122492508@qq.com pq是不可战胜的

总分: 100 / 100

编程题 总分: 100 / 100

7-1 Sexy Primes 答案正确 得分: 20 / 20

Sexy primes are pairs of primes of the form $(p, p + 6)$, so-named since "sex" is the Latin word for "six". (Quoted from http://mathworld.wolfram.com/SexyPrimes.html)

Now given an integer, you are supposed to tell if it is a sexy prime.

Input Specification:

Each input file contains one test case. Each case gives a positive integer N ($\leq 10^8$).

Output Specification:

For each case, print in a line `Yes` if N is a sexy prime, then print in the next line the other sexy prime paired with N (if the answer is not unique, output the smaller number). Or if N is not a sexy prime, print `No` instead, then print in the next line the smallest sexy prime which is larger than N.

Sample Input 1:

```
47
```

Sample Output 1:

```
Yes
41
```

Sample Input 2:

```
21
```

Sample Output 2:

```
No
23
```

```
/**
* 分析:
**/

#include <bits/stdc++.h>
using namespace std;

bool isPrime(int n) {
        if(n<=1)
                return false;
        int sqr=(int)sqrt(n*1.0);
        for(int i=2; i<=sqr; i++)
                if(n%i==0)
                        return false;
        return true;
}

bool isSexy(int n) {
        if(isPrime(n)) {
                if(isPrime(n-6)||isPrime(n+6))
```

```
                        return true;
                else
                        return false;
        } else
                return false;
}

int main() {
        int n;
        scanf("%d",&n);
        if(isSexy(n)) {
                printf("Yes\n");
                if(isPrime(n-6))
                        printf("%d\n",n-6);
                else
                        printf("%d\n",n+6);
        } else {
                printf("No\n");
                while(!isSexy(n))
                        n++;
                printf("%d\n",n);
        }
        return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	3 ms	416KB
1	答案正确	3 ms	460KB
2	答案正确	3 ms	540KB
3	答案正确	3 ms	512KB
4	答案正确	3 ms	512KB
5	答案正确	3 ms	368KB

7-2 Anniversary 答案正确 得分: 25 / 25

Zhejiang University is about to celebrate her 122th anniversary in 2019. To prepare for the celebration, the alumni association （校友会） has gathered the ID's of all her alumni. Now your job is to write a program to count the number of alumni among all the people who come to the celebration.

Input Specification:

Each input file contains one test case. For each case, the first part is about the information of all the alumni. Given in the first line is a positive integer N ($\le 10^5$). Then N lines follow, each contains an ID number of an alumnus. An ID number is a string of 18 digits or the letter `X`. It is guaranteed that all the ID's are distinct.

The next part gives the information of all the people who come to the celebration. Again given in the first line is a positive integer M ($\le 10^5$). Then M lines follow, each contains an ID number of a guest. It is guaranteed that all the ID's are distinct.

Output Specification:

First print in a line the number of alumni among all the people who come to the celebration. Then in the second line, print the ID of the oldest alumnus -- notice that the 7th - 14th digits of the ID gives one's birth date. If no alumnus comes, output the ID of the oldest guest instead. It is guaranteed that such an alumnus or guest is unique.

Sample Input:

```
5
372928196906118710
610481197806202213
440684198612150417
13072819571002001X
150702193604190912
6
530125197901260019
150702193604190912
```

```
220221196701020034
610481197806202213
440684198612150417
370205198709275042
```

Sample Output:

```
3
150702193604190912
```

```
/**
* 分析：
**/

#include <bits/stdc++.h>
using namespace std;
const int maxn = 100010;

int n,m,cnta=0,cntg=0,ans=0;
string s;
unordered_map<string,bool> isalumni;

struct guest {
        string id;
        string date;
} a[maxn], g[maxn];

bool cmp(guest a,guest b) {
        return a.date<b.date;
}

int main() {
        scanf("%d",&n);
        for(int i=0; i<n; i++) {
                cin>>s;
                isalumni[s]=true;
                a[cnta].id=s;
                a[cnta].date=s.substr(6,8);
                cnta++;
        }
        scanf("%d",&m);
        for(int i=0; i<m; i++) {
                cin>>s;
                if(isalumni.find(s)!=isalumni.end())
                        ans++;
                g[cntg].id=s;
                g[cntg].date=s.substr(6,8);
                cntg++;
        }
        sort(g,g+cntg,cmp);
        printf("%d\n",ans);
        if(ans!=0) {
                for(int i=0; i<cntg; i++) {
                        if(isalumni.find(g[i].id)!=isalumni.end()) {
                                printf("%s\n",g[i].id.c_str());
                                break;
                        }
                }
        } else
                printf("%s\n",g[0].id.c_str());
        return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	10 ms	12832KB
1	答案正确	61 ms	12832KB
2	答案正确	79 ms	12828KB
3	答案正确	472 ms	32432KB
4	答案正确	576 ms	32584KB

7-3 Telefraud Detection 答案正确 得分: 25 / 25

Telefraud（电信诈骗） remains a common and persistent problem in our society. In some cases, unsuspecting victims lose their entire life savings. To stop this crime, you are supposed to write a program to detect those suspects from a huge amount of phone call records.

A person must be detected as a suspect if he/she makes more than K short phone calls to **different** people everyday, but no more than 20% of these people would call back. And more, if two suspects are calling each other, we say they might belong to the same gang. A makes a **short** phone call to B means that the total duration of the calls from A to B is no more than 5 minutes.

Input Specification:

Each input file contains one test case. For each case, the first line gives 3 positive integers K (≤ 500, the threshold（阈值） of the amount of short phone calls), N ($\le 10^3$, the number of different phone numbers), and M ($\le 10^5$, the number of phone call records). Then M lines of one day's records are given, each in the format:

```
caller receiver duration
```

where `caller` and `receiver` are numbered from 1 to N, and `duration` is no more than 1440 minutes in a day.

Output Specification:

Print in each line all the detected suspects in a gang, in ascending order of their numbers. The gangs are printed in ascending order of their first members. The numbers in a line must be separated by exactly 1 space, and there must be no extra space at the beginning or the end of the line.

If no one is detected, output `None` instead.

Sample Input 1:

```
5 15 31
1 4 2
1 5 2
1 5 4
1 7 5
1 8 3
1 9 1
1 6 5
1 15 2
1 15 5
3 2 2
3 5 15
3 13 1
3 12 1
3 14 1
3 10 2
3 11 5
5 2 1
5 3 10
5 1 1
5 7 2
5 6 1
5 13 4
5 15 1
11 10 5
12 14 1
6 1 1
6 9 2
6 10 5
6 11 2
6 12 1
6 13 1
```

Sample Output 1:

```
3 5
6
```

Note: In sample 1, although `1` had 9 records, but there were 7 distinct receivers, among which `5` and `15` both had conversations lasted more than 5 minutes in total. Hence `1` had made 5 short phone calls and didn't exceed the threshold 5, and therefore is not a suspect.

Sample Input 2:

```
5 7 8
1 2 1
1 3 1
1 4 1
1 5 1
1 6 1
1 7 1
2 1 1
3 1 1
```

Sample Output 2:

```
None
```

```
/**
* 分析：给不同的人打超过k个短电话，只有不超过20%的人回应，短电话为5min以内的
*       先判断是不是嫌疑人，然后在通话记录中找有没有另一个嫌疑人，有的话加入团伙
**/

#include <bits/stdc++.h>
using namespace std;
const int maxn = 1010;

int k,n,m;
int father[maxn];
int tol[maxn][maxn];
set<int> to[maxn]; //播出的短电话
set<int> tos[maxn]; //播出的电话 统计嫌疑人用
vector<int> sus; //嫌疑人
vector<int> ans[maxn];

int findRoot(int x) {
        int y=x;
        while(x!=father[x])
                x=father[x];
        while(y!=father[y]) {
                int z=y;
                y=father[y];
                father[z]=x;
        }
        return x;
}

void Union(int a,int b) {
        int ra=findRoot(a);
        int rb=findRoot(b);
        if(ra!=rb) {
                if(ra>rb)
                        father[ra]=rb;
                else
                        father[rb]=ra;
        }
}

int main() {
        int a,b,d;
        scanf("%d%d%d",&k,&n,&m);
        for(int i=0; i<m; i++) {
                scanf("%d%d%d",&a,&b,&d);
```

```
                tol[a][b]+=d;
//              if(d<=5)
//                      to[a].insert(b);
                tos[a].insert(b);
        }
        for(int i=1; i<=n; i++) {
                for(auto j=tos[i].begin(); j!=tos[i].end(); j++) {
                        int a=i,b=*j;
                        if(tol[a][b]<=5)
                                to[a].insert(b);
                }
        }
        for(int i=1; i<=n; i++) {
                if(to[i].size()>k) {
                        int cnt=0; //计算回应数
                        for(auto j:to[i]) {
                                if(tos[j].find(i)!=tos[j].end())
                                        cnt++;
                        }
                        if(cnt*1.0/to[i].size()<=0.2)
                                sus.push_back(i);
                }
        }
        if(sus.size()!=0) {
//              int cnt=0; //团伙数
//              for(int i=0; i<sus.size(); i++) {
//                      int a=sus[i];
//                      if(id[a]==-1) {
//                              g[cnt].mem.insert(a);
//                              g[cnt].first=a;
//                              id[a]=cnt;
//                              cnt++;
//                      }
//                      for(int j=i+1; j<sus.size(); j++) {
//                              int b=sus[j];
//                              if(tos[a].find(b)!=tos[a].end()&&tos[b].find(a)!=tos[b].end()) {
//                                      g[id[a]].mem.insert(b);
//                                      id[b]=id[a];
//                              }
//                      }
//              }
//              sort(g,g+cnt,cmp);
//              for(int i=0; i<cnt; i++) {
//                      int l=0,len=g[i].mem.size();
//                      for(auto j:g[i].mem) {
//                              printf("%d",j);
//                              l++;
//                              if(l!=len)
//                                      printf(" ");
//                      }
//                      printf("\n");
//              }
                for(int i=1; i<=n; i++)
                        father[i]=i;
                for(int i=0; i<sus.size(); i++) {
                        for(int j=i+1; j<sus.size(); j++) {
                                int  a=sus[i],b=sus[j];
                                if(tos[a].find(b)!=tos[a].end()&&tos[b].find(a)!=tos[b].end()) {
                                        Union(a,b);
                                }
                        }
                }
                for(auto i:sus)
                        ans[findRoot(i)].push_back(i);
                for(int i=1;i<=n;i++) {
                        if(ans[i].size()!=0) {
                                int l=0,len=ans[i].size();
                                for(int j:ans[i]) {
                                        printf("%d",j);
                                        l++;
                                        if(l!=len)
                                                printf(" ");
                                        else
                                                printf("\n");
                                }
                        }
                }
        } else
                printf("None\n");
```

```
    return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	4 ms	596KB
1	答案正确	4 ms	496KB
2	答案正确	4 ms	496KB
3	答案正确	4 ms	552KB
4	答案正确	89 ms	11904KB

7-4 Structure of a Binary Tree 答案正确 得分: 30 / 30

Suppose that all the keys in a binary tree are distinct positive integers. Given the postorder and inorder traversal sequences, a binary tree can be uniquely determined.

Now given a sequence of statements about the structure of the resulting tree, you are supposed to tell if they are correct or not. A statment is one of the following:

- A is the root
- A and B are siblings
- A is the parent of B
- A is the left child of B
- A is the right child of B
- A and B are on the same level
- It is a full tree

Note:

- Two nodes are **on the same level**, means that they have the same depth.
- A **full binary tree** is a tree in which every node other than the leaves has two children.

Input Specification:

Each input file contains one test case. For each case, the first line gives a positive integer N (≤ 30), the total number of nodes in the binary tree. The second line gives the postorder sequence and the third line gives the inorder sequence. All the numbers in a line are no more than 10^3 and are separated by a space.

Then another positive integer M (≤ 30) is given, followed by M lines of statements. It is guaranteed that both `A` and `B` in the statements are in the tree.

Output Specification:

For each statement, print in a line `Yes` if it is correct, or `No` if not.

Sample Input:

```
9
16 7 11 32 28 2 23 8 15
16 23 7 32 11 2 28 15 8
7
15 is the root
8 and 2 are siblings
32 is the parent of 11
23 is the left child of 16
28 is the right child of 2
7 and 11 are on the same level
It is a full tree
```

Sample Output:

```
Yes
No
Yes
No
Yes
Yes
Yes
```

```
/**
* 分析:
**/

#include <bits/stdc++.h>
using namespace std;
const int maxn = 31;

int n,m,root;
int post[maxn],in[maxn];
string s;
unordered_map<int,int> id;

struct Node {
        int data,lchild,rchild;
        int parent;
        int level;
} node[maxn];

int create(int postl,int postr,int inl,int inr,int parent,int level) {
        if(postl>postr)
                return -1;
        int root=postr;
        node[root].data=post[postr];
        node[root].parent=parent;
        node[root].level=level;
        int k;
        for(k=inl; k<=inr; k++) {
                if(in[k]==post[postr])
                        break;
        }
        int numleft=k-inl;
        node[root].lchild=create(postl,postl+numleft-1,inl,k-1,root,level+1);
        node[root].rchild=create(postl+numleft,postr-1,k+1,inr,root,level+1);
        id[node[root].data]=root;
        return root;
}

bool isfull;
void isFull(int root) {
        int l=node[root].lchild,r=node[root].rchild;
        if((l==-1&&r!=-1)||(l!=-1&&r==-1))
                isfull=false;
        if(root==-1)
                return;
        isFull(l);
        isFull(r);
}

int main() {
        scanf("%d",&n);
        for(int i=0; i<n; i++)
                scanf("%d",&post[i]);
        for(int i=0; i<n; i++)
                scanf("%d",&in[i]);
        root=create(0,n-1,0,n-1,-1,0);
        scanf("%d",&m);
        getchar();
        while(m--) {
                getline(cin,s);
                if(s.find("root")!=-1) { // A is the root
                        string ts;
                        for(int i=0; i<s.size(); i++) {
                                if(s[i]!=' ')
                                        ts+=s[i];
                                else
```

```
                        break;
                }
                int temp=stoi(ts);
                if(temp==node[root].data)
                        printf("Yes\n");
                else
                        printf("No\n");
        } else if(s.find("siblings")!=-1) {
                string as,bs;
                int i;
                for(i=0; i<s.size(); i++) {
                        if(s[i]!=' ')
                                as+=s[i];
                        else
                                break;
                }
                for(; i<s.size(); i++) {
                        if(isdigit(s[i]))
                                bs+=s[i];
                }
                int a=stoi(as),b=stoi(bs);
                if(node[id[a]].parent==node[id[b]].parent)
                        printf("Yes\n");
                else
                        printf("No\n");
        } else if(s.find("parent")!=-1) {
                string as,bs;
                int i;
                for(i=0; i<s.size(); i++) {
                        if(s[i]!=' ')
                                as+=s[i];
                        else
                                break;
                }
                for(; i<s.size(); i++) {
                        if(isdigit(s[i]))
                                bs+=s[i];
                }
                int a=stoi(as),b=stoi(bs);
                if(node[id[b]].parent==id[a])
                        printf("Yes\n");
                else
                        printf("No\n");
        } else if(s.find("left")!=-1) {
                string as,bs;
                int i;
                for(i=0; i<s.size(); i++) {
                        if(s[i]!=' ')
                                as+=s[i];
                        else
                                break;
                }
                for(; i<s.size(); i++) {
                        if(isdigit(s[i]))
                                bs+=s[i];
                }
                int a=stoi(as),b=stoi(bs);
                if(node[id[b]].lchild==id[a])
                        printf("Yes\n");
                else
                        printf("No\n");
        } else if(s.find("right")!=-1) {
                string as,bs;
                int i;
                for(i=0; i<s.size(); i++) {
                        if(s[i]!=' ')
                                as+=s[i];
                        else
                                break;
                }
                for(; i<s.size(); i++) {
                        if(isdigit(s[i]))
                                bs+=s[i];
                }
                int a=stoi(as),b=stoi(bs);
                if(node[id[b]].rchild==id[a])
                        printf("Yes\n");
                else
                        printf("No\n");
        } else if(s.find("level")!=-1) {
```

```
                        string as,bs;
                        int i;
                        for(i=0; i<s.size(); i++) {
                                if(s[i]!=' ')
                                        as+=s[i];
                                else
                                        break;
                        }
                        for(; i<s.size(); i++) {
                                if(isdigit(s[i]))
                                        bs+=s[i];
                        }
                        int a=stoi(as),b=stoi(bs);
                        if(node[id[a]].level==node[id[b]].level)
                                printf("Yes\n");
                        else
                                printf("No\n");
                } else if(s.find("full")!=-1) {
                        isfull=true;
                        isFull(root);
                        if(isfull)
                                printf("Yes\n");
                        else
                                printf("No\n");
                }
        }
        return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	4 ms	384KB
1	答案正确	3 ms	352KB
2	答案正确	3 ms	496KB
3	答案正确	2 ms	356KB
4	答案正确	3 ms	356KB