第五章 贪心算法

骆吉洲
计算机科学与工程系

提要

5.1 *Elements of Greedy Algorithms*
5.2 *An activity-selection problem*
5.3 *Huffman codes*
5.4 *Theoretical foundations of Greedy Algorithms*
5.5 *A task-scheduling problem*
5.6 *Minimal spanning tree problem*
5.7 *Single-sourse shortest path problem*

参考资料

《Introduction to Algorithms》
- 第16章: 16.1, 16.2, 16.3, 16.4, 16.5
  23.1, 23.2

《计算机算法设计与分析》
- 第4章: 4.1, 4.2, 4.4, 4.5, 4.6, 4.8

《网站资料》
- 第五章

5.1 *Elements of Greedy Algorithms*

- *Greedy*算法的基本概念
- *Greedy*选择性
- 优化子结构
- 与动态规划方法的比较
- *Greedy*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

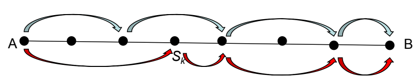
编号i	1	2	3	4	5	6
价值v_i	60	100	120	140	30	40
重量w_i	10	20	30	35	10	20
v_i/w_i	6	5	4	4	3	2

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

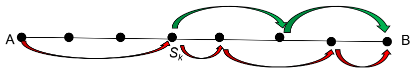
  从A出发不加油最远到达加油S_k
  必存在最优加油策略在S_k首次加油



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
  - Greedy-choice-property
  - Optimal substructure

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
- Greedy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

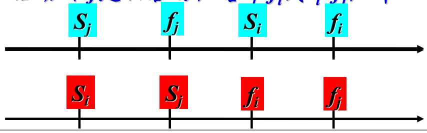
- 证明算法所求解的问题具有贪心选择性
- 证明算法所求解的问题具有优化子结构
- 证明算法确实按照贪心选择性进行局部优化选择

5.2 An activity-selection problem

- 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$S=\{1,2,\ldots,n\}$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，$s_i \le f_i$
- 相容活动
  - 活动i和j是相容的，若$s_j \ge f_i$或$s_i \ge f_j$，即



- 问题定义
  - —输入：$S=\{1, 2, \ldots, n\}$，$F=\{ [s_i, f_i] \}$，$n \ge i \ge 1$
  - —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$S=\{1,2,\ldots,n\}$是n个活动集合，$[s_i, f_i]$是活动的起始终止时间，且$f_1 \le f_2 \le \ldots \le f_n$，$S$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$k \ne 1$，令$B=A-\{k\} \cup \{1\}$,

由于A中活动相容，$f_1 \le f_k \le s_j$, 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$f_1 \le f_2 \le \ldots \le f_n$，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$S' = \{i \in S | s_i \ge f_1\}$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$\forall i \in S'$, $s_i \ge f_1$, 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.

- Greedy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$S_i=\{ j \in S \mid s_j \ge f_{i-1}\}$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$f_1 \le \ldots \le f_n$，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$S_2=\{ j \in S \mid s_j \ge f_1\}$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$f_1 \leq f_2 \leq ... \leq f_n$已排序）

  Greedy-Activity-Selector(S, F)

  $n \leftarrow$lenyth(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

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*Greedy*选择性
  - 活动选择问题具有优化子结构
  - 算法按照Greedy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Greedy选择性进行局部优化选择.

HIT CS&E

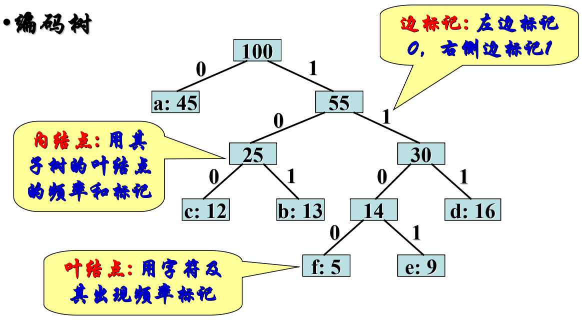
5.3 Huffman codes

- 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

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

•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 $C=\{c_1, c_2, ..., c_n\}$,

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

优化解的结构分析

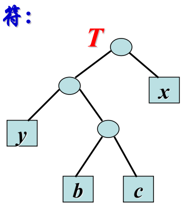
- 我们需要证明
  - 优化前缀树问题具有Greedy选择性
  - 优化前缀树问题具有优化子结构

- Greedy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.

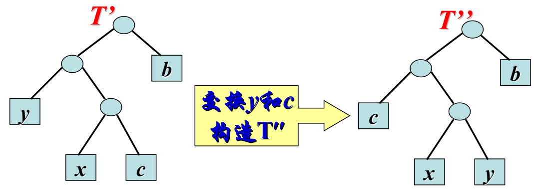
HIT CS&E

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':

HIT CS&E



往证T''是最优化前缀树.

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

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

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

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

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

$\because f(b)\geq f(x)$, $d_T(b)\geq d_T(x)$ (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$B(T)\geq B(T'')$.

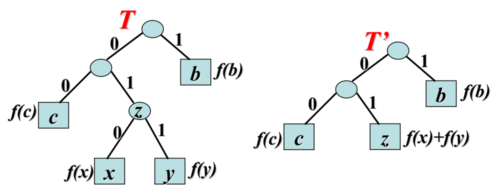
由于T是最优化的，所以$B(T)\leq B(T'')$.

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

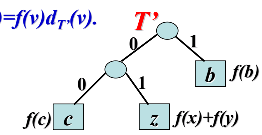
$\forall v\in C-\{x,y\}$, $d_T(v)=d_{T'}(v)$, $f(v)d_T(v)=f(v)d_{T'}(v)$.

由于 $d_T(x)=d_T(y)=d_{T'}(z)+1$,

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

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

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



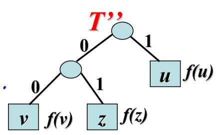
由于 $f(x)+f(y)=f(z)$, $f(x)d_T(x)+f(y)d_T(y)=f(z)d_{T'}(z)+(f(x)+f(y))$.

于是$B(T)=B(T')+f(x)+f(y)$.

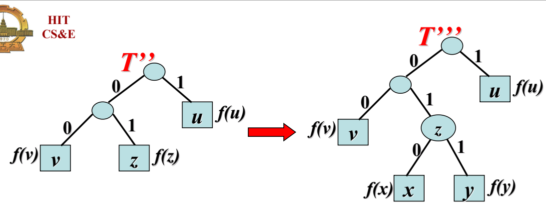
若T'不是C'的优化前缀编码树，则必存在T''，使$B(T'')<B(T')$.

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''':



HIT CS&E



T'''代价为：

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

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

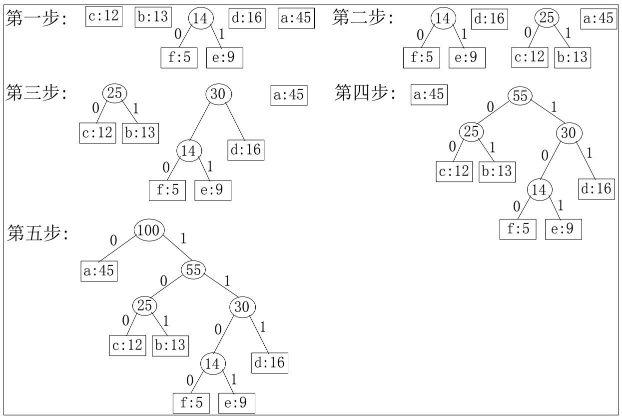
$= B(T'')+f(x)+f(y)<B(T')+f(x)+f(y)= B(T)$

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←|C|;
2.  Q←C;  /* 用BUILD-HEAP建立堆 */
3.  FOR i←1 To n-1 Do
4.      z←Allocate-Node( );
5.      x←left[z]←Extract-MIN(Q); /* 堆操作*/
6.      y←right[z]←Extract-MIN(Q); /* 堆操作*/
7.      f(z)←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Huffman算法按照引理2的Greedy选择性确定的规则进行局部优化选择，所以Huffman算法产生一个优化前缀编码树。

HIT CS&E

5.4 Minimal spanning tree problem

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

HIT CS&E

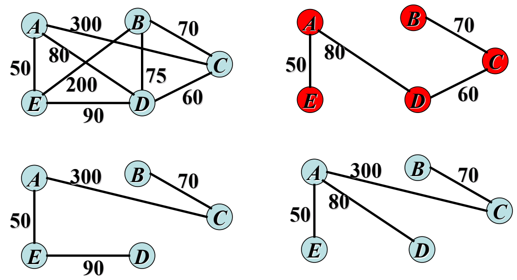
问题的定义

- 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

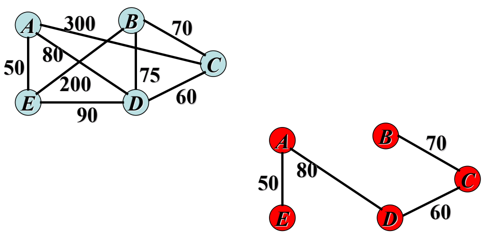
  输入: 无向连通图$G=(V, E)$, 权函数W

  输出: G的最小生成树

- 实例



- 算法思想



HIT CS&E

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

HIT CS&E

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

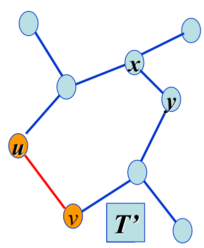
  第5-8步执行$O(m)$个Find-Set和Union操作

  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: $O(m\log m)$

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；

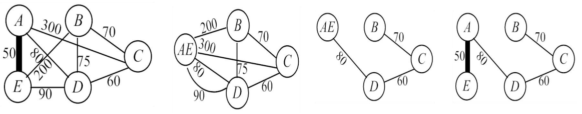
否则，如右图所示

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'。

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

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$W:E\to R$, $uv\in E$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。

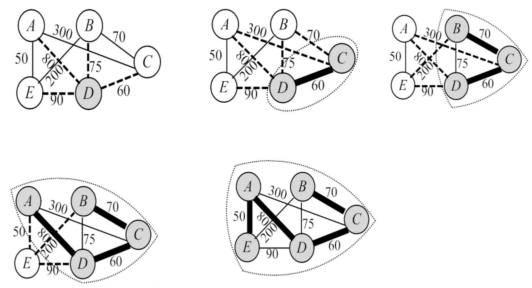
算法正确性

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



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

```
C←{r}，T←∅;
建堆Q维护C与V-C之间的边
 While C≠V  do
      uv ←Extract_Min(Q)              //u∈C,v∈V-C
      C←C∪{v};     T← T∪{uv};
      for ∀x∈Adj[v]  do
            if x∈C then  将vx从Q中删除
            Else          将vx插入Q
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

```
For  ∀ v∈V[G]   Do
      key[v]←+∞
      π[v] ←null
key[r] ←0
Q ←V[G]
While Q≠∅ do
      u ←Extract_Min(Q)                    //找到安全轻边
      for ∀v∈Adj[u]  do
         if v∈Q 且 w(u,v)<key[v]   then
                  π[v]    ←u
                  key[v] ←w(u,v)           //更新信息
Return  A={(v, π[v])| v ∈V[G] -r}
```

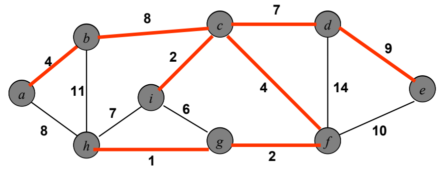
log V

2E遍

常数时间

logV

实例



	π[v]	key
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

- $A=\{(v,\pi(v))\mid v\in V\text{-}r\text{-}Q\}$
- 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]<+∞,且key[v]是将v连接到当前生成树需要的最小权值

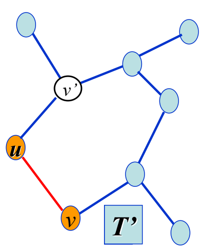
算法复杂性

假设用最小堆实现Q

总的时间开销为O(VlogV+ElogV)=O(ElogV)

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；

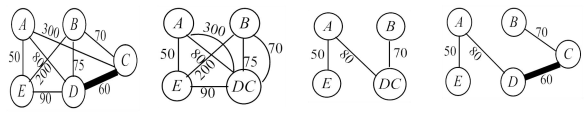
否则，如右图所示

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'。

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

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$W:E\rightarrow R$, $uv\in E$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.

HIT CS&E

5.5 Theoretical foundations of Greedy Algorithms

- Matroid (拟阵)
- Matroid 实例
- Matroid的性质
- 加权Matroid上的Greedy算法
- 任务调度问题

HIT CS&E

Matroid（拟阵）

- Matroid定义

Matroid是一个序对$M=(S, I)$，满足:

(1) S是一个非空有限集合.

(2) I是非空的S子集的集族，I中的子集称为S的独立子集.

(3) 遗传性: 如果$A\in I$，$B\subseteq A$，则$B\in I$

(4) 交换性: 如果$A\in I$，$B\in I$，$|A|<|B|$，则$\exists x\in B\text{-}A$使得$A\cup\{x\}\in I$.

HIT CS&E

- 实例(Graphic Matroid)

定义1. 设$G=(V, E)$是一个无向图，由G确定的$M_G=(S_G, I_G)$定义如下: S_G是G的边集合E，$I_G=\{A\,|\,A\subseteq E, (V, A)$是森林$\}$.

定理1. 如果G是一个无向连通图且$|V|>1$，则$M_G=(S_G, I_G)$是一个Matroid.

证. ① $S_G=E$是一个有限集合.

② $\forall e\in E$, $(V, \{e\})$是一个森林，$\{e\}\in I_G$. 于是，I_G是S_G的非空集族.

HIT CS&E

③ M_G满足遗传性: 如果$B\in I_G$，$A\subseteq B$，则(V, A)是一个森林. 于是，$A\in I_G$，M_G满足遗传性.

④ M_G满足交换性: 设$A\in I_G$, $B\in I_G$, $|A|<|B|$.

如果B的任意一条边都包含在A的同一棵树中，则B的边数不大于A的边数，与$|A|<|B|$矛盾.

于是，B必包含一条边(u,v)，(u,v)在A的不相同树中.

$(u,v)\in A\text{-}B$连接A的两棵不同树.

$(V, A\cup\{(u,v)\})$是森林，$A\cup\{(u,v)\}\in I_G$.

于是, M_G满足交换性.

- Matroid的性质

定义2. 设$M=(S, I)$是一个Matroid, $A\in I$. 如果$A\cup\{x\}\in I$, $x\notin A$, x称为A的一个extension.

定义3. 设$M=(S, I)$是Matroid, $A\in I$. 若A没有extension, 则称A为极大独立子集合.

定理2. 一个Matroid的所有极大独立子集合都具有相同大小.

证. 设A是Matroid M的极大独立子集合，而且存在M的另一个独立子集合B, $|A|<|B|$.

根据M的交换性, $\exists x\in B\text{-}A$使$A\cup\{x\}\in I$, 与A最大矛盾.

HIT CS&E

加权Matroid上的Greedy算法

- Matroid的优化子集

实际背景:

很多可用Greedy算法获得最优解的问题可以归结为在加权Matroid中寻找优化子集的问题, 即给定$M=(S, I)$和权函数W, 搜索独立子集$A\in I$, 使得$W(A)$最大。

- **实例：最小生成树问题**
  - 问题定义
    - 输入：无向图$G=(V, E)$，权函数$W: E\rightarrow$正整数集
    - 输出：边子集合$A\subseteq E$, (V, A)是一棵树，$W(A)$最小
  - 转换为加权Matroid上寻找优化子集问题
    - 构造：
      - ① $M_G = (S_G, I_G)$是图Matroid
      - ② $W'(e) = W_0 - W(e)$，$W_0 > \max\{W(e)\}$
        - $\forall e\in E$，$W'(e)>0$.
        - W'扩展为$W'(A)=|V|W_0 - W(A)$，$\forall A\subseteq E$
    - M_G的最优子集A满足：
      - (V,A)是森林
      - $W'(A)$最大，即$W(A)$最小.
    - 最小生成树问题可以由求M_G的最优子集的算法来求解

- **加权Matroid优化子集问题的定义**
  - 输入：Matroid $M=(S, I)$, M的加权函数W
  - 输出：M的最优子集

- **算法**

Greedy(M, W)

1 $A=\Phi$;
2 按权W值大小排序S;
3 For $\forall x\in S$（按$W(x)$递减顺序）DO
4 If $A\cup\{x\}\in I$ /* 选择目前$W(x)$最大的x */
5 Then $A\cup\{x\}$;
6 Return A.

- **时间复杂性**

step 2: $O(|s|\log|s|)$

step 4: $O(f(|s|))$

$T(|s|) = O(\ |s|\log|s|+|s|\ f(|s|)\ )$

- **算法正确性**

引理1. Greedy算法总是返回一个独立子集合.

我们需要证明A是最优子集，即证明
- 加权Matroid最优子集问题具有Greedy选择性
- 加权Matroid最优子集问题具有优化子结构
- 算法按照优化子结构和选择规则选择最优子集

- **Greedy选择性**

引理2. 设$M=(S, I)$是一个加权Matroid, W是M的权函数, S按W值递减排序. 若x是S中首个满$\{x\}\in I$的元素，则存在一个M的优化子集A, $x\in A$.

证. 设S第一个元素x满足$\{x\}\in I$. 若存在优化子集A包含x，则引理得证. 否则, 设B是任意非空优化子集, $x\notin B$.
显然, $\forall y\in B$，$W(x)\geq W(y)$. 如下构造含x的优化子集A:
初始：$A=\{x\}\in I$;
用交换性：$\forall y\in B-A$, 若$A\cup\{y\}\in I$, $A=A\cup\{y\}$, 直至$|A|=|B|$.
显然，$\exists z\in B$，$A=(B-\{z\})\cup\{x\}$.
于是, $W(A)=W(B)-W(y)+W(x)\geq W(B)$.
因为B是优化子集, 所以$W(A)\leq W(B)$, $W(A)=W(B)$.
A是优化子集, 且$x\in A$.

引理3. 设$M=(S, I)$是一个Matroid. 如果$x\in S$不是空集Φ的extension, 则x不是任何独立子集的extension.

证. 反证.设$x\in S$是独立子集A的extension但不是Φ的extension.
由于x是A的extension, $A\cup\{x\}\in I$. 由M的遗传性，$\{x\}\in I$, 即$\{x\}$是Φ的 extension, 矛盾.

推论1. 任何元素一旦不能被选中, 则永远不会被选中.

推论2. Greedy算法不会由于不再考虑未被初始选中的元素而产生错误.

- 优化子结构

引理4. 设x是第一个被Greedy算法选中的元素. 包含x的优化子集A包含子问题$M'=(S', I')$的优化子集$A'=A-\{x\}$, $M'=(S', I')$定义如下:

$S'=\{y\in S|\{x, y\}\in I\}$, $I'=\{B\subseteq S-\{x\}|B\cup\{x\}\in I\}$
而且M'的权函数与M的权函数相同.

证. 若A是M的优化子集且$x\in A$, 则$A'=A-\{x\}\subseteq A$.
因为$A=A'\cup\{x\}\in I$, 所以$A'=A-\{x\}\in I'$.
若A'不是M'的优化子集, 则存在M'的一个优化子集B使得$W(B)>W(A')$.
由于$B\cup\{x\}\in I$, $W(A)=W(A')+W(x)$,
$W(B\cup\{x\})=W(B)+W(x)>W(A')+W(x)=W(A)$,
与$W(A)$优化矛盾.

- 算法正确性

定理1. 设$M=(S, I)$是一个Matroid, W是M的权函数, Greedy(M,W)返回一个M的优化子集.

证. ①. 引理3说明, 任何没有被Greedy选中的S元素, 以后不会被选中, 可以不再考虑.
②. 一旦S的第一个x被选中, x可以加到A, 因为引理2说明存在一个包含x的优化子集.
③. 引理4意味着余下的问题是在M'中求解最优子集的问题.
Greedy算法是按照上述三个规则工作的,
所以Greedy(M,W)返回一个M的优化子集.

HIT CS&E

5.6 A task-scheduling problem

HIT CS&E

问题定义

- 单位时间任务
  需要一个单位时间就能够完成的任务
- 单位时间任务的调度问题
  输入:
  单位时间任务集$S=\{1, 2, \ldots, n\}$
  正整数任务期限$D=\{d_1,d_2,\ldots,d_n\}$, 任务i须在d_i前完成
  非负权集$W=\{w_1,w_2,\ldots,w_n\}$, 任务i不在d_i前完成罚款w_i
  输出:
  S的一个调度(S的一个执行顺序), 具有最小总罚款

- 转换为加权Matroid的优化子集问题

定义1. 设S是一个任务调度. 一个任务在S中是迟的如果它在规定的期限之后完成; 否则是早的.

迟$d_{i_k}<k$ $k+1\le d_{i_{k+1}}$早
i_1, $i_2, \ldots, i_{k-1}, i_k$, i_{k+1}, $i_{k+1}, \ldots, i_n$

i_1, $i_2, \ldots, i_{k-1}$, i_{k+1}, i_k, $i_{k+1}, \ldots, i_n$
早$k\le d_{i_{k+1}}$ $d_{i_k}<k$迟

定义2. 如果在一个调度中, 早任务总是先于迟任务, 则称该调度具有早任务优先形式.

命题1: 问题存在早任务优先形式的优化解

HIT CS&E

早d_{i_k} > $d_{i_{k+1}}$早
i_1, $i_2, \ldots, i_{k-1}, i_k$, i_{k+1}, $i_{k+1}, \ldots, i_n$

i_1, $i_2, \ldots, i_{k-1}$, i_{k+1}, i_k, $i_{k+1}, \ldots, i_n$
早 $k+1\le d_{i_{k+1}}<d_{i_k}$早

定义3. 如果一个调度具有早任务优先形式而且按期限单调递增顺序执行各任务, 则称该调度具有规范化形式.

命题2: 问题存在规范化形式的优化解

HIT CS&E

- **任务调度的规范化**

  第一步：将调度安排成早任务优先形式，即如果早任务x跟在迟任务y之后，交换x和y的位置；

  第二步：如果任务i和j是早任务，而且分别完成于时间k和$k+1$，$d_j<d_i$，交换i和j的位置。

  - 调度优先形式不改变任何任务的早或迟状态
  - 调度规范形式不改变任何任务的早或迟状态

HIT CS&E

寻找最优调度问题成为寻找在该最优调度中的早任务集合A的问题。一旦A被确定后，就可以按期限单调递增序列出A中的所有元素，然后按任意顺序列出迟任务(即$S-A$)

HIT CS&E

定义4. 任务集合A称为独立的如果存在一个关于A的调度，使得A中的任务皆非迟任务。

例. 一个优化调度的早任务集合是独立独立任务集合.

以下：

用I表示所有独立任务集合的集族

用$N_t(A)$表示A中期限小于等于t的任务数

HIT CS&E

引理1. 对于任何任务集合A，下边的命题等价：

1. A是独立集合，
2. 对于$t=1, 2, \ldots, n$, $N_t(A)\leq t$,
3. 如果按照期限递增顺序调度A中任务，则A中无迟任务.

证. $1\rightarrow2.$ 如果$N_t(A)>t$，则有多于t个任务需要在t时间内完成，不存在使得A中无迟任务的调度.

$2\rightarrow3.$ 若A中任务依其期限递增排列，则2意味着排序后，在时间t之前必须完成的A中任务数至多为t. 于是，按期限递增顺序调度A中任务，A无迟任务.

$3\rightarrow1.$ 显然.

定理1. 若S是一个带期限的单位时间任务的集合，且I为所有独立任务集构成的集族，则$M=(S, I)$是一个Matroid.

证明. 1. S是非空有限集合.

2. I是S的子集的非空集族，因为单个任务集属于I.

3. **遗传性**：若$A\in I$, $B\subseteq A$，则$B\in I$.

4. **交换性**：设$A, B\in I$, $|B|>|A|$, $k=\max_{1\leq t\leq n}\{t \mid N_t(B)\leq N_t(A)\}$.

$N_t(B)\leq N_t(A)$ k $N_j(B)>N_j(A)$

于是，B中包含了比A中更多的具有期限$k+1$的任务. 设$x\in B-A$, x具有期限$k+1$. 令$A'=A\cup\{x\}$. 往证A'独立.

对于$1\leq t\leq k$, $N_t(A')=N_t(A)\leq t$，因为A是独立的.

对于$k<t\leq n$, $N_t(A')\leq N_t(B)\leq t$，因为B是独立的.

于是，A'是独立的.

HIT CS&E

最后，任务调度问题转换为$M=(S, I)$上寻找最优子集问题，M的加权函数为W(罚款)