HIT CS&E

第五章 贪心算法

骆吉洲
计算机科学与技术学院

HIT CS&E

提要

HIT CS&E

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

HIT CS&E

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

HIT CS&E

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

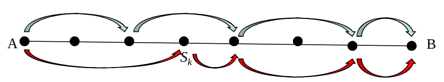
HIT CS&E

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

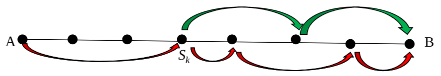


HIT CS&E

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



HIT CS&E

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

HIT CS&E

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

HIT CS&E

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

HIT CS&E

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

HIT CS&E

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

HIT CS&E

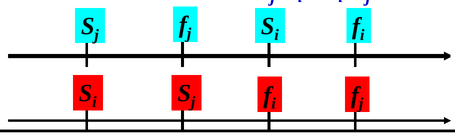
5.2 活动选择问题

- 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

HIT CS&E

问题的定义

- 活动
  - 设$S=\{1,2,\ldots,n\}$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



HIT CS&E

- 问题定义
  - 输入：$S=\{1, 2, \ldots, n\}$，$F=\{ [s_i, f_i] \}$，$n \geq i \geq 1$
  - 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

HIT CS&E

优化解结构分析

引理1 设$S=\{1,2,\ldots,n\}$是n个活动集合，$[s_i, f_i]$是活动的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$，$S$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$S=\{1, 2, \ldots, n\}$是n个活动集合，$[s_i, f_i]$是活动i的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$，设$A$是$S$的调度问题的一个优化解且包括活动1，则$A' = A-\{1\}$是$S' = \{i \in S | s_i \geq f_1\}$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 $S=\{1, 2, \ldots, n\}$ 是 n 个活动集合，$f_0=0$, l_i是 $S_i=\{ j \in S \mid s_j \geq f_{i-1}\}$ 中具有最小结束时间 f_{l_i} 的活动. 设A是S的包含活动1的优化解, 其中 $f_1 \leq \ldots \leq f_n$，则$A= \bigcup_{i=1}^{k} \{l_i\}$

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，$A_1= \bigcup_{i=2}^{k} \{l_i\}$. 于是，$A= \bigcup_{i=1}^{k} \{l_i\}$.

HIT CS&E

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

HIT CS&E

- 算法

  （设$f_1 \le f_2 \le \ldots \le f_n$已排序）

```
Greedy-Activity-Selector(S, F)
n←lenyth(S);
A←{1}
j←1
For  i←2  To  n  Do
     If  s_i ≥ f_j
     Then  A←A∪{i}; j←i;
Return  A
```

HIT CS&E

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

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

HIT CS&E

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

HIT CS&E

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.

HIT CS&E

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

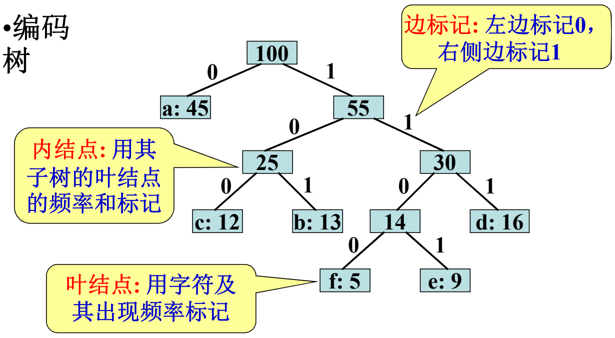
HIT CS&E

问题的定义

- 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

HIT CS&E

•编码树



HIT CS&E

- 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:

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

HIT CS&E

- 优化编码树问题

输入: 字母表 $C = \{c_1, c_2, ..., c_n\}$,

频率表 $F = \{f(c_1), f(c_2), ..., f(c_n)\}$

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

HIT CS&E

优化解的结构分析

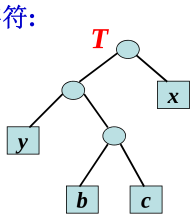
- 我们需要证明
  - 优化前缀树问题具有贪心选择性
  - 优化前缀树问题具有优化子结构

HIT CS&E

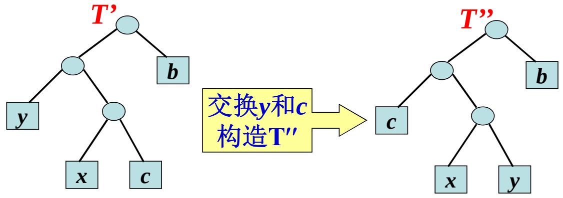
- 贪心选择性

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$f(b)\le f(c)$, $f(x)\le f(y)$. 因x与y是具有最低频率的字符, $f(b)\ge f(x)$, $f(c)\ge f(y)$.交换T的b和x, 从T构造T':



往证T''是最优化前缀树.

$B(T)-B(T')$

$= \sum_{c\in C} f(c)d_T(c) - \sum_{c\in C} f(c)d_{T'}(c)$

$= f(x)d_T(x) + f(b)d_T(b) - f(x)d_{T'}(x) - f(b)d_{T'}(b)$

$= f(x)d_T(x) + f(b)d_T(b) - f(x)d_T(b) - f(b)d_T(x)$

$= (f(b)-f(x))(d_T(b)-d_T(x))$.

$\because f(b)\ge f(x)$, $d_T(b)\ge d_T(x)$ (因为b的深度最大)

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

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

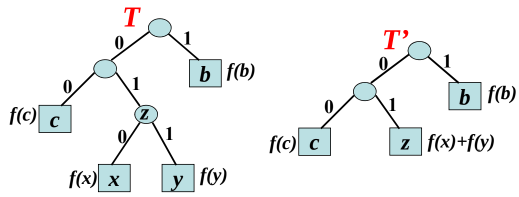
由于T是最优化的，所以$B(T)\le B(T'')$.

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$B(T)=B(T')+f(x)+f(y)$.

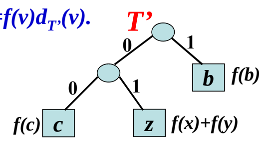
$\forall v\in C-\{x,y\}$, $d_T(v)=d_{T'}(v)$, $f(v)d_T(v)=f(v)d_{T'}(v)$.

由于 $d_T(x)=d_T(y)=d_{T'}(z)+1$,

$f(x)d_T(x)+f(y)d_T(y)$

$=(f(x)+f(y))(d_{T'}(z)+1)$

$=(f(x)+f(y))d_{T'}(z)+(f(x)+f(y))$



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$B(T)=B(T')+f(x)+f(y)$.

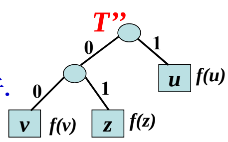
若T'不是C'的优化前缀编码树，

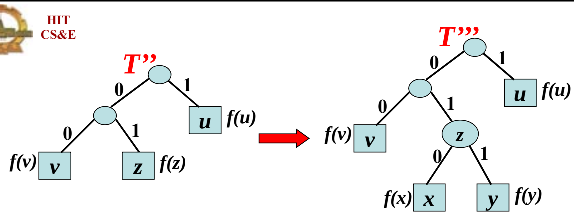
则必存在T''，使$B(T'')<B(T')$.

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:

$B(T''')=\ldots\ldots+(f(x)+f(y))(d_{T''}(z)+1)$

$=\ldots\ldots+f(z)d_{T''}(z)+(f(x)+f(y))$ $(d_{T''}(z)=d_{T'}(z))$

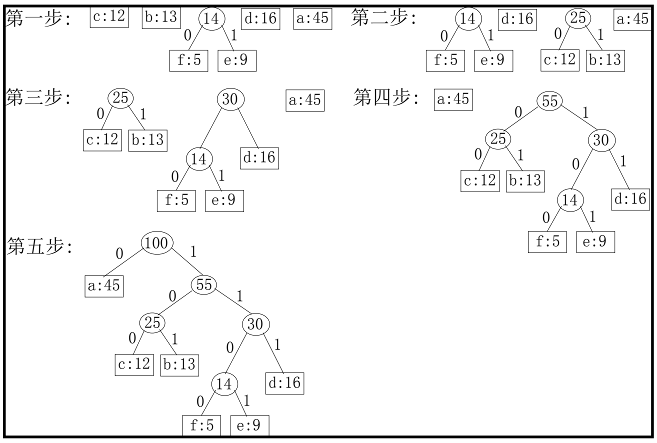
$=B(T'')+f(x)+f(y)<B(T')+f(x)+f(y)=B(T)$

与T是优化的矛盾，故T'是C'的优化编码树.

HIT CS&E

算法的设计

- 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. $z \leftarrow$ Allocate-Node( );
5. $x \leftarrow left[z] \leftarrow$ Extract-MIN(Q); /* 堆操作*/
6. $y \leftarrow right[z] \leftarrow$ 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: $O(n\log n)$
- $T(n)=O(n)+O(n\log n) =O(n\log n)$

HIT CS&E

正确性证明

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

HIT CS&E

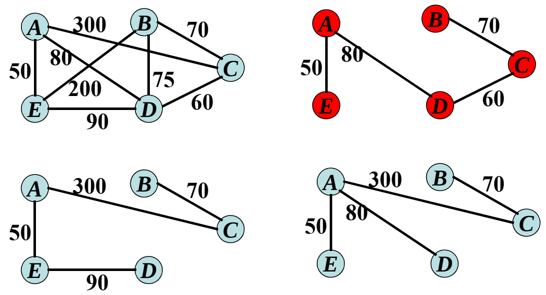
问题的定义

- 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 $W: E\rightarrow\{实数\}$ 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

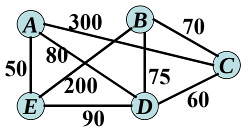
  输入: 无向连通图$G=(V, E)$, 权函数W

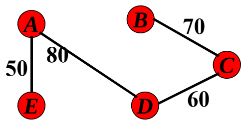
  输出: G的最小生成树

- 实例



- 算法思想

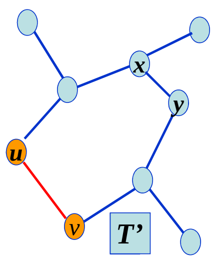




HIT CS&E

贪心选择性

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

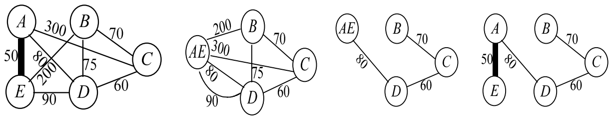
在T中添加uv边，产生环

删除环中不同于uv的权值最小的边xy,得到T'。

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

HIT CS&E

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

HIT CS&E

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$T\cdot uv$是$G.uv$的一棵最小生成树.

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

若不然，存在$G\cdot uv$的生成树T'使得$W(T')<W(T\cdot uv)$。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,$W(T'')= W(T')+W(uv) <W(T\cdot uv)+W(uv)=W(T)$,这与$T$是$G$的最小生成树矛盾。

Kruskal算法

MST-Kruskal(*G*,*W*)
1. A=ϑ;
2. For ∀v∈*V[G]* Do
3. Make-Set(*v*); /*
4. 按照*W*值的递增顺序排序*E[G]*;
5. For ∀(*u, v*)∈*E[G]* (按*W*值的递增顺序) Do
6. If Find-Set(*u*)≠Find-Set(*v*)
7. Then A=A∪{(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(mlogm)*
- 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), $\alpha(n)$=logn=logm
- 总时间复杂性: *O(mlogm)*

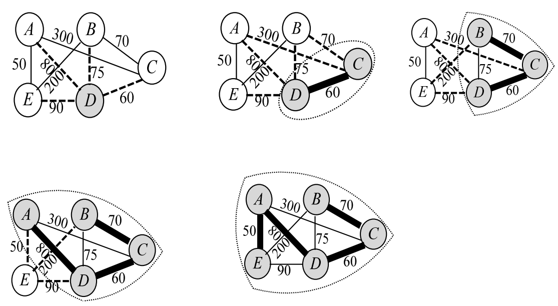
算法正确性

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.

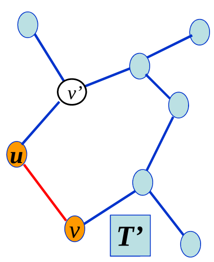
Prim算法

•算法思想



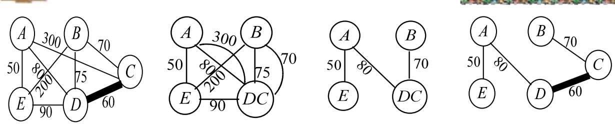
贪心选择性

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* ∈*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'* ∈*T*.
删除环中边*uv'*,得到*T'*。
$w(T')=w(T)-w(xy)+w(uv)\leq w(T)$

优化子结构



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

算法描述(2)

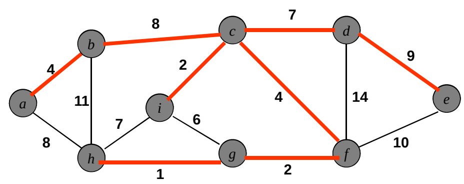
MST-Prim(G,W,r)

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.   key[v]$\leftarrow+\infty$
3. π[v] $\leftarrow$null
4. key[r] $\leftarrow$0
5. $Q\leftarrow V[G]$
6. While $Q\neq\varnothing$ do
7. $u\leftarrow$Extract_Min(Q)   //找到安全轻边   log V
8.   for $\forall v\in$Adj[u] do   2E遍
9.   if $v\in Q$ 且 $w(u,v)<$key[v] then   常数时间
10. π[v] $\leftarrow u$
11.   key[v] $\leftarrow$w(u,v)   //更新信息   logV
12. Return $A=\{(v,\pi[v])\mid v\in V[G]-r\}$

实例



	π[v]	key[v]
a	null	0
b	a	4
c	b	8
d	c	7
e	d	9
f	c	4
g	f	2
h	g	1
i	c	2

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$\pi(v)\neq$null

  则key[v]$<+\infty$,且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(VlogV+ElogV)=O(ElogV)

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.