集合

1. 并查集

**亲戚：**如果x和y是亲戚，y和z是亲戚，那么x和z也是亲戚。如果x和y是亲戚，那么x的亲戚都是y的亲戚，y的亲戚也都是x的亲戚。现在给出某个亲戚关系图，求任意给出的两个人是否具有亲戚关系。

#include<bits/stdc++.h>  
using namespace std;  
#define MAXN 5010  
int n,m,p,x,y;  
int fa[MAXN];  
//查询是否是同一个集合   
int find(int x){  
 if(x==fa[x])return x;  
 return fa[x]=find(fa[x]);  
}  
//连接两个集合   
void join(int c1,int c2){  
 int f1=find(c1),f2=find(c2);  
 if(f1!=f2)fa[f1]=f2;  
}  
int main(){  
 cin>>n>>m>>p;  
 for(int i=1;i<=n;++i)fa[i]=i;  
 for(int i=0;i<m;++i){  
 cin>>x>>y;  
 join(x,y);  
 }  
 for(int i=0;i<p;++i){  
 cin>>x>>y;  
 if(find(x)==find(y))  
 cout<<"yes"<<endl;  
 else  
 cout<<"no"<<endl;  
 }  
 return 0;  
}  
//6 5 3  
//1 2  
//1 5  
//3 4  
//5 2  
//1 3  
//1 4  
//yes  
//2 3  
//yes  
//5 6  
//no

二、Hash表

**字符串哈希：**给定N(N<=10000)个字符串，第i个字符串长度为Mi(Mi<=1500)，字符串内包含数字、大小写字母(大小写敏感)，请求出N个字符串中共有多少个不同的字符串。

**1.简化版问题：**

给定N个自然数，值域是[0,10^9]，求出这N个自然数中共有多少个不同的自然数。

(1)开放地址法：

#include<bits/stdc++.h>  
using namespace std;  
#define mod 233333  
int n,x,ans,a[mod+2];  
int main(){  
 cin>>n;  
 for(int i=1;i<=n;++i){  
 cin>>x;  
 x%=mod;  
 if(!a[x])a[x]=1,++ans;  
 }  
 cout<<ans<<endl;  
 return 0;  
}

(2)链地址法：

#include<bits/stdc++.h>  
using namespace std;  
#define mod 233333  
int n,x,ans;  
vector<int> linker[mod+2];  
inline void insert(int x){  
 for(int i=0;i<linker[x%mod].size();++i)  
 if(linker[x%mod][i]==x)  
 return;  
 linker[x%mod].push\_back(x);  
 ++ans;  
}  
int main(){  
 cin>>n;  
 for(int i=1;i<=n;++i){  
 cin>>x;  
 insert(x);  
 }  
 cout<<ans<<endl;  
 return 0;  
}

**2.字符串哈希代码如下：**

#include<bits/stdc++.h>  
using namespace std;  
#define MAXN 1510  
#define base 261  
#define mod 23333  
int n,ans;  
char s[MAXN];  
vector<string> linker[mod+2];  
inline void insert(){  
 int hash=1;  
 for(int i=0;s[i];++i)  
 hash=(hash\*111\*base+s[i])%mod;//计算出字符串的hash值  
 string t=s;  
 for(int i=0;i<linker[hash].size();++i)  
 //遍历hash值为当前字符串hash值的hash链表，以检查这个字符串是否已经存在   
 if(linker[hash][i]==t)  
 return;//如果找到了一个同样的字符串，那么新的这个字符串不计入答案   
 linker[hash].push\_back(t);//否则把这个字符串计入答案   
 ++ans;  
}  
int main(){  
 cin>>n;  
 for(int i=1;i<=n;++i)  
 cin>>s,insert();  
 cout<<ans<<endl;  
 return 0;  
}

1. 集合应用实例

**set的本质是红黑树(一种比较优秀的平衡二叉树)。**

1. set<int>ds：建立一个名字叫做ds的、元素类型为int的集合。
2. ds.insert(x)：在集合中插入一个元素，如果这个元素已经存在，则什么都不干。
3. ds.erase(x)：在集合中删除元素x，如果这个数不存在，则什么都不干。
4. ds.erase(it)：删除集合中地址为it的元素。
5. ds.end()：返回集合中最后一个元素的下一个元素的地址。不过这个很少直接使用，而是配合其他方法进行比较，以确认某个元素是否存在。
6. ds.find(x)：查询x在集合中的地址，如果这个数不存在，则返回ds.end()。
7. ds.lower\_bound(x)：查询不小于x的最小的数在集合中的地址，如果这个数不存在，则返回ds.end()。
8. ds.upper\_bound(x)：查询大于x的最小的数在集合中的地址，如果这个数不存在，则返回ds.end()。
9. ds.empty()：如果集合是空的，则返回1，否则返回0。
10. ds.size()：返回集合中元素的个数。

**map关联集合的本质也是一棵红黑树，可以看作一个下标可以是任意类型的数组。**

1. map<A,B>ds：建立一个名字叫做ds、下标类型为A，元素类型为B的映射表，例如map<string,int>就是一个将string映射到int的映射表。
2. ds[A]=B:把这个“数组”中下标为A的位置的值变成B，这里下标可以是任意类型，不一定限定为大于0的整数，比如map<string,string>ds，就可以进行ds[“kkksc03”]=“mascot”的操作。
3. ds[A]：访问这个“数组”中下标为A的元素，比如可以进行cout<<ds[“kkksc03”]<<endl;这样的操作。
4. ds.end()：返回映射表中最后一个元素的下一个元素的地址。这个很少直接单独使用，而是配合其他方法进行比较，以确认某个元素是否存在。
5. ds.find(x)：查询x在映射表中的地址，如果这个数不存在，则返回ds.end()。
6. ds.empty()：如果映射表是空的，则返回1，否则返回0。
7. ds.size()：返回映射表中的元素个数。

**pair的实现是一个[结构体](https://so.csdn.net/so/search?q=%E7%BB%93%E6%9E%84%E4%BD%93&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/sevenjoin/article/details/_blank)**

#include<bits/stdc++.h>  
using namespace std;  
int main(){  
 pair<int,double> p[3];  
 p[0].first =1;  
 p[0].second =2;  
 p[1].first =3;  
 p[1].second =4;  
 p[2].first =5;  
 p[2].second =6;  
 if(p[1].first>p[0].first){  
 int tmp=p[1].second;  
 p[1].second=p[0].second;  
 p[0].second=tmp;  
 }  
 for(int i=0;i<3;++i) {  
 cout<<p[i].first<<' '<<p[i].second<<endl;  
 }  
 return 0;  
}