算法设计与分析

LECTURE 8

Lecture 8

图优化-贪心

Outline

- 贪心策略
- 最小生成树问题
  - Prim 算法
  - Kruskal 算法
- 给定源点最短路径问题
  - Dijkstra 算法

贪心策略

剑指 Offer II 103. 最少的硬币数目

难度 中等 13 收藏 分享 切换为英文 接收动态 反馈

给定不同面额的硬币 `coins` 和一个总金额 `amount`。编写一个函数来计算可以凑成总金额所需的最少的硬币个数。如果没有任何一种硬币组合能组成总金额，返回 `-1`。

你可以认为每种硬币的数量是无限的。

示例 1:

```
输入：coins = [1, 2, 5], amount = 11
输出：3
解释：11 = 5 + 5 + 1
```

示例 2:

```
输入：coins = [2], amount = 3
输出：-1
```

贪心策略

- 零钱兑换问题：
  - 候选：不同面额的硬币，如 1，5，10，25
  - 限制：总金额
  - 优化：最少的硬币个数

- 贪心策略
  - 每次尽可能选择金额最大的硬币

贪心策略

- 贪心策略不一定保证最优解

- 例如，总金额 15，可用硬币 {1，5，12}
  - 贪心选择：{12，1，1，1}
  - 最优解：{5，5，5}

最小生成树 MST

- 无向图连通G的生成树T是其子图，满足：
  - T 包含图 G 中的所有顶点
  - T 是连通无环图，即一棵树

- 定义一棵生成树的权为其所有边权的和。
- 最小生成树为权重最小的生成树。

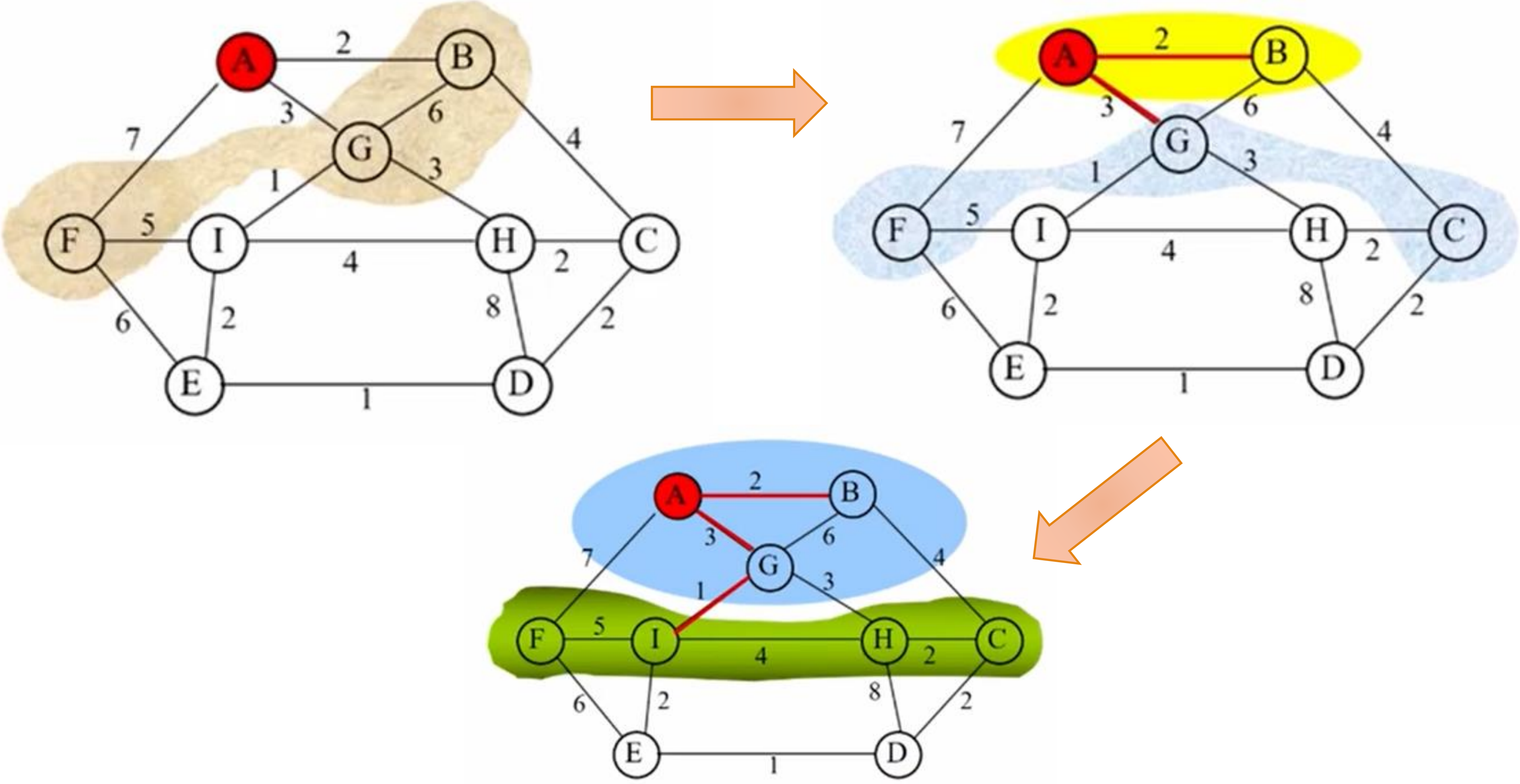
- 图中的最小生成树不一定是唯一的。

MST 贪心求解

- Prim 算法：
  - Select: 局部最优选择（无环）

- Kruskal 算法：
  - Select: 权重最小的边
  - Check: 无环

Prim 算法

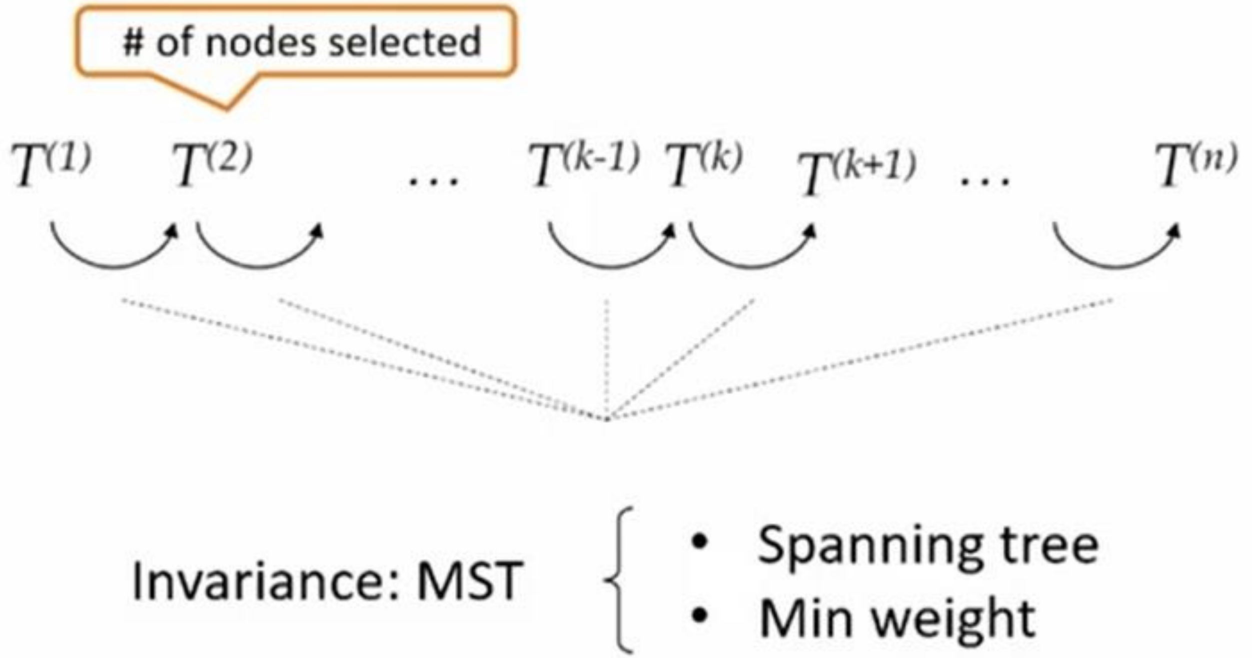

A B G F I H C E D
2 3 7 6 4 1 3 5 4 2 6 2 8 2 1
A B G F I H C E D
2 3 7 6 4 1 3 5 4 2 6 2 8 2 1
A B G F I H C E D
2 3 7 6 4 1 3 5 4 2 6 2 8 2 1

Prim 算法

- 构建一棵生成树
  - 局部生成树不断生长，直至包含图中所有节点的过程
- 保证生成树的权重最小
  - 贪心策略：贪心地选择权重最小的边加入

Prim 算法的正确性

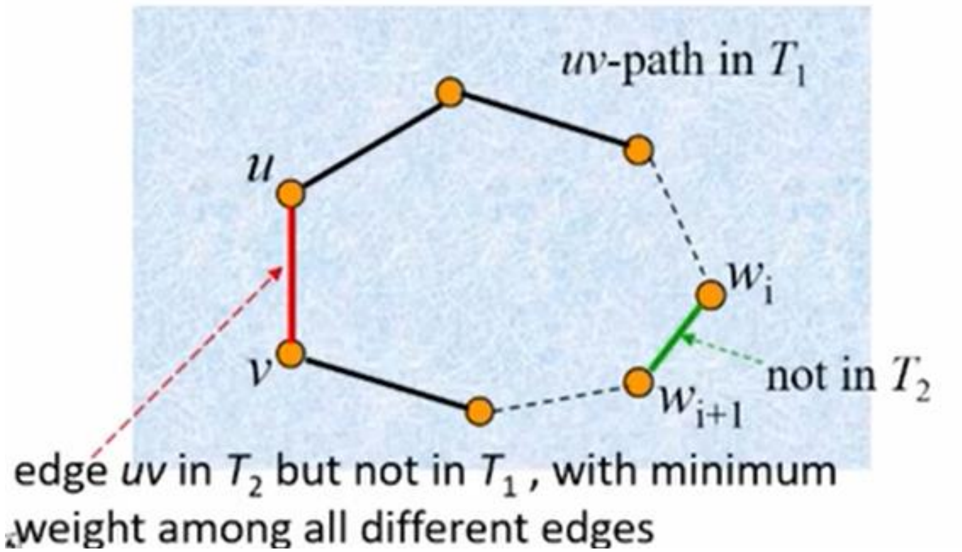
- 数学归纳法证明
- 归纳不变式：局部生成树总是局部最小生成树



最小生成树性质

- (最小生成树-间接定义) 给定图 G 的生成树 T，定义 T是图G的最小生成树，如果它满足“最小生成树性质”：对于任意不在 T 中的边 e，$T\cup\{e\}$ 含有一个环，并且 e 是环中最大权值的边（可能不唯一）。

- 引理：所有满足最小生成树性质的生成树T都具有相同的权值。



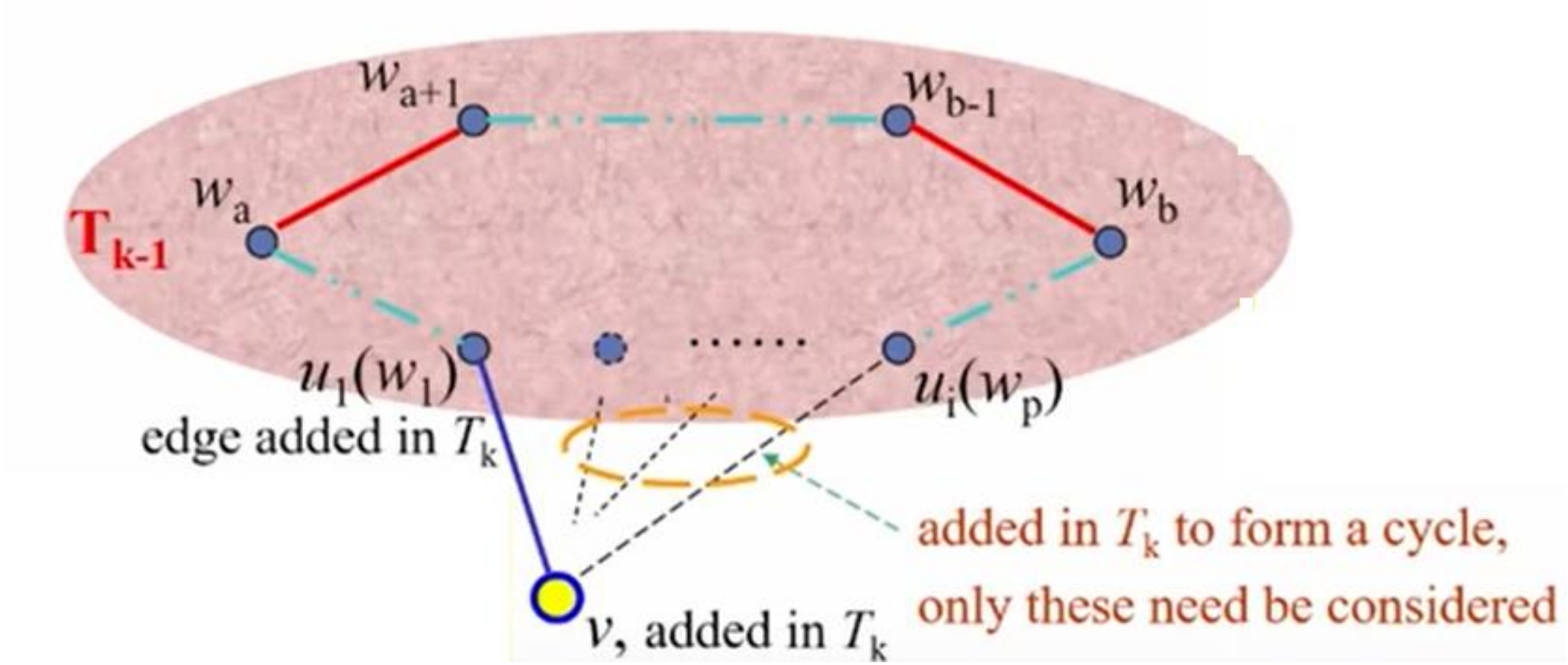
最小生成树性质

- 定理10.1 T 是最小生成树当且仅当 T 具有最小生成树性质

- 证明：
  - =>：反证假设它不满足最小生成树性质，则存在边 $e \notin T, T \cup \{e\}$中存在一个环，而环中存在比 e 权重大的边 e’，此时将 e’ 去掉得到 T’，T’.weight < T.weight，矛盾。
  - <=：根据引理10.1，满足最小生成树性质的生成树具有相同权值，易证具有最小生成树性质的生成树是最小生成树。

Prim 算法的正确性

- 归纳不变式：局部生成树总是局部最小生成树（满足最小生成树性质）



Prim 算法的实现

- Fringe 节点维护成一个优先队列

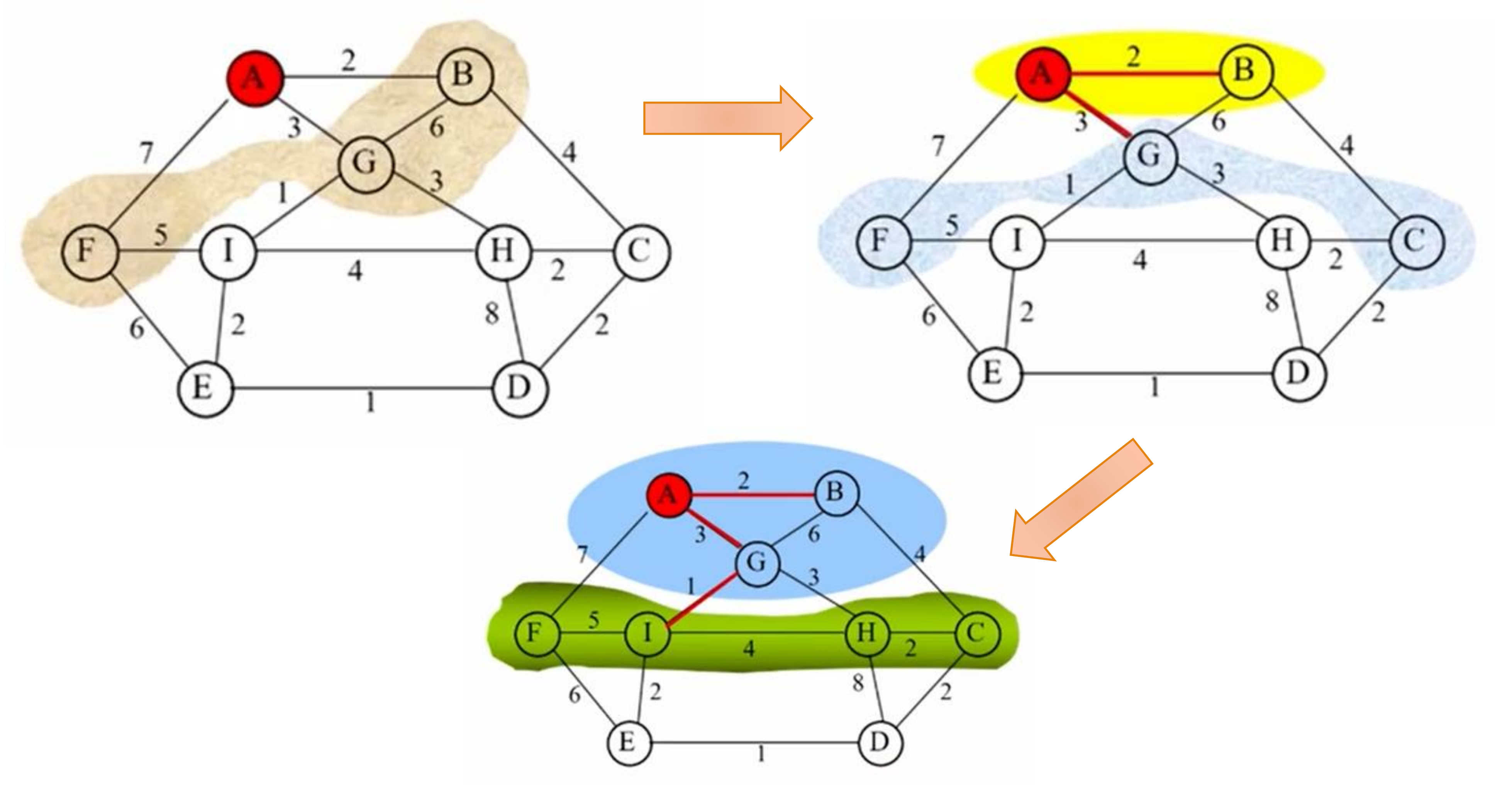
- 对于已经在优先队列中的点，算法需要随时检查是否需要更新它的优先级

Main Procedure

```
primMST(G,n)
  Initialize the priority queue pq as empty;
  Select vertex s to start the tree;
  Set its candidate edge to (-1,s,0);
  insert(pq,s,0);
  while (pq is not empty)
    v=getMin(pq); deleteMin(pq);
    add the candidate edge of v to the tree;
    updateFringe(pq,G,v);
  return
```

getMin(pq) always be the vertex with the smallest key in the fringe set.

Prim 算法



Prim 算法的分析

- 从节点的角度
  - 执行 n 次 insert 以及 getMin, deleteMin 操作
- 从边的角度
  - 已经在优先队列中的要检查优先级是否需要调整，执行至多 m 次 decreaseKey 操作

- Prim 算法的代价：

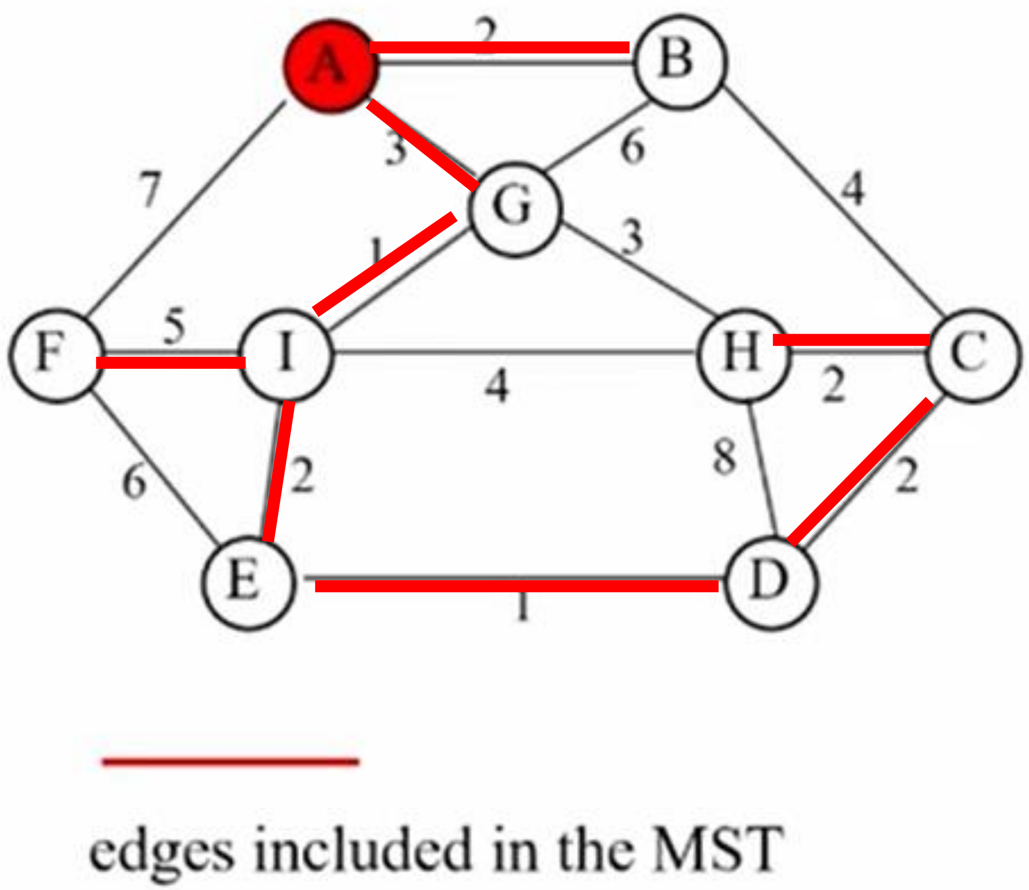
$$T(n,m) = O(n \cdot C_{EXTRACT-MIN} + n \cdot C_{INSERT} + m \cdot C_{DECREASE-KEY})$$

Prim 算法的分析

- 基于数组实现优先队列：$O(n^2 + m)$
- 基于堆实现优先队列：$O((m + n)\log n)$

操作	数组	堆
INSERT	O(1)	O(log n)
EXTRACT-MIN	O(n)	O(log n)
DECREASE-KEY	O(1)	O(log n)

Kruskal 算法



Kruskal 算法实现与分析

```
• kruskalMST(G,n,F) //outline
•     int count;
•     Build a minimizing priority queue, pq, of edges of G, prioritized by weight.
•     Initialize a Union-Find structure, sets, in which each vertex of G is in its own
  set.
•
• F=ϕ;
•     while (isEmpty(pq) == false)
•         vwEdge = getMin(pq);
•         deleteMin(pq);
•         int vSet = find(sets, vwEdge.from);
•         int wSet = find(sets, vwEdge.to);
•         if (vSet ≠ wSet)
•             Add vwEdge to F;
•             union(sets, vSet, wSet)
•     return
```

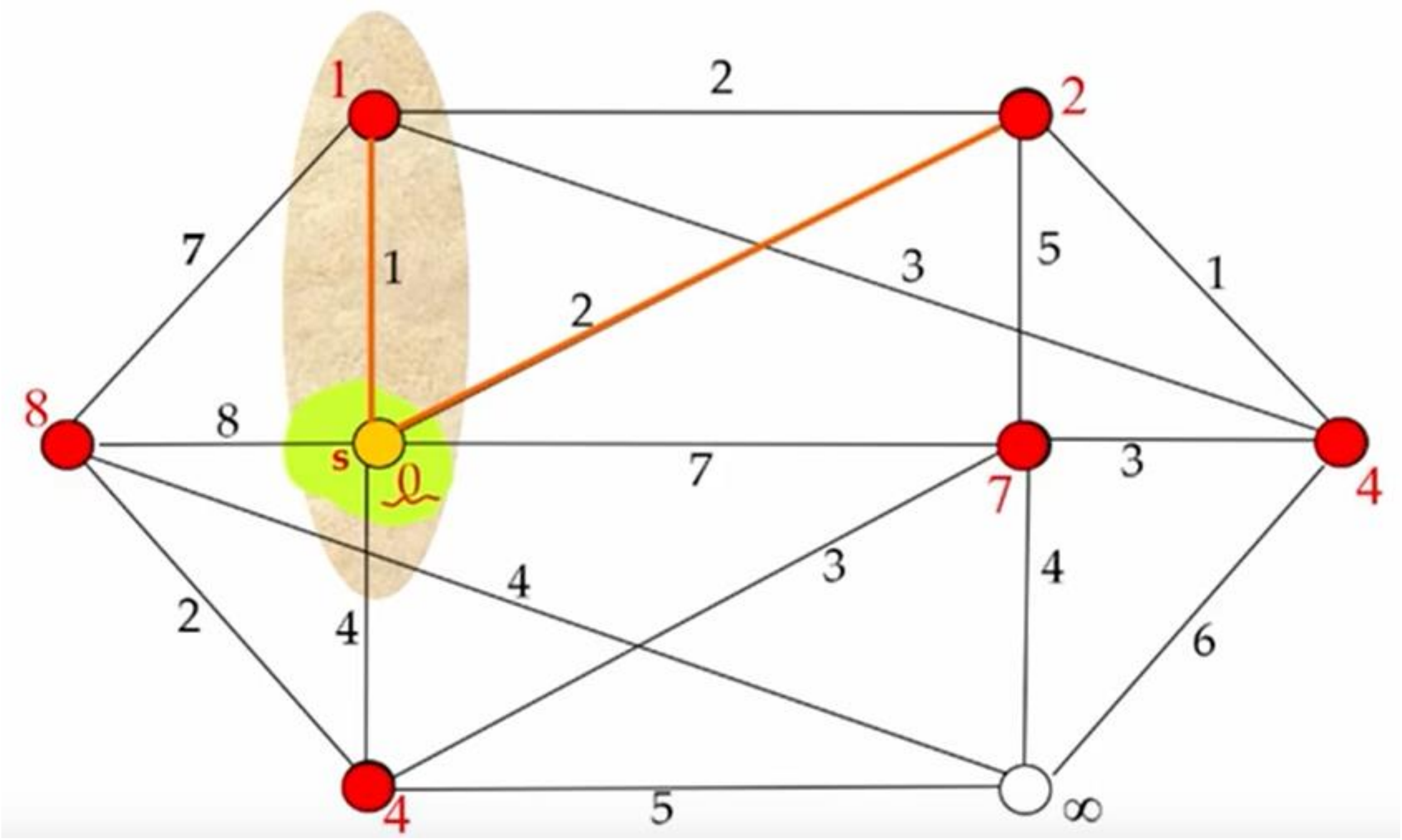
O(m log m)

Kruskal 算法的正确性

- 数学归纳法
  - 归纳不变式：Kruskal 算法得到的局部生成森林 $F^{(k)}$总是最小生森林

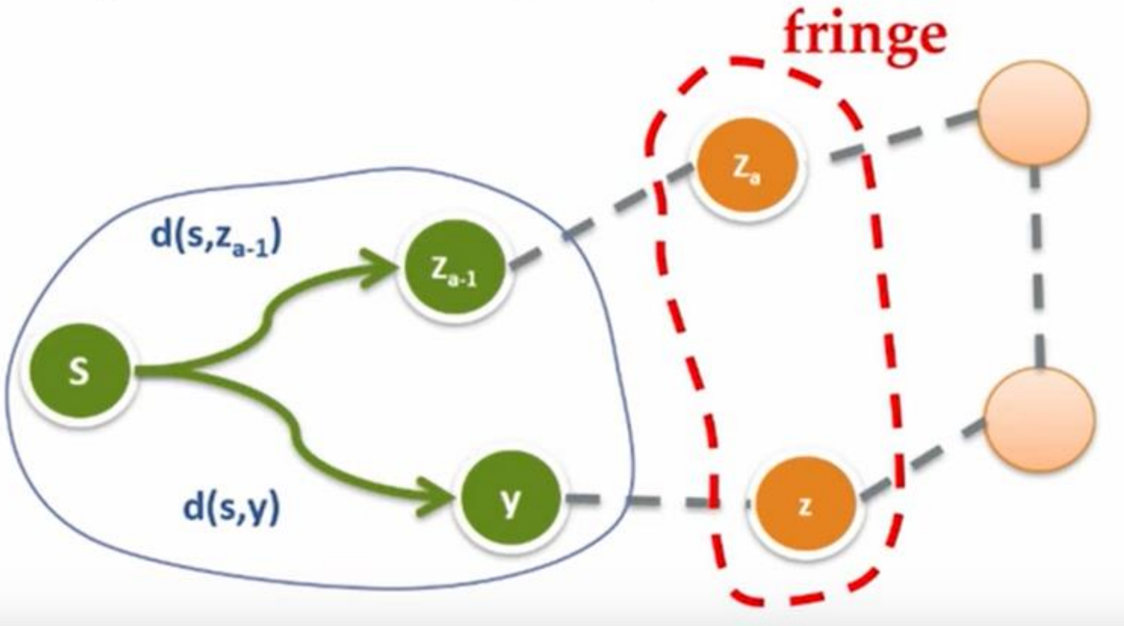
- 假设 $F^{(k)} = F^{(k-1)} \cup \{e\}$，如果 $e \in T$, 易证；假设 $e \notin T$，则 $T \cup \{e\}$中有环。下面证必有边 e’ 与 e 权值相等。
  - 根据最小生成树性质，e是环上权最大的边之一，$e'.weight \le e.weight$
  - 根据 Kruskal 算法贪心原则，$e'.weight \ge e.weight$

Dijkstra 算法



Dijkstra 算法正确性

- $W(s\text{->}y\text{->}z) < W(s\text{->}z_{a-1}\text{->}z_a\text{->}z)$



贪心遍历框架 BestFS

- 在图中贪心地去搜索某种最优结构（如：最小生成树、最短路径等）
- 贪心搜索（Best-First Search）框架

- 节点三种状态：
  - Fresh: 贪心搜索尚未涉及的节点。
  - Fringe: 进行贪心选择的所有候选节点。
  - Finished: 已完成贪心搜索，无须后续处理的节点。

贪心遍历框架 BestFS

- 当点 x 完成从 Fringe 向 Finished 的转换时，它的所有邻居一一被更新。

- 邻居节点为Fresh：
  - 加入到当前已发现的候选节点集合。
- 邻居节点为Fringe:
  - 需要检查节点 x 的状态变化是否导致节点 y 的贪心指标的更新
    - Prim：更新边权重的信息
    - Dijkstra：更新节点的最短路径

Thanks