Water 题解

——LeavingZ

题目大意

给出一个 N 点 M 边的无向图，求一个位置（可以在某一条边上的某一点）使得该点到所有顶点的距离中的最大值最小。

题解

猜想函数形式

我们对每一条边 (u,v,w) 都去寻找其上的最优点（定义一条边的最优点为该边上到图中所有顶点最大距离最小的点）

定义一个点的权值为到到图中所有顶点的最大距离。

设 $f(x)$ 表示当我们选择这条边上距离端点 u 距离 x 的一点时该点的权值

对于一个点，计算其权值相当简单

```
int m[207][207];//我们假设你已经做过一次Flyod得到了这个数组
double calc(int u,int v,int w,double x)////计算边(u,v,w)上距离u为x的一点的权值
{
    double res=0;
    for(int i=1;i<=N;i++)
        res=max(res,min(m[u][i]+x,m[v][i]+w-x));
    //从这一点p走到图中另一点只有两种路径：p->u->i 或者 p->v->i，二者取最小值即为p->i的距离
    //res为距离最大值
    return res;
}
```

这时候你可以选择打表来观察这个函数的性质，方法也很简单，写一个暴力程序，

```
double calc(int u,int v,int w,double x)//计算边(u,v,w)上距离u为x的一点的权值
{
    double res=0;
    for(int i=1;i<=N;i++)
        res=max(res,min(m[u][i]+x,m[v][i]+w-x));
    return res;
}
int main()
{
    //.......省略一部分
    int u,v,w;
    for(int i=1;i<=M;i++)
```

```
    {
        u=from[i],v=to[i],w=wi[i];
        for(double x=0;x<=w;x+=0.1)//大致描绘出f(x)的形状
            printf("%.2f ",calc(u,v,w,x));
        printf("\n");
    }
}
```

通过观察你会发现这个函数是一个单谷函数

你所要求的就是最小值点对应的函数值，如果能用较高的效率实现这个操作的话，我们对每条边都进行一次计算取最小值即可。

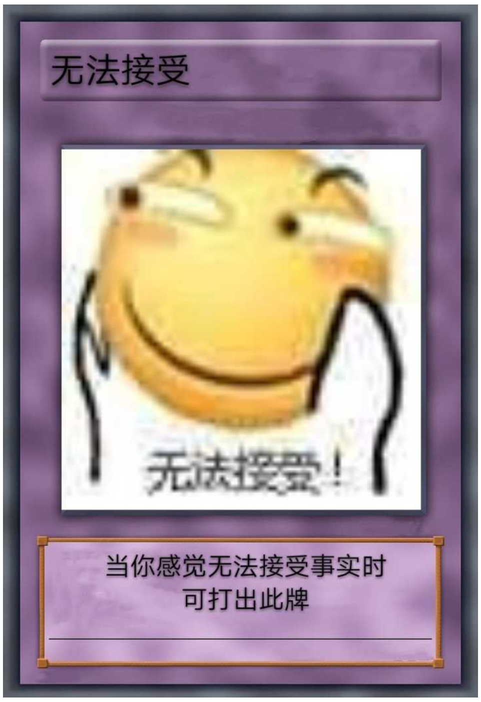
三分法

三分法用于高效率的求解单峰/单谷函数的极值

这一部分不专门讲解，可以去洛谷模板题处进行学习

高效计算

如果我们使用上面的朴素方法计算 $f(x)$ 那么时间复杂度将达到 $O(N^3+MN\log w)$



问题出在哪里呢

每次三分出一个点 x ，我们都要枚举 $1\sim N$ 来计算其权值，这很慢。

所以我们尝试优化这一过程

设 d_i 为顶点 i 到我们当前选择的这一点的距离

可以写出 $d_i=\min(dis_{u,i}+x,dis_{v,i}+w-x)$

当 $dis_{u,i}+x\le dis_{v,i}+w-x$ 时上式取 $dis_{u,i}+x$

即，当 $x\le\dfrac{dis_{v,i}+w-dis_{u,i}}{2}$ 时上式取 $dis_{u,i}+x$ 否则取 $dis_{v,i}+w-x$

你发现 $\dfrac{dis_{v,i}+w-dis_{u,i}}{2}$ 这个值和 x 没有关系

所以我们便有了办法

处理每一条边时，我们令 $lim_i = \dfrac{dis_{v,i} + w - dis_{u,i}}{2}$ ，那么

$$d_i = \begin{cases} dis_{u,i} + x, x < lim_i \\ dis_{v,i} + w - x, x \geq lim_i \end{cases}$$

该点权值即为 $\max\limits_{i=1}^{n} d_i$

求出 $lim_{1\sim N}$ 并且排序，每次想要计算 $f(x)$ 时，进行二分查找，对于 $lim_i \leq x$ 的顶点，求 $\max(dis_{v,i}) + w - x$，其他顶点，求 $\max(dis_{u,i}) + x$。

采用预处理前/后缀最大值实现高效计算，代码如下

```
int dis[maxn][maxn];
struct node{
    double lim;
    int d1,d2,id;
}l[maxn];
bool operator < (const node &x,const node &y) {return x.lim<y.lim;}
int pre[maxn],suf[maxn];
double calc(double x,int w)//高效计算f(x)
{
    node tmp;
    tmp.lim=x;
    int p=std::upper_bound(l+1,l+1+N,tmp)-l;
    //pre[p-1]就是满足lim[i]<=x的max(dis[v][i]),在solve函数中已经预处理了前缀最大值
    //suf[p]就是满足lim[i]>x的max(dis[u][i]),在solve函数中已经预处理了后缀最大值
    return max(pre[p-1]+w-x,suf[p]+x);
}
double solve(int u,int v,int w)
{
    for(int i=1;i<=N;i++)
        l[i].id=i,l[i].lim=(dis[v][i]+w-dis[i][u])/2.0,l[i].d1=dis[u]
[i],l[i].d2=dis[v][i];
    sort(l+1,l+1+N);
    pre[1]=l[1].d2;
    for(int i=2;i<=N;i++)//预处理前缀最大的dis[v][i]
        pre[i]=max(pre[i-1],l[i].d2);
    suf[N]=l[N].d1;
    for(int i=N-1;i>0;i--)//预处理后缀最大的dis[u][i]
        suf[i]=max(suf[i+1],l[i].d1);
    //还没写完
}
```

最终形态

现在你可以快速计算 $f(x)$ ，那么对每条边进行三分求最小值，取所有边上最优点权值的最小值作为答案

但是要注意数量级，本题要求输出保留两位小数，为了保持精度同时不超时，建议三分的限度为 $10^{-4} \sim 10^{-5}$

最终时间复杂度为 $O(N^3 + M(N + \log w \log N))$

最终代码如下（呃，直接看原来代码也可以，这里是我重写的一版）

```
#include<cstdio>
#include<algorithm>
```

```
#include<cstring>
#include<cctype>
#include<cstdlib>
#include<ctime>
using std::max;
using std::min;
using std::sort;
using std::swap;
const int maxn=507;
const int maxm=20007;
const double del=5e-4;
struct E{
    int u,v,w;
}e[maxm];
int first[maxn],nt[maxm],ES;
int N,M;
inline void addE(int u,int v,int w)
{
    e[++ES]=(E){u,v,w};
    nt[ES]=first[u];
    first[u]=ES;
    return ;
}
int dis[maxn][maxn];
struct node{
    double lim;
    int d1,d2,id;
}l[maxn];
bool operator < (const node &x,const node &y) {return x.lim<y.lim;}
int pre[maxn],suf[maxn];
double calc(double x,int w)//高效计算f(x)
{
    node tmp;
    tmp.lim=x;
    int p=std::upper_bound(l+1,l+1+N,tmp)-l;
    //pre[p-1]就是满足lim[i]<=x的max(dis[v][i]),在solve函数中已经预处理了前缀最大值
    //suf[p]就是满足lim[i]>x的max(dis[u][i]),在solve函数中已经预处理了后缀最大值
    return max(pre[p-1]+w-x,suf[p]+x);
}
double solve(int u,int v,int w)
{
    for(int i=1;i<=N;i++)
        l[i].id=i,l[i].lim=(dis[v][i]+w-dis[i][u])/2.0,l[i].d1=dis[u]
[i],l[i].d2=dis[v][i];
    sort(l+1,l+1+N);
    pre[1]=l[1].d2;
    for(int i=2;i<=N;i++)//预处理前缀最大的dis[v][i]
        pre[i]=max(pre[i-1],l[i].d2);
    suf[N]=l[N].d1;
    for(int i=N-1;i>0;i--)//预处理后缀最大的dis[u][i]
        suf[i]=max(suf[i+1],l[i].d1);
    double L=0,R=w,lx,rx,lv,rv,res=3e9;
    while(R-L>=del)
    {
        lx=L+(R-L)/3;rx=R-(R-L)/3;
        lv=calc(lx,w);rv=calc(rx,w);
        if(lv<=rv) R=rx,res=min(res,lv);
        else L=lx,res=min(res,rv);
```

```
    }
    return res;
}
int main()
{
    freopen("water.in","r",stdin);
    freopen("water.out","w",stdout);
    scanf("%d%d",&N,&M);
    int u,v,w;
    memset(dis,0x3f,sizeof(dis));
    for(int i=1;i<=M;i++)
        scanf("%d%d%d",&u,&v,&w),addE(u,v,w),dis[u][v]=dis[v][u]=min(dis[u]
[v],w);
    for(int i=1;i<=N;i++) dis[i][i]=0;
    for(int k=1;k<=N;k++)
        for(int i=1;i<=N;i++)
            for(int j=1;j<=N;j++)
                dis[i][j]=min(dis[i][k]+dis[k][j],dis[i][j]);
    double ans=3e9;
    for(int i=1;i<=M;i++)
        ans=min(ans,solve(e[i].u,e[i].v,e[i].w));
    printf("%.2f",ans);
    return 0;
}
```