[P2169]正则表达式

Algorithm	Tarjan 缩点 最短路
Created	@Aug 9, 2020 6:01 PM
Difficulty	提高+/省选-
Related to 近期更新 (Property)	[P2169]正则表达式
URL	https://www.luogu.com.cn/problem/P2169

题目链接：

正则表达式

小Z童鞋一日意外的看到小X写了一个正则表达式的高级程序，这个正则表达式程序仅仅由字符"0","1"，"."和"*"构成，但是他能够匹配出所有在OJ上都AC的程序的核心代码！小Z大为颇感好奇，于是他决定入侵小X的电脑上去获得这个正则表达式的高级程序。 在Internet网络中的每台电脑并不是直接一对一

https://www.luogu.com.cn/problem/P2169

题解：

由题意得，发现同一个强连通分量内的点互相到达费用为零，唯一会产生费用的地方就是不同强连通分量之间的连边。那么可以将强连通分量求出后缩点用$Dijkstra$算法求最短路。

直接上代码吧，看变量名字就能懂了：

```
//File: P2169.cpp
//Author: yanyanlongxia
//Date: 2020/8/9
//
#include <bits/stdc++.h>

using namespace std;
int n,m,dft,tot,ntot,root,sccn,head[1000005],edge[1000005],nedge[1000005],nxt[100
0005],ver[1000005],dfn[1000005],low[1000005],bel[1000005],nhead[1000005],nver[100
0005],nnxt[1000005],dist[1000005];
stack<int>st;
bool in[1000005],vis[1000005];
void add(int x,int y,int z)
{
    ver[++tot]=y;
    nxt[tot]=head[x];
    edge[tot]=z;
    head[x]=tot;
}
void nadd(int x,int y,int z)
{
    nver[++ntot]=y;
    nnxt[ntot]=nhead[x];
    nedge[ntot]=z;
    nhead[x]=ntot;
}
void tarjan(int x)
{
    dfn[x]=low[x]=++dft;
    st.push(x);
    in[x]=true;
    for(int i=head[x];i;i=nxt[i])
    {
        int y=ver[i];
        if(!dfn[y])
        {
            tarjan(y);
            low[x]=min(low[x],low[y]);
        }
        else
            if(in[y])
                low[x]=min(low[x],dfn[y]);
    }
    if(low[x]==dfn[x])
    {
        int y;
        sccn++;
        do {
            y=st.top();
```

```
            st.pop();in[y]=false;
            bel[y]=sccn;
        }while (x!=y);
    }
}
void dijkstra()
{
    memset(dist,0x3f3f3f3f,sizeof(dist));
    priority_queue<pair<int,int> >q;
    dist[root]=0;
    q.push(make_pair(0,root));
    while (!q.empty())
    {
        int x=q.top().second;
        q.pop();
        if(vis[x])
            continue;
        vis[x]=true;
        for(int i=nhead[x];i;i=nnxt[i])
        {
            int y=nver[i],z=nedge[i];
            if(dist[y]>dist[x]+z)
            {
                dist[y]=dist[x]+z;
                q.push(make_pair(-dist[y],y));
            }
        }
    }
}
int main() {
    int x,y,z;
    scanf("%d%d",&n,&m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d%d",&x,&y,&z);
        add(x,y,z);
    }
    for(int i=1;i<=n;i++)
        if(!dfn[i])
            tarjan(i);
    for(int i=1;i<=n;i++)
        for(int j=head[i];j;j=nxt[j])
            if(bel[i]!=bel[ver[j]])
                nadd(bel[i],bel[ver[j]],edge[j]);
    if(bel[1]==bel[n])
    {
        printf("0\n");
        return 0;
    }
    root=bel[1];
    dijkstra();
    printf("%d\n",dist[bel[n]]);
    return 0;
}
```