贪心算法

Jun Wu

wujun@yzu.edu.cn

April 18, 2018

活动场所选择

- n个活动$S = \{a_1, \cdots, a_n\}$共用同一场所，每个活动对场所的使用是独占的
- 每个活动有两个参数：开始时间和结束时间，s_i，f_i
- 若两个活动的时间不重叠，称该两个活动兼容
- 问题：找出活动两两兼容的最大子集？

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

图：活动场所选择问题实例

最优子结构

- 将所有活动按活动的结束时间的升序排序；
- 不妨设$f_1 \leq f_2 \leq \cdots \leq f_n$；
- 给定最优解$OPT = \{a_{i_1}, a_{i_2}, \cdots, a_{i_m}\}$，若$a_{i_k} \in OPT$；
- 根据a_{i_k}的开始时间和结束时间，定义子问题实例如下：
- $L = \{a_j \in S | f_j \leq s_{i_k}\}$; $R = \{a_j \in S | s_j \geq f_{i_k}\}$。

Lemma 1

$$OPT_L = \{a_{i_1}, \cdots, a_{i_{k-1}}\}$$

and

$$OPT_R = \{a_{i_{k+1}}, \cdots, a_{i_m}\}.$$

递归关系

- 为了描述方便，我们增加两个虚拟活动：
- a_0:$f_0 = 0$;
- a_{n+1}:$s_{n+1} = \infty$。
- 这样我们可以定义子问题：

$$S_{ij} = \{a_k \in S | f_i \le s_k < f_k \le s_j\}.$$

- 求解活动场所选择问题最优解值的递归关系如下：

$$c[i,j] = \begin{cases} 0, & if\ S_{ij} = \emptyset \\ \max_{a_k \in S_{ij}} \{c[i,k] + c[k,j] + 1\}, & otherwise \end{cases}.$$

贪心选择性质

Lemma 2

设$f_m = \min_{a_k \in S_{ij}} \{f_k\}$，那么下述结论成立：

1. $S_{im} = \emptyset$;
2. 存在某个最优解OPT，使得$a_m \in OPT$。

Proof.

1. 若存在$a_k \in S_{im}$,那么$s_k < f_k \leq s_m < f_m \Rightarrow$ 矛盾。
2. 任意取一最优解OPT'，分两种情况讨论：

- case 1：若$a_m \in OPT'$，引理得证。
- case 2：若$a_m \notin OPT'$：设a_k是OPT'中结束时间最早的。
- 那么令$OPT = OPT' \setminus \{a_k\} \cup \{a_m\}$，$OPT$仍是最优解。

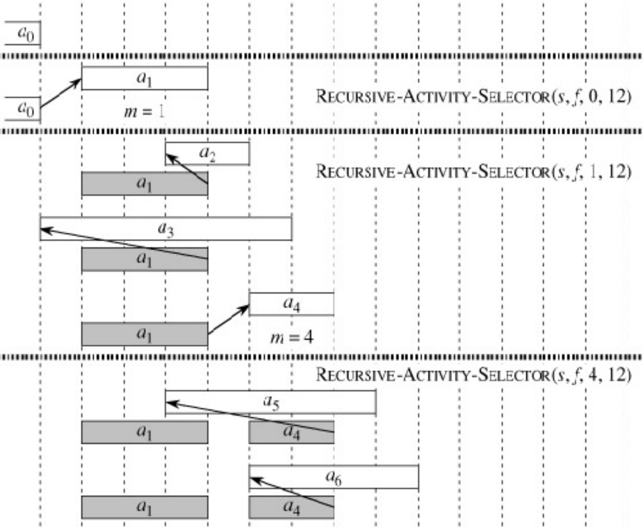
$\square$

RECURSIVE-ACTIVITY-SELECTOR(s, f, i, j)

$m \leftarrow i + 1$
while $m < j$ and $s_m < f_i$ **do**
    $m \leftarrow m + 1$
end while
if $m < j$ **then**
    RETURN $\{a_m\} \cup$ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, j)
else
    RETURN $\emptyset$
end if

例子

k	s_k	f_k
0	–	0
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9



例子

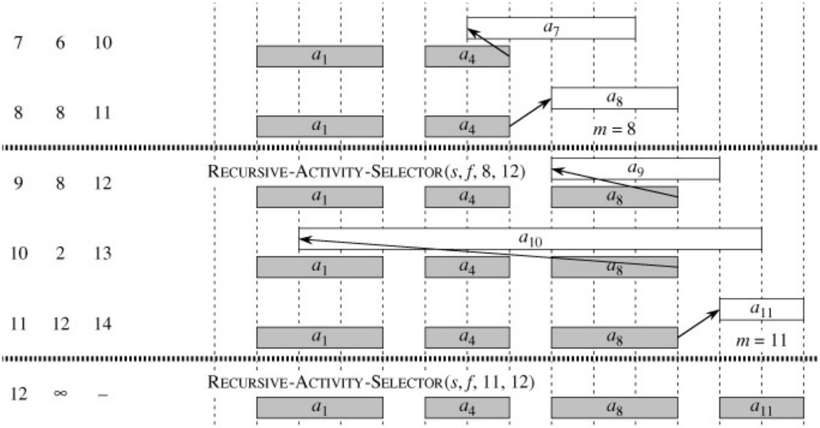


图: 活动场所选择算法执行过程

- 算法复杂度：$O(n \log n)$。

GREEDY-ACTIVITY-SELECTOR(s, f)

```
n ← length[s]
A ← {a_1}
i ← 1
for m ← 2 to n do
    if s_m ≥ f_i then
        A ← A ∪ {a_m}
        i ← m
    end if
end for
RETURN A
```

背包问题的定义

- 给定一背包，背包容量为C；
- 给定一组商品$\{a_1, \cdots, a_n\}$;
- 每件商品$a_i (1 \leq i \leq n)$有两个参数：重量w_i和价值v_i;
- 背包问题：选择商品装入背包，使得所装商品的价值最大化。

可分背包问题：

$$\max \sum_i v_i x_i$$

$s.t.$

$$\sum_i w_i x_i \leq C$$

$$x_i \in [0, 1], 1 \leq i \leq n$$

0-1背包问题：

$$\max \sum_i v_i x_i$$

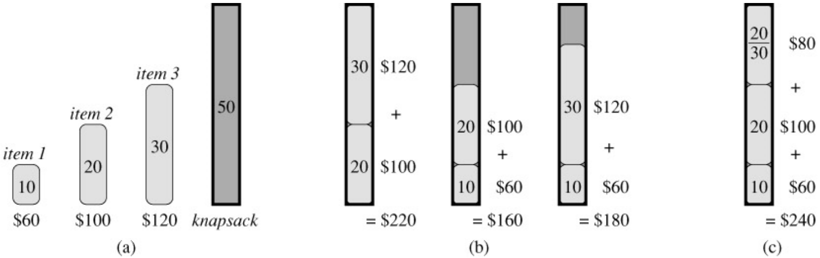
$s.t.$

$$\sum_i w_i x_i \leq C$$

$$x_i \in \{0, 1\}, 1 \leq i \leq n$$

可分背包问题的贪心算法

- 最优子结构性质？
- 贪心选择性质？
- 贪心算法？是否适用于0-1背包问题？
- 算法的复杂度？



贪心算法要点

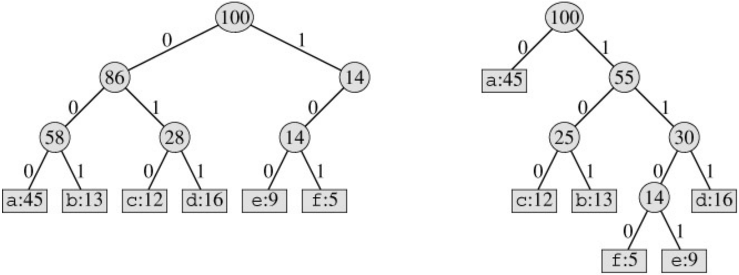
1. 最优子结构性质
    - 分析方法：
    - 考察任意最优解
    - 定义子问题
    - 采用剪切-粘贴法证明
2. 贪心选择性质：存在最优解包含了贪心选择出的结果
    - 分析方法：
    - 任取一最优解
    - 若该最优解包含了贪心选择出的结果，贪心选择性质显然成立
    - 若不包含，调整该最优解，得到另一最优解包含贪心选择出的结果
    - 注意：必须证明调整后的解为最优解

问题定义

- 信源产生k条消息，这些消息由n种字符(C)组成。
- 已知各种字符在k条消息中出现的频率，$\forall c \in C$，$f(c)$表示c的频率。
- **问题：** 设计C的二元前缀编码，使得记录k条消息所需的比特数最少。
- 前缀码：任意$x, y \in C$，x的码字不是y码字的前缀。

C	a	b	c	d	e	f
频率	45	13	12	16	9	5
定长码	000	001	010	011	100	101
变长码	0	101	100	111	1101	1100

二元前缀编码与二叉树一一对应。



(a) (b)

文件记录的长度可以表示为:

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

贪心选择性质

Lemma 3

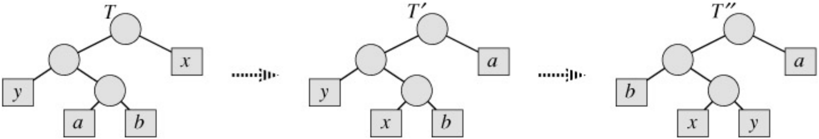
设x和y是C中频率最小的两个字符。那么存在一个最优编码使得码字x和码字y的长度相同且只有最后一位不同。

Proof.

- 即存在最优编码二叉树，使得x和y在离树根最远处的一对兄弟。
- 任取一个最优编码二叉树。
- 若该树不满足引理，则作系列调整得到新的编码二叉树。
- 证明新的编码二叉树为满足引理的最优编码树。

$\square$

贪心选择性质的证明



$$
\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c) \\
&= f(a) d_T(a) + f(x) d_T(x) - f(a) d_{T'}(a) - f(x) d_{T'}(x) \\
&= f(a) d_T(a) + f(x) d_T(x) - f(a) d_T(x) - f(x) d_T(a) \\
&= f(a)(d_T(a) - d_T(x)) - f(x)(d_T(a) - d_T(x)) \\
&= (f(a) - f(x))(d_T(a) - d_T(x)) \\
&\geq 0
\end{aligned}
$$

最优子结构性质

Lemma 4

设$x, y \in C$为频率最低的一对字符，字符$z \notin C$。令$f(z) = f(x) + f(y)$。定义子问题$C' = C \cup \{z\} \setminus \{x, y\}$。若$T$是$C$的最优编码树且$x, y$是兄弟，那么删去叶结点$x, y$后的树$T'$是子问题$C'$的最优编码树。

Proof.

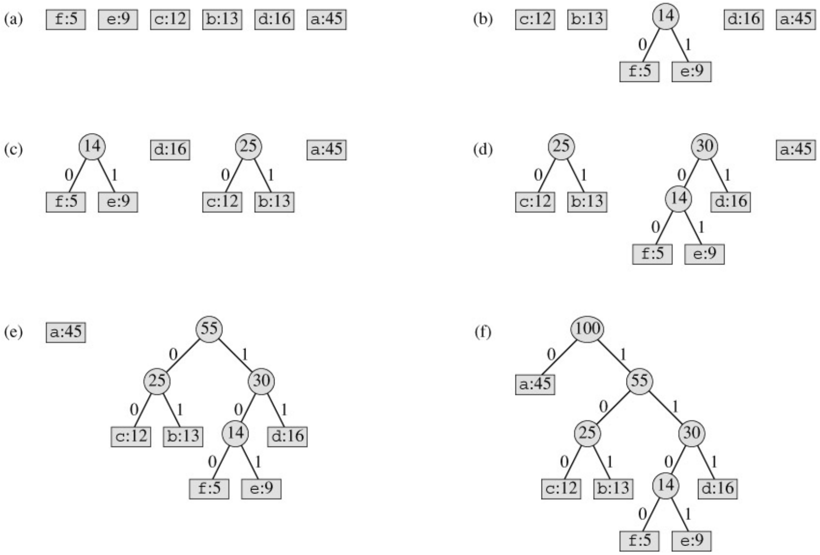
- 若T'不是C'的最优解，则存在最优解T''。将x, y作为T''中z的儿子，得到C的解T'''。

$$
\begin{aligned}
B(T''') &= \textstyle\sum_{c \in C} f(c) d_{T'''}(c) \\
&= \textstyle\sum_{c \in C \setminus \{x,y\}} f(c) d_{T''}(c) + f(x) d_{T'''}(x) + f(y) d_{T'''}(y) \\
&= \textstyle\sum_{c \in C \setminus \{x,y\}} f(c) d_{T''}(c) + f(z) d_{T''}(z) + f(x) + f(y) \\
&= B(T'') + f(x) + f(y) \\
&< B(T') + f(x) + f(y) = B(T)
\end{aligned}
$$

- 与T是C的最优解矛盾。

$\square$

哈夫曼算法执行过程



(a)
f:5
e:9
c:12
b:13
d:16
a:45
(b)
c:12
b:13
14
0
1
f:5
e:9
d:16
a:45
(c)
14
0
1
f:5
e:9
d:16
25
0
1
c:12
b:13
a:45
(d)
25
0
1
c:12
b:13
30
0
1
14
d:16
0
1
f:5
e:9
a:45
(e)
a:45
55
0
1
25
30
0
1
c:12
b:13
0
1
14
d:16
0
1
f:5
e:9
(f)
100
0
1
a:45
55
0
1
25
30
0
1
c:12
b:13
0
1
14
d:16
0
1
f:5
e:9

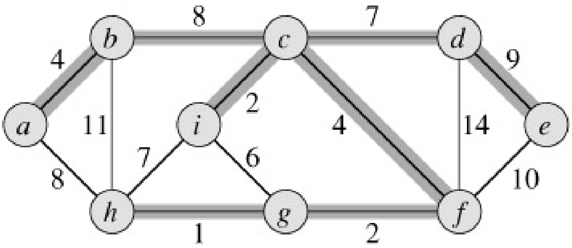
HUFFMAN(C, f)

$n \leftarrow |C|$
$Q \leftarrow C$
for $i \leftarrow 1$ **to** $n-1$ **do**
    allocate a new node z
    left$[z] \leftarrow x \leftarrow$ EXTRACT-MIN(Q)
    right$[z] \leftarrow y \leftarrow$ EXTRACT-MIN(Q)
    $f[z] \leftarrow f[x] + f[y]$
    INSERT(Q, z)
end for
return EXTRACT-MIN(Q)

- 算法复杂度：$O(n \log n)$。

最小费用生成树问题

- 设$G = (V, E)$为连通无向图
- $w : E \to R^+$为边的权重
- $T \subseteq E$，T称作G的生成树当且仅当$|T| = |V| - 1$并且$G_T = (V, T)$是连通的。
- 生成树的费用：$w(T) = \sum_{e \in T} w(e)$。
- **问题：** 寻找G的生成树T使得$w(T)$最小。



最优子结构性质

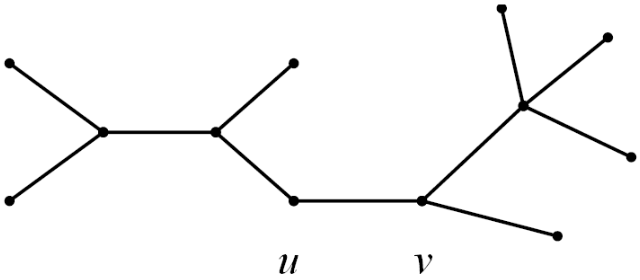
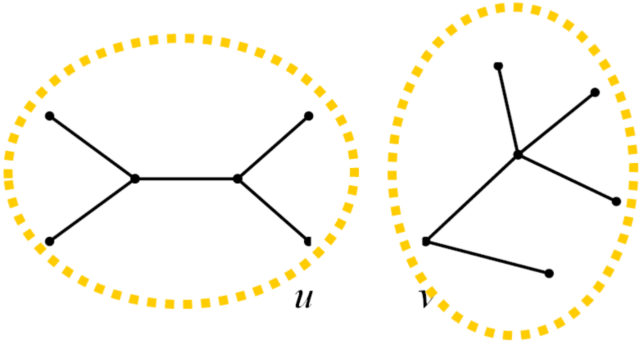


图: 一棵MST T

- $(u, v) \in T$。定义子问题?

最优子结构性质

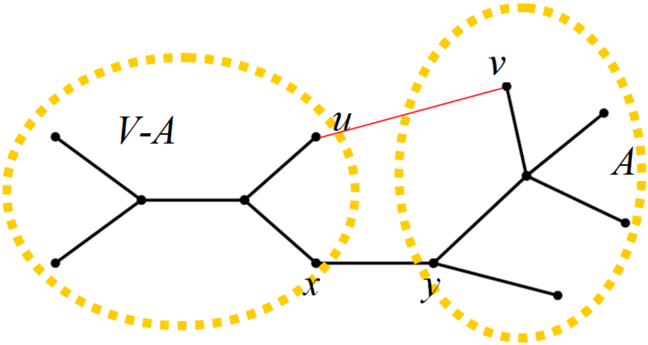


- 每个黄色圈中的点在图G上的导出子图构成一个小规模的MST问题实例。
- 剪切-粘贴法证明

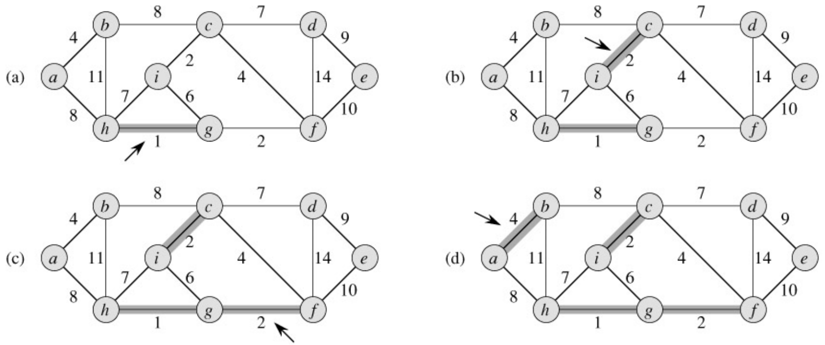
贪心选择性质

Lemma 5

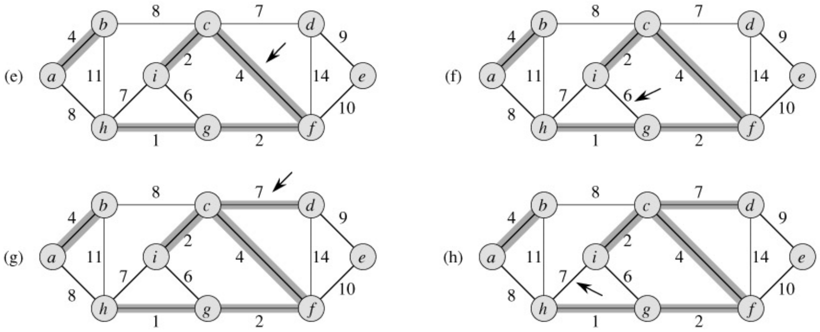
设$G = (V, E)$，$A \subset V$，而$(u, v) \in E$是连接A和$V \setminus A$的费用最小的边。那么存在*MST* T使得$(u, v) \in T$。



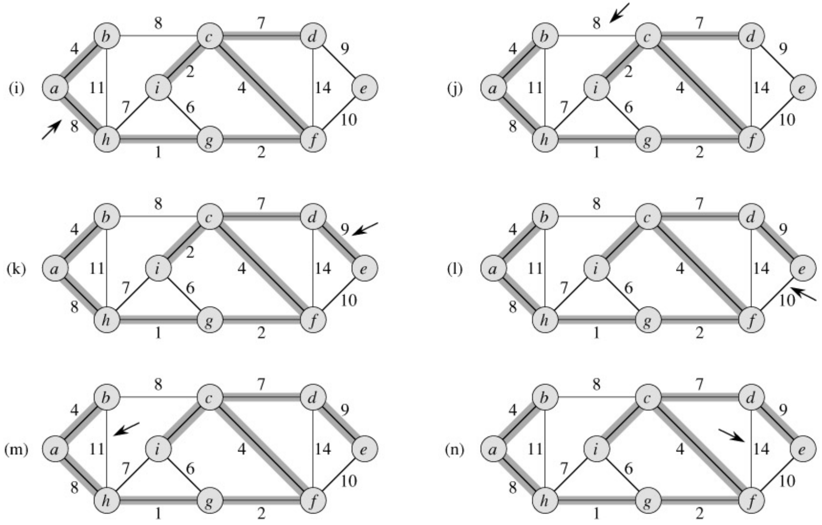
Kruskal算法执行过程



Kruskal算法执行过程



Kruskal算法执行过程

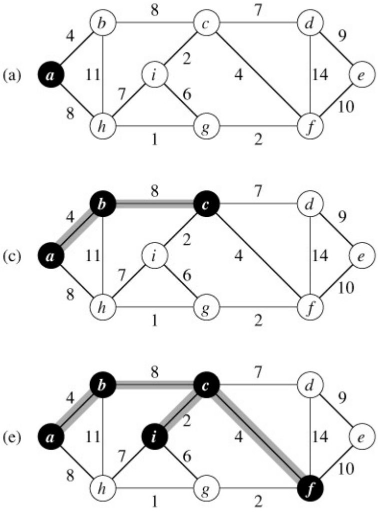


MST-KRUSKAL(G, w)

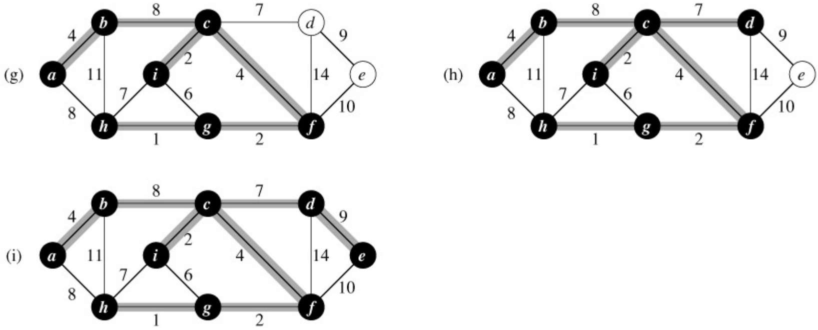
$A \leftarrow \emptyset$
for each vertex $v \in V$ **do**
  MAKE-SET(v)
end for
sort the edges of E into nondecreasing order by weight w
for each edge $(u, v) \in E$, taken in nondecreasing order by weight **do**
  if FIND-SET$(u) \neq$ FIND-SET(v) **then**
    $A \leftarrow A \cup \{(u, v)\}$
    UNION(u, v)
  end if
end for
return A

- 复杂度：$O(|E|)$次FIND-SET以及$O(|V|)$次UNION和MAKE-SET操作，总共需要$O(E \log |V|)$。

Prim算法执行过程



Prim算法执行过程



MST-PRIM(G, w, r)

```
for each u ∈ V do
    key[u] ← ∞
    π[u] ← nil
end for
key[r] ← 0
Q ← V
while Q ≠ ∅ do
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u] do
        if v ∈ Q and w(u, v) < key[v] then
            π[v] ← u
            key[v] ← w(u, v)
        end if
    end for
end while
```

- 复杂度：$O(|E|)$次堆优先值调整需要$O(|E|\log|V|)$。采用Fibonacci堆，可以降低为$O(|E|)$。因此，总的复杂度为$O(|E|+|V|\log|V|)$。