

第五章 贪心算法

骆吉洲
计算机科学与技术学院

提要

- 5.1 贪心算法原理
- 5.2 活动选择问题
- 5.3 哈夫曼编码
- 5.4 最小生成树问题

参考资料

《Introduction to Algorithms》

- 第16章: 16.1, 16.2, 16.3, 16.4, 16.5
23.1, 23.2

《课件》

- 第五章

5.1 贪心算法原理

- 贪心算法的基本概念
- 贪心选择性
- 优化子结构
- 与动态规划方法的比较
- 贪心算法正确性证明方法

贪心算法的基本概念

- 贪心算法的基本思想
 - 求解最优化问题的算法包含一系列步骤
 - 每一步都有一组选择
 - 作出在当前看来最好的选择
 - 希望通过作出局部优化选择达到全局优化选择

考虑背包容量为50的如下0-1背包问题

- 每次选价值最大的物品
- 每次选单位重量价值最大的物品

| 编号 i | 1 | 2 | 3 | 4 | 5 | 6 |
|-----------|----|-----|-----|-----|----|----|
| 价值 v_i | 60 | 100 | 120 | 140 | 30 | 40 |
| 重量 w_i | 10 | 20 | 30 | 35 | 10 | 20 |
| v_i/w_i | 6 | 5 | 4 | 4 | 3 | 2 |

贪心算法的基本概念

- 贪心算法的基本思想
 - 求解最优化问题的算法包含一系列步骤
 - 每一步都有一组选择
 - 作出在当前看来最好的选择
 - 希望通过作出局部优化选择达到全局优化选择
- 考虑生活常识: 司机利用贪心策略总使加油次数最小
- 第一次加油位置是合理的
- 从A出发不加油最远到达加油 S_k
必存在最优加油策略在 S_k 首次加油





贪心算法的基本概念

贪心算法的基本思想

- 求解最优化问题的算法包含一系列步骤
- 每一步都有一组选择
- 作出在当前看来最好的选择
- 希望通过作出局部优化选择达到全局优化选择

考虑生活常识: 司机利用贪心策略总使加油次数最小

- 第一次加油位置是合理的
- 贪心选择和剩下子问题的解一起构成原问题的解
- 数学归纳法



贪心算法的基本概念

贪心算法的基本思想

- 求解最优化问题的算法包含一系列步骤
- 每一步都有一组选择
- 作出在当前看来最好的选择
- 希望通过作出局部优化选择达到全局优化选择

- 贪心算法不一定总产生优化解
- 贪心算法是否产生优化解，需严格证明

贪心算法产生优化解的条件

- 贪心选择性
- 优化子结构



贪心选择性

贪心选择性

若一个优化问题的全局优化解可以通过局部优化选择得到，则该问题称为具有 Greedy 选择性。

一个问题是否具有贪心选择性需证明

- 证明贪心选择的合理性 贪心选择性
- 证明优化子结构
- 数学归纳法 过程相同，不是本质



优化子结构

若一个优化问题的优化解包含它的子问题的优化解，则称其具有优化子结构



与动态规划方法的比较

动态规划方法可用的条件

- 优化子结构
- 子问题重叠性
- 子问题空间小

贪心方法可用的条件

- 优化子结构
- 贪心选择性

- 可用贪心方法时，动态规划方法可能不适用
- 可用动态规划方法时，贪心方法可能不适用



贪心算法正确性证明方法

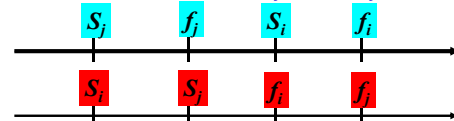
- 证明算法所求解的问题具有贪心选择性
- 证明算法所求解的问题具有优化子结构
- 证明算法确实按照贪心选择性进行局部优化选择

5.2 活动选择问题

- 问题的定义
- 优化解的结构分析
- 算法设计
- 算法复杂性
- 算法正确性证明

问题的定义

- 活动
 - 设 $S=\{1,2,\dots,n\}$ 是 n 个活动的集合，各个活动使用同一个资源，资源在同一时间只能为一个活动使用
 - 每个活动 i 有起始时间 s_i ，终止时间 f_i ， $s_i \leq f_i$
- 相容活动
 - 活动 i 和 j 是相容的，若 $s_j \geq f_i$ 或 $s_i \geq f_j$ ，即



问题定义

- 输入: $S=\{1, 2, \dots, n\}$, $F=\{[s_i, f_i]\}$, $n \geq 1$
- 输出: S 的最大相容集合

贪心思想:

为了选择最多的相容活动，每次选 f_i 最小的活动，使我们能够选更多的活动

优化解结构分析

引理1 设 $S=\{1,2,\dots,n\}$ 是 n 个活动集合， $[s_i, f_i]$ 是活动的起始终止时间，且 $f_1 \leq f_2 \leq \dots \leq f_n$ ， S 的活动选择问题的某个优化解包括活动1。

证 设 A 是一个优化解，按结束时间排序 A 中活动，设其第一个活动为 k ，第二个活动为 j 。
如果 $k=1$ ，引理成立。
如果 $k \neq 1$ ，令 $B=A-\{k\} \cup \{1\}$ ，
由于 A 中活动相容， $f_1 \leq f_k \leq s_j$ ， B 中活动相容。
因为 $|B|=|A|$ ，所以 B 是一个优化解，且包括活动1。

引理2. 设 $S=\{1, 2, \dots, n\}$ 是 n 个活动集合， $[s_i, f_i]$ 是活动 i 的起始终止时间，且 $f_1 \leq f_2 \leq \dots \leq f_n$ ，设 A 是 S 的调度问题的一个优化解且包括活动1，则 $A \setminus \{1\}$ 是 $S'=\{i \in S \mid s_i \geq f_1\}$ 的调度问题的优化解。

引理2说明活动选择问题具有优化子结构

令 $B=\{1\} \cup B'$ 。对于 $\forall i \in S'$, $s_i \geq f_1$, B 中活动相容。
 B 是 S 的一个解。

由于 $|A|=|A'|+1$, $|B|=|B'|+1 > |A'|+1=|A|$ ，与 A 最大矛盾。

贪心选择性

引理3. 设 $S=\{1, 2, \dots, n\}$ 是 n 个活动集合， $f_0=0$, I_i 是 $S_i=\{j \in S \mid s_j \geq f_{i-1}\}$ 中具有最小结束时间 f_{i-1} 的活动。设 A 是 S 的包含活动1的优化解，其中

$$f_1 \leq \dots \leq f_n, \text{ 则 } A = \bigcup_{i=1}^k I_i$$

证. 对 $|A|$ 作归纳法。

当 $|A|=1$ 时，由引理1，命题成立。

设 $|A|<k$ 时，命题成立。

当 $|A|=k$ 时，由引理2， $A=\{1\} \cup A_1$ ，

A_1 是 $S_2=\{j \in S \mid s_j \geq f_1\}$ 的优化解。

由归纳假设， $A_1 = \bigcup_{i=2}^k I_i$ 。于是， $A = \bigcup_{i=1}^k I_i$ 。



算法的设计

- 贪心思想

为了选择最多的相容活动，每次选 f_i 最小的活动，使我们能够选更多的活动



- 算法

(设 f_1, f_2, \dots, f_n 已排序)

Greedy-Activity-Selector(S, F)

$n \leftarrow \text{length}(S);$

$A \leftarrow \{1\}$

$j \leftarrow 1$

For $i \leftarrow 2$ To n Do

 If $s_i \geq f_j$

 Then $A \leftarrow A \cup \{i\}; j \leftarrow i;$

Return A



复杂性设计

- 如果结束时间已排序

$T(n) = \theta(n)$

- 如果 结束时间未排序

$T(n) = \theta(n) + \theta(n \log n) = \theta(n \log n)$



算法正确性证明

- 需要证明

- 活动选择问题具有贪心选择性
- 活动选择问题具有优化子结构
- 算法按照贪心选择性计算解



定理. Greedy-Activity-Selector算法能够产生最优解.

证. Greedy-Activity-Selector算法按照引理3的贪心选择性进行局部优化选择.

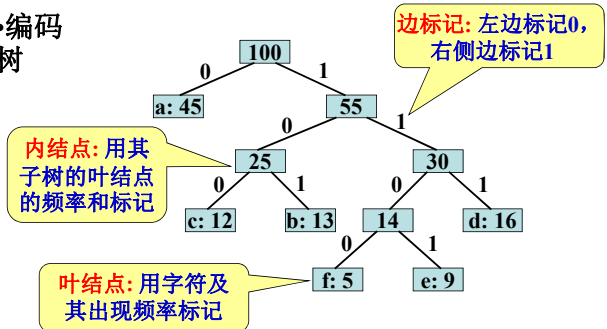


5.3 哈夫曼编码

- 问题的定义
- 优化解的结构分析
- 算法设计
- 算法复杂性分析
- 算法正确性证明

- 二进制字符编码
 - 每个字符用一个二进制0、1串来表示.
- 固定长编码
 - 每个字符都用相同长的0、1串表示.
- 可变长编码
 - 经常出现的字符用短码, 不经常出现的用长码
- 前缀编码
 - 无任何字符的编码是另一个字符编码的前缀

• 编码树



- 编码树 T 的代价
 - 设 C 是字母表, $\forall c \in C$
 - $f(c)$ 是 c 在文件中出现的频率
 - $d_T(c)$ 是叶子 c 在树 T 中的深度, 即 c 的编码长度
 - T 的代价是编码一个文件的所有字符的代码位数:

$$B(T) = \sum_{c \in C} f(c) d_T(c)$$

• 优化编码树问题

输入: 字母表 $C = \{c_1, c_2, \dots, c_n\}$,
频率表 $F = \{f(c_1), f(c_2), \dots, f(c_n)\}$
输出: 具有最小 $B(T)$ 的 C 前缀编码树

贪心思想:

循环地选择具有最低频率的两个结点,
生成一棵子树, 直至形成树

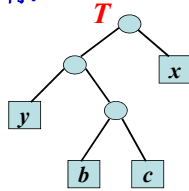
优化的结构分析

- 我们需要证明
 - 优化前缀树问题具有贪心选择性
 - 优化前缀树问题具有优化子结构

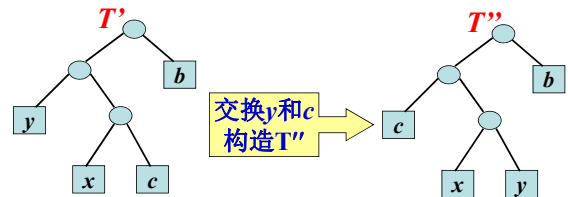
• 贪心选择性

引理1. 设 C 是字母表, $\forall c \in C$, c 具有频率 $f(c)$, x, y 是 C 中具有最小频率的两个字符, 则存在一个 C 的优化前缀树, x 与 y 的编码具有相同长度, 且仅在最末一位不同.

证: 设 T 是 C 的优化前缀树, 且 b 和 c 是具有最大深度的两个兄弟字符:



不失一般性, 设 $f(b) \leq f(c)$, $f(x) \leq f(y)$. 因 x 与 y 是具有最低频率的字符, $f(b) \geq f(x)$, $f(c) \geq f(y)$. 交换 T 的 b 和 x , 从 T 构造 T' :



往证 T' 是最优化前缀树.

$$\begin{aligned} B(T) - B(T') &= \sum_{c \in C} f(c)d_T(c) - \sum_{c \in C} f(c)d_{T'}(c) \\ &= f(x)d_T(x) + f(b)d_T(b) - f(x)d_{T'}(x) - f(b)d_{T'}(b) \\ &= f(x)d_T(x) + f(b)d_T(b) - f(x)d_T(b) - f(b)d_T(x) \rightarrow \text{按} d_T(x) \text{整理} \\ &= (f(b) - f(x))(d_T(b) - d_T(x)). \end{aligned}$$

$\therefore f(b) \geq f(x)$, $d_T(b) \geq d_T(x)$ (因为 b 的深度最大)

$\therefore B(T) - B(T') \geq 0$, $B(T) \geq B(T')$

同理可证 $B(T') \geq B(T'')$. 于是 $B(T) \geq B(T'')$.

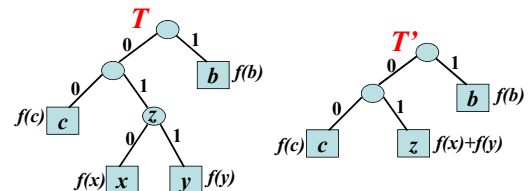
由于 T 是最优化的, 所以 $B(T) \leq B(T'')$.

于是, $B(T) = B(T'')$, T'' 是 C 的最优化前缀编码树.

在 T'' 中, x 和 y 具有相同长度编码, 且仅最后一位不同.

• 优化子结构

引理2. 设 T 是字母表 C 的优化前缀树, $\forall c \in C$, $f(c)$ 是 c 在文件中出现的频率. 设 x 、 y 是 T 中任意两个相邻叶结点, z 是它们的父结点, 则 z 作为频率是 $f(z) = f(x) + f(y)$ 的字符, $T' = T - \{x, y\}$ 是字母表 $C' = C - \{x, y\} \cup \{z\}$ 的优化前缀编码树.



证. 往证 $B(T) = B(T') + f(x) + f(y)$.

$\forall v \in C - \{x, y\}$, $d_T(v) = d_{T'}(v)$, $f(v)d_T(v) = f(v)d_{T'}(v)$.

由于 $d_T(x) = d_T(y) = d_T(z) + 1$,

$$\begin{aligned} f(x)d_T(x) + f(y)d_T(y) &= (f(x) + f(y))(d_T(z) + 1) \\ &= (f(x) + f(y))d_T(z) + (f(x) + f(y)) \end{aligned}$$

由于 $f(x) + f(y) = f(z)$, $f(x)d_T(x) + f(y)d_T(y) = f(z)d_T(z) + (f(x) + f(y))$.

于是 $B(T) = B(T') + f(x) + f(y)$.

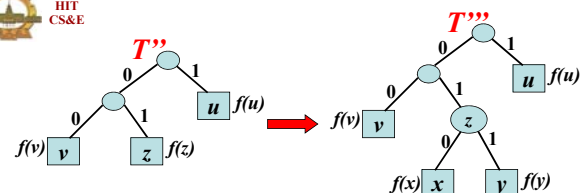
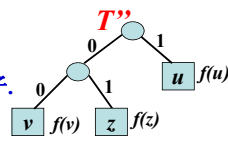
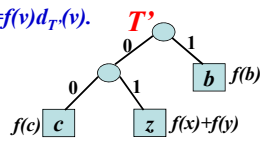
若 T' 不是 C' 的优化前缀编码树,

则必存在 T'' , 使 $B(T') < B(T'')$.

因为 z 是 C' 中字符, 它必为 T'' 中的叶子.

把结点 x 与 y 加入 T'' , 作为 z 的子结点,

则得到 C 的一个如下前缀编码树 T''' :



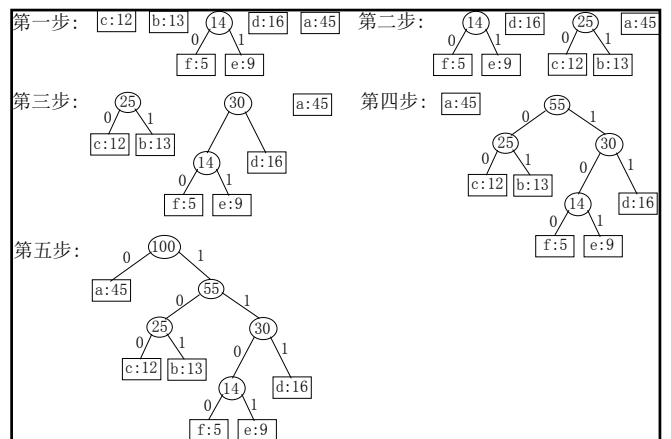
T''' 代价为:

$$\begin{aligned} B(T''') &= \dots + (f(x) + f(y))(d_{T''}(z) + 1) \\ &= \dots + f(z)d_{T''}(z) + (f(x) + f(y)) \quad (d_{T''}(z) = d_{T'}(z)) \\ &= B(T') + f(x) + f(y) < B(T') + f(x) + f(y) = B(T) \end{aligned}$$

与 T 是优化的矛盾, 故 T' 是 C' 的优化编码树.

• 基本思想

- 循环地选择具有最低频率的两个结点，生成一棵子树，直至形成树
- 初始: $f:5, e:9, c:12, b:13, d:16, a:45$



• Greedy算法(使用堆操作实现)

Huffman(C, F)

1. $n \leftarrow |C|$;
2. $Q \leftarrow C$; /* 用BUILD-HEAP建立堆 */
3. FOR $i \leftarrow 1$ TO $n-1$ DO
4. $z \leftarrow \text{Allocate-Node}()$;
5. $x \leftarrow \text{left}[z] \leftarrow \text{Extract-MIN}(Q)$; /* 堆操作 */
6. $y \leftarrow \text{right}[z] \leftarrow \text{Extract-MIN}(Q)$; /* 堆操作 */
7. $f(z) \leftarrow f(x) + f(y)$;
8. Insert(Q, z); /* 堆操作 */
9. Return



HIT
CS&E

复杂性分析

- 设 Q 由一个堆实现
- 第2步用堆排序的BUILD-HEAP实现: $O(n)$
- 每个堆操作要求 $O(\log n)$, 循环 $n-1$ 次: $O(n \log n)$
- $T(n) = O(n) + O(n \log n) = O(n \log n)$

定理. Huffman算法产生一个优化前缀编码树

证. 由于引理1、引理2成立,而且哈夫曼算法按照引理2的贪心选择性确定的规则进行局部优化选择,所以哈夫曼算法产生一个优化前缀编码树。



HIT
CS&E

5.4 最小生成树

- 问题的定义
- 优化解结构分析
- 贪心选择性
- Kruskal算法
- 算法复杂性
- 算法正确性证明

•生成树

- 设 $G=(V, E)$ 是一个边加权无向连通图. G 的生成树是无向树 $S=(V, T)$, $T \subseteq E$, 以下用 T 表示 S .
- 如果 $W: E \rightarrow \{\text{实数}\}$ 是 G 的权函数, T 的权值定义为 $W(T) = \sum_{(u,v) \in T} W(u,v)$.

•最小生成树

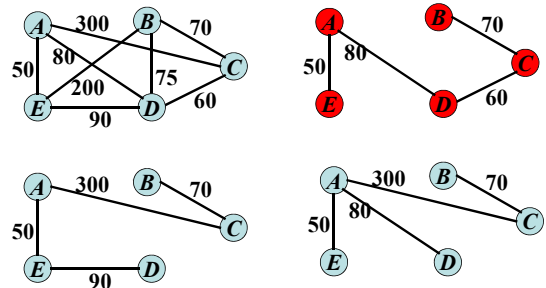
- G 的最小生成树是 $W(T)$ 最小的 G 之生成树.

•问题的定义

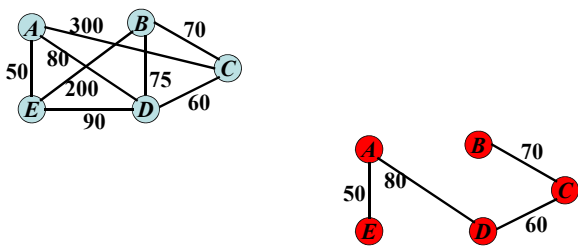
输入: 无向连通图 $G=(V, E)$, 权函数 W

输出: G 的最小生成树

•实例



•算法思想



定理1. 设 uv 是 G 中权值最小的边, 则必有一棵最小生成树包含边 uv .

证明: 设 T 是 G 的一棵 MST

若 $uv \in T$, 结论成立;

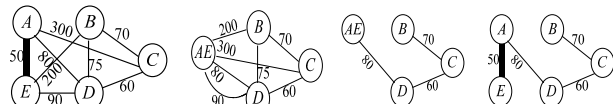
否则, 如右图所示

在 T 中添加 uv 边, 产生环

删除环中不同于 uv 的权值最小的边 xy , 得到 T' .

$$w(T') = w(T) - w(xy) + w(uv) \leq w(T)$$

T'



收缩图 G 的边 uv — G_{uv}

- 用新顶点 C_{uv} 代替边 uv
- 将 G 中原来与 u 或 v 关联的边与 C_{uv} 关联
- 删除 C_{uv} 到其自身的边

上述操作的逆操作称为 **扩张**

定理1. 给定加权无向连通图 $G=(V, E)$, 权值函数为 $W: E \rightarrow R$, $uv \in E$ 是 G 中权值最小的边. 设 T 是 G 的包含 uv 的一棵最小生成树, 则 $T-uv$ 是 G_{uv} 的一棵最小生成树.

证明. 由于 $T-uv$ 是不含回路的连通图且包含了 G_{uv} 的所有顶点, 因此, $T-uv$ 是 G_{uv} 的一棵生成树. 下面证明 $T-uv$ 是 G_{uv} 的代价最小的生成树.

若不然, 存在 G_{uv} 的生成树 T' 使得 $W(T') < W(T-uv)$. 显然, T' 中包含顶点 C_{uv} 且是连通的, 因此 $T'' = T' \circ C_{uv}$ 包含 G 的所有顶点且不含回路, 故 T'' 是 G 的一棵生成树. 但, $W(T'') = W(T') + W(uv) < W(T-uv) + W(uv) = W(T)$, 这与 T 是 G 的最小生成树矛盾.

扩张



P86

Kruskal算法

MST-Kruskal(G, W)

1. $A = \emptyset$;
2. For $\forall v \in V/G$ Do
3. Make-Set(v); /*
4. 按照 W 值的递增顺序排序 E/G ;
5. For $\forall (u, v) \in E/G$ (按 W 值的递增顺序) Do
6. If Find-Set(u) \neq Find-Set(v)
7. Then $A = A \cup \{(u, v)\}$; Union(u, v);
8. Return A



算法复杂性

- 令 $n = |V|$, $m = |E|$
- 第4步需要时间: $O(m \log m)$
- 第2-3步执行 $O(n)$ 个Make-Set操作
- 第5-8步执行 $O(m)$ 个Find-Set和Union操作
- 需要时间: $O((n+m)\alpha(n))$
- $m \geq n-1$ (因为 G 连通), $\alpha(n) = \log n = \log m$
- 总时间复杂性: $O(m \log m)$



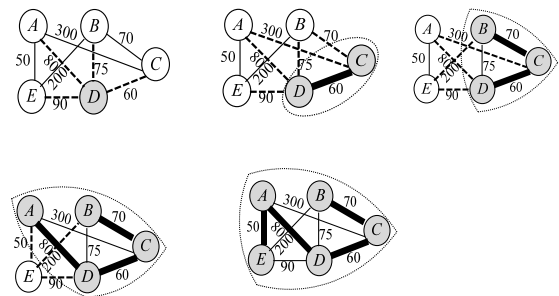
算法正确性

定理2. MST-Kruskal(G, W)算法能够产生图 G 的最小生成树。

证. 因为算法按照贪心选择性进行局部优化选择。

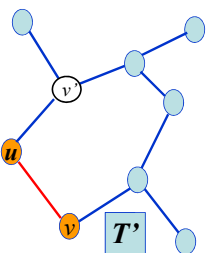
• 算法思想

Prim算法



贪心选择性

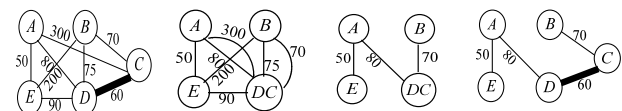
定理1. 设 uv 是 G 中与顶点 u 关联的权值最小的边，则必有一棵最小生成树包含边 uv 。



证明: 设 T 是 G 的一棵MST
若 $uv \in T$, 结论成立;
否则, 如右图所示
在 T 中添加 uv 边, 产生环, 环中顶点 u 的度为2, 即存在 $uv' \in T$.
删除环中边 uv' , 得到 T' .
 $w(T') = w(T) - w(uv') + w(uv) \leq w(T)$



优化子结构



收缩图 G 的边 uv — $G \bullet uv$

- 用新顶点 C_{uv} 代替边 uv
- 将 G 中原来与 u 或 v 关联的边与 C_{uv} 关联
- 删除 C_{uv} 到其自身的边

上述操作的逆操作称为**扩张**



定理1. 给定加权无向连通图 $G=(V,E)$, 权值函数为 $W:E \rightarrow \mathbb{R}$, $uv \in E$ 是 G 中顶点 u 关联的权值最小的边。设 T 是 G 的包含 uv 的一棵最小生成树, 则 $T \cdot uv$ 是 $G \cdot uv$ 的一棵最小生成树。
证明. 同Kruskal算法优化子结构的证明。



算法描述(1)

MST-Prim(G, W, r)

Input 连通图 G , 权值函数 W , 树根 r

Output G 的一棵以 r 为根的生成树

```

1.  $C \leftarrow \{r\}$ ,  $T \leftarrow \emptyset$ ;
2. 建堆  $Q$  维护  $C$  与  $V-C$  之间的边
3. While  $C \neq V$  do
4.    $uv \leftarrow \text{Extract\_Min}(Q)$  //  $u \in C, v \in V-C$ 
5.    $C \leftarrow C \cup \{v\}$ ;  $T \leftarrow T \cup \{uv\}$ ;
6.   for  $\forall x \in \text{Adj}[v]$  do
7.     if  $x \in C$  then 将  $vx$  从  $Q$  中删除
8.     Else          将  $vx$  插入  $Q$ 
9. Return  $T$ 

```

log E

2E 遍

log E



算法描述(2)

MST-Prim(G, W, r)

Input 连通图 G , 权值函数 W , 树根 r

Output G 的一棵以 r 为根的生成树

```

1. For  $\forall v \in V[G]$  Do
2.    $\text{key}[v] \leftarrow +\infty$ 
3.    $\pi[v] \leftarrow \text{null}$ 
4.  $\text{key}[r] \leftarrow 0$ 
5.  $Q \leftarrow V[G]$ 
6. While  $Q \neq \emptyset$  do
7.    $u \leftarrow \text{Extract\_Min}(Q)$  // 找到安全轻边
8.   for  $\forall v \in \text{Adj}[u]$  do
9.     if  $v \in Q$  且  $w(u,v) < \text{key}[v]$  then
10.       $\pi[v] \leftarrow u$ 
11.       $\text{key}[v] \leftarrow w(u,v)$  // 更新信息
12. Return  $A = \{(v, \pi[v]) \mid v \in V[G] - r\}$ 

```

log V

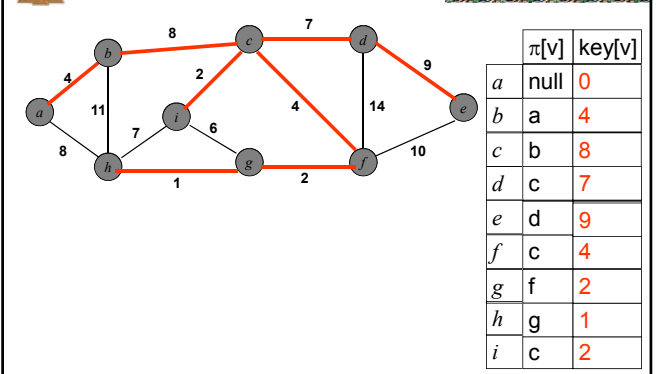
2E 遍

常数时间

log V



实例



算法分析

算法正确性

证明算法第6-11步的while循环具有如下的循环不变量

- $A = \{(v, \pi[v]) \mid v \in V - r - Q\}$
- 已经位于生成树中的顶点集为 $V - Q$
- $\forall v \in Q$, 如果 $\pi[v] \neq \text{null}$
则 $\text{key}[v] < +\infty$, 且 $\text{key}[v]$ 是将 v 连接到当前生成树需要的最小权值

算法复杂性

假设用最小堆实现 Q

总的时间开销为 $O(V \log V + E \log V) = O(E \log V)$



算法正确性

定理2. MST-Prim(G, W) 算法能够产生图 G 的最小生成树。

证. 因为算法按照贪心选择性进行局部优化选择。