

图应用

Prim算法：实现

11-E5

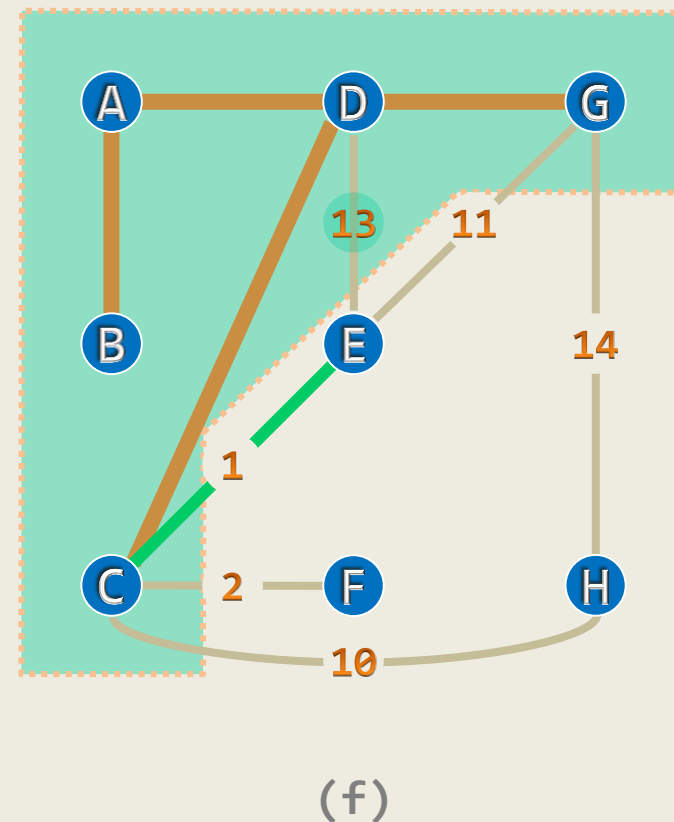
旁边一将，圆睁环眼，倒竖虎须，挺丈八蛇矛，飞马大叫：“三姓家奴休走！燕人张飞在此！” 吕布见了，弃了公孙瓒，便战张飞。

邓俊辉

deng@tsinghua.edu.cn

PFS

- ❖ $\forall v \notin V_k$, let $\text{priority}(v) = \|s, v\| \leq \infty$
- ❖ 于是套用PFS框架, 为将 T_k 扩充至 T_{k+1} , 只需
 - 选出优先级最高的跨边 e_k 及其对应顶点 v_k , 并将其加入 T_k
 - 随后, 更新 $V \setminus V_{k+1}$ 中所有顶点的优先级 (数)
- ❖ 注意: 优先级数随后可能改变 (降低) 的顶点, 必与 v_k 邻接
- ❖ 因此, 只需枚举 v_k 的每一邻接顶点 u , 并取
$$\text{priority}(u) = \min(\text{priority}(u), \|v_k, u\|)$$
- ❖ 以上完全符合PFS的框架, 唯一要做的工作无非是
按照prioUpdater()规范, 编写一个优先级 (数) 更新器...



Priority Updater ~ PrimPU

```
g->pfs( 0, PrimPU<char, Rank>() ); //从顶点0出发, 启动Prim算法
```

```
template <typename Tv, typename Te> struct PrimPU { //Prim算法的顶点优先级更新器
```

```
    virtual void operator()( Graph<Tv, Te>* g, Rank v, Rank u ) { //对v的每个
```

```
        if ( UNDISCOVERED != g->status(u) ) return; //尚未被发现的邻居u, 按
```

```
        if ( g->priority(u) > g->weight(v, u) ) { //Prim
```

```
            g->priority(u) = g->weight(v, u); //策略
```

```
            g->parent(u)    = v; //做松弛
```

```
        }
```

```
    }
```

```
};
```