bfs queue

#include<stdio.h>//上下电梯 输入5 1 53 3 1 2 5 0

#include<string.h>

#include<queue>

using namespace std;

int st,endd,all;

struct lift

{

int num;

int step;

} ;

int np[205];

bool map[205];

int bfs()

{

lift a,next;

queue<lift> q;

a.num=st;

a.step=0;

q.push(a);

map[st]=1;

while(!q.empty())

{

a=q.front();

if(a.num==endd)

return a.step;

q.pop() ;

next=a;

next.num=a.num+np[a.num];

if(!map[next.num]&&next.num>0&&next.num<=all)

{

next.step=a.step+1;

map[next.num]=1;

q.push(next);

}

next.num=a.num-np[a.num];

if(!map[next.num]&&next.num>0&&next.num<=all)

{

next.step=a.step+1;

map[next.num]=1;

q.push(next);

}

}

return -1;

}

int main()

{

while(scanf("%d",&all),all)

{

int i;

memset(map,0,sizeof(map));

scanf("%d%d",&st,&endd);

for(i=1;i<=all;i++)

{

scanf("%d",&np[i]);

}

printf("%d\n",bfs());

}

}

Gcd

#include<stdio.h>

int gcd(int a,int b)

{

return !b? a : gcd(b,a%b);

}

int main()

{

int a,b;

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

{

printf("%d\n",gcd(a,b));

}

}

next\_permutation(a, a + 10)

#include<stdio.h>

#include<algorithm>

using namespace std;

int main()

{

int a[10];

long long ans=0;

for(int i=0;i<10;i++)

a[i]=i;

do

{

ans++;

// for(int i=0;i<10;i++)

// printf("%d",a[i]);//必do while

// printf("\n");

}

while(next\_permutation(a, a + 10)) ;//顺序过 看7行 一定注意范围

printf("%d",ans);

a[0]=3;a[1]=2;a[2]=1;

do

{

for(int i=0;i<3;i++)

printf("%d",a[i]);//必do while

printf("\n");

}

while(prev\_permutation(a, a + 3)) ;//逆序过 看19行

}

并查集

#include<iostream>

#include<cstring>

#include<cstdio>

#include<cstdlib>

using namespace std;

int father[50002],a,b,m,n,p;

int find(int x){

if(father[x]!=x)

father[x]=find(father[x]);

/\*

x代表例题中的人，father[x]中所存的数代表这一集合中所有人都与一个人有亲戚关系

相当于例题中第一个集合所有的元素都与第一个元素有亲戚关系

搜索时只要找元素所指向的father[x]=x的元素(即父元素)

然后比较两个元素的父元素是否相同就可以判断其关系

\*/

return father[x];

}

int main()

{

int i;

scanf("%d%d%d",&n,&m,&p);

for(i=1;i<=n;i++)

father[i]=i;

for(i=1;i<=m;i++)

{

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

a=find(a);

b=find(b);

father[a]=b;

}

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

{

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

a=find(a);

b=find(b);

if(a==b)

printf("Yes\n");

else

printf("No\n");

}

return 0;}

大小排序选择法1

#include<stdio.h>

int main()

{//小到大

int a[10],i,j,k,t;

for(i=0;i<10;i++)

scanf("%d",&a[i]);

for(i=0;i<10;i++)

{ k=i;

for(j=i+1;j<10;j++)

if(a[j]<a[k]) k=j;

if(k!=i)

{ t=a[i];a[i]=a[k];a[k]=t; }

}

for(i=0;i<10;i++)

printf("%d ",a[i]);

return 0 ;

}

扩展欧几里德算法1

#include<iostream>

#include<cstdio>

using namespace std;

int x,y,q;

void extend\_Eulid(int a,int b){

if(b==0){

x=1;y=0;q=a;

return;

}

extend\_Eulid(b,a%b);

int temp=x;

x=y;

y=temp-a/b\*y;

return ;

}

int main()

{

int a,b;

cin>>a>>b;

extend\_Eulid(a,b);

printf("%d=(%d)\*%d+(%d)\*%d\n",q,x,a,y,b);

return 0;

}

质数

#include<stdio.h>

int main()

{

int a,b,i=0,n;

int z[10000];

int ifz[10000];//判断是否素数

for(a=2;a<10000;a++)

{

for(b=2;b<a;b++)

{

if((a%b)==0) break;

};

if(b==a)

{

z[i]=a;

ifz[z[i]]=1;

i++;

} ;

}

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

printf("%d ",z[n]);

printf("%d\n",n) ;

// while(scanf("%d",&n),n)

// printf("%d\n",ifz[n]);

}

中国剩余定理111

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

using namespace std;

\_\_int64 x,y;

\_\_int64 extended\_euclid(int s,int t) //??????

{

\_\_int64 i,temp;

if(t==0)

{

x=1;

y=0;

return s;

}

else

{

i=extended\_euclid(t,s%t);

temp=x;

x=y;

y=temp-(s/t\*y);

return i;

}

}

int main()

{

\_\_int64 a1,a2,b1,b2,t,c,n,m,i,j;

while(scanf("%I64d",&t)!=EOF)

{

scanf("%I64d%I64d",&a1,&b1);

int ok=0;

for(i=0;i<t-1;i++)

{

scanf("%I64d%I64d",&a2,&b2);

if(ok) continue;

int e=extended\_euclid(a1,a2);

c=b2-b1;

if(c%e)

{

ok=1;

continue;

}

int q=a2/e;

x=(x\*c/e%q+q)%q; //????????

b1=a1\*x+b1; // b=x;

a1=a1\*a2/e; // a?a1?a2???????

}

if(ok) printf("-1\n");

else printf("%lld\n",b1);

}

return 0;

}

最小生成树Kruskal算法

#include <iostream>

#include <algorithm>

#include <cstring>

#include <cstdlib>

#include <cstdio>

#include <cmath>

using namespace std;

const int DMAXN = 105;

const int EMAXN = 1005;

int c, s, q, x, y, val;

int p[DMAXN], u[EMAXN], v[EMAXN], r[EMAXN], dis[EMAXN];

int cmp(const int i, const int j)

{

return dis[i] < dis[j];

}

int find(int i)

{

return p[i] == i ? i : (p[i] = find(p[i]));

}

void Kruskal()

{

int px, py;

for(int i=1; i<=c; i++)

p[i] = i;

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

{

int e = r[i];

int xx = find(u[e]);

int yy = find(v[e]);

if(xx != yy)

p[xx] = yy;

px = find(x); // ??x??

py = find(y); // ??y??

if(px == py) // ??????,?????????

{

val = dis[e];

return ;

}

}

}

int main()

{

int fx, fy, wi, ff = 0, t = 0;

while(scanf("%d%d%d", &c, &s, &q))

{

if(!c && !s&& !q)

break;

if(ff)

puts("");

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

{

scanf("%d%d%d", &fx, &fy, &wi);

u[i] = fx;

v[i] = fy;

dis[i] = wi;

r[i] = i;

}

sort(r, r+s, cmp);

printf("Case #%d\n", ++t);

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

{

val = -1;

scanf("%d%d", &x, &y);

Kruskal();

if(val > 0)

printf("%d\n", val);

else

printf("no path\n");

}

ff = 1;

}

return 0;

}

一年第几天

#include <stdio.h>

int main()

{

int day,month,year,sum,leap;

printf("\nplease input year monthday\n");

scanf("%d%d%d",&year,&month,&day);

printf("%d,%,d%d",year,month,day);

switch(month)/\*先计算某月以前月份的总天数\*/

{

case 1:

sum=0;break;

case 2:

sum=31;break;

case 3:

sum=59;break;

case 4:

sum=90;break;

case 5:

sum=120;break;

case 6:

sum=151;break;

case 7:

sum=181;break;

case 8:

sum=212;break;

case 9:

sum=243;break;

case 10:

sum=273;break;

case 11:

sum=304;break;

case 12:

sum=334;break;

default:

printf("data error");

break;

}

sum=sum+day; /\*再加上某天的天数\*/

if(year%400==0||(year%4==0&&year%100!=0))/\*判断是不是闰年\*/

leap=1;

else

leap=0;

if(leap==1&&month>2)/\*如果是闰年且月份大于2,总天数应该加一天\*/

sum++;

printf("It is the %d day.",sum);

return 0;

}