

最短路

POJ 2139: Six Degrees of Cowvin Bacon

直接上floyd即可，注意三重循环中，k在最外层循环。代码如下

```
/*

*/

#define method_1
#ifdef method_1
/*
直接上floyd即可，注意三重循环中，k在最外层循环。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"  "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=300+5;
const int INF=0x3f3f3f3f;
int n,m,d[maxn][maxn],a[maxn],ans=INF;
int main() {
    ios::sync_with_stdio(false);
    //freopen("Six Degrees of Cowvin Bacon.in","r",stdin);
    cin>>n>>m;
    memset(d,INF,sizeof(d));
    for(int i=1;i<=n;i++) d[i][i]=0;
    for(int i=1;i<=m;i++){
        int cnt;cin>>cnt;
        for(int j=1;j<=cnt;j++) cin>>a[j];
        for(int j=1;j<=cnt;j++) for(int k=1;k<=cnt;k++) if(j!=k)
d[a[j]][a[k]]=1;
    }
}
```

```

        for(int k=1;k<=n;k++) for(int i=1;i<=n;i++) for(int j=1;j<=n;j++)
d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
        for(int i=1;i<=n;i++){
            int sum=0;
            for(int j=1;j<=n;j++) if(i!=j) sum+=d[i][j];
            ans=min(ans,sum);
//            D(sum);
        }
        cout<<ans*100/(n-1);
        return 0;
    }
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```

POJ 3259: Wormholes

因为图可能不连通，所以暴力枚举每个点作为起点，**spfa**判断是否存在负环即可。

代码如下

```

/*

*/

#define method_1
#ifdef method_1
/*

因为图可能不连通，所以暴力枚举每个点作为起点，spfa判断是否存在负环即可。

*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>

```

```

#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=500+5;
const int maxm=2500*2+200+5;
const int INF=0x3f3f3f3f;
int F,n,m,w,head[maxn],d[maxn],cnt[maxn],tot,vis[maxn];
queue<int>q;
struct node{
    int from,to,c;
}edge[maxn];
void add(int from,int to,int c){
    edge[++tot].from=head[from],head[from]=tot,edge[tot].to=to,edge[tot].c=c;
}
void init(){
    memset(head,0,sizeof(head));
    tot=1;
}
void spfainit(int s){
    memset(d,INF,sizeof(d));
    memset(cnt,0,sizeof(cnt));
    memset(vis,0,sizeof(vis));
    d[s]=0;vis[s]=1;
    while(q.size()) q.pop();
    q.push(s);
}
bool spfa(){
    for(int i=1;i<=n;i++){ //依次考虑用每个节点作为起点
        spfainit(i);
        while(q.size()){
            int x=q.front();q.pop();vis[x]=0;
            for(int i=head[x];i;i=edge[i].from){
                int y=edge[i].to,c=edge[i].c;
                if(d[x]+c<d[y]){
                    d[y]=d[x]+c;
                    cnt[y]=cnt[x]+1;
                    if(cnt[y]>n) return true;
                    if(!vis[y]){
                        q.push(y);
                        vis[y]=1;
                    }
                }
            }
        }
    }
}

```

```

    }
    return false;
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Wormholes.in", "r", stdin);
    cin >> F;
    while (F--) {
        init();
        cin >> n >> m >> w;
        int from, to, c;
        for (int i = 1; i <= m; i++) cin >> from >> to >> c, add(from, to, c), add(to, from, c);
        //注意非虫洞是双向路径
        for (int i = 1; i <= w; i++) cin >> from >> to >> c, add(from, to, -c);
        spfa()? cout << "YES" << endl : cout << "NO" << endl;
    }
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```

POJ 3268: Silver Cow Party

以X为原点跑一次spfa，然后将边反向后再跑一次spfa，所有节点的d值和答案取最大值即可。代码如下

```

/*

*/

#define method_1
#ifdef method_1
/*

以X为原点跑一次spfa，然后将边反向后再跑一次spfa，所有节点的d值和答案取最大值即可。
*/
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cmath>

```

```

#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"  "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=1000+5;
const int maxm=100000+5;
const int INF=0x3f3f3f3f;
int n,m,x,head[maxn],d[2][maxn],ans=-INF,tot,vis[maxn];
int from[maxm],to[maxm],c[maxm];
queue<int>q;
struct node {
    int from,to,c;
} edge[maxm];
void add(int from,int to,int c) {
    edge[++tot].from=head[from],head[from]=tot,edge[tot].to=to,edge[tot].c=c;
}
void init(int t,int s) {
    memset(head,0,sizeof(head));
    tot=1;
    memset(d[t],INF,sizeof(d[t]));
    memset(vis,0,sizeof(vis));
    d[t][s]=0;
    vis[s]=1;
    while(q.size()) q.pop();
    q.push(s);
}
void spfa(int t,int s) {
    while(q.size()) {
        int x=q.front();q.pop();vis[x]=0;
        for(int i=head[x]; i; i=edge[i].from) {
            int y=edge[i].to,c=edge[i].c;
            if(d[t][x]+c<d[t][y]) {
                d[t][y]=d[t][x]+c;
                if(!vis[y]) {
                    q.push(y);
                    vis[y]=1;
                }
            }
        }
    }
}

```

```

    }
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Silver Cow Party.in", "r", stdin);
    cin>>n>>m>>x;
    init(0,x);
    for(int i=1;i<=m;i++) cin>>from[i]>>to[i]>>c[i],add(from[i],to[i],c[i]);
    spfa(0,x);
    init(1,x);
    for(int i=1;i<=m;i++) add(to[i],from[i],c[i]);
    spfa(1,x);
    for(int i=1;i<=n;i++) ans=max(ans,d[0][i]+d[1][i]);
    cout<<ans;
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```

最小生成树

[POJ 1258: Agri-Net](#) mst模板题。代码如下

```

/*

*/
#define method_1
#ifdef method_1
/*
mst模板题。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>

```

```

#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=100+5;
const int INF=0x3f3f3f3f;
int n,tot,ans,f[maxn];
struct node{
    int from,to,c; //不是链式前向星
    bool operator<(const node& h)const{return c<h.c;}
}edge[maxn*maxn];
void add(int from,int to,int c){
    edge[++tot].from=from,edge[tot].to=to,edge[tot].c=c;
}
void init(){
    tot=0,ans=0;
    for(int i=1;i<=n;i++) f[i]=i;
}
int find(int x){
    return f[x]==x?f[x]:f[x]=find(f[x]);
}
void kruskal(){
    sort(edge+1,edge+n*n+1);
    int cnt=0; //mst中的边数
    for(int i=1;i<=n*n;i++){
        if(cnt>=n-1) break;
        int x=edge[i].from,y=edge[i].to,c=edge[i].c;
        x=find(x),y=find(y);
        if(x!=y){
            f[x]=y;ans+=c;cnt++;
        }
    }
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Agri-Net.in","r",stdin);
    while(cin>>n){
        init();
        int temp;
        for(int i=1;i<=n;i++) for(int j=1;j<=n;j++) cin>>temp,add(i,j,temp);
    }
}

```

```

        kruskal();
        cout<<ans<<endl;
    }
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```

[POJ 2377: Bad Cowtractors](#) 求最大生成树，顺便判断图的连通性。代码如下

```

/*

*/

#define method_1
#ifdef method_1
/*
求最大生成树，顺便判断图的连通性。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"  "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=1000+5;

```

```

const int maxm=20000+5;
const int INF=0x3f3f3f3f;
int n,tot,m,ans,f[maxn];
int cnt=0; //mst中的边数
struct node{
    int from,to,c; //不是链式前向星
    bool operator<(const node& h)const{return c>h.c;} //求最大生成树
}edge[maxm];
void add(int from,int to,int c){
    edge[++tot].from=from,edge[tot].to=to,edge[tot].c=c;
}
void init(){
    tot=0,ans=0;
    for(int i=1;i<=n;i++) f[i]=i;
}
int find(int x){
    return f[x]==x?f[x]:f[x]=find(f[x]);
}
void kruskal(){
    sort(edge+1,edge+m+1);
    for(int i=1;i<=m;i++){
        if(cnt>=n-1) break;
        int x=edge[i].from,y=edge[i].to,c=edge[i].c;
        x=find(x),y=find(y);
        if(x!=y){
            f[x]=y;ans+=c;cnt++;
        }
    }
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Bad Cowtractors.in","r",stdin);
    cin>>n>>m;
    init();
    int from,to,c;
    for(int i=1;i<=m;i++) cin>>from>>to>>c,add(from,to,c);
    kruskal();
    cnt==n-1?cout<<ans:cout<<-1;
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3

```

```

/*
*/

#endif

```

[POJ 2395: Out of Hay](#) 求mst中最大的边。代码如下

```

/*
*/
#define method_1
#ifdef method_1
/*
求mst中最大的边。
*/
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<set>
#include<map>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cstdlib>
#include<iomanip>
#include<ctime>
#include<string>
#define D(x) cout<<#x<<" = "<<x<<"  "
#define E cout<<endl
using namespace std;
typedef long long ll;
typedef pair<int,int>pii;
const int maxn=2000+5;
const int maxm=10000+5;
const int INF=0x3f3f3f3f;
int n,tot,m,ans,f[maxn];
int cnt=0; //mst中的边数
struct node{
    int from,to,c; //不是链式前向星
    bool operator<(const node& h)const{return c<h.c;}
}edge[maxm];
void add(int from,int to,int c){
    edge[++tot].from=from,edge[tot].to=to,edge[tot].c=c;
}
void init(){

```

```

        tot=0;
        for(int i=1;i<=n;i++) f[i]=i;
    }
    int find(int x){
        return f[x]==x?x:f[x]=find(f[x]);
    }
    void kruskal(){
        sort(edge+1,edge+m+1);
        for(int i=1;i<=m;i++){
            if(cnt>=n-1) break;
            int x=edge[i].from,y=edge[i].to,c=edge[i].c;
            x=find(x),y=find(y);
            if(x!=y){
                f[x]=y;cnt++;ans=c;
            }
        }
    }
}
int main() {
    ios::sync_with_stdio(false);
    //freopen("Out of Hay.in","r",stdin);
    cin>>n>>m;
    init();
    int from,to,c;
    for(int i=1;i<=m;i++) cin>>from>>to>>c,add(from,to,c);
    kruskal();
    cout<<ans;
    return 0;
}
#endif
#ifdef method_2
/*

*/

#endif
#ifdef method_3
/*

*/

#endif

```