

## 6. 图

### Prim算法 实现

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

❖ 对于  $V_k$  之外的每一顶点  $v$  , 令 :

$$\text{priority}(v) = \text{v到}V_k\text{的距离}$$

于是套用 **优先级遍历** 算法框架...

❖ 为将  $T_k$  扩充至  $T_{k+1}$  , 可以

- 选出优先级最高的 ( **极短** ) 跨边  $e_k$  , 及其对应顶点  $u_k$  , 并将其加入  $T_k$
- 随后 , 更新  $V_{k+1}$  之外每一顶点的优先级 ( 数 )

❖ 注意：优先级数随后可能改变（降低）的顶点，必与 $u_k$ 邻接

❖ 因此，只需

- 枚举 $u_k$ 的每一邻接顶点 $v$ ，并取
- $\text{priority}(v) = \min( \text{priority}(v), |u_k, v| )$

❖ 以上完全符合PFS的框架，唯一要做的工作无非是

按照`prioUpdater()`模式

编写一个优先级（数）更新器...

## PrioUpdater()

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

❖ `template <typename Tv, typename Te>` //顶点类型、边类型

`struct PrimPU {` //Prim算法的顶点优先级更新器

`virtual void operator()( Graph<Tv, Te>* g, int uk, int v ) {` //对 $u_k$ 的每个

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

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

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

`g->parent(v) = uk;` //做松弛 ( $v \sim 1v = 吕$ )

`}`

`}`

`};`