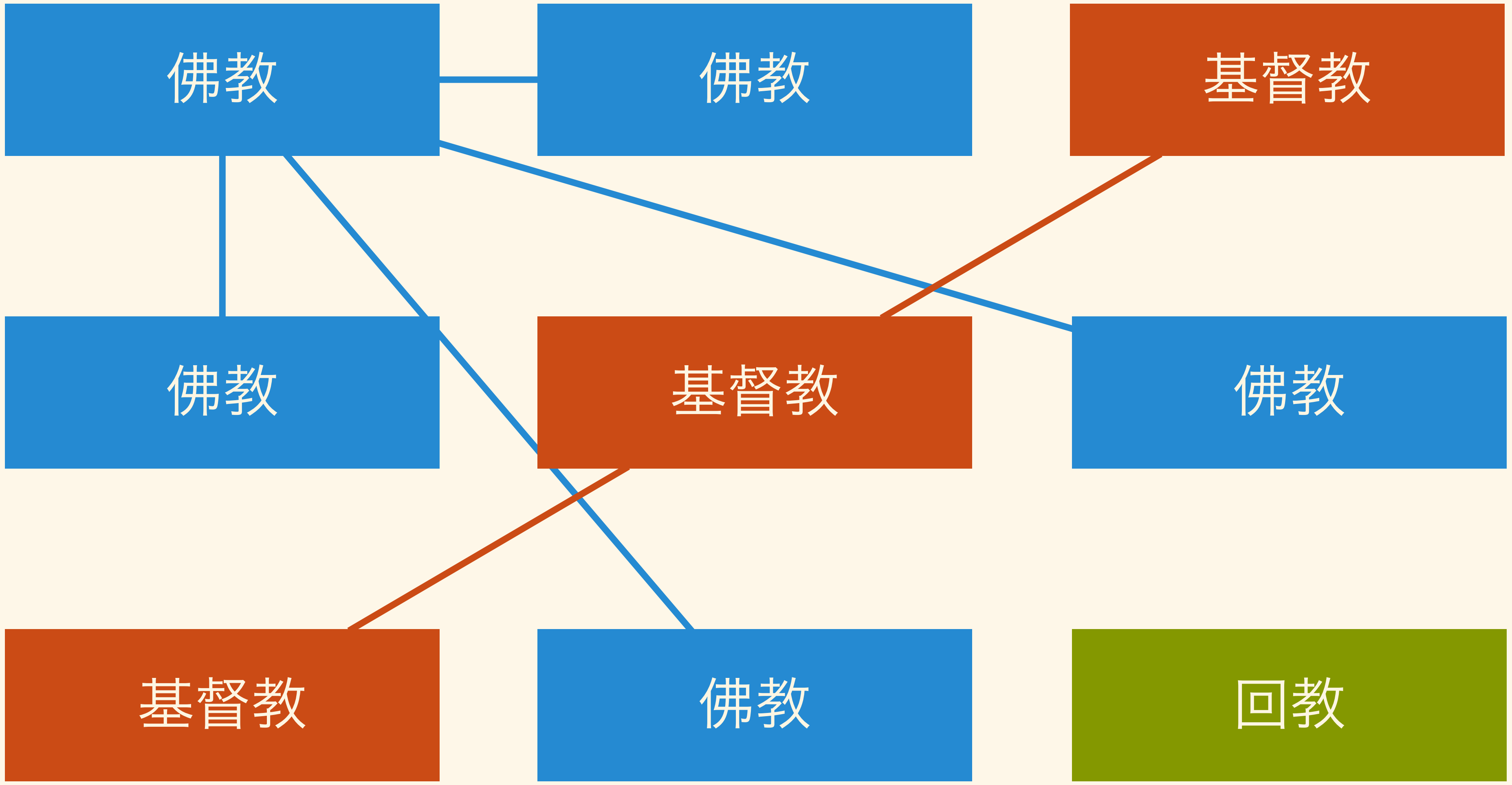
10583 - Ubiquitous Religions



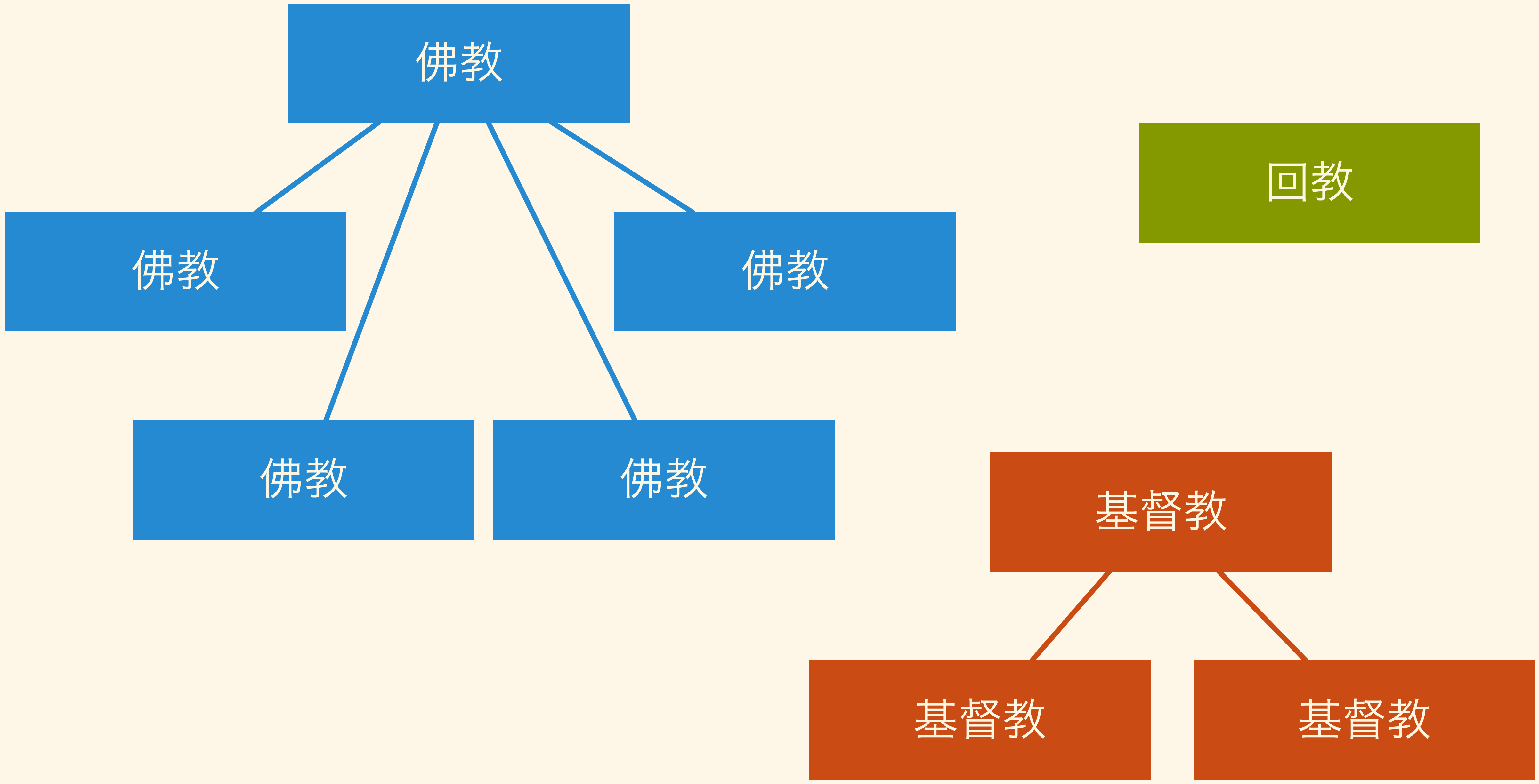
10583 - Ubiquitous Religions



10583 - Ubiquitous Religions



10583 - Ubiquitous Religions



Source Code

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 50010
int parent[ MAXN ], height[ MAXN ];

int main() {
   int n, t = 0, e, a, b;
   while ( scanf( “%d %d”, &n, &e ) && n && e ) {
      for ( int i = 1; i <= n; ++i )
         parent[ i ] = i;
         height[ i ] = 1;
      for ( int i = 0; i < e; ++i ) {
         scanf( “%d %d”, &a, &b );
         unionSet( findSet( a ), findSet( b ) );
      }
      int ret = 0;
      for ( int i = 1; i <= n; ++i )
         if ( parent[ i ] == i )
            ++ret;
      printf( “Case %d: %d\n”, ++t, ret );
   }
   return 0;
}
```

Practice Now

[UVa] 10608 - Friends

Thank You for Your Listening.