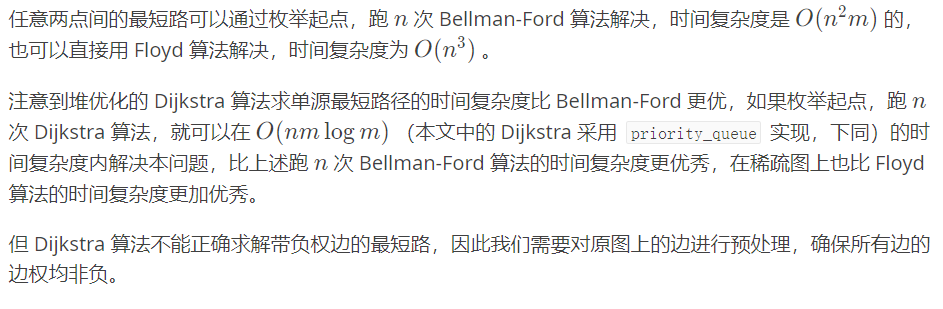
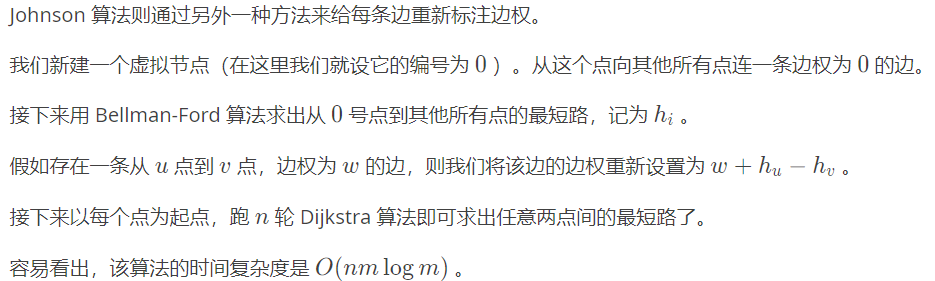


Johnson 和 Floyd 一样，是一种能求出无负环图上任意两点间最短路径的算法。





Bellman-Ford算法可用spfa代替

值得注意的是，Johnson算法可以用来求出有向图中的所有环，具体算法也很简单，如果没有负边的话，就不需要spfa，然后跑n次dij之后，

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

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

if(d[i][j]!=inf&&d[j][i]!=inf&&i!=j)

这样i到j之间就是一个简单环了

#include<bits/stdc++.h>

#define ll long long

#define inf 1e9//inf是1e9

#define mn 10005

using namespace std;

ll ans,cnt=0,h[mn],used[mn],

bj[mn],dis[mn],

n,m,d[mn];

//dis表示的是0这个虚点到所有点的最短路长度，d才是从某个起点s到所有点的最短路长度

struct ww//结构体式写法

{

ll x,d;

friend bool operator<(ww a,ww b){return a.d>b.d;}//<为从大到小排列，>为从小到大排列

};

priority\_queue<ww>q;

struct w{ll next,to,z;}a[mn];//mn至少开到n+m

void add(ll x,ll y,ll z)

{

a[++cnt]=(w){h[x],y,z};

h[x]=cnt;

}

queue<ll>p;

ll spfa(ll s)

{ll i,x,y;

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

dis[i]=inf;

dis[s]=0;bj[s]=1;

p.push(s);

while(p.empty()==0)

{

x=p.front();

p.pop();

bj[x]=0;

for(i=h[x];i;i=a[i].next)

{

y=a[i].to;

if(dis[y]>dis[x]+a[i].z)

{

dis[y]=dis[x]+a[i].z;

if(bj[y]==0)

{

p.push(y);

bj[y]=1;

used[y]++;

if(used[y]>=n+1)return 0;

}

}

}

}

return 1;

}

void dij(ll s)

{ll i,y,x;

for(i=1;i<=n;i++){d[i]=inf;bj[i]=0;}

d[s]=0;

q.push((ww){s,d[s]});

while(!q.empty())

{

ww u=q.top();

q.pop();

x=u.x;

if(bj[x])continue;

bj[x]=1;

for(i=h[x];i;i=a[i].next)

{

y=a[i].to;

if(!bj[y]&&d[y]>d[x]+a[i].z)

{

d[y]=d[x]+a[i].z;

q.push((ww){y,d[y]});

}

}

}

}

int main()

{

ll x,y,z,i,j,k;

char ch;

cin>>n>>m;

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

{

scanf("%lld%lld%lld",&x,&y,&z);

add(x,y,z);

}

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

add(0,i,0);

if(spfa(0)==0){printf("-1");return 0;}

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

for(i=h[x];i;i=a[i].next)

a[i].z+=dis[x]-dis[a[i].to];

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

{

dij(i);

ans=0;

for(j=1;j<=n;j++)//最后的d[j]就是从起点i到j的最短路距离

if(d[j]==inf)ans+=j\*inf;

else ans+=j\*(d[j]+dis[j]-dis[i]);

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

}

return 0;

}