银河英雄传说

考查内容：

并查集

题目分析：

考虑到题目让维护的操作与并查集非常类似，我们用并查集维护操作。

首先要实现的功能是判断两个节点是否属于同一列，用一个f数组记录每个节点所属列的列首，按照并查集的方式合并和查询即可。

对于查询两个节点之间距离的功能，需要再维护两个数组deep和tail，deep表示该元素在所在的这列的位置（从0开始计算），tail代表该元素所在列的列尾，很显然，deep[tail]+1=该列大小。

由于每列的f是每个元素都可以查询到的，所以维护tail的时候，只需要维护每列列首的tail即可，如果将A列合并到B列的末尾，那么tail[f_B]=tail[f_A]；将A列合并到B列末尾时，假设A列上所有元素的f和deep都是正确的，那么f和deep合并后是怎样变化的呢？设原先A列中有一个元素i，则deep[i]=deep[i]+f[i],f[i]=find(f[i])，故，只要f和deep同步，在find时就可以直接将祖先的deep加给自己由于deep从0开始计算，所以不会出现无限增多的情况。

代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/*************************
*创建时间：2018 08 03
*文件类型：源代码文件
*题目来源：COGS
*当前状态：已通过
*备忘录：并查集
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
using namespace std;
namespace IO
{
```

```
#define SIZE 1
int T=0,end=0;
char s[SIZE+1];
int in()
{
    T=0;
    return end=fread(s,1,SIZE,stdin);
}
int read(int &a)
{
    int i=a=0,flag=1;
    while(1)
    {
        if(T==end&&!in())return i;
        if(s[T]==' '||s[T]=='\n'||s[T]=='\r')
        {
            if(i)return i;
        }
        else if(s[T]=='-')
        {
            flag=-1;
        }
        else
        {
            a=a*flag*10+s[T]-'0';
            ++i;
        }
        ++T;
    }
}
int read(char &a)
{
    a=0;
    while(!a)
    {
        if(T==end&&!in())   return 0;
        if(s[T]!=' '&&s[T]!='\n'&&s[T]!='\r')a=s[T];
        ++T;
    }
    return 1;
```

```
    }
}
#define MAXN 30001
int T,f[MAXN],tail[MAXN],deep[MAXN];
/* Variable explain:

*/
void read()
{
    // freopen("galaxy.in","r",stdin);
    // freopen("galaxy.out","w",stdout);
    IO::read(T);
    for(int i=1;i<=30000;++i)f[i]=tail[i]=i;
    return;
}
int find(int x)
{
    if(x==f[x])return x;
    int fx=find(f[x]);
    // printf("f[%d]=%d\ndeep[%d]:%d-->%d\n",x,fx,x,deep[x],deep[x]+deep[f[x]]);
    deep[x]+=deep[f[x]];
    return f[x]=fx;
}
void merge(int a,int b)
{
    a=find(a),b=find(b);
    f[a]=b;
    find(tail[b]);
    deep[a]=deep[tail[b]]+1;
    // printf("tail=%d\ndeep[%d]=%d\n",tail[b],a,deep[tail[b]]+1);
    tail[b]=tail[a];
    return;
}
int main()
{
    read();
    while(T--)
    {
        char ls1;
```

```
        int ls2,ls3;
        IO::read(ls1);
        IO::read(ls2);
        IO::read(ls3);
        if(ls1=='M')
        {
            merge(ls2,ls3);
        }
        else
        {
            printf("%d\n",find(ls2)==find(ls3)?max(abs(deep[ls2]-
deep[ls3])-1,0):-1);
            // printf("%d %d\n",deep[ls2],deep[ls3]);
        }
    }
    return 0;
}
/*
C 3 1
C 5 1
C 3 4
C 3 5
C 5 1
C 4 2

*/
```

<题目跳转> <查看代码>