#Dijkstra

```
#include<stdio.h>
#define INFINITY 2000000000
int n,m,s,a,b,c;
int wfound[100];
int Sweight[100];
int Weights[100][100];
int Spath[100];
int i,j;
void Dijkstra()
{
    int i, j, v, minweight;
    for(i=1; i<=n; i++) { Sweight[i] = Weights[1][i]; Spath[i] = 1; }  //初始化数组
Sweight和Spath
    Sweight [1] = 0;
    wfound [1] = 1;
    for(i=1; i<= n-1; i++) {  //迭代VNUM-1次
        minweight = INFINITY;
        for(j=1; j <= n;  j++)   //找到未标记的最小权重值顶点
            if( !wfound[j] && ( Sweight[j] < minweight) ) {
                v = j;
                minweight = Sweight[v];
            }
        wfound[v] = 1;  //标记该顶点为已找到最短路径
        for(j =1; j <= n; j++) //找到未标记顶点且其权值大于v的权值+(v,j)的权值，更新其
权值
            if( !wfound[j]  &&  (minweight + Weights[v][j] < Sweight[j] )) {
                Sweight[j] = minweight + Weights[v][j];
                Spath[j] = v;  //记录前驱顶点
            }
    }
}
int main()
{
        scanf("%d%d%d",&n,&m,&s);
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        Weights[i][j]=INFINITY;
                }
        }
        for(i=0;i<m;i++){
                scanf("%d%d%d",&a,&b,&c);
                Weights[a][b]=c;
        }
        Dijkstra();
        for(i=1;i<=n;i++){
                printf("%d ",Sweight[i]);
        }
}
```

#Dinic1

```
#include<bits/stdc++.h>
using namespace std;
const int INF=0x3f3f3f3f;
const int MAXN=1000005;
const int MAXM=1000005;
int d[MAXN],n,m,p[MAXN],eid,S,T,x[MAXN],y[MAXN],z[MAXN],a[105][105],sz,sum;
struct A{  //灰常正常的最大流（Dinic）
        int v,c,next;
}e[MAXM];
void init(){
        memset(p,-1,sizeof(p));
        eid=0;
}
void add(int u,int v,int c){
        e[eid].v=v;
        e[eid].c=c;
        e[eid].next=p[u];
        p[u]=eid++;
}
void insert(int u,int v,int c){
        add(u,v,c);
        add(v,u,0);
}
int bfs(){
        memset(d,-1,sizeof(d));
        queue<int>q;
        d[S]=0;
        q.push(S);
        while(!q.empty()){
                int u=q.front();
                q.pop();
                for(int i=p[u];i!=-1;i=e[i].next){
                        int v=e[i].v;
                        if(e[i].c>0&&d[v]==-1){
                                d[v]=d[u]+1;
                                q.push(v);
                        }
                }
        }
        return (d[T]!=-1);
}
int dfs(int u,int flow){
        if(u==T) return flow;
        int ret=0;
        for(int i=p[u];i!=-1;i=e[i].next){
                int v=e[i].v;
                if(e[i].c>0&&d[v]==d[u]+1){
                        int tmp=dfs(v,min(flow,e[i].c));
                        e[i].c-=tmp;
                        e[i^1].c+=tmp;
                        flow-=tmp;
```

```
                        ret+=tmp;
                        if(!flow) break;
                }
        }
        if(!ret) d[u]=-1;
        return ret;
}
int Dinic(){
        int ret=0;
        while(bfs()){
                ret+=dfs(S,INF);
        }
        return ret;
}
int main(){    //以下开始码风突变（中了yjq的膜法）
        init();
        scanf("%d%d", &m, &n);
        S=0;T=n*m+1;//建立源点和汇点
        for(int i = 1; i <= m; i++){
                for(int j = 1; j <= n; j++){
                        scanf("%d", &a[i][j]);
                        sum += a[i][j];
                        sz ++;
                        if((i + j) % 2){
                                insert(S, sz, a[i][j]);//连向源点
                                if(j < n) insert(sz, sz + 1, INF);//把有限制条件的
连起来，边权注意要尽量大
                                if(j > 1) insert(sz, sz - 1, INF);
                                if(i < m) insert(sz, sz + n, INF);
                                if(i > 1) insert(sz, sz - n, INF);
                        } else {
                                insert(sz,T,a[i][j]);//连向汇点
                        }
                }
        }
        printf("%d",sum - Dinic());//总的边权 - 最大流（最小割）
        return 0;
}
```

#Dinic2

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<queue>
#define INF 2147483647
#define max 10005
struct node{
        int a,b,c,num;
}p[105];
int min(int a,int b){
        if(a<b) return a;
```

```
        return b;
}
struct Edge{
        int v;
        int c;
        int next;
}e[max];
int head[max],e_num=-1;
int n,m,S,T;
int cmp(const void *a,const void *b){
        struct node c,d;
        c=*(struct node*)a;
        d=*(struct node*)b;
        return c.c-d.c;
}
void add(int u,int v,int c){
        e_num++;
        e[e_num].v=v;
        e[e_num].c=c;
        e[e_num].next=head[u];
        head[u]=e_num;
}
void insert(int u,int v,int c){
        add(u,v,c);
        add(v,u,c);
}
int depth[max];// 层次网络
bool bfs()
{
    std::queue<int> q;//定义一个bfs寻找分层图时的队列
    while (!q.empty()) q.pop();
    memset(depth,-1,sizeof(depth));
    depth[S]=0;//源点深度为0
    q.push(S);
    while(!q.empty()){
        int u=q.front();
        q.pop();
        for(int i=head[u];i!=-1;i=e[i].next){
            int v=e[i].v;
            if(e[i].c>0&&depth[v]==-1){
                q.push(v);
                depth[v]=depth[u]+1;
            }
        }
    }
    return (depth[T]!=-1);
}
int dfs(int u,int flow){        //flow表示当前搜索分支的流量上限
    if(u==T){
        return flow;
    }
    int res=0;
    for(int i=head[u];i!=-1;i=e[i].next){
        int v=e[i].v;
```

```
        if(e[i].c>0&&depth[u]+1==depth[v]){
            int tmp=dfs(v,min(flow,e[i].c));    // 递归计算顶点 v，用 c(u, v) 来更新
当前流量上限
            flow-=tmp;
            e[i].c-=tmp;
            res+=tmp;
            e[i^1].c+=tmp;      // 修改反向弧的容量
            if(flow==0){        // 流量达到上限，不必继续搜索了
                break;
            }
        }
    }
    if(res==0){     // 当前没有经过顶点 u 的可行流，不再搜索顶点 u
        depth[u]=-1;
    }
    return res;
}
int dinic(){        // 函数返回值就是最大流的结果
    int res=0;
    while(bfs()){
        res+=dfs(S,INF);    // 初始流量上限为 INF
    }
    return res;
}
int main(){
        int i,j;
        int ans[505]={};
    scanf("%d%d",&n,&m);//m为边
    for(i=0;i<m;i++){
        scanf("%d%d%d",&p[i].a,&p[i].b,&p[i].c);
        p[i].num=i;
    }
    qsort(p,m,sizeof(struct node),cmp);
    for(i=0;i<m;i++){
        memset(head,-1,sizeof(head));
        memset(e,0,sizeof(e));
        e_num=-1;
        S=p[i].a,T=p[i].b;
        for(j=i;j>=0;j--){
                if(p[j].c==p[i].c) continue;
                insert(p[j].a,p[j].b,1);
                }
                ans[p[i].num]=dinic();
        }
        for(i=0;i<m;i++){
                printf("%d ",ans[i]);
        }
    return 0;
}
```

#EK

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
const int INF=0x7fffffff;

queue <int> q;
int n,m,x,y,s,t,g[201][201],pre[201],flow[201],maxflow;
//g邻接矩阵存图，pre增广路径中每个点的前驱，flow源点到这个点的流量
int a,b;
inline int bfs(int s,int t)
{
    while (!q.empty()) q.pop();
    for (int i=1; i<=n; i++) pre[i]=-1;
    pre[s]=0;
    q.push(s);
    flow[s]=INF;
    while (!q.empty())
    {
        int x=q.front();
        q.pop();
        if (x==t) break;
        for (int i=1; i<=n; i++)
          //EK一次只找一个增广路
          if (g[x][i]>0 && pre[i]==-1)
          {
            pre[i]=x;
            flow[i]=min(flow[x],g[x][i]);
            q.push(i);
          }
    }
    if (pre[t]==-1) return -1;
    else return flow[t];
}

//increase为增广的流量
void EK(int s,int t)
{
    int increase=0;
    while (bfs(s,t)!=-1)
    {//迭代
        int k=t;
        while (k!=s)
        {
            int last=pre[k];//从后往前找路径
            g[last][k]-=increase;
            g[k][last]+=increase;
            k=last;
        }
        maxflow+=increase;
    }
}
```

```
int main()
{
    scanf("%d%d%d%d",&a,&m,&b,&n);
    for (int i=1; i<=m; i++)
    {
        int z;
        scanf("%d%d%d",&x,&y,&z);
        g[x][y]+=z;//此处不可直接输入，要+=
    }
    EK(b,n);
    printf("%d",maxflow);
    return 0;
}
```

#Floyd

```
#include<cstdio>
using namespace std;
int t;
int n,m;
int a,b,c;
int i,j,k;
int s[205][205];
int w;
int main()
{
        scanf("%d",&t);
        while(t--){
                scanf("%d%d",&n,&m);
                w=0;
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                s[i][j]=-1;
                        }
                }
                for(i=0;i<m;i++){
                        scanf("%d%d%d",&a,&b,&c);
                        s[a][b]=c;
                }
                for(k=1;k<=n;k++){
                        for(i=1;i<=n;i++){
                                for(j=1;j<=n;j++){
                                        if(s[i][j]>s[i][k]+s[k][j]&&s[i]
[k]!=-1&&s[k][j]!=-1){
                                                s[i][j]=s[i][k]+s[k][j];
                                        }
                                        else if(s[i][j]==-1&&s[i][k]!=-1&&s[k]
[j]!=-1&&i!=j){
                                                s[i][j]=s[i][k]+s[k][j];
                                        }
                                }
```

```
                        }
                }
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                if(w<s[i][j]){
                                        w=s[i][j];
                                }
                        }
                }
                //printf("%d",w);
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                if(w==s[i][j]&&i!=j){
                                        printf("%d %d\n",i,j);
                                }
                        }
                }
        }
}
```

#凸包

```
#include<cstdio>
#include<algorithm>
#include<cmath>
#define rint register int
using namespace std;

struct node {
        double x,y;
} a[100005];
int n,p,st[100005],top;
double ans,miny=2e9,minx=2e9;

int cmp(node b,node c) { //极角排序
        if (fabs((b.y-miny)*(c.x-minx)-(c.y-miny)*(b.x-minx))<=1e-8) return
fabs(minx-b.x)<fabs(minx-c.x);
        return (b.y-miny)*(c.x-minx)<(c.y-miny)*(b.x-minx);
}

int check(int b,int c,int d) { //叉积判断
        return ((a[b].x*a[c].y)+(a[c].x*a[d].y)+(a[d].x*a[b].y)-(a[b].x*a[d].y)-
(a[c].x*a[b].y)-(a[d].x*a[c].y))>0;
}

double dist(double x1,double y1,double x2,double y2) { //计算两点间的距离
        return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}

int main() {
        rint i;
        scanf("%d",&n);
```

```
        for (i=1; i<=n; ++i) {
                scanf("%lf%lf",&a[i].x,&a[i].y);
                if (a[i].y<miny) { //寻找最下方的点
                        miny=a[i].y;
                        minx=a[i].x;
                }
        }
        sort(a+1,a+1+n,cmp); //极角排序
        st[1]=1;
        st[2]=2;
        top=2; //将两个点加入栈中
        for (i=3; i<=n; ++i) { //扫描
                while (!check(st[top-1],st[top],i)) top--;
                st[++top]=i;
        }
        for (i=2; i<=top; ++i) //计算答案
                ans+=dist(a[st[i-1]].x,a[st[i-1]].y,a[st[i]].x,a[st[i]].y);
        ans+=dist(a[st[top]].x,a[st[top]].y,a[1].x,a[st[1]].y);
        double area=0;
        for(i=1;i<top;i++){
                area+=(a[st[i]].x*a[st[i+1]].y-a[st[i+1]].x*a[st[i]].y);
        }
        area+=(a[st[top]].x*a[st[1]].y-a[st[1]].x*a[st[top]].y);
        area/=2;
        printf("%.2lf %.2lf",ans,area);
        return 0;
}
```

#图1

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#define max 300005
using namespace std;
typedef long long ll;
struct node {
        int a,b,c;
} ver[max];
struct Edge {
        int v;
        int c;
        int next;
} e[max];
int head[max],e_num=0;
int n,m,S,T;
ll mid;
int color[max],vis[max];
void add(int u,int v,int c) {
        e[e_num].v=v;
        e[e_num].c=c;
```

```
        e[e_num].next=head[u];
        head[u]=e_num;
        e_num++;
}
void insert(int u,int v,int c) {
        add(u,v,c);
        add(v,u,c);
}
bool dfs(int u, int c)
{
        vis[u]=1;
    color[u]=c;
    for(int i=head[u];~i;i=e[i].next)
    {
        int j=e[i].v;
        if(!color[j])
        {
            if(!dfs(j, 3-c)) return false;
        }
        else if(color[j]==c) return false;
    }
    return true;
}
bool check() {
        memset(vis,0,sizeof(vis));
        memset(head,-1,sizeof(head));
        memset(color,0,sizeof(color));
        for(int i=1; i<=m; i++) {
                if(ver[i].c>mid) insert(ver[i].a,ver[i].b,ver[i].c);//如果大则连边
        }
        for(int i=1; i<=n; i++){
                if(!vis[i]){
                        if(!dfs(i,1)) return false;
                }
        }
        return true;
}
int main() {
        int a,b,c;
        ll ans;
        scanf("%d%d",&n,&m);
        for(int i=1; i<=m; i++) {
                scanf("%d%d%d",&ver[i].a,&ver[i].b,&ver[i].c);
        }
        ll l=0,r=1e14;
        while(l<=r){
                mid=((l+r)>>1);
                if(check()) ans=mid,r=mid-1;
                else l=mid+1;
        }
        printf("%lld",ans);
}
```