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。显然这个连通子树是一棵树就够了，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 $\textbf{Dijkstra}$ 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：$dp[i][S]=\min\{dp[i][T]+dp[i][S-T]\boldsymbol{-}\mathbf{a[i]}\mid T\subseteq S\}$ 以及 $dp[j][S]=\min(dp[j][S],dp[i][S]\boldsymbol{+}\mathbf{a[j]})$.

Code：

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

int main(){
    scanf("%d%d%d", &n, &m, &k);
    for(int i = 1; i <= m; i++){
        int u, v, w; scanf("%d%d%d", &u, &v, &w);
        addEdge(u, v, w), addEdge(v, u, w);
    }
    for(int i = 1; i <= n; i++)
        for(int j = 0; j < (1 << k); j++)
            dp[i][j] = INF;
    for(int i = 1; i <= k; i++){
        scanf("%d", &keys[i]);
        dp[keys[i]][1<<(i-1)] = 0;
    }
    for(int S = 1; S < (1 << k); S++){
        for(int i = 1; i <= n; i++){
            for(int T = S; T; T = (T - 1) & S)
                dp[i][S] = min(dp[i][S], dp[i][T] + dp[i][S ^ T]);
```

```
            if(dp[i][S] != INF) q.push(make_pair(dp[i][S], i));
        }
        dijkstra(S);
    }
    printf("%lld\n", dp[keys[1]][(1<<k)-1]);
    return 0;
}
```