# 离散数学大作业合集

### 作业1

```

#include <iostream>

using namespace std;

#include <cstdio>

void rela(int n){

int people[n][n];

// 初始化关系矩阵，自己的位置填入-1，其余为0

for(int i=0;i<n;++i){

for (int j = 0; j < n; ++j){

if (i==j){

people[i][j] = -1;

} else{

people[i][j] = 0;

}}}

// 将输入的有朋友关系的位置改为数值1

while(true){

int a = 0, b =-2;

scanf(", (%d,%d)", &a, &b);

if (b == -2)break;

else {

people[a - 1][b - 1] = 1;

people[b - 1][a - 1] = 1;

}}

// 判断关系

for (int i = 0; i < n; ++i) {

for (int j = i + 1; j < n; ++j) {

//位置为0的地方有朋友的朋友和其他情况

if (people[i][j]==0){

//筛选出朋友的朋友并输出结果

for (int k = 0; k < n; ++k) {

if (people[i][k]==1&&people[j][k]==1){

cout << "(" << i + 1 << "," << j + 1 << ")";

break;

}}}}}

// cout << endl;

// for (int i = 0; i < n; ++i){

// for (int j = 0; j < n; ++j) cout <<people[i][j] << "\t";

// cout << endl;}

}

int main() {

int x;

cin >> x;

rela(x);

return 0;

}

```

#### 作业2

![image-20220522105424507](C:\Users\Lenovo\AppData\Roaming\Typora\typora-user-images\image-20220522105424507.png)

java版

算法:

算法的话就类似于一种叫“查并集”的算法，意思就是一种树型的数据结构，用于处理一些不相交集合的合并及查询问题。. 它的思想是用一个数组表示了整片森林，树的根节点唯一标识了一个集合，我们只要找到了某个元素的的树根，就能确定它在哪个集合里。

我们根据这个算法就可以解出题目要求

```

import java.util.Scanner;

public class Main {

static int[] f;

static int[][] g;

public static int find(int x) {

if (x == f[x])

return x;

else

return f[x] = find(f[x]);

}

​ public static void merge(int x, int y) {

​ int fx = find(x);

​ int fy = find(y);

​ if (fx == fy)

​ return;

​ f[fx] = fy;

​ }

​

​ public static boolean same(int x, int y) {

​ return find(x) == find(y);

​ }

​

​ public static void main(String[] args) {

​ Scanner scanner = new Scanner(System.in);

​ while (scanner.hasNext()) {

​ String s = scanner.nextLine();

​ String[] si = s.split(",");

​ int n = Integer.parseInt(si[0]);

​ f = new int[n + 1];

​ g = new int[n + 1][n + 1];

​ for (int i = 1; i <= n; i++) {

​ for (int j = 1; j <= n; j++) {

​ g[i][j] = 0;

​ }

​ }

​ for (int i = 0; i <= n; i++)

​ f[i] = i;

​ for (int i = 1; i < si.length; i += 2) {

​ int a = Integer.parseInt(si[i].replace("<", "").replace(">", "").replace(" ", ""));

​ int b = Integer.parseInt(si[i + 1].replace("<", "").replace(">", "").replace(" ", ""));

​ merge(a, b);

​ g[a][b] = g[b][a] = 1;

​ }

​ for (int i = 1; i <= n; i++) {

​ for (int j = i + 1; j <= n; j++) {

​ if (g[i][j] != 1 && same(i, j)) {

​ System.out.print("<" + i + "," + j + ">");

​ }

​ }

​ }

​ System.out.println();

​ }

​ }

}

```

#### 作业3

![image-20220522111541881](C:\Users\Lenovo\AppData\Roaming\Typora\typora-user-images\image-20220522111541881.png)

![image-20220522111555741](C:\Users\Lenovo\AppData\Roaming\Typora\typora-user-images\image-20220522111555741.png)

![image-20220522111613926](C:\Users\Lenovo\AppData\Roaming\Typora\typora-user-images\image-20220522111613926.png)

```

#include <bits/stdc++.h>

#define maxn 100005

using namespace std;

const string s="DESTINYABCFGHJKLMOPQRUVWXZ";

char inv[maxn];

string ch;

int main()

{

getline(cin,ch); int len=ch.length();

for(int i=0;i<25;i++) inv[s[i]-'A']=i+'A';

for(int i=0;i<len;i++)

{

if(ch[i]<='z'&&ch[i]>='a') ch[i]-='a',ch[i]+='A';

if(ch[i]==' ') printf(" ");

else printf("%c",inv[ch[i]-'A']);

}

return 0;

}

```

#### 作业4

第一种:

```

#include "stdio.h"

int main()

{

int p;

int q;

int r;

int s;//定义四个变量作为四个人

for(p=0;p<=1;p++)

//1表示是某人做的,0表示不是,每个人都有做与不做两种情况,用循环进行遍历

for(q=0;q<=1;q++)

for(r=0;r<=1;r++)

for(s=0;s<=1;s++)

if(((r==1)+(q==0)+(r==0)+(p==1)==3)&&(p+q+r+s==1))

{

if(p==1)

printf("r");

else if(q==1)

printf("q干的");

else if(r==1)

printf("r干的");

else if(s==1)

printf("s干的");

}

​ return 0;

}

```

第二种:

```

#include<iostream>

using namespace std;

int main()

{

char thisman = '/0';

for(thisman='P' ; thisman < 'S';thisman++)

{

if(3 == ((thisman !='Q')+(thisman == 'P')+(thisman == 'R')+(thisman != 'R')))

{

cout<<thisman<<"\n";

}

}

return 0;

}

```

#### 作业5

```

#include <iostream>

#include <cmath>

#include <stack>

#include <deque>

#include <vector>

using namespace std;

const int MAXN=10000;

int sub\_len,node\_num;

bool edge[MAXN][MAXN];

class node

{

public:

int data[MAXN]={0},vis=0;

vector<int> start,end;

};

node v[MAXN];

stack<int> t;

deque<int> node\_stack;

void build\_node()

{

int num;

for (int i=0; i<node\_num; i++)

{

num=i;

while (!t.empty())

t.pop();

while (num)

{

t.push(num%2);

num/=2;

}

for (int j=sub\_len-1-(int)t.size(); j<sub\_len-1; j++)

{

v[i].data[j]=t.top();

t.pop();

}

for (int j=1; j<sub\_len-1; j++)

v[i].start.push\_back(v[i].data[j]);

for (int j=0; j<sub\_len-2; j++)

v[i].end.push\_back(v[i].data[j]);

}

}

void dfs(int n)

{

v[n].vis++;

node\_stack.push\_back(n);

if (v[n].start==v[n].end&&v[n].vis<2&&edge[n][n]==0)

{

edge[n][n]=true;

dfs(n);

return;

}

for (int i=0; i<node\_num; i++)

{

if (v[i].vis>=1||edge[n][i]==true||i==n||v[n].start!=v[i].end)

continue;

edge[n][i]=true;

dfs(i);

return;

}

for (int i=0; i<node\_num; i++)

{

if (v[i].vis>=2||edge[n][i]==true||i==n||v[n].start!=v[i].end)

continue;

edge[n][i]=true;

dfs(i);

}

}

void print\_sequence()

{

//cout<<"对应的布鲁因序列为：";

for (int i=0; i<sub\_len-1; i++) {

cout<<v[node\_stack.front()].data[i];

}

node\_stack.pop\_front();

while (node\_stack.size()>sub\_len-2) {

cout<<v[node\_stack.front()].data[sub\_len-2];

node\_stack.pop\_front();

}

// while (!node\_stack.empty()) {

// for (int i=0; i<sub\_len-1; i++)

// cout<<v[node\_stack.front()].data[i];

// cout<<" ";

// node\_stack.pop\_front();

// }

//打印节点顺序

}

int main()

{

//cout<<"输入子序列的长度：";

cin>>sub\_len;

if (sub\_len==1) {

cout<<"对应的布鲁因序列为：01"<<endl;

return 0;

}

node\_num=pow(2, sub\_len-1);

build\_node();

node\_stack.clear();

dfs(0);

// cout<<node\_stack.size()<<endl;

// for (int i=0; i<node\_num; i++) {

// cout<<v[i].vis<<" ";

// }

// cout<<endl;

//打印每个节点的情况

print\_sequence();

cout<<endl;

// for (int i=0; i<node\_num; i++) {

// for (int j=0; j<sub\_len-1; j++) {

// cout<<v[i].data[j];

// }

// cout<<" ";

// }

//打印节点序列

}

```