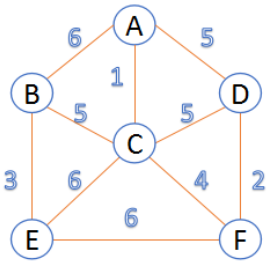
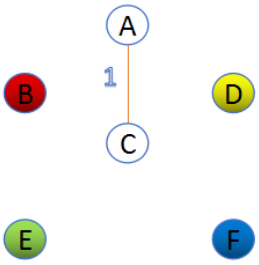
最小生成树

一颗有n个顶点的生成树有且仅有n-1条边

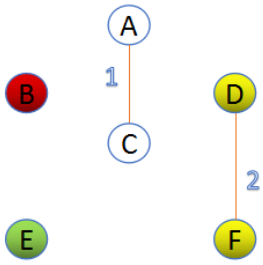
kruskal



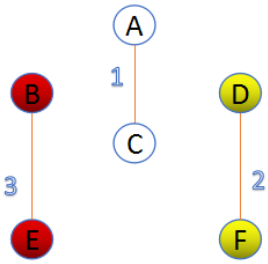
连通网G



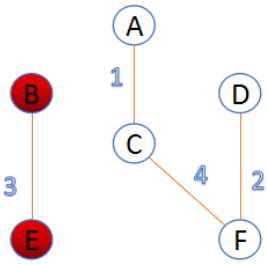
1.选择代价最小的边(A,C);并保证A,C不在同一颗树上,然后合并A,C



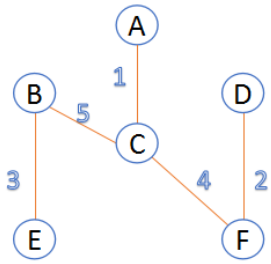
2.选择代价最小的边(D,F);并保证D,F不在同一颗树上,然后合并D,F



3.选择代价最小的边(B,E);并保证B,E不在同一颗树上,然后合并B,E



4.选择代价最小的边(C,F);并保证C,F不在同一颗树上,然后合并C,F所在的树

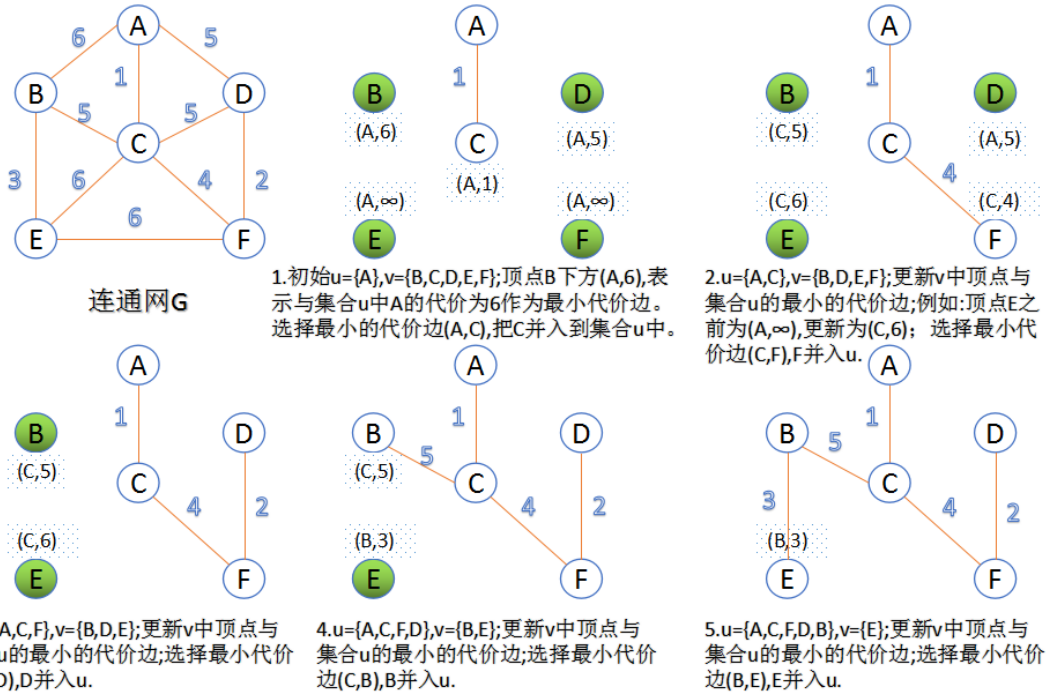


5.选择代价最小的边(A,D),顶点A,D在同一颗树上,丢弃;
选择最小的边(C,D),顶点C,D在同一颗树上,丢弃;
选择最小的边(B,C),顶点B,C不在同一颗树上,加入此边,然后合并B,C所在的树,此时所有顶点在同一颗树上，返回;

```
//P3366 【模板】最小生成树    cruskal
#include<bits/stdc++.h>
using namespace std;
#define maxn 5005
#define maxm 200005

int n,m;
int f[maxn];
struct edge{
    int u,v,w;
}e[maxm];

bool cmp(edge x,edge y){
    return x.w<y.w;
}

int find(int x){
    if(x==f[x]) return f[x];
    return f[x]=find(f[x]);
}

int main(){
    cin>>n>>m;
    int i;
    for(i=1;i<=m;i++){
        scanf("%d%d%d",&e[i].u,&e[i].v,&e[i].w);
```

```
    }
    sort(e+1,e+m+1,cmp);
    for(int i=1;i<=n;i++) f[i]=i;
    int tot=0;
    long long ans=0;
    for(i=1;i<=m;i++){
        int u=e[i].u,v=e[i].v;
        int fu=find(u),fv=find(v);
        if(fu==fv) continue;
        tot++;
        f[fv]=fu;
        ans+=e[i].w;
        if(tot==n-1) break;
    }
    if(tot!=n-1) cout<<"orz"<<endl;
    else cout<<ans<<endl;
    return 0;
}
```

prim



邻接矩阵

```
//P3366 【模板】最小生成树  prim  邻接矩阵
#include<bits/stdc++.h>
using namespace std;
#define maxn 5005
#define maxm 200005
#define INF 0x7fffffff

int n,m;
```

```
int mp[maxn][maxn];
bool vis[maxn];
int dis[maxn];

int prim(){
    for(int i=1;i<=n;i++) dis[i]=INF;
    dis[1]=1;
    vis[1]=1;
    int now=1;
    for(int i=1;i<=n;i++){
        dis[i]=min(dis[i],mp[now][i]);
    }
    int tot=0;
    int sum=0;
    while(tot<n-1){
        int minPath=INF;
        for(int i=1;i<=n;i++){
            if(!vis[i]&&dis[i]<minPath){
                now=i;
                minPath=dis[i];
            }
        }
        if(minPath==INF) return -1;
        tot++;
        sum+=minPath;
        vis[now]=1;
        for(int i=1;i<=n;i++){
            if(mp[now][i]!=INF&&!vis[i]){
                dis[i]=min(dis[i],mp[now][i]);
            }
        }
    }
    return sum;
}

int main(){
    cin>>n>>m;
    int i,j,u,v,w;
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            mp[i][j]=INF;
        }
    }
    for(i=1;i<=m;i++){
        scanf("%d%d%d",&u,&v,&w);
        mp[u][v]=min(mp[u][v],w);
        mp[v][u]=min(mp[v][u],w);
    }
    int ans=prim();
    if(ans==-1) cout<<"orz";
    else cout<<ans;
    return 0;
}
```

链式前向星

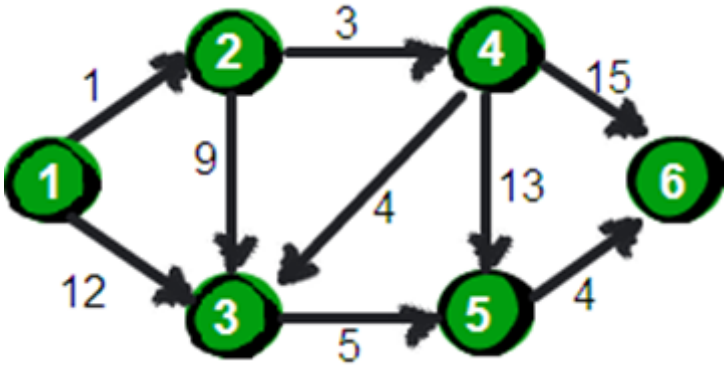
```
//P3366 【模板】最小生成树  prim  链式前向星
```

```cpp
#include<bits/stdc++.h>
using namespace std;
#define maxn 5005
#define maxm 200005
#define INF 0x7fffffff

int n,m;
bool vis[maxn];
int dis[maxn];
int head[maxn],cnt;
struct edge{
    int v,w,next;
}e[maxm<<1];

void add(int u,int v,int w){
    e[++cnt].v=v;
    e[cnt].w=w;
    e[cnt].next=head[u];
    head[u]=cnt;
}

int prim(){
    for(int i=1;i<=n;i++) dis[i]=INF;
    dis[1]=0;vis[1]=1;
    int sum=0,now=1;
    int tot=0;
    for(int i=head[now];i;i=e[i].next){
        int v=e[i].v;
        dis[v]=min(dis[v],e[i].w);
    }
    while(tot<n-1){
        int minPath=INF;
        for(int i=1;i<=n;i++){
            if(!vis[i]&&dis[i]<minPath){
                now=i;
                minPath=dis[i];
            }
        }
        if(minPath==INF) return -1;
        vis[now]=1;
        sum+=minPath;
        tot++;
        for(int i=head[now];i;i=e[i].next){
            int v=e[i].v;
            if(vis[v]) continue;
            dis[v]=min(dis[v],e[i].w);
        }
    }
    return sum;
}

int main(){
    cin>>n>>m;
    int u,v,w;
    for(int i=1;i<=m;i++){
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,w);
        add(v,u,w);
```

```
    }
    int ans=prim();
    if(ans==-1) cout<<"orz";
    else cout<<ans;
    return 0;
}
```

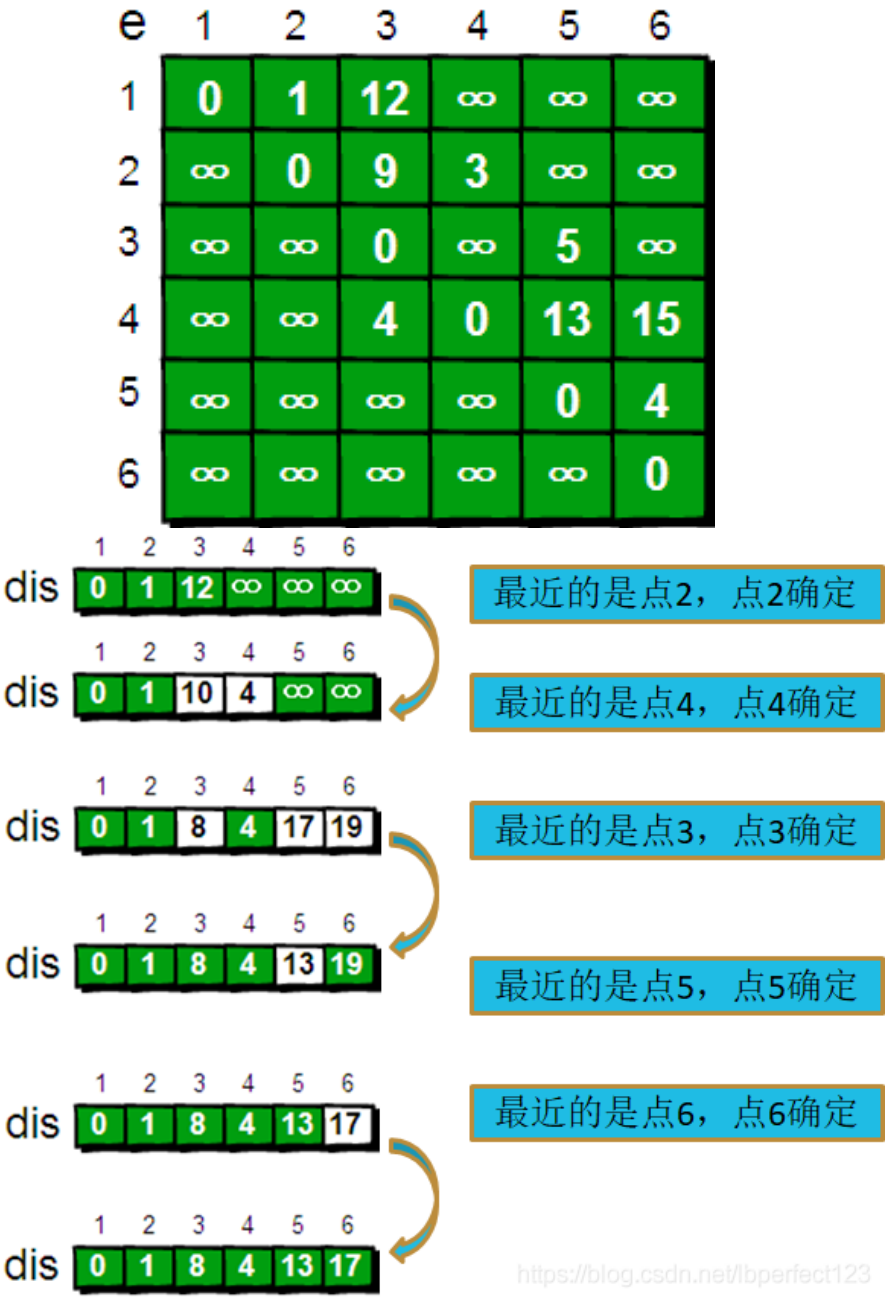
最短路

dijkstra



e	1	2	3	4	5	6
1	0	1	12	∞	∞	∞
2	∞	0	9	3	∞	∞
3	∞	∞	0	∞	5	∞
4	∞	∞	4	0	13	15
5	∞	∞	∞	∞	0	4
6	∞	∞	∞	∞	∞	0

	1	2	3	4	5	6
dis	0	1	12	∞	∞	∞




```cpp
//P4779 【模板】单源最短路径（标准版）    堆优化
#include<bits/stdc++.h>
using namespace std;
#define maxn 100005
#define maxm 200005
#define INF 0x7fffffff

int n,m,s;
long long dis[maxn];
bool vis[maxn];
int head[maxn],cnt=0;
struct edge{
    int v,next;
    long long w;
}e[maxm];

void add(int u,int v,long long w){
    e[++cnt].v=v;
```

```
    e[cnt].w=w;
    e[cnt].next=head[u];
    head[u]=cnt;
}

struct node{
    int id;
    long long dis;
    bool operator<(const node &tem)const{
        if(this->dis==tem.dis) return this->id<tem.id;
        return this->dis>tem.dis;
    }
};

priority_queue<node> q;
void dijkstra(){
    for(int i=1;i<=n;i++) dis[i]=INF;
    dis[s]=0;

    q.push(node{s,dis[s]});
    while(q.size()){
        int now=q.top().id;
        q.pop();
        if(vis[now]) continue;
        vis[now]=1;
        for(int i=head[now];i;i=e[i].next){
            int v=e[i].v;
            if(vis[v]) continue;
            if(dis[now]+e[i].w<dis[v]){
                dis[v]=dis[now]+e[i].w;
                q.push(node{v,dis[v]});
            }
        }
    }
}

int main(){
    cin>>n>>m>>s;
    int u,v;
    long long w;
    for(int i=1;i<=m;i++){
        scanf("%d%d%lld",&u,&v,&w);
        add(u,v,w);
    }
    dijkstra();
    for(int i=1;i<=n;i++){
        cout<<dis[i]<<" ";
    }
    return 0;
}
```