

斯坦纳树

Steiner Tree

给定一个无向连通图 $G = (V, E)$ ，含有 n 个点和 m 条边。其中有 k 个点为特殊点，求出原图中的一个连通子图，使这些特殊点连通。显然这个连通子图是一棵树就够了，称为斯坦纳树。

求边权和最小的连通这些特殊点的树，称为最小斯坦纳树。

DP 求解最小斯坦纳树

Idea: 设 $dp[i][S]$ 表示以点 i 为根、已连通的特殊点构成集合 S 的最小边权和，则转移分两个阶段：

- 对连通的子集进行转移： $dp[i][S] = \min\{dp[i][T] + dp[i][S - T] \mid T \subseteq S\}$.
- 在当前连通的子集状态下，用已更新的 dp 值进行松弛操作： $dp[j][S] = \min(dp[j][S], dp[i][S] + w(i, j))$.

松弛操作采用 **Dijkstra** 算法进行。

Complexity: $O(3^k \cdot n + 2^k(n + m) \lg m)$

Extended: 若问点权和最小的，只需略微修改 dp 方程：

$dp[i][S] = \min\{dp[i][T] + dp[i][S - T] - a[i] \mid T \subseteq S\}$ 以及
 $dp[j][S] = \min(dp[j][S], dp[i][S] + a[j])$.

Code:

```
1 LL dp[N][2005];
2 priority_queue< pair<LL, int>, vector<pair<LL, int>>,
  greater<pair<LL, int>>> > q;
3 void dijkstra(int S){
4     vector<bool> vis(n+5);
5     while(!q.empty()){
6         auto cur = q.top(); q.pop();
7         if(vis[cur.second]) continue;
8         vis[cur.second] = true;
9         for(int i = head[cur.second]; i; i = edge[i].nxt){
10             if(dp[edge[i].to][S] > dp[cur.second][S] + edge[i].dis){
```

```

11         dp[edge[i].to][S] = dp[cur.second][S] + edge[i].dis;
12         q.push(make_pair(dp[edge[i].to][S], edge[i].to));
13     }
14 }
15 }
16 }
17
18 int main(){
19     scanf("%d%d%d", &n, &m, &k);
20     for(int i = 1; i <= m; i++){
21         int u, v, w; scanf("%d%d%d", &u, &v, &w);
22         addEdge(u, v, w), addEdge(v, u, w);
23     }
24     for(int i = 1; i <= n; i++)
25         for(int j = 0; j < (1 << k); j++)
26             dp[i][j] = INF;
27     for(int i = 1; i <= k; i++){
28         scanf("%d", &keys[i]);
29         dp[keys[i]][1<<(i-1)] = 0;
30     }
31     for(int S = 1; S < (1 << k); S++){
32         for(int i = 1; i <= n; i++){
33             for(int T = S; T; T = (T - 1) & S)
34                 dp[i][S] = min(dp[i][S], dp[i][T] + dp[i][S ^ T]);
35             if(dp[i][S] != INF) q.push(make_pair(dp[i][S], i));
36         }
37         dijkstra(S);
38     }
39     printf("%lld\n", dp[keys[1]][(1<<k)-1]);
40     return 0;
41 }

```