NCKU Programming Contest Training Course
Course 2-3
2016/03/02

Chien-Wen Chen
ai281918@gmail.com

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

made by electron & free999 & ai281918

Disjoint Set

- Disjoint Set
  - We have a collection of disjoint sets of elements. Each set is identified by a representative element. We want to perform union operations, and tell which set something is in. This is useful in a minimum spanning tree algorithm and many other applications. Formally, we have the following operations.
- **Basic Operation**
  - MAKE-SET(x) ：Create new set {x} with representative x.
  - UNION(x,y)：x and y are elements of two sets. Remove these sets and add their union. Choose a representative for it.
  - FIND-SET(x)：return the representative of the set containing x.

Disjoint Set

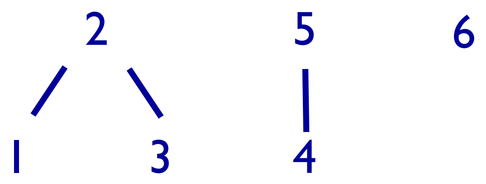
- Example

MAKE-SET(1)	{1}	
MAKE-SET(2)	{2}	
MAKE-SET(3)	{3}	
MAKE-SET(4)	{4}	
FIND(3)	**(returns 3)**	
FIND(2)	**(returns 2)**	
UNION(1,2)		{1,2}
FIND(2)	**(returns 1)**	
FIND(1)	**(returns 1)**	
UNION(4,3)		{3,4}
FIND(4)	**(returns 4)**	
FIND(3)	**(returns 4)**	
UNION(1,3)		{1,2,3,4}
FIND(2)	**(returns 1)**	
FIND(1)	**(returns 1)**	
FIND(4)	**(returns 1)**	
FIND(3)	**(returns 1)**	

Disjoint Set

- Forest Implementation

Here we represent each set as a tree, and the representative is the root . For example, the following forest represents the set {1,2,3}, {4,5}, {6} :

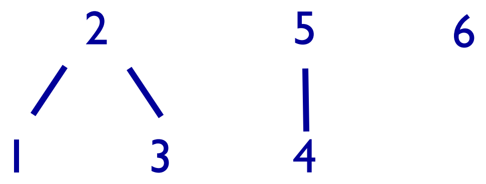


Implementation

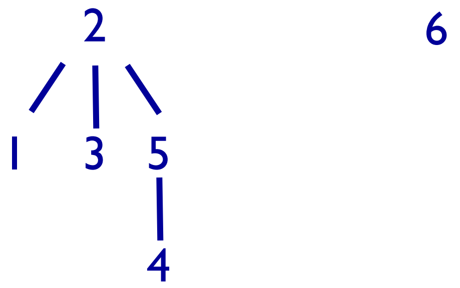
MAKE-SET(x)	Create a tree
FIND-SET(x)	Return the root
UNION(x,y)	Combine two trees

Disjoint Set

- Forest Implementation



Thus we would get the following form UNION(1,4)



This representation does not improve the running time in the worst case over the linked list representation.

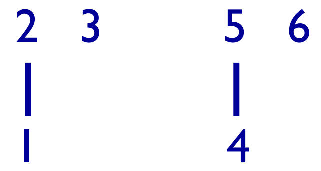
Disjoint Set

- Example

MAKE-SET(1) … MAKE-SET(6)

1 2 3 4 5 6 RANKS = 0

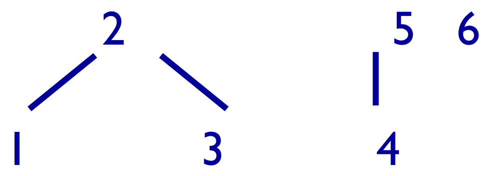
UNION(1,2) UNION(4,5)



RANK(2)=1

RANK(5)=1

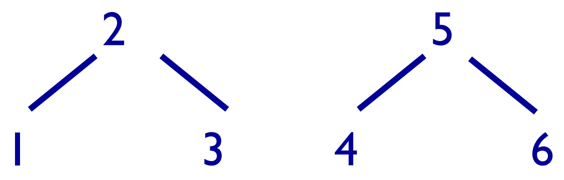
UNION(1,3)



RANK(2)=1

RANK(5)=1

UNION(5,6)



RANK(2)=1

RANK(5)=1

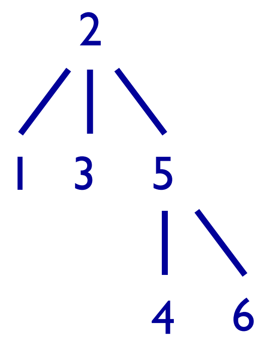
Disjoint Set

- Path Compaction and Rank
  - These are refinements of the forest representation which make it significantly faster
    - FIND-SET: Do path compression
    - UNION: Use ranks
  - "Path compression" means that when we do FIND-SET(X), we make all nodes encountered point directly to the representative element for x. Initially, all elements have rank 0. The ranks of representative elements are updated so that if two sets with representatives of the same rank are unioned, then the new representative is incremented by one.

Disjoint Set

- Example

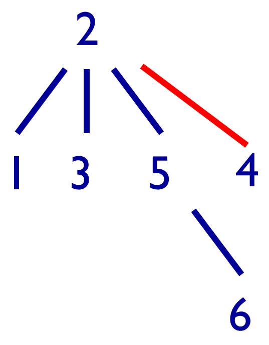
UNION(4,3)



RANK(2)=2

RANK(5)=1

FIND(4) (path compression)

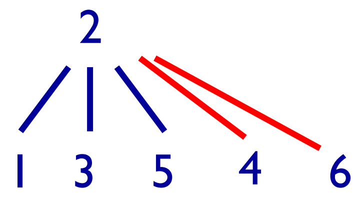


Disjoint Set

- Example

FIND(3) no change

FIND(6) (path compression)



Disjoint Set

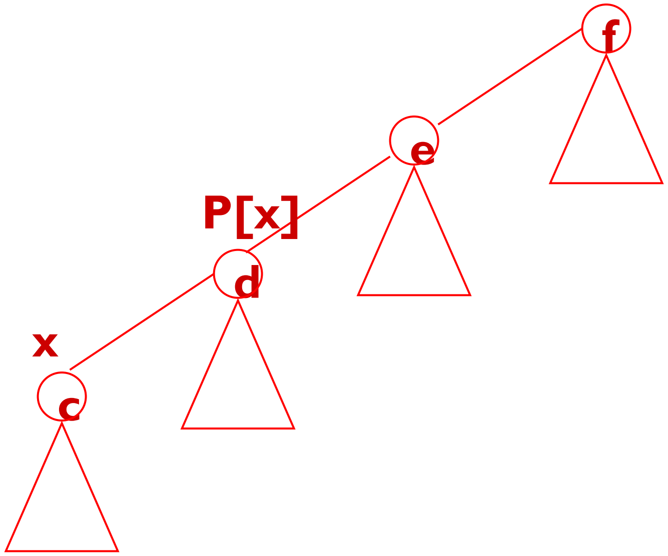
- MakeSet and Union

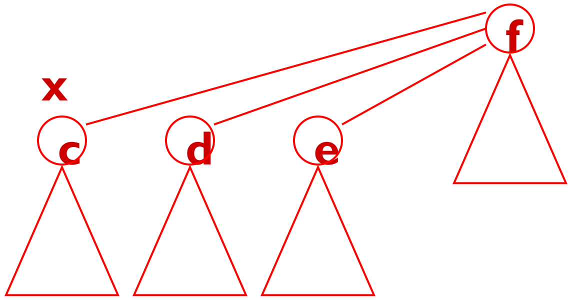
```
void MakeSet(int x)
{
   p[x]    = x;
   rank[x] = 0;
}

void Union(int x,int y)
{
    Link(FindSet(x),FindSet(y));
}
```

Disjoint Set

- FindSet

```
int FindSet(int x)
{
    if(x!=p[x])
        p[x] = FindSet(p[x]);
    return p[x];
}
```

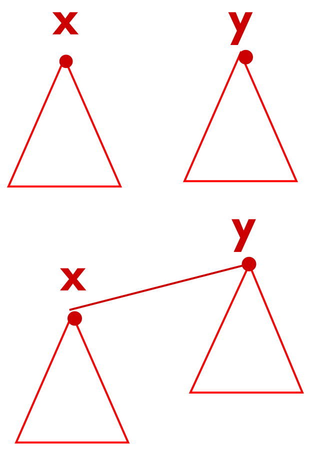




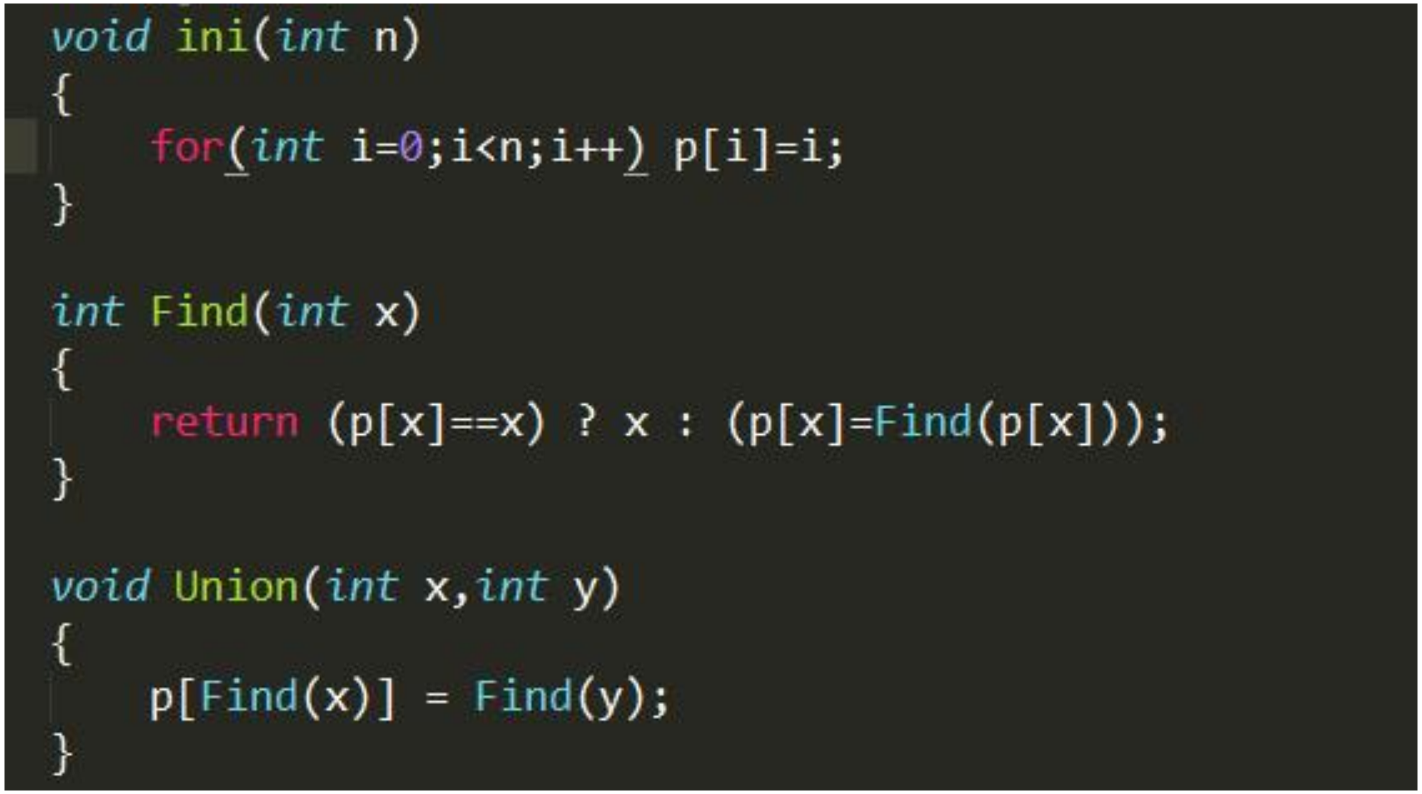
Disjoint Set

- Link

```
void Link(int x,int y)
{
    if(rank[x]>rank[y])
        p[y] = x;
    else
    {
        p[x] = y;
        if(rank[x]==rank[y])
            rank[y]++;
    }
}
```



Disjoint Set



```
void ini(int n)
{
    for(int i=0;i<n;i++) p[i]=i;
}

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

void Union(int x,int y)
{
    p[Find(x)] = Find(y);
}
```

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.

Example - 1

UVa 10583 - Ubiquitous Religions

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.

Example - 2

UVa 10685 - Nature

Problem Description

In nature, there are alimentary chains. At the basis of this chain, we generally have the vegetals. Small animals eat those vegetals and bigger animals eat the smaller. There can be cycles in the chain, as when some animal dies he starts a decomposition process which will transform its body into minerals that are a source of energy for the vegetals. In this problem you will have to find the largest alimentary chain for a given group of creatures. You can consider that if A is predator of B then they are in the same chain.

Example - 2

UVa 10685 - Nature

Input

The input file contains several input sets. The description of each set is given below: Each set starts with two integers C (1 ≤ C ≤ 5000), the number of creatures, and R (0 ≤ R ≤ 5000), the number of relations. Follow C lines with the names of the creatures, each consisting of lower case letters (a, b, . . . , z). No name is longer than 30 letters. Then there will be R lines describing the relations. Each line will have 2 names of creatures, meaning that the second creature is a predator of the first one. You can assume that no creature is a predator of himself. Input is terminated by a set where C = R = 0. This set should not be processed. There is a blank line beteween two input sets.

Output

For each input set produce one line of output, the size of the largest alimentary chain.

Homework 6

Total 6 Problems

- uva (5)
  - 793, 879, 10583, 10685, 11987
- poj(1)
  - 1703

Thank for Your Attention