

图应用

Dijkstra算法：实现

07-C5

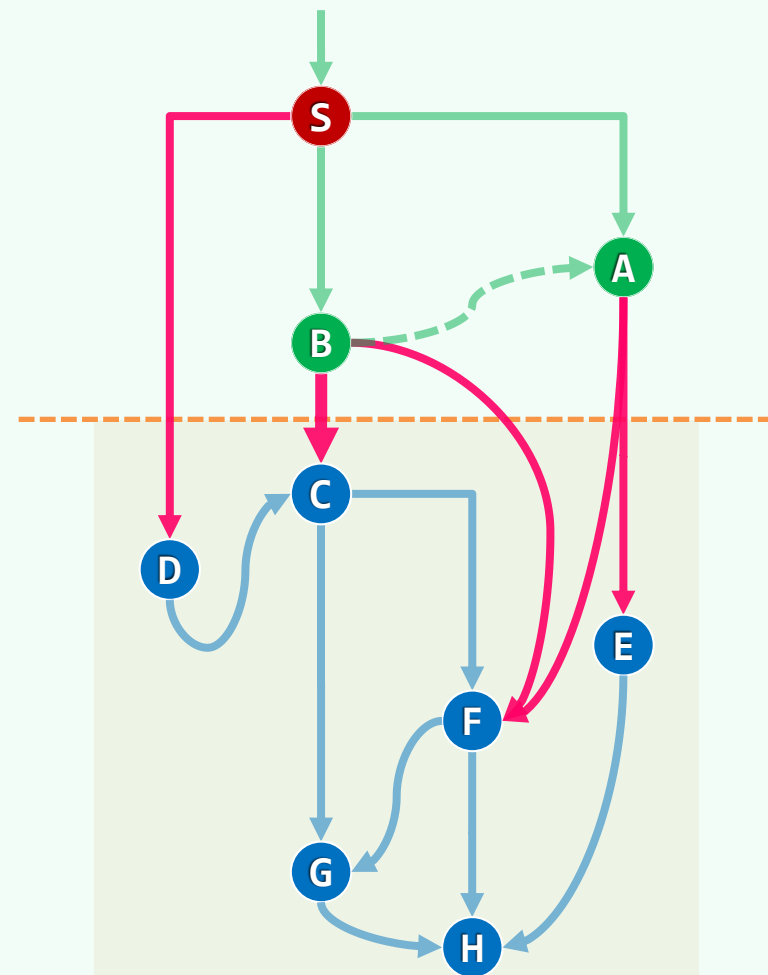
他永远也不能了解，也不能向自己解释，他当时已经疲惫不堪、精疲力竭了，他理应走最短最直的路回家，而他为什么偏要走没有必要经过的干草市场回家去呢？虽然绕的路并不远，但是这显然完全没有必要。

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  $u_k$ ，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  $u_k$  邻接
- ❖ 因此，只需枚举  $u_k$  的每一邻接顶点  $v$ ，并取
$$\text{priority}(v) = \min(\text{priority}(v), \text{priority}(u_k) + \|u_k, v\|)$$
- ❖ 以上完全符合PFS的框架，唯一要做的工作无非是  
按照prioUpdater()规范，编写一个优先级（数）更新器...



## PrioUpdater()

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

❖ template <typename Tv, typename Te> struct DijkPU { //Dijkstra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->priority(uk) + g->weight(uk, v) ) { //Dijkstra

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

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

        }

    }

};
```