**变换队列：**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

int main()

{

int n;

while(scanf("%d",&n)!=EOF)

{

int a[100],maxhigh=0,b[100];

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

{

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

b[n-1-i]=a[i];

if(a[i]>maxhigh)

maxhigh=a[i];

}

int startA,startB;

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

{

if(a[i]>maxhigh/2)

{

startA=i;

break;

}

}

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

{

if(b[i]>maxhigh/2)

{

startB=i;

break;

}

}

int ans=0;

for(int i=startA,j=startB;2\*a[i]>=maxhigh&&i<n;i++,j++)

{

ans+=abs(a[i]-b[j]);

}

ans/=2;

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

{

if(a[i]>maxhigh/2)

continue;

else ans+=a[i];

}

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

}

//cout << "Hello world!" << endl;

return 0;

}

**第二题**

#include <iostream>

#include <stdio.h>

#define INF 0x7ffffff

#include <string.h>

#include <algorithm>

using namespace std;

struct E{

char a[500];

int len;

bool operator < (const E &A) const

{

return len<A.len;

}

}str[5];

int main()

{

for(int i=0;i<5;i++)

{

scanf("%s",str[i].a);

str[i].len=strlen(str[i].a);

}

sort(str,str+5);

// for(int i=0;i<5;i++)

// puts(str[i].a);

for(int i=0;i<5;i++)

{

puts(str[i].a);

if(str[i].len==str[i+1].len&&(i+1)<5)

continue;

else break;

}

for(int i=4;i>-1;i--)

{

puts(str[i].a);

if(str[i].len==str[i-1].len&&(i-1)>=0)

continue;

else break;

}

return 0;

}

**第四题（阈值）**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define N 100

struct E{

int x,y;

}edge[100];

int tree[N];

int findRoot(int x)

{

if(tree[x]==-1) return x;

else {

int tmp=findRoot(tree[x]);

tree[x]=tmp;

return tmp;

}

}

int distance1(int i,int j)

{

int dis=(edge[i].x-edge[j].x)\*(edge[i].x-edge[j].x)+(edge[i].y-edge[j].y)\*(edge[i].y-edge[j].y);

return dis;

}

int main()

{

int n,r;

while(scanf("%d%d",&n,&r)!=EOF)

{

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

{

tree[i]=-1;

scanf("%d%d",&edge[i].x,&edge[i].y);

}

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

{

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

{

//printf("(%d,%d)=%d\n",i,j,distance1(i,j));

if(distance1(i,j)<=r\*r)

{

int a=findRoot(i),b=findRoot(j);

if(a!=b)

tree[a]=b;

}

}

}

int ans=0;

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

if(tree[i]==-1)

ans++;

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

}

cout << "Hello world!" << endl;

return 0;

}

**课程选择：**

#include <iostream>

#include <stdio.h>

#include <vector>

#include <math.h>

#include <string.h>

using namespace std;

#define num 100

vector<int> conflict[num];//冲突组

int tree[num];

int dp[num];

int findRoot(int x)

{

if(tree[x]==-1)

return x;

else {

int tmp=findRoot(tree[x]);

tree[x]=tmp;

return tmp;

}

}

struct E{

int cnum;

int ci;

int pi;

}className[100];

void groupConflict(int tree[],int n,int &group)//分组

{

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

{

int x=className[i].cnum;

if(tree[x]==-1)

{

int flag=0;

for(int j=1;j<=group;j++)

if(conflict[j].front()==x)

flag=1;

if(flag==0)

{

group++;

conflict[group].push\_back(x);

}

}

else

{

int a=findRoot(x),flag=0;

for(int j=1;j<=group;j++)

{

if(conflict[j].front()==a)

{

flag=1;

conflict[j].push\_back(x);

//printf("group:%d+%d\n",group,x);

}

}

if(flag==0)

{

group++;

conflict[group].push\_back(a);

conflict[group].push\_back(x);

//printf("group:%d+%d %d\n",group,a,x);

}

}

}

}

void groupPack(int dp[],int n,int group,int money)

{

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

for(int j=money;j>=0;j--)

for(int k=0;k<conflict[i].size();k++)

{

int newp=conflict[i][k];

int nowPi=className[newp].pi,nowCi=className[newp].ci;

if(j-nowPi>=0)//必须加上！

dp[j]=max(dp[j],dp[j-nowPi]+nowCi);

}

printf("dp[%d]=%d\n",money,dp[money]);

}

int main()

{

int T;

while(scanf("%d",&T)!=EOF)

{

while(T--)

{

for(int i=1;i<100;i++)

conflict[i].clear();

int n,money,group=0;

scanf("%d%d",&n,&money);

printf("依次课程号，冲突课，CI,PI(冲突写0 为不冲突)\n");

memset(tree,-1,sizeof(tree));

memset(dp,0,sizeof(dp));

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

{

int a,b,aCi,aPi;

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

className[i].cnum=a;

className[i].ci=aCi;

className[i].pi=aPi;

a=findRoot(a);

if(b==0)//b=0 表示没有冲突 这门课

continue;

b=findRoot(b);

if(a!=b)

{

tree[a]=b;

}

}

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

{

int sp=className[i].cnum;

printf("(%d %d) ",sp,findRoot(sp));

}

groupConflict(tree,n,group);

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

groupPack(dp,n,group,money);

}

}

// cout << "Hello world!" << endl;

return 0;

}