Algorithm Codelet

TieWay59

October 16, 2019

Contents

1 图论

1.1 dijstra 优化版.txt

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

const int mod = (int) 1e9+7;

int n,m,dist[205],head[205],now;
struct edge{
        int to,next,val;
}e[20005];
void init(int x,int y,int v){
        e[++now].to=y,e[now].val=v,e[now].next=head[x],head[x]=now;
        e[++now].to=x,e[now].val=v,e[now].next=head[y],head[y]=now;
}

struct node{
        int id,val;
        node(){}
        node(int id,int val):id(id),val(val){}
        bool operator < (const node &x)const{
                return val > x.val;
        }
};

void dij(int x){
        dist[x]=0;
        priority_queue<node> q;
        q.push(node(x,0));
        while(!q.empty()){
                node u=q.top();
                q.pop();
                if(u.val!=dist[u.id])continue;
                for(int i=head[u.id];~i;i=e[i].next){
                        int v=e[i].to;
                        if(dist[v]-e[i].val>dist[u.id]){
                                dist[v]=dist[u.id]+e[i].val;
                                q.push(node(v,dist[v]));
                        }
                }
        }
}

int main()
{
    while(~scanf("%d%d",&n,&m))
    {

        if(n==0&&m==0)break;
        memset(head,-1,sizeof(head));
        fill(dist,dist+200,mod);
        now=0;
        int x,y,z;
```

```
        for(int i=1;i<=m;i++)
        {
            scanf("%d%d%d",&x,&y,&z);
            init(x,y,z);
        }
        dij(1);
        printf("%d\n",dist[n]);
    }
}
```

1.2 KM.txt

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int qwq=0x7fffffff;
int w[1000][1000];  //w 数组记录边权值
int line[1000],usex[1000],usey[1000],cx[1000],cy[1000];  //line 数组记录右边端点所连的左端点， usex，usey 数组记录是否曾访问过，也是判断是否在增广路上，cx，cy 数组就是记录点的顶标
int n,ans,m;  //n 左 m 右
bool find(int x){
    usex[x]=1;
    for (int i=1;i<=m;i++){
        if ((usey[i]==0)&&(cx[x]+cy[i]==w[x][i])){   //如果这个点未访问过并且它是子图里面的边
            usey[i]=1;
            if ((line[i]==0)||find(line[i])){   //如果这个点未匹配或者匹配点能更改
                line[i]=x;
                return true;
            }
        }
    }
    return false;
}
int km(){
    for (int i=1;i<=n;i++){  //分别对左边点依次匹配
        while (true){
            int d=qwq;
            memset(usex,0,sizeof(usex));
            memset(usey,0,sizeof(usey));
            if (find(i)) break;  //直到成功匹配才换下一个点匹配
            for (int j=1;j<=n;j++){
                if (usex[j]){
                    for (int k=1;k<=m;k++)
                    if (!usey[k]) d=min(d,cx[j]+cy[k]-w[j][k]);  //计算 d 值
                }
            }
            if (d==qwq) return -1;
            for (int j=1;j<=n;j++)
                if (usex[j]) cx[j]-=d;
            for (int j=1;j<=m;j++)
                if (usey[j]) cy[j]+=d;      //添加新边
        }
```

```
    }
    ans=0;
    for (int i=1;i<=m;i++)
    ans+=w[line[i]][i];
    return ans;
}
int main(){
    while (~scanf("%d%d",&n,&m)){
        memset(cy,0,sizeof(cy));
        memset(w,0,sizeof(w));
        memset(cx,0,sizeof(cx));
        for (int i=1;i<=n;i++){
            int d=0;
            for (int j=1;j<=n;j++){
                scanf("%d",&w[i][j]);
                d=max(d,w[i][j]);   //此处顺便初始化左边点的顶标
            }
            cx[i]=d;
        }
        memset(line,0,sizeof(line));
        printf("%d\n",km());
    }
    return 0;
}

匈牙利算法 https://blog.csdn.net/cillyb/article/details/55511666
KM 算法 http://www.cnblogs.com/wenruo/p/5264235.html
```

1.3 scc+ 缩点.txt

```
#include<stdio.h>
#include<iostream>
#include<string.h>
using namespace std;
typedef long long ll;
const int N=2e3+5;

int head[N],head2[N],cnt;
int indegree[N],outdegree[N];

struct node{
    int v,nxt;
}edge[N<<2],edge2[N<<2];


void addedge(int u,int v){
    edge[++cnt].v=v;
    edge[cnt].nxt=head[u];
    head[u]=cnt;
}

void addedge2(int u,int v){
    edge2[++cnt].v=v;
    edge2[cnt].nxt=head2[u];
    head2[u]=cnt;
```

```
}

int dfn[N],low[N],st[N],scc[N],top,tt,sccid;

void tarjan(int u){
    if(dfn[u])return;
    low[u]=dfn[u]=++tt;
    st[++top]=u;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!dfn[v]){
            tarjan(v);
            low[u]=min(low[u],low[v]);
        }
        else if(!scc[v]){
            low[u]=min(low[u],dfn[v]);
        }
    }
    if(dfn[u]==low[u]){
        sccid++;
        while(top){
            int now=st[top--];
            scc[now]=sccid;
            if(now==u)break;
        }
    }
}

int main(){
    int n;
    scanf("%d",&n);
    memset(head,-1,sizeof(head));
    for(int i=1;i<=n;i++){
        int v;
        while(true){
            scanf("%d",&v);
            if(!v)break;
            addedge(i,v);
        }
    }
    for(int i=1;i<=n;i++){
        tarjan(i);
    }
    memset(head2,-1,sizeof(head2));
    cnt=0;
    for(int i=1;i<=n;i++){
        for(int j=head[i];~j;j=edge[j].nxt){
            int v=edge[j].v;
            if(scc[i]==scc[v])continue;
            addedge2(scc[i],scc[v]);
            //printf("%d %d\n",scc[i],scc[v]);
            indegree[scc[v]]++;
            outdegree[scc[i]]++;
        }
    }
    int ans1=0,ans2=0,tp1=0,tp2=0;
```

```
    for(int i=1;i<=sccid;i++){
        if(!indegree[i]){
            ans1++;
            tp1++;
        }
        if(!outdegree[i]){
            tp2++;
        }
    }
    ans2=max(tp1,tp2);
    if(sccid==1){
        printf("1\n0\n");
    }
    else{
        printf("%d\n%d\n",ans1,ans2);
    }
    return 0;
}
```

1.4 tarjan.txt

```
有向边 tarjan

void tarjan(int root){
        dfn[root]=low[root]=++index;
        st[++top]=root;
        gson(i,root){
                int v=edge[i].v;
                if(!dfn[v]){
                        tarjan(v);
                        low[root]=min(low[root],low[v]);
                }
                else if(!scc[v])low[root]=min(low[root],dfn[v]);
        }
        if(low[root]==dfn[root]){
                sccnum++;
                for(;;){
                        int x=st[top--];
                        scc[x]=sccnum;
                        if(x==root)break;
                }
        }
}

无向图 tarjan

void tarjan(int root){
        dfn[root]=low[root]=++index;
        st[++top]=root;
        gson(i,root){
                int v=edge[i].v;
                if(ef[i])continue;
                ef[i]=ef[i^1]=1;
                if(!dfn[v]){
                        tarjan(v);
```

```
                        low[root]=min(low[root],low[v]);
                        if(dfn[root]<low[v]){
                                //桥
                        }
                }
                else low[root]=min(low[root],dfn[v]);
        }
        if(low[root]==dfn[root]){
                bccnum++;
                for(;;){
                        int x=st[top--];
                        bcc[x]=bccnum;
                        if(x==root)break;
                }
        }
}
```

1.5 一些结论.txt

```
最小点覆盖数 = 最大匹配数
最大独立集 = 顶点数 - 最大匹配数
最小路径覆盖数 = 顶点数 - 最大匹配数
```

1.6 匈牙利算法.txt

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
using namespace std;
#include<string.h>
int match[1005], vis[1005];
int n, m;
int mp[1005][1005];
bool dfs(int u)
{
    for(int v=1;v<=m;v++)
    if(mp[u][v]&&!vis[v])
    {
        vis[v]=true;
        if( match[v] ==-1 || dfs(match[v]))
        {
            match[v]=u;
            return true;
        }
    }
    return false;
}
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d", &n, &m);
```

```
        if(n>m) return 0*printf("NO\n");
        memset(mp, 0, sizeof(mp));
        for(int i=1;i<=n;i++)
        {
            int k,x;
            scanf("%d", &k);
            while(k--)
            {
                scanf("%d", &x);
                mp[i][x] = 1;
            }
        }
        int sum=0;
        memset(match,-1,sizeof(match));
        for (int i=1; i<=n; i++)
        {
            memset(vis,0,sizeof(vis));
            if (dfs(i)) sum++;
        }
        if(sum==n)printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```

1.7 最大流.txt

```
const int N=505;

const int MAXN = 1<<26;
struct Edge{
    int u,v,c;
    int nxt;
}edge[N*N];

int n,m;
int head[N],edn;
int d[N];
int sp,tp;

int to[N];

void add_edge(int u,int v,int c)
{
    edge[edn].u=u; edge[edn].v=v; edge[edn].c=c;
    edge[edn].nxt=head[u]; head[u]=edn++;

    edge[edn].u=v; edge[edn].v=u; edge[edn].c=0;
    edge[edn].nxt=head[v]; head[v]=edn++;
}
int bfs()
{
    queue <int> q;
    memset(d,-1,sizeof(d));
    d[sp]=0;
```

```
    q.push(sp);
    while(!q.empty())
    {
        int cur=q.front();
        q.pop();
        for(int i=head[cur];i!=-1;i=edge[i].nxt)
        {
            int v=edge[i].v;
            if(d[v]==-1 && edge[i].c>0)
            {
                d[v]=d[cur]+1;
                q.push(v);
            }
        }
    }
    return d[tp] != -1;
}
int dfs(int a,int b)
{
    int r=0;
    if(a==tp)return b;
    for(int i=head[a];i!=-1 && r<b;i=edge[i].nxt)
    {
        int v=edge[i].v;
        if(edge[i].c>0 && d[v]==d[a]+1)
        {
            int x=min(edge[i].c,b-r);
            x=dfs(v,x);
            to[a]=v;
            r+=x;
            edge[i].c-=x;
            edge[i^1].c+=x;
        }
    }
    if(!r)d[a]=-2;
    return r;
}

int dinic(int sp,int tp)
{
    int total=0,t;
    while(bfs())
    {
        while(t=dfs(sp,MAXN))
            total+=t;
    }
    return total;
}
```

1.8 点 bcc.txt

```
#include<stdio.h>
#include<iostream>
#include<string.h>
using namespace std;
```

```cpp
typedef long long ll;
const int N=1e5+3;

int head[N],cnt;
int ans1,ans2;
struct node{
    int u,v,nxt;
}edge[N<<1];

void addedge(int u,int v){
    edge[++cnt].v=v;
    edge[cnt].u=u;
    edge[cnt].nxt=head[u];
    head[u]=cnt;
}

int dfn[N],low[N],st[N],bcc[N],bj[N],top,tt,bccid;
bool vis[N<<1],iscut[N<<1];

void tarjan(int u,int fa){
    dfn[u]=low[u]=++tt;
    //新点初始化
    int child=0;
    //初始节点需要两个以上儿子且 dfn[root]<=low[v] 才是割点
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(vis[i])continue;
        vis[i]=vis[i^1]=1;
        st[++top]=i;//边入栈
        if(!dfn[v]){
            child++;
            tarjan(v,u);
            low[u]=min(low[u],low[v]);
            if(dfn[u]<low[v])ans1++;//判桥
            if(dfn[u]<=low[v]){
                iscut[u]=1;
                bccid++;
                int num1=0,num2=0;//记录点 bcc 中边的数量和点的数量
                for(;;){
                    num1++;
                    int j=st[top--];
                    if(bj[edge[j].v]!=bccid){bj[edge[j].v]=bccid;num2++;}
                    if(bj[edge[j^1].v]!=bccid){bj[edge[j^1].v]=bccid;num2++;}
                    bcc[(j>>1)+1]=bccid;//标记边所属的 bcc
                    if(i==j)break;
                }
                if(num1>num2)ans2+=num1;
            }
        }
        else low[u]=min(low[u],dfn[v]);
    }
    if(u==fa&&child<2)iscut[u]=0;
    //如果初始节点没有两个以上的儿子，标记清零 isCut[i]=1 表示该点是割点
}

void init(){
```

```
    memset(bcc,0, sizeof(bcc));
    memset(dfn,0, sizeof(dfn));
    memset(vis,0, sizeof(vis));
    memset(iscut,0, sizeof(iscut));
    memset(head,-1, sizeof(head));
    memset(bj,0, sizeof(bj));
    cnt=-1;
    top=tt=bccid=0;
    ans1=ans2=0;
}

int main(){
    int n,m;
    while (~scanf("%d%d",&n,&m)){
        if(!n&&!m)break;
        init();
        for(int i=1,u,v;i<=m;i++){
            scanf("%d%d",&u,&v);
            addedge(u,v);
            addedge(v,u);
        }
        for(int i=0;i<n;i++){
            if(!dfn[i]){
                tarjan(i,-1);
            }
        }
        printf("%d %d\n",ans1,ans2);
    }
    return 0;
}



//有自环时不加自环的边
//缩点方法：清空路径，枚举 E[] 数组中存储的路径，建立双向边
void tarjan(int root,int fa){
    dfn[root]=low[root]=++idx;
        //新点初始化
    int child=0;
//初始节点需要两个以上儿子且 dfn[root]<=low[v] 才是割点
    for(int i=head[u];~i;i=edge[i].nxt{
        int v=edge[i].v;
        if(ef[i])continue;
        ef[i]=ef[i^1]=1;
        st[++top]=i;//边入栈
        if(!dfn[v]){
            child++;
            tarjan(v,root);
            low[root]=min(low[root],low[v]);
            //if(dfn[root]<low[v]) 桥 ++
            if(dfn[root]<=low[v]){
//此点是割点，需注意初始节点要有两个儿子
                N++;
//注意这里 N++, 建数组时要注意开至少两倍大
               for(;;){
                    int j=st[top--];
```

```
//bj[] 数组用来标记节点所属的 bcc，割点会改变，无意义。E[] 存新图的边，esum 是其数量，
↪  tarjan 结束后建双向边
                    if(bj[edge[j].v]!=N){
                        bj[edge[j].v]=N;
                        E[++esum]=make_pair(edge[j].v,N);
                    }
                    if(bj[edge[j^1].v]!=N){
                        bj[edge[j^1].v]=N;
                        E[++esum]=make_pair(edge[j^1].v,N);
                    }
                    belong[(j>>1)+1]=N;//标记边所属的 bcc
                    if(i==j)break;
                }
            }
        }
        else low[root]=min(low[root],dfn[v]);
    }
    if(root==fa&&child<2)isCut[root]=0;
    //如果初始节点没有两个以上的儿子，标记清零 isCut[i]=1 表示该点是割点
}
```

1.9 费用流.txt

```
const   int oo=1e9;  //无穷
const   int mm=11111111;  //边
const   int mn=888888;  //点
int node,src,dest,edge;
int ver[mm],flow[mm],cost[mm],nex[mm];
int head[mn],dis[mn],p[mn],q[mn],vis[mn];
/** 这些变量基本与最大流相同，增加了
 cost 表示边的费用，
 p 记录可行流上节点对应的反向边
 */
void prepare(int _node,int _src,int _dest)  //预处理   点的个数  起点  终点
{
    node=_node,src=_src,dest=_dest;
    for(int i=0; i<node; i++)head[i]=-1,vis[i]=0;
    edge=0;
}
void addedge(int u,int v,int f,int c)
{
    ver[edge]=v,flow[edge]=f,cost[edge]=c,nex[edge]=head[u],head[u]=edge++;
    ver[edge]=u,flow[edge]=0,cost[edge]=-c,nex[edge]=head[v],head[v]=edge++;
}
/** 以上同最大流 */
/**spfa 求最短路，并用 p 记录最短路上的边 */
bool spfa()
{
    int i,u,v,l,r=0,tmp;
    for(i=0; i<node; ++i)dis[i]=oo;
    dis[q[r++]=src]=0;
    p[src]=p[dest]=-1;
    for(l=0; l!=r; (++l>=mn)?l=0:l)
        for(i=head[u=q[l]],vis[u]=0; i>=0; i=nex[i])
            if(flow[i]&&dis[v=ver[i]]>(tmp=dis[u]+cost[i]))
```

```
            {
                dis[v]=tmp;
                p[v]=i^1;
                if(vis[v]) continue;
                vis[q[r++]=v]=1;
                if(r>=mn)r=0;
            }
    return p[dest]>-1;
}
/** 源点到汇点的一条最短路即可行流，不断的找这样的可行流 */
int SpfaFlow()
{
    int i,ret=0,delta;
    while(spfa())
    {
        for(i=p[dest],delta=oo; i>=0; i=p[ver[i]])
            if(flow[i^1]<delta)delta=flow[i^1];
        for(i=p[dest]; i>=0; i=p[ver[i]])
            flow[i]+=delta,flow[i^1]-=delta;
        ret+=delta*dis[dest];
    }
    return ret;
}
```

1.10 边 bcc+ 缩点.txt

```
#include<stdio.h>
#include<iostream>
#include<string.h>
using namespace std;
typedef long long ll;
const int N=5e3+3;

int head[N],head2[N],cnt;
int indegree[N];

struct node{
    int v,nxt;
}edge[N<<2],edge2[N<<2];


void addedge(int u,int v){
    edge[++cnt].v=v;
    edge[cnt].nxt=head[u];
    head[u]=cnt;
}

void addedge2(int u,int v){
    edge2[++cnt].v=v;
    edge2[cnt].nxt=head2[u];
    head2[u]=cnt;
}

int dfn[N],low[N],st[N],bcc[N],top,tt,bccid;
bool vis[N<<1];
```

```
30
31  void tarjan(int u){
32      if(dfn[u])return;
33      low[u]=dfn[u]=++tt;
34      st[++top]=u;
35      for(int i=head[u];~i;i=edge[i].nxt){
36          int v=edge[i].v;
37          if(vis[i])continue;
38          vis[i]=vis[i^1]=true;
39          if(!dfn[v]){
40              tarjan(v);
41              low[u]=min(low[u],low[v]);
42              /*if(dfn[u]<low[v]){u-v 为桥
43              }*/
44          }
45          else{
46              low[u]=min(low[u],dfn[v]);
47          }
48      }
49      if(dfn[u]==low[u]){
50          bccid++;
51          while(top){
52              int now=st[top--];
53              bcc[now]=bccid;
54              if(now==u)break;
55          }
56      }
57  }
58
59  int main(){
60      int n,m;
61      scanf("%d%d",&n,&m);
62      memset(head,-1,sizeof(head));
63      cnt=-1;
64      for(int i=1;i<=m;i++){
65          int x,y;
66          scanf("%d%d",&x,&y);
67          addedge(x,y);
68          addedge(y,x);
69      }
70      for(int i=1;i<=n;i++){
71          tarjan(i);
72      }
73      //for(int i=1;i<=n;i++)printf("%d\n",bcc[i]);
74      memset(head2,-1,sizeof(head2));
75      cnt=-1;
76      for(int i=1;i<=n;i++){
77          for(int j=head[i];~j;j=edge[j].nxt){
78              int v=edge[j].v;
79              if(bcc[i]==bcc[v])continue;
80              addedge2(bcc[i],bcc[v]);
81              addedge2(bcc[v],bcc[i]);
82              //printf("%d %d\n",scc[i],scc[v]);
83              indegree[bcc[v]]++;
84              indegree[bcc[i]]++;
85          }
```

```
    }
    int ans=0;
    for(int i=1;i<=bccid;i++){
        if(indegree[i]==2){
            ans++;
        }
    }
    printf("%d\n",(ans+1)/2);
    return 0;
}
```

2 字符串

2.1 ac 自动机.txt

```
struct Tire{
    static const int NODENUM=(int)1e6+5,R=26;
    int nxt[NODENUM][R],fail[NODENUM],ed[NODENUM];
    //nxt 指针，fail 指针，ed：在某个结点串结束的数量
    int rt,tot;
    int newnode(){
        for(int i=0;i<R;i++)nxt[tot][i]=-1;
        ed[tot]=0;
        return tot++;
    }
    void init(){
        tot=0;
        rt=newnode();
    }
    void insert(char *s){//建字典树
        int now=rt,len=strlen(s);
        for(int i=0;i<len;i++){
            int val=s[i]-'a';
            if(nxt[now][val]==-1)nxt[now][val]=newnode();
            now=nxt[now][val];
        }
        ed[now]++;
    }
    void build(){//bfs 求解 fail 指针
        queue<int>q;
        fail[rt]=rt;
        for(int i=0;i<R;i++){
            if(nxt[rt][i]==-1)nxt[rt][i]=rt;
            else {
                fail[nxt[rt][i]]=rt;
                q.push(nxt[rt][i]);
            }
        }
        while(!q.empty()){
            int now=q.front();q.pop();
            for(int i=0;i<R;i++){
                if(nxt[now][i]==-1)nxt[now][i]=nxt[fail[now]][i];
                else {
                    fail[nxt[now][i]]=nxt[fail[now]][i];
```

```
                    q.push(nxt[now][i]);
                }
            }
        }
    }
    int query(char *s){//查询某个串所有前缀出现的次数
        int now=rt,res=0,len=strlen(s);
        for(int i=0;i<len;i++){
            int val=s[i]-'a';
            now=nxt[now][val];
            int tmp=now;
            while(tmp!=rt){
                res+=ed[tmp];
                ed[tmp]=0;
                tmp=fail[tmp];
            }
        }
        return res;
    }
}ac;
```

2.2 ekmp.txt

```
void get_next(char *a,int len){
        int k=0,i=1;
        next[0]=len;
        while(k<len && a[k] == a[k+1])++k;
        next[1]=k;
        k=1;
        while(++i<len){
                int maxr=k+next[k]-1;
                next[i]=min(next[i-k],max(maxr-i+1,0));
                while(i+next[i]<len && a[next[i]] == a[i+next[i]])++next[i];
                if(i+next[i]>k+next[k])k=i;
        }
}

void EKMP(char *a,char *b){
        int lena=strlen(a),lenb=strlen(b),k=0,i=0;
        get_next(a,lena);
        b[lenb]='*';//b[lenb] 重置为 a,b 里没出现的字符
        while(a[k] == b[k])++k;//如果没有重置 b[lenb], 应为 k<lenb && a[k] == b[k]
        extend[0]=k;
        k=0;
        while(++i<lenb){
                int maxr=k+extend[k]-1;
                extend[i]=min(next[i-k],max(maxr-i+1,0));
                while(a[extend[i]] == b[i+extend[i]])++extend[i];//如果没有重置
                ↪  b[lenb], 需判断 i+extend[i]<lenb
                if(i+extend[i]>k+extend[k])k=i;
        }
}
```

2.3 kmp.txt

```
int nxt[100005];
char s[100005],t[100005];

void kmp_next(char *T,int *nt){
    nt[0]=-1;
    for(int i=0,j=-1,m=strlen(T);i<m;){
        if(j==-1||T[i]==T[j]){
                i++,j++;
                        if(T[i]!=T[j])nt[i]=j;
                        else nt[i]=nt[j];
        }
        else j=nt[j];
    }
}

int kmp(char *S,char *T,int *nt){//返回 T 在 S 中出现几次
    kmp_next(T,nt);
    int ans=0,sn=strlen(S),tn=strlen(T);
    for(int i=0,j=0;i<sn;){
        if(j==-1||S[i]==T[j])i++,j++;
        else j=nt[j];
        if(j==tn)ans++;
    }
    return ans;
}
```

2.4 后缀数组.txt

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

//sa[x]：排名为 x 的后缀的第一个字符在原串中的下标
//rk[id]：在原串中第一个字符下标为 id 的后缀的排名
//h[i]=height[rk[i]]：排名为 i 的后缀和排名为 i-1 的后缀的公共前缀的长度
//w~ 数组为桶排序用的数组
const int MAXN =(int)1e6+10;
int wa[MAXN],wb[MAXN],wv[MAXN],we[MAXN],rk[MAXN];

int cmp(int *r,int a,int b,int l){return r[a]==r[b]&&r[a+l]==r[b+l];}
void build_sa(int *r,int *sa,int n,int m){
        int i,j,p,*x=wa,*y=wb,*t;
        for(i=0;i<m;i++)we[i]=0;
        for(i=0;i<n;i++)we[x[i]=r[i]]++;
        for(i=1;i<m;i++)we[i]+=we[i-1];
        for(i=n-1;i>=0;i--)sa[--we[x[i]]]=i;
        for(j=1,p=1;p<n;j*=2,m=p){
                for(p=0,i=n-j;i<n;i++)y[p++]=i;
                for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;
                for(i=0;i<n;i++)wv[i]=x[y[i]];
                for(i=0;i<m;i++)we[i]=0;
                for(i=0;i<n;i++)we[wv[i]]++;
                for(i=1;i<m;i++)we[i]+=we[i-1];
```

```
                for(i=n-1;i>=0;i--)sa[--we[wv[i]]]=y[i];
                for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
                x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
        }
}
int height[MAXN];
void calheight(int *r,int *sa,int n){
        int i,j,k=0;
        for(i=1;i<=n;i++)rk[sa[i]]=i;
        for(i=0;i<n;height[rk[i++]]=k){
                for(k?k--:0,j=sa[rk[i]-1];r[i+k]==r[j+k];k++);
        }
}
int sa[MAXN],a[MAXN];
char str[MAXN];
int main()
{
        scanf("%s",str);
        int n=strlen(str);
        for(int i=0;i<n;i++)a[i]=str[i];
        a[n]=0;
        build_sa(a,sa,n+1,128);
        calheight(a,sa,n);
        for(int i=1;i<=n;i++)printf("%d ",sa[i]+1);
        printf("\n");
        for(int i=2;i<=n;i++)printf("%d ",height[i]);
        printf("\n");
        return 0;
}
```

2.5 后缀自动机.txt

```
const int M=1000100;
char s[M];
int epos[M],len[M],nxt[M][26],link[M],f[M];
int w[M],q[M];
int n,T,K;

struct sam{
    int root,last,cnt;
    sam(){root=last=++cnt;}
    void insert(int c){
        int np=++cnt,p=last;last=np;
        epos[np]=1;len[np]=len[p]+1;
        for(;p&&!nxt[p][c];nxt[p][c]=np,p=link[p]);
        if (!p) link[np]=root;
        else if (len[nxt[p][c]]==len[p]+1) link[np]=nxt[p][c];
        else{
            int nq=++cnt,q=nxt[p][c];len[nq]=len[p]+1;
            memcpy(nxt[nq],nxt[q],sizeof nxt[q]);link[nq]=link[q];
            link[np]=link[q]=nq;
            for(;p&&nxt[p][c]==q;nxt[p][c]=nq,p=link[p]);
        }
    }
    void build(){
```

```
        scanf("%s",s+1);n=strlen(s+1);
        for(int i=1;i<=n;i++)insert(s[i]-'a');
    }
    void topsort(){
        for(int i=1;i<=cnt;i++)w[len[i]]++;
        for(int i=1;i<=n;i++)w[i]+=w[i-1];
        for(int i=1;i<=cnt;i++)q[w[len[i]]--]=i;
    }
}SAM;
```

2.6 回文树.txt

```
const int MAXN = 100005 ;
const int N = 26 ;

struct Palindromic_Tree {
    int next[MAXN][N] ;//next 指针，next 指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN] ;//fail 指针，失配后跳转到 fail 指针指向的节点
    int cnt[MAXN] ;//cnt[i] 表示节点 i 表示的本质不同的串的个数，需跑 count 函数
    int num[MAXN] ;//表示以节点 i 表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[MAXN] ;//len[i] 表示节点 i 表示的回文串的长度
    int S[MAXN] ;//存放添加的字符
    int last ;//指向上一个字符所在的节点，方便下一次 add
    int n ;//字符数组指针
    int p ;//节点指针
        int pos[MXAN] ;//某种本质的回文串的一个右端点
    int newnode ( int l ) {//新建节点
        for ( int i = 0 ; i < N ; ++ i ) next[p][i] = 0 ;
        cnt[p] = 0 ;
        num[p] = 0 ;
        len[p] = l ;
        return p ++ ;
    }

    void init () {//初始化
        p = 0 ;
        newnode (  0 ) ;
        newnode ( -1 ) ;
        last = 0 ;
        n = 0 ;
        S[n] = -1 ;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1 ;
    }

    int get_fail ( int x ) {//和 KMP 一样，失配后找一个尽量最长的
        while ( S[n - len[x] - 1] != S[n] ) x = fail[x] ;
        return x ;
    }

    void add ( int c ) {
        c -= 'a' ;
        S[++ n] = c ;
        int cur = get_fail ( last ) ;//通过上一个回文串找这个回文串的匹配位置
```

```
        if ( !next[cur][c] ) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
            int now = newnode ( len[cur] + 2 ) ;//新建节点
            fail[now] = next[get_fail ( fail[cur] )][c] ;//和 AC 自动机一样建立 fail 指针，以便失配后跳转
            next[cur][c] = now ;
            num[now] = num[fail[now]] + 1 ;
        }
        last = next[cur][c] ;
        cnt[last] ++ ;
        pos[last] = n;
    }

    void count () {
        for ( int i = p - 1 ; i >= 0 ; -- i ) cnt[fail[i]] += cnt[i] ;
        //父亲累加儿子的 cnt，因为如果 fail[v]=u，则 u 一定是 v 的子回文串!
    }
} ;
```

2.7 字典树.txt

```
int tree[maxn][30];
int tsize[maxn];
int tot;

void add_edge(char *s){
    int len=strlen(s);
    int rt=0;
    for(int i=0;i<len;i++){
        int xb=s[i]-'a';
        if(!tree[rt][xb])tree[rt][xb]=++tot;
        tsize[tree[rt][xb]]++;
        rt=tree[rt][xb];
    }
}

int query(char *s){//返回 s 是多少串的前缀
    int len=strlen(s);
    int rt=0;
    for(int i=0;i<len;i++){
        int xb=s[i]-'a';
        if(!tree[rt][xb])return 0;
        rt=tree[rt][xb];
    }
    return tsize[rt];
}
```

2.8 最小表示法.txt

```
// 循环同构中字典序最小的那个
char x[maxn];

int main(){
    int t=rd();
```

```
    while(t--){
        gets(x);
        int len=strlen(x);
        int i=0,j=1,k=0;
        while(i<len&&j<len&&k<len){
            int cmp=x[(i+k)%len]-x[(j+k)%len];
            if(!cmp){
                k++;
            }
            else{
                if(cmp>0)
                    i+=k+1;
                else
                    j+=k+1;
                if(i==j)
                    j++;
                k=0;
            }
        }
        int pos=min(i,j); // 解的位置
        printf("%d\n",pos+1);
    }
    return 0;
}
```

2.9 马拉车.txt

```
//abc -> *#a#b#a#\0
//下标 x->2*x+2
void manacher(char *s){
        int len=strlen(s);
        for(int i=len;i>=0;--i){
                s[i+i+2]=s[i];
                s[i+i+1]='#';
        }
        s[0]='*';
        int k=1,maxlen=0;
        for(int i=2;i<len+len+1;++i){
                int maxr=k+p[k]-1;
                p[i]=min(p[2*k-i],max(maxr-i+1,1));
                while(s[i-p[i]] == s[i+p[i]])++p[i];
                if(i+p[i]>k+p[k])k=i;
                if(p[i]>maxlen)maxlen=p[i];
        }
}
```

3 数据结构

3.1 st.txt

```
int mp[N];
int stmaxs[N][20],stmins[N][20];


```

```
void build(int n,)
{
    for (int i=1;i<=n;i++){
        stmaxs[i][0]=mp[i];
        stmins[i][0]=mp[i];
    }
    for (int j=1;(1<<j)<=n;j++)
        for (int i=1;i+(1<<j)-1<=n;i++){
            stmaxs[i][j]=max(stmaxs[i][j-1],stmaxs[i+(1<<(j-1))][j-1]);
            stmins[i][j]=min(stmins[i][j-1],stmins[i+(1<<(j-1))][j-1]);
        }
}

int query_max(int l,int r)
{
    int len=(int)(log( double(r-l+1)/log(2.0)));//算出区间的长度是 2 的几次
    return max(stmaxs[l][len],stmaxs[r-(1<<len)+1][len]);
}
```

3.2 主席树第 k 大.txt

```
#include<stdio.h>
#include<algorithm>
#include<map>
using namespace std;
#define N 100005
int num[N],ls[N];
int root[N];
int tsize[N*25],lchild[N*25],rchild[N*25];
int tot;


void update(int last,int cur,int l,int r,int k)
{
    tsize[cur]=tsize[last]+1;
    lchild[cur]=lchild[last];
    rchild[cur]=rchild[last];
    if(l==r)return;
    int mid=(l+r)>>1;
    if(k<=mid)update(lchild[last],lchild[cur]=++tot,l,mid,k);
    else update(rchild[last],rchild[cur]=++tot,mid+1,r,k);
}

int query(int last,int cur,int L,int R,int k)
{
    if(L==R)return L;
    int mid=(L+R)>>1;
    int lson=tsize[lchild[cur]]-tsize[lchild[last]];
    if(lson>=k)return query(lchild[last],lchild[cur],L,mid,k);
    else return query(rchild[last],rchild[cur],mid+1,R,k-lson);
}


int main()
{
```

```
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n,m;
        scanf("%d%d",&n,&m);
        for(int i=1;i<=n;i++)
            scanf("%d",&num[i]),ls[i]=num[i];
        sort(num+1,num+1+n);
        int sum=unique(num+1,num+1+n)-num-1;
        tot=0;
        for(int i=1;i<=n;i++)
                update(root[i-1],root[i]=++tot,1,sum,
                        lower_bound(num+1,num+1+sum,ls[i])-num);
        int l,r,k;
        for(int i=1;i<=m;i++)
        {
            scanf("%d%d%d",&l,&r,&k);
            printf("%d\n",num[query(root[l-1],root[r],1,sum,k)]);
        }
    }
}
```

4 数论

4.1 几何补充.txt

```
#define pi acos(-1.0)

typedef struct node
{
        int x;
        int y;
}point;
 //圆面积交
double AREA(point a, double r1, point b, double r2)
{
        double d = sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
        if (d >= r1+r2)
                return 0;
        if (r1>r2)
        {
                double tmp = r1;
                r1 = r2;
                r2 = tmp;
        }
        if(r2 - r1 >= d)
                return pi*r1*r1;
        double ang1=acos((r1*r1+d*d-r2*r2)/(2*r1*d));
        double ang2=acos((r2*r2+d*d-r1*r1)/(2*r2*d));
        return ang1*r1*r1 + ang2*r2*r2 - r1*d*sin(ang1);
}


```

```
28
29  最小圆覆盖
30  double eps=1e-9;
31  const double pi=acos(-1);
32
33  struct node{
34      double x,y;
35      node(){}
36      node(double x,double y):x(x),y(y){}
37  };
38  typedef node Vector;
39  typedef node Point;
40  typedef node point;
41  Vector operator + (Vector a, Vector b){//向量加法
42      return Vector(a.x + b.x, a.y + b.y);
43  }
44  Vector operator - (Vector a, Vector b){//向量减法
45      return Vector(a.x - b.x, a.y - b.y);
46  }
47  Vector operator * (Vector a, double p){//向量数乘
48      return Vector(a.x*p, a.y*p);
49  }
50  Vector operator / (Vector a, double p){//向量数除
51      return Vector(a.x / p, a.y / p);
52  }
53  double Dot(Vector a, Vector b){//内积
54      return a.x*b.x + a.y*b.y;
55  }
56  double Length(Vector a){//模
57      return sqrt(Dot(a, a));
58  }
59  double Cross(Vector a, Vector b){//外积
60      return a.x*b.y - a.y*b.x;
61  }
62  const double Pi=acos(-1.0);
63
64  int dcmp(double x)
65  {
66      if (fabs(x)<eps) return 0;
67      else if (x<0) return -1;
68      else return 1;
69  }
70
71  double lenth(node a) {return sqrt(Dot(a,a));}
72
73  node rotate(node a,double t)    //向量旋转
74  {
75      return node(a.x*cos(t)-a.y*sin(t),a.x*sin(t)+a.y*cos(t));
76  }
77
78  node jiao(node p,node v,node q,node w)
79  {
80      node u=p-q;
81      double t=Cross(w,u)/Cross(v,w);
82      return p+v*t;
83  }
```

```
84
85  node get_c(node a,node b,node c)
86  {
87      node p=(a+b)/2;    //ad 中点
88      node q=(a+c)/2;    //ac 中点
89      node v=rotate(b-a,Pi/2.0),w=rotate(c-a,Pi/2.0);   //中垂线的方向向量
90      if (dcmp(Cross(v,w))==0)    //平行
91      {
92          if (dcmp(lenth(a-b)+lenth(b-c)-lenth(a-c))==0)
93             return (a+c)/2;
94          if (dcmp(lenth(b-a)+lenth(a-c)-lenth(b-c))==0)
95             return (b+c)/2;
96          if (dcmp(lenth(a-c)+lenth(c-b)-lenth(a-b))==0)
97             return (a+b)/2;
98      }
99      return jiao(p,v,q,w);
100 }
101 node P[290];
102 node c;
103 double r;
104
105 void min_circular(int n)
106 {
107     random_shuffle(P+1,P+n+1);    //随机化
108     c=P[1],r=0;
109     //c 圆心
110     //r 半径
111     for (int i=2;i<=n;i++)
112         if (dcmp(lenth(c-P[i])-r)>0)    //不在圆内
113         {
114             c=P[i],r=0;
115             for (int j=1;j<i;j++)
116                 if (dcmp(lenth(c-P[j])-r)>0)
117                 {
118                     c=(P[i]+P[j])/2.0;
119                     r=lenth(c-P[i]);
120                     for (int k=1;k<j;k++)
121                         if (dcmp(lenth(c-P[k])-r)>0)
122                         {
123                             c=get_c(P[i],P[j],P[k]);
124                             r=lenth(c-P[i]);
125                         }
126                 }
127         }
128 }
129
130 int main(){
131     freopen("robots.in","r",stdin);
132     int t;scanf("%d",&t);
133     while(t--){
134         int n;double R,rr;scanf("%d%lf%lf",&n,&R,&rr);
135         P[1]=node(0,0);
136         n++;
137         for(int i=2;i<=n;i++){
138             double x,y;scanf("%lf%lf",&x,&y);
139             P[i]=P[i-1]+node(x,y);
```

```
        }
        min_circular(n);
        printf("%.9f %.9f\n",-c.x,-c.y);
    }
}


极角排序
point Tmp;//选好的起点
int Quadrant(point a){// 象限
    if(a.x>0&&a.y>=0)  return 1;
    if(a.x<=0&&a.y>0)  return 2;
    if(a.x<0&&a.y<=0)  return 3;
    if(a.x>=0&&a.y<0)  return 4;
}
bool cmp(point a,point b){
    if(Quadrant(a-Tmp)==Quadrant(b-Tmp)){
        LL ans=Cross(a-Tmp,b-Tmp);
        if(ans==0)return a.x<b.x;
        return ans>0;
    }
    return Quadrant(a-Tmp)<Quadrant(b-Tmp);
}




最近圆对
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef double db;
const int N=5e4+7;
const db eps=1e-8;
const db inf=2e5;
int sign(db k){if(k>eps)return 1;if(k<-eps) return -1;return 0;}
int dcmp(db k1,db k2){return sign(k1-k2);}
int t,n;
struct Point{
    db x,y;
    Point operator -(const Point k)const{return (Point){x-k.x,y-k.y};}
    db abs2()const{return x*x+y*y;}
    db abs()const{return sqrt(abs2());}
    db dis(const Point k)const{return ((*this)-k).abs();}
    void input(){scanf("%lf%lf",&x,&y);}
    void output(){printf("(%f,%f)\n",x,y);}
};
struct Circle{
    Point o;db r;
    bool operator<(const Circle k)const{return o.y<k.o.y;}
    void input(){o.input();scanf("%lf",&r);}
}c[N];
struct Node{
    db pos;
    int id;
    bool operator <(const Node k)const{return pos<k.pos;}
```

```
}a[N],b[N];
set<int>st;
set<int>::iterator it,jt;
bool jg(int id,db mid,set<int>::iterator it){
    it++;
    Circle tmp;
    if(it!=st.end()){
        tmp=c[*it];
        if(dcmp(tmp.o.dis(c[id].o),mid*2+c[id].r+tmp.r)<=0)
                return true;
    }
    it--;
    if(it!=st.begin()){
        it--;
        tmp=c[*it];
        if(dcmp(tmp.o.dis(c[id].o),mid*2+c[id].r+tmp.r)<=0)
                return true;
    }
    return false;
}
bool ins(int id,db mid){
    it=st.insert(id).first;
    return jg(id,mid,it);
}
bool ers(int id,db mid){
    jt=it=st.find(id);
    if(jg(id,mid,it)) return true;
    it++;
    if(it!=st.end()&&jt!=st.begin()){
        jt--;
        Circle c1=c[*it],c2=c[*jt];
        if(dcmp(c1.o.dis(c2.o),mid*2+c1.r+c2.r)<=0) return true;
    }
    st.erase(id);
    return false;
}
bool ck(db mid){
    st.clear();
    int i=1,j=1;
    while(i<=n&&j<=n){
        if(a[i].pos-mid<=b[j].pos+mid){
            if(ins(a[i++].id,mid)) return true;
        }
        else{
            if(ers(b[j++].id,mid)) return true;
        }
    }
    while(i<=n){
        if(ins(a[i++].id,mid)) return true;
    }
    while(j<=n){
        if(ers(b[j++].id,mid)) return true;
    }
    return false;
}
int main()
```

```
{
    scanf("%d",&t);
    while(t--){
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            c[i].input();
        }
        sort(c+1,c+1+n);
        for(int i=1;i<=n;i++){
            a[i]=(Node){c[i].o.x-c[i].r,i};
            b[i]=(Node){c[i].o.x+c[i].r,i};
        }
        sort(a+1,a+1+n);
        sort(b+1,b+1+n);
        db lo=0,hi=c[1].o.dis(c[2].o)-c[1].r-c[2].r;
        while(hi-lo>eps){
            db mid=(lo+hi)/2;
            if(ck(mid)) hi=mid;
            else lo=mid;
        }
        printf("%.6f\n",lo*2);
    }
}
```