斯坦纳树

给定 nn 个点 A1,A2,⋯,AnA1,A2,⋯,An，试求连接此 nn 个点，总长最短的直线段连接系统，并且任意两点都可由系统中的直线段组成的折线连接起来。他们将此新问题称为 **斯坦纳树问题。**

斯坦纳树问题是组合优化问题，与最小生成树相似，是最短网络的一种。最小生成树是在给定的点集和边中寻求最短网络使所有点连通。而最小斯坦纳树**允许在给定点外增加额外的点**，使生成的最短网络开销最小。

将指定点集合中的所有点连通，且边权总和最小的生成树称为**最小斯坦纳树**（Minimal Steiner Tree）

算法推导

这是一个组合优化问题，可以用**状压DP**来解决。

首先有已经结论：**答案的子图一定是树。**

我们首先钦定一个树根，设dp(i,S)表示以i为根，包含S点集的最小代价。

考虑状态转移：

$$若i的度数等于1，则dp(j,s)+w(j,i)->dp(i,s)$$
$$若i的度数大于1，则dp(i,T)+dp(i,S-T)->dp(i,S)(T\subseteq S)$$

状态转移时对每个S，将图做松弛操作，采用dijkstra实现。

总的时间复杂度 $O(n\times 3^k+mlogm\times 2^k)$

（洛谷P6192【模板】最小斯坦纳树 图示）

代码

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f3f3f3f3f
using namespace std;
const int maxn=510;
const int INF=0x3f3f3f3f;
typedef long long ll;
typedef pair<int,int> P;
int n,m,k,u,v,w;
struct E{
    int to,next,dis;
```

```
}edge[maxn<<1];

int head[maxn<<1],tree[maxn<<1],cnt;
int dp[maxn][5005],vis[maxn];
//dp[i][j]表示以i为根的一棵树，包含集合j中所有点的最小边权值和
int key[maxn];  //关键点
priority_queue<P,vector<P>,greater<P> >q;

void addedge(int from,int to,int dis){
    edge[++cnt].next=head[from];
    edge[cnt].to=to;
    edge[cnt].dis=dis;
    head[from]=cnt;
    tree[cnt]=to;
}

void dijkstra(int s){  //迪杰斯特拉堆优化算法
    memset(vis,0,sizeof(vis));
    while(!q.empty()){
        P fro=q.top();
        q.pop();
        if(vis[fro.second]) continue;
        vis[fro.second]=1;
        for(int i=head[fro.second];i;i=edge[i].next){
            if(dp[tree[i]][s]>dp[fro.second][s]+edge[i].dis){
                dp[tree[i]][s]=dp[fro.second][s]+edge[i].dis;
                //再当前子集连通状态下进行边的松弛操作
                q.push(P(dp[tree[i]][s],tree[i]));
            }
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    memset(dp,INF,sizeof(dp));
    cin>>n>>m>>k;
    for(int i=1;i<=m;i++){ //建无向图
        cin>>u>>v>>w;
        addedge(u,v,w);
        addedge(v,u,w);
    }
    for(int i=1;i<=k;i++){
        cin>>key[i];
        dp[key[i]][1<<(i-1)]=0;
    }
    for(int s=1;s<(1<<k);s++){ //表示点集
        for(int i=1;i<=n;i++){  //中间部分
            for(int subs=s&(s-1);subs;subs=s&(subs-1)) //子图
                dp[i][s]=min(dp[i][s],dp[i][subs]+dp[i][s^subs]);
            if(dp[i][s]!=INF) q.push(P(dp[i][s],i));
        }
        dijkstra(s);
    }
    cout<<dp[key[1]][(1<<k)-1]<<endl;
    return 0;
}
```

参考资料

https://www.luogu.com.cn/problem/solution/P6192