算法基础 HW9

PB20111686 黄瑞轩

1

假设原来的最小生成树是 T，去掉的边为 e，若 $e\notin E(T)$，则直接返回 T；若 $e\in E(T)$，则去掉 e 后 T 成为两个连通分量 T_1,T_2，这时遍历所有非 T 的边，找到能连接 T_1,T_2 且权重最小的边 e'，则 $T_1\cup T_2\cup e'$ 就是新的生成树（Kruskal 算法原理保证正确性），此时代价是 $O(|E|)$。

2

假设原来的最小生成树是 T，增加的边为 $e=(u,v)$，由最小生成树的性质，$T\cup e$ 上存在且仅存在一个包含 e 的环，于是可以找到一条 u 到 v 的非平凡路径 P。从 u 开始沿着 P 向前检查路过的边，若这些边权重都大于 $w(e)$，则把权重最大的那条边删除，得到新的生成树 T'，否则原来的 T 仍是新图的生成树，此时代价是 $O(|V|)$。

3

假设用例中有 n 个起点，其组成的点集为 S，我们可以对这 n 个点集调用 Dijkstra 算法 n 次。为了取得路径，可将原 Dijkstra 算法做如下修改：

```
INITIALIZE_SINGLE_SOURCE(G, s)
    for v in G.V
        v.d[s] = ∞
        v.π[s] = ∞
    s.d[s] = 0
    s.π[s] = s

RELAX(u, v, s)
    if v.d[s] > u.d[s] + w(u, v)
        v.π[s] = u
```

所需要的算法：

```
MULTI_SOURCE_DIJSKTRA(G, S)
    for v in S
        DIJKSTRA(G, v)

_PRINT_PATH(G, v, s)
    if v != s
        _PRINT_PATH(G, v.π[s], s)
    print(v)

PRINT_PATH(G, s)
    for v in G.V
        _PRINT_PATH(G, v, s)
```

4

增加一个虚拟节点 s，其对 S 中任意顶点 v 有且仅有一条出边 $s \to v$，且 $w(s, v) = 0$，这样只需要考虑对任意 $u \in T$，$\min_u d(s, u)$ 为所求，这只需要调用一次 Dijkstra 算法即可，最坏情况下时间复杂度为 $O(E \log V)$，得到的最短路径中删除 s 节点即可。