本周教学内容

贪心法的设计要素

得不到最优解的处理：
找零钱问题的参数化分析

贪心法证明:按步骤归纳
活动选择

贪心法证明:按规模归纳
装载问题

贪心法证明:交换论证
最小延迟调度

贪心法的例子：活动选择问题

贪心法的例子：
活动选择问题

活动选择问题

输入：$S = \{1, 2, \ldots, n\}$为n 项活动的集合， s_i，f_i 分别为活动 i 的开始和结束时间.

活动 i 与 j 相容 $\Leftrightarrow s_i \geq f_j$ 或 $s_j \geq f_i$.

求：最大的两两相容的活动集 A

输入实例：

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

解：{1, 4, 8}

贪心算法

挑选过程是多步判断，每步依据某种“短视”的策略进行活动选择，选择时注意满足相容性条件.

策略1：开始时间早的优先

排序使 $s_1 \le s_2 \le \ldots \le s_n$，从前向后挑选

策略2：占用时间少的优先

排序使得 $f_1 - s_1 \le f_2 - s_2 \le \ldots \le f_n - s_n$，从前向后挑选

策略3：结束早的优先

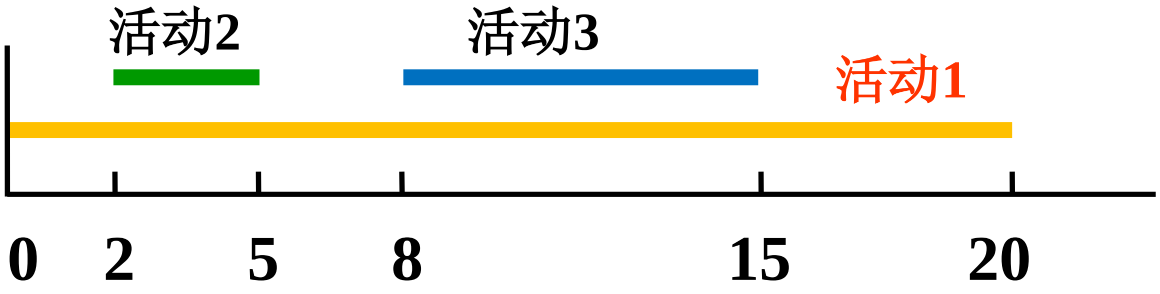
排序使 $f_1 \le f_2 \le \ldots \le f_n$，从前向后挑选

策略1的反例

策略1：开始早的优先

反例：S ={1,2,3 }

s_1=0，f_1=20，s_2=2，f_2=5，s_3=8，f_3=15

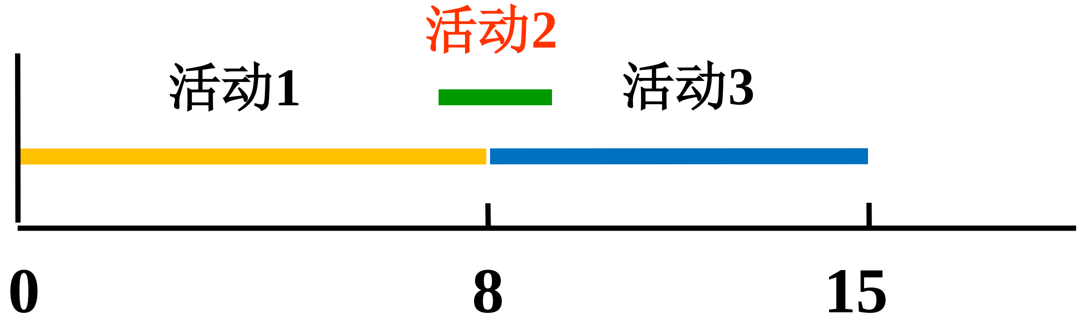


策略2的反例

策略2：占时少的优先

反例：　$S = \{ 1, 2, 3 \}$

$s_1=0,\ f_1=8,\ s_2=7,\ f_2=9,\ s_3=8,\ f_3=15$



策略3伪码

算法 **Greedy Select**

输入：活动集 S，s_i, f_i，$i = 1, 2, \ldots, n$，$f_1 \le \ldots \le f_n$

输出：$A \subseteq S$，选中的活动子集

$n \leftarrow length[S]$

$A \leftarrow \{1\}$

$j \leftarrow 1$ 已选入的最后标号

for $i \leftarrow 2$ **to** n **do**

  if $s_i \ge f_j$ 判相容

  then $A \leftarrow A \cup \{ i \}$

    $j \leftarrow i$

return A

完成时间 $t = \max \{ f_k : k \in A \}$

运行实例

输入：$S = \{ 1, 2, \ldots , 10 \}$

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

解：$A = \{1, 4, 8\}$，$t = 11$

时间复杂度

$$O(nlogn) + O(n) = O(nlogn)$$

如何证明该算法对所有的实例都得到正确的解？

贪心算法的特点

设计要素：

(1) 贪心法适用于组合优化问题.

(2) 求解过程是多步判断过程，最终的判断序列对应于问题的最优解.

(3) 依据某种“短视的”贪心选择性质判断，性质好坏决定算法的成败.

(4) 贪心法必须进行正确性证明.

(5) 证明贪心法不正确的技巧：举反例.

贪心法的优势：算法简单，时间和空间复杂性低

贪心法正确性
证明：活动选择

一个数学归纳法的例子

例：证明对于任何自然数n，

$$1+2+\ldots+n = n\,(n+1)/2$$

证　$n=1$, 左边=1, 右边=$1\times(1+1)/2=1$

假设对任意自然数 n 等式成立，则

$$\begin{aligned} & 1+2+\ldots+(n+1) \\ &= (1+2+\ldots+n) + (n+1) \\ &= n(n+1)/2 + (n+1) \\ &= (n+1)\,(n/2+1) \\ &= (n+1)\,(n+2)/2 \end{aligned}$$

归纳假设代入

第一数学归纳法

适合证明涉及自然数的命题 $P(n)$

归纳基础：证明$P(1)$为真 (或$P(0)$为真).

归纳步骤：若对所有n有$P(n)$为真，证明 $P(n+1)$为真

$$\forall n, P(n) \to P(n+1)$$

$$P(1)$$

$n=1$， $P(1) \Rightarrow P(2)$

$n=2$, $P(2) \Rightarrow P(3)$

…

第二数学归纳法

适合证明涉及自然数的命题 $P(n)$

归纳基础：证明 $P(1)$为真 (或$P(0)$为真).

归纳步骤：若对所有小于n 的 k 有 $P(k)$真，
证明 $P(n)$为真

$$\forall k\,(\,k<n \wedge P(k)\,) \rightarrow P(n)$$

$$P(1)$$

$n=2$， $P(1) \Rightarrow P(2)$

$n=3$, $P(1) \wedge P(2) \Rightarrow P(3)$

…

两种归纳法的区别

归纳基础一样　　　$P(1)$为真

归纳步骤不同

证明逻辑

归纳法1：　$P(1) \Rightarrow P(2) \Rightarrow P(3) \ldots$

归纳法2：

$$P(1) \Rightarrow \left.\begin{array}{l} P(1) \\ P(2) \end{array}\right\} \Rightarrow \left.\begin{array}{l} P(1) \\ P(2) \\ P(3) \end{array}\right\} \Rightarrow P(4)\ \ldots$$

算法正确性归纳证明

证明步骤：

1. 叙述一个有关自然数*n*的命题，该命题断定该贪心策略的执行最终将导致最优解. 其中自然数 *n* 可以代表算法步数或者问题规模 .

2. 证明命题对所有的自然数为真 .
   归纳基础(从最小实例规模开始)
   归纳步骤(第一或第二数学归纳法)

活动选择算法的命题

命题

算法 Select执行到第 k 步，选择 k 项活动

$$i_1 = 1,\ i_2,\ \ldots,\ i_k$$

则存在最优解 A包含活动 $i_1=1, i_2, \ldots, i_k$.

根据上述命题：对于任何 k，算法前 k 步的选择都将导致最优解，至多到第 n 步将得到问题实例的最优解

归纳证明：归纳基础

令 $S=\{1,2,\ldots,n\}$是活动集,且$f_1 \le \ldots \le f_n$

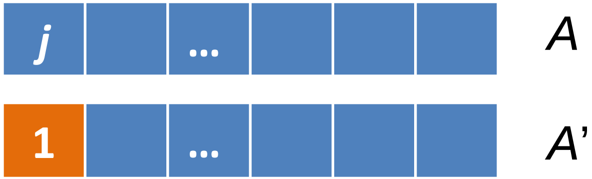
归纳基础：$k=1$, 证明存在最优解包含活动 1

证　任取最优解A，　A中活动按截止时间递增排列．　如果A的第一个活动为j，$j \neq 1$，用1替换A的活动j得到解A'，即

$$A' = (A-\{j\}) \cup \{1\},$$

由于$f_1 \le f_j$，A'也是最优解，且含有1.

$f_1 \le f_j$



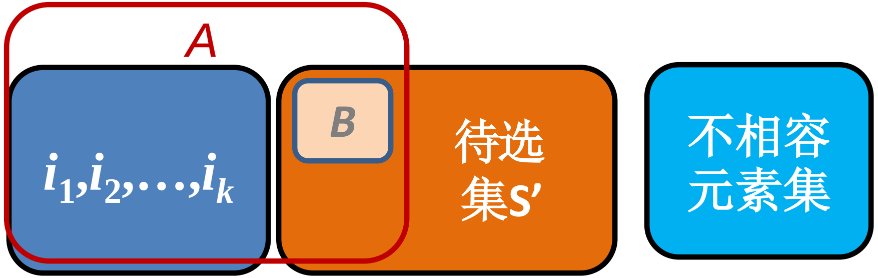
归纳步骤

假设命题对k为真，证明对$k+1$也为真.

证　算法执行到第 k 步, 选择了活动$i_1=1$, $i_2, \ldots, i_k$, 根据归纳假设存在最优解 A包含$i_1=1,i_2,\ldots,i_k$，A中剩下活动选自集合S'

$$S' = \{\, i \mid i \in S,\ s_i \geq f_k \,\}$$

$$A = \{\, i_1, i_2, \ldots, i_k \,\} \cup B$$

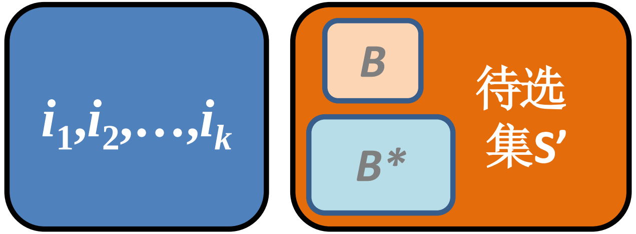


归纳步骤（续）

B是 S'的最优解.（若不然，S' 的最优解为B^*, B^*的活动比 B多，那么

$$B^* \cup \{1, i_2, \ldots, i_k\}$$

是 S 的最优解，且比 A的活动多,与 A 的最优性矛盾.）



归纳步骤(续)

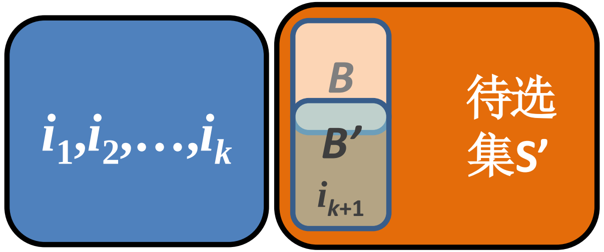
将S'看成子问题，根据归纳基础，

存在 S' 的最优解B'有S'中的第一个活动 i_{k+1}, 且 $|B'| = |B|$, 于是

$$\{ i_1, i_2, \dots , i_k \} \cup B'$$

$$= \{ i_1, i_2, \dots , i_k , i_{k+1} \} \cup (B' - \{ i_{k+1} \})$$

也是原问题的最优解.



小结

- 贪心法正确性证明方法：数学归纳法
  第一数学归纳法、第二数学归纳法
- 活动选择问题的贪心法证明：
  叙述一个涉及步数的算法正确性命题
  证明归纳基础
  证明归纳步骤

最优装载问题

最优装载问题

问题：

n 个集装箱1, 2, … , n 装上轮船，集装箱 i 的重量 w_i, 轮船装载重量限制为C, 无体积限制. 问如何装使得上船的集装箱最多？不妨设每个箱子的重量 $w_i \le C$.

该问题是0-1背包问题的子问题. 集装箱相当于物品，物品重量是 w_i，价值 v_i 都等于1，轮船载重限制 C 相当于背包重量限制 b.

建模

设 $\langle x_1, x_2, \ldots, x_n\rangle$ 表示解向量，$x_i = 0,1$,
$x_i = 1$当且仅当第 i 个集装箱装上船

目标函数 $\max \sum_{i=1}^{n} x_i$

约束条件 $\sum_{i=1}^{n} w_i x_i \le C$

$x_i = 0,1 \quad i = 1,2,\ldots,n$

算法设计

- 贪心策略：轻者优先
- 算法设计：
  将集装箱排序，使得

$$w_1 \le w_2 \le \ldots \le w_n$$

  按照标号从小到大装箱，直到装入下一个箱子将使得集装箱总重超过轮船装载重量限制，则停止.

正确性证明思路

- 命题：对装载问题任何规模为 n 的输入实例，算法得到最优解.

- 设集装箱从轻到重记为1, 2, … , n.

  归纳基础　证明对任何只含 1个箱子的输入实例，贪心法得到最优解. 显然正确.

- 归纳步骤　证明：假设对于任何n个箱子的输入实例贪心法都能得到最优解，那么对于任何n+1个箱子的输入实例贪心法也得到最优解.

归纳步骤证明思路

$N=\{1,2,\ldots,n+1\}$, $w_1 \le w_2 \le \ldots \le w_{n+1}$

⇓

去掉箱子1，令 $C' = C - \{w_1\}$，

得到规模 n 的输入 $N' = \{2,3,\ldots,n+1\}$

⇓

关于输入 N' 和C'的最优解 I'

⇓

在 I' 加入箱子1，得到 I

⇓

证明 I 是关于输入 N 的最优解

正确性证明

假设对于 n 个集装箱的输入，贪心法都可以得到最优解，考虑 输入

$$N = \{ 1, 2, \dots , n+1 \}$$

其中 $w_1 \le w_2 \le \dots \le w_{n+1}$.

由归纳假设，对于

$$N' = \{2, 3, \dots, n+1\}，\quad C' = C - w_1,$$

贪心法得到最优解 I'. 令

$$I = I' \cup \{1\}$$

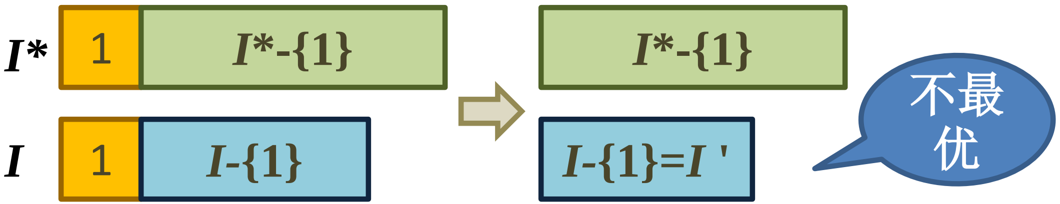
正确性证明(续)

***I* (算法解)是关于 *N* 的最优解.**

若不然，存在包含 1 的关于 *N* 的最优解 I^*(如果 I^* 中没有1，用 1 替换 I^* 中的第一个元素得到的解也是最优解), 且 $|I^*|>|I|$；那么$I^*–\{1\}$是 N' 和C' 的解且

$$|I^*-\{1\}| > |I-\{1\}| = |I'|$$

与 I'是关于N' 和C' 的最优解矛盾.



小结

- 装载问题是0-1背包的子问题 (每件物品重量为1)，NP难的问题存在多项式时间可解的子问题.
- 贪心法证明：对规模归纳

最小延迟调度问题

最小延迟调度

问题：

客户集合A，$\forall i \in A$，t_i为服务时间，d_i为要求完成时间，t_i，d_i为正整数。一个调度$f: A \to N$，$f(i)$为客户 i 的开始时间。求最大延迟达到最小的调度，即求 f 使得

$$\min_f \{\max_{i \in A}\{f(i)+t_i-d_i\}\}$$

$$\forall i, j \in A, i \neq j, f(i)+t_i \leq f(j)$$

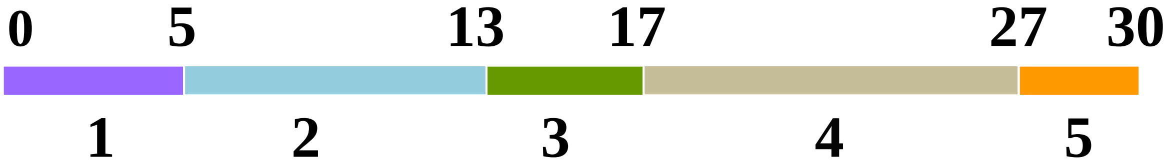
$$\text{or } f(j)+t_j \leq f(i)$$

实例：调度1

$A = \{1, 2, 3, 4, 5\}$, $T = \langle 5, 8, 4, 10, 3 \rangle$,
$D = \langle 10, 12, 15, 11, 20 \rangle$

调度1：顺序安排

$f_1(1)=0$, $f_1(2)=5$, $f_1(3)=13$, $f_1(4)=17$, $f_1(5)=27$



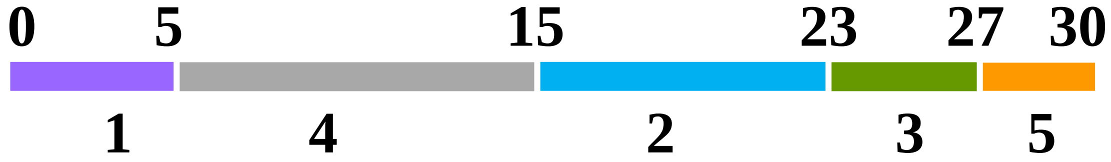
各任务延迟：0, 1, 2, **16**, 10;
最大延迟：16

更优的调度2

$A = \{1, 2, 3, 4, 5\}$, $T = <5, 8, 4, 10, 3>$,
$D = <10, 12, 15, 11, 20>$

调度2：按截止时间从前到后安排

$f_2(1)=0$, $f_2(2)=15$, $f_2(3)=23$, $f_2(4)=5$, $f_2(5)=27$



各任务延迟：0, 11, 12, 4, 10;
最大延迟：12

贪心策略

贪心策略1：按照 t_i 从小到大安排

贪心策略2：按照 $d_i - t_i$ 从小到大安排

贪心策略3：按照 d_i 从小到大安排

策略1 对某些实例得不到最优解.

反例：$t_1=1, d_1=100, t_2=10, d_2=10$

策略2 对某些实例得不到最优解.

反例：$t_1=1, d_1=2, t_2=10, d_2=10$

策略3伪码

算法 **Schedule**

输入：A, T, D

输出：f

1. 排序 A 使得 $d_1 \le d_2 \le \ldots \le d_n$
2. $f(1) \leftarrow 0$ 从0时刻起
3. $i \leftarrow 2$
4. **while** $i \le n$ **do**
5. $\quad f(i) \leftarrow f(i-1) + t_{i-1}$ 没有空闲
6. $\quad i \leftarrow i + 1$

设计思想：按完成时间从早到晚安排任务，没有空闲.

交换论证：正确性证明

证明思路：

- 分析一般最优解与算法解的区别（成分,排列顺序不同）
- 设计一种转换操作（替换成分或交换次序），可以在有限步将任意一个普通最优解逐步转换成算法的解
- 上述每步转换都不降低解的最优性质

贪心算法的解的性质：

没有空闲时间，没有逆序.

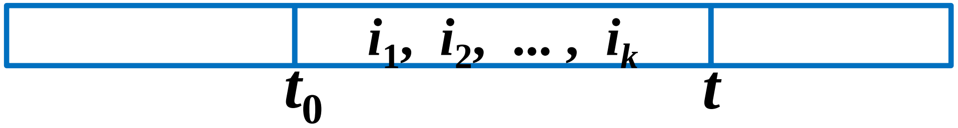
逆序 (i, j): $f(i) < f(j)$ 且 $d_i > d_j$

引理

引理1 所有没有逆序、没有空闲时间的调度具有相同的最大延迟.

证： 设 f 没有逆序，在 f 中具有相同完成时间 d 的客户 $i_1, i_2, \ldots, i_k$ 连续安排, 其开始时刻为 t_0, 完成这些任务的时刻是 t，最大延迟为最后任务延迟 $t-d$, 与 $i_1, i_2, \ldots, i_k$ 的排列次序无关.

$$t = t_0 + (t_{i_1} + t_{i_2} + \ldots + t_{i_k})$$



证明要点

从一个没有空闲的最优解出发，逐步转变成没有逆序的解．根据引理 1，这个解和算法解具有相同的最大延迟．

(1) 如果一个最优调度存在逆序，那么存在 $i<n$ 使得$(i, i+1)$ 构成一个逆序，称为相邻的逆序．

(2) 交换相邻逆序i 和j，得到的解仍旧最优．

(3) 每次交换后逆序数减 1，至多经过 $n(n-1)/2$ 次交换得到一个没有逆序的最优调度——等价于算法的解．

交换相邻逆序仍旧最优

设 f_1是一个任意最优解，存在相邻逆序 (i, j). 交换 i 和 j 的顺序, 得到解 f_2. 那么 f_2的最大延迟不超过 f_1的最大延迟.

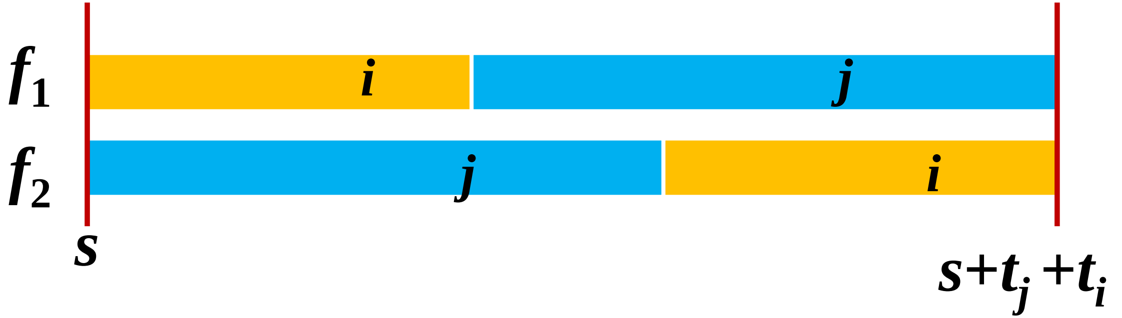
理由：

(1) 交换 i, j 与其他客户延迟时间无关

(2) 交换后不增加 j 的延迟，但可能增加 i 的延迟

(3) i 在 f_2的延迟小于 j 在 f_1 的延迟，因此小于f_1的最大延迟 r

i 在 f_2的延迟不超过 j 在f_1 的延迟



$$\text{delay}(f_2, i) = \underline{s + t_j + t_i} - d_i$$

$$\text{delay}(f_1, j) = \underline{s + t_i + t_j} - d_j$$

$$d_j < d_i$$

$$\text{delay}(f_2, i) < \text{delya}(f_1, j) \le r$$

小结

贪心法正确性证明方法：交换论证

- 分析算法解与一般最优解的区别，找到把一般解改造成算法解的一系列操作(替换成份、交换次序)
- 证明操作步数有限
- 证明每步操作后的得到解仍旧保持最优

得不到最优解的处理方法

得不到最优解的处理

- 输入参数分析：
  考虑输入参数在什么取值范围内使用贪心法可以得到最优解

- 误差分析：
  估计贪心法——近似算法所得到的解与最优解的误差（对所有的输入实例在最坏情况下误差的上界）

找零钱问题

问题 设有 n 种零钱， 重量分别为w_1, w_2, ... , w_n, 价值分别为$v_1 = 1$, v_2, ... , v_n. 需要付的总钱数是 y .不妨设币值和钱数都为正整数. 问：如何付钱使得所付钱的总重最轻?

实例

v_1=1, v_2=5, v_3=14, v_4=18, w_i=1, i=1,2,3,4.
y=28

最优解：x_3=2， x_1=x_2=x_4=0， 总重2

建模

令选用第 i 种硬币的数目是 x_i,

$i = 1, 2, \dots, n$

$$\min\{\sum_{i=1}^{n} w_i x_i\}$$

$$\sum_{i=1}^{n} v_i x_i = y, \quad x_i \in \mathrm{N}, \quad i = 1,2,\dots,n$$

动态规划算法

设$F_k(y)$表示用前 k种零钱，总钱数为 y 的最小重量

$$F_k(y) = \min_{0 \le x_k \le \left\lfloor \frac{y}{v_k} \right\rfloor} \{F_{k\text{-}1}(y - v_k x_k) + w_k x_k\}$$

$$F_1(y) = w_1 \left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

贪心法

单位价值重量轻的货币优先，设

$$\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \dots \geq \frac{w_n}{v_n}$$

使用前 k 种零钱，总钱数为 y

贪心法的总重为 $G_k(y)$，

$$G_k(y) = w_k \left\lfloor \frac{y}{v_k} \right\rfloor + G_{k\text{-}1}(y \bmod v_k) \quad k > 1$$

$$G_1(y) = w_1 \left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

n=1,2贪心法是最优解

$n = 1$ 只有一种零钱，$F_1(y) = G_1(y)$

$n = 2$，x_2越大，得到的解越好

$$\boxed{F_2(y) = \min_{0 \le x_2 \le \lfloor y/v_2 \rfloor} \{F_1(y - v_2 x_2) + w_2 x_2\}}$$

$$\begin{aligned}
&F_1(y - v_2(\underline{x_2+\delta})) + w_2(\underline{x_2+\delta})] \\
&\quad - [F_1(y - v_2 \underline{x_2}) + w_2 \underline{x_2}] \\
&= [w_1(\underline{y - v_2 x_2} - v_2\delta) + \underline{w_2 x_2} + w_2\delta)] \\
&\quad - [w_1(\underline{y - v_2 x_2}) + \underline{w_2 x_2}] \\
&= -w_1 v_2 \delta + w_2 \delta \\
&= \delta(-w_1 v_2 + w_2) \le 0
\end{aligned}$$

$w_1 \ge w_2/v_2$

判别条件

定理 对每个正整数 k，假设对所有非负整数 y 有$G_k(y) = F_k(y)$，且存在 p 和 δ 满足

$$\boxed{v_{k+1} = pv_k - \delta,}$$

其中 $0 \le \delta < v_k$， $v_k \le v_{k+1}$, p为正整数，

则下面的命题等价：

(1) $G_{k+1}(y) = F_{k+1}(y)$ 对一切正整数 y；

(2) $G_{k+1}(pv_k) = F_{k+1}(pv_k)$；

(3) $w_{k+1} + G_k(\delta) \le pw_k$.

几点说明

- 根据条件(1)与(3)的等价性，可以对 $k = 3, 4, \ldots, n$，依次利用条件(3)对贪心法是否得到最优解做出判别.
- 条件(3)验证 1 次需 $O(k)$ 时间，$k = O(n)$，整个验证时间$O(n^2)$
- 条件(2)是条件(1) 在 $y = pv_k$ 时的特殊情况. 若条件(1)成立，显然有条件(2)成立. 反之，若条件(2)不成立，则条件(1)不成立，钱数 $y = pv_k$ 恰好提供了一个贪心法不正确的反例.

验证实例

$$v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in Z^+$$
$$w_{k+1} + G_k(\delta) \le pw_k$$

例 $v_1=1, v_2=5, v_3=14, v_4=18, w_i=1$, $i=1,2,3,4$. 对一切 y 有

$$G_1(y)=F_1(y), \quad G_2(y)=F_2(y).$$

验证 $G_3(y) = F_3(y)$

$$v_3 = pv_2 - \delta \Rightarrow \boxed{p = 3, \delta = 1.}$$

$$w_3+G_2(\delta)=1+1=2$$

$$pw_2=3\times1=3$$

$$\boxed{w_3+G_2(\delta)\le pw_2}$$

贪心法对于 $n=3$ 的实例得到最优解

验证实例

$$v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in Z^+$$
$$w_{k+1} + G_k(\delta) \le pw_k$$

例 $v_1=1, v_2=5, v_3=14, v_4=18$, $w_i=1$, $i=1,2,3,4$. 对一切 y 有

$$G_1(y)=F_1(y), G_2(y)=F_2(y), G_3(y)=F_3(y)$$

验证 $G_4(y) = F_4(y)$

$$v_4 = pv_3 - \delta \Rightarrow \boxed{p=2,\ \delta=10}$$

$$w_4 + G_3(\delta)=1+2=3$$

$$pw_3 = 2\times 1=2$$

$$\boxed{w_4 + G_3(\delta) > pw_3}$$

$n=4$, $y=pv_3=28$,

最优解: $x_3=2$, 贪心法: $x_4=1$, $x_2=2$

小结

- 贪心策略不一定得到最优解，在这种情况下可以有两种处理方法：
  (1) 参数化分析：分析参数取什么值
  可得到最优解
  (2) 估计贪心法得到的解在最坏情况
  下与最优解的误差

- 一个参数化分析的例子:找零钱问题

本周教学内容

重要的贪心算法

哈夫曼算法的证明

最优前缀码及哈夫曼算法

Kruskal算法

Prim算法

最小生成树

Dijkstra算法的正确性证明

单源最短路径Dijkstra算法

最优前缀码

二元前缀码

- 二元前缀码

  用0-1字符串作为代码表示字符，要求任何字符的代码都不能作为其它字符代码的前缀

- 非前缀码的例子

  a: 001, *b*: 00, *c*: 010, *d*: 01

- 解码的歧义，例如字符串 0100001

  解码1: 01, 00, 001 *d*, *b*, *a*

  解码2: 010, 00, 01 *c*, *b*, *d*

前缀码的二叉树表示

前缀码：

{00000, 00001, 0001, 001, 01,100,101,11}

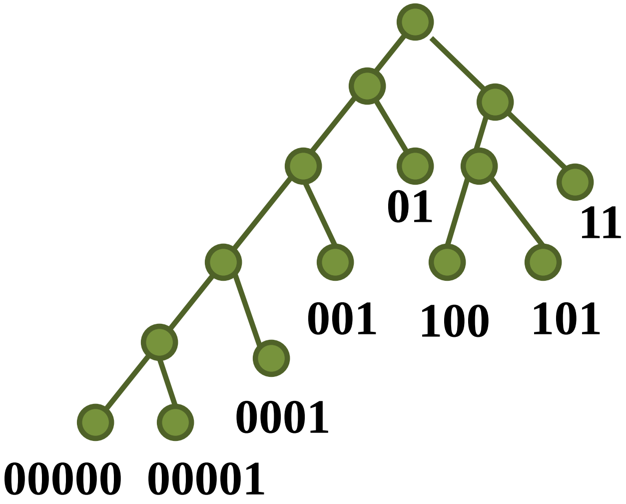
构造树：

0 -左子树

1-右子树

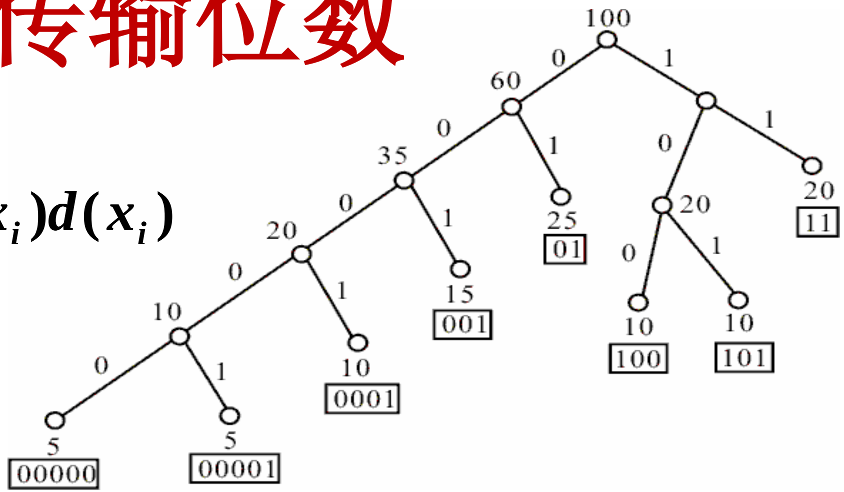
码对应一片树叶

最大位数为树深



平均传输位数

$$B=\sum_{i=1}^{n} f(x_i)d(x_i)$$



B=[(5+5)×5+10×4+(15+10+10)×3

+(25+20)×2]/100=2.85

问题：给定字符集$C=\{x_1,x_2,\ldots,x_n\}$和每个字符的频率$f(x_i)$，$i=1,2,\ldots,n$. 求关于C的一个最优前缀码(平均传输位数最小).

哈夫曼树算法伪码

算法 Huffman(C)

输入： C ={x_1, x_2,…, x_n}, $f(x_i)$, i=1,2,…,n.

输出： Q / /队列

1. n←|C |
2. Q←C //频率递增队列Q
3. for i←1 to n–1 do
4. z←Allocate-Node() //生成结点 z
5. $z.left$←Q中最小元 //最小作z左儿子
6. $z.right$←Q中最小元 //最小作z右儿子
7. $f(z)$←$f(x)$+$f(y)$
8. Insert(Q,z) // 将 z 插入Q
9. return Q

实例

输入 *a*:45; *b*:13; *c*:12; *d*:16; *e*:9; *f*:5

编码：

f--0000,

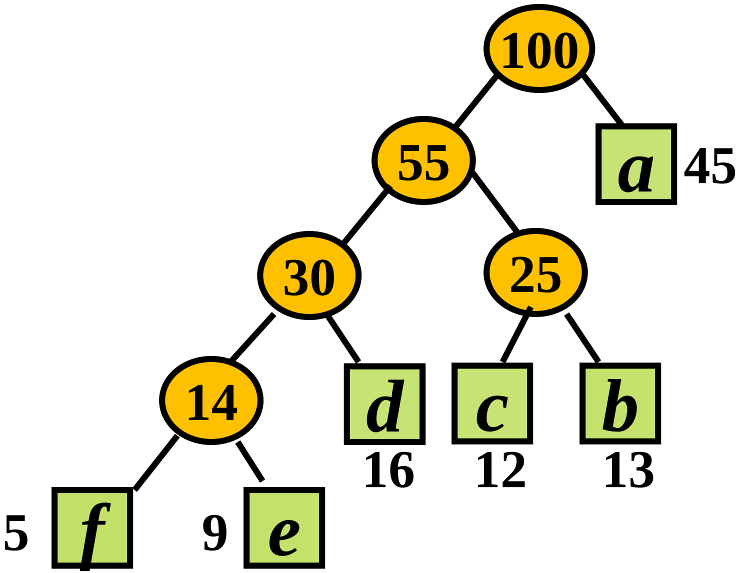
e--0001,

d--001,

c--010,

b—011,

a--1



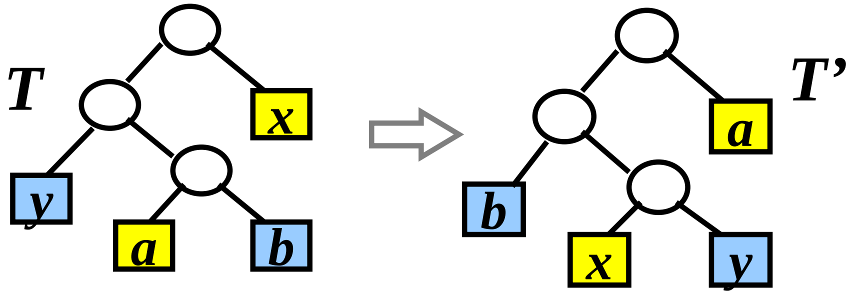
平均位数：

$4\times(0.05+0.09)+3\times(0.16+0.12+0.13)+1\times0.45=2.24$

最优前缀码性质：引理1

引理1: C是字符集，$\forall c\in C$, $f(c)$为频率, $x,y\in C$, $f(x)$, $f(y)$频率最小, 那么存在最优二元前缀码使得 x, y 码字等长且仅在最后一位不同.

$f(x)\le f(a)$
$f(y)\le f(b)$



$$B(T)-B(T')=\sum_{i\in C} f[i]d_T(i)-\sum_{i\in C} f[i]d_{T'}(i)\ge 0$$

其中$d_T(i)$为i 在T 中的层数(i到根的距离)

引理2

引理 设T是二元前缀码的二叉树,$\forall\, x,y \in T$, x, y是树叶兄弟，z 是 x, y的父亲, 令

$$T' = T-\{x, y\}$$

且令 z 的频率

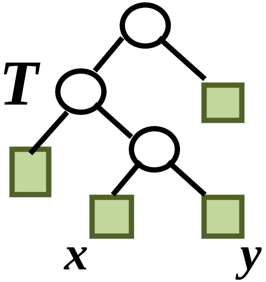
$$f(z) = f(x)+f(y)$$

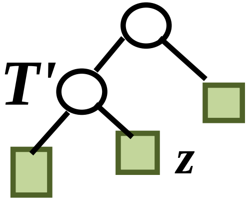
T'是对应二元前缀码

$$C' = (C-\{\, x, y\,\})\cup\{\, z\,\}$$

的二叉树，那么

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





引理2证明

证 $\forall c \in C-\{x,y\}$,有

$$d_T(c) = d_{T'}(c) \Rightarrow f(c)d_T(c) = f(c)d_{T'}(c)$$

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

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

$$= \sum_{i \in T, i \neq x,y} f(i) d_T(i) + f(x) d_T(x) + f(y) d_T(y)$$

$$= \sum_{i \in T', i \neq z} f(i) d_{T'}(i) + f(z) d_{T'}(z) + (f(x) + f(y))$$

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

小结

- 二元前缀码及其二叉树表示
- 给定频率下的平均传输位数计算公式
- 最优前缀码——平均传输位数最少
- 哈夫曼算法
- 前缀码的性质

哈夫曼算法
的证明及应用

两个引理

引理1:设C是字符集，$\forall c\in C$, $f(c)$为频率，$x, y\in C$, $f(x)$, $f(y)$频率最小，那么存在最优二元前缀码使得 x, y 码字等长，且仅在最后一位不同.

引理2 设 T 是二元前缀码所对应的二叉树，$\forall\, x, y\in T$, x, y是树叶兄弟，z 是 x, y 的父亲，令$T'= T-\{x, y\}$, 且令 z 的频率$f(z)=f(x)+f(y)$，T'是对应于二元前缀码 $C' =(C-\{x, y\})\cup\{z\}$的二叉树，那么 $B(T)=B(T')+f(x)+f(y)$.

算法正确性证明思路

定理　**Huffman** 算法对任意规模为*n*（*n*≥2）的字符集*C* 都得到关于*C* 的最优前缀码的二叉树.

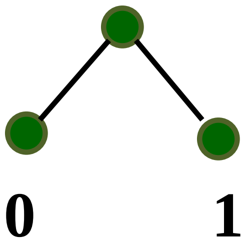
归纳基础　证明：对于*n*=2的字符集, **Huffman**算法得到最优前缀码.

归纳步骤　证明：假设**Huffman**算法对于规模为 *k* 的字符集都得到最优前缀码，那么对于规模为*k*+1的字符集也得到最优前缀码.

归纳基础

n=2，字符集　$C=\{x_1, x_2\}$，

对任何代码的字符至少都需要1位二进制数字. Huffman算法得到的代码是 0 和 1，是最优前缀码.



归纳步骤

假设Huffman算法对于规模为 k 的字符集都得到最优前缀码. 考虑规模为 k+1的字符集

$$C = \{x_1, x_2, \ldots, x_{k+1}\},$$

其中 $x_1, x_2 \in C$是频率最小的两个字符.

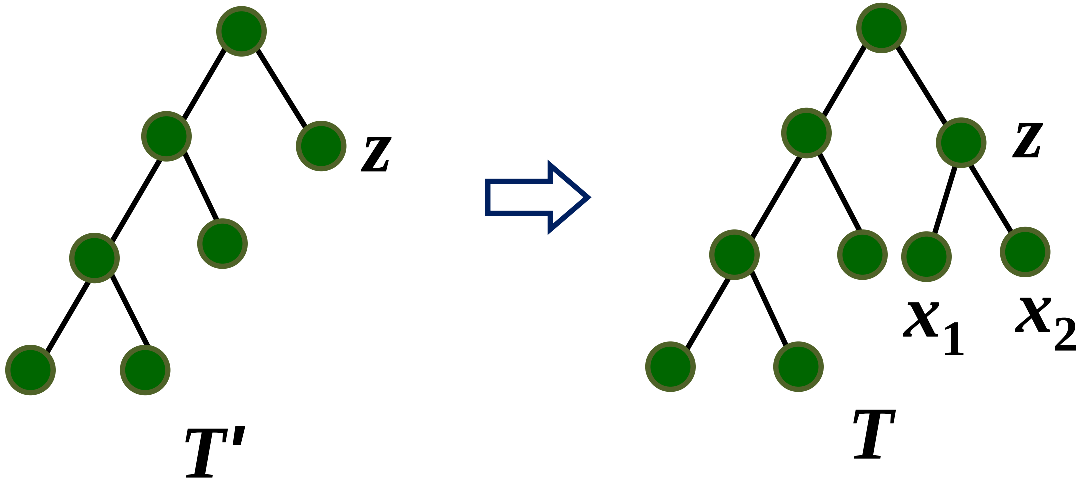
令

$$\boxed{\begin{gathered} C' = (C - \{x_1, x_2\}) \cup \{z\}, \\ f(z) = f(x_1) + f(x_2) \end{gathered}}$$

根据归纳假设，算法得到一棵关于字符集 C',频率$f(z)$和$f(x_i)$ (i =3,4, ...,k+1)的最优前缀码的二叉树T'.

归纳步骤(续)

把 x_1, x_2作为 z 的儿子附到 T'上，得到树T，那么T是关于$C=(C'-\{z\})\cup\{x_1, x_2\}$的最优前缀码的二叉树.



归纳步骤(续)

如若不然, 存在更优树 T^*, $B(T^*)<B(T)$, 且由引理1，其树叶兄弟是 x_1和 x_2.

去掉 T^*中 x_1和 x_2，得到$T^{*\prime}$. 根据引理2

$$\begin{aligned} B(T^{*\prime}) &= B(T^*) - (\underline{f(x_1)+f(x_2)}) \\ &< B(T) - (\underline{f(x_1)+f(x_2)}) \\ &= B(T') \end{aligned}$$

与 T'是一棵关于 C'的最优前缀码的二叉树矛盾.

应用：文件归并

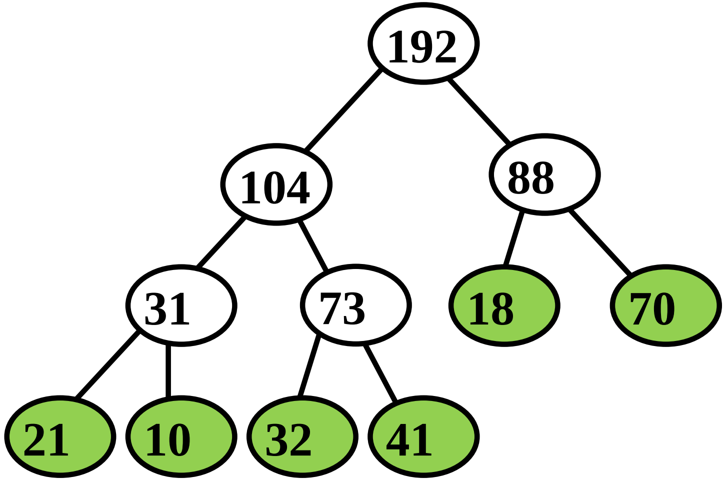
问题：给定一组不同长度的排好序文件构成的集合 $S = \{ f_1, \ldots , f_n\}$，其中 f_i 表示第 i 个文件含有的项数. 使用二分归并将这些文件归并成一个有序文件.

归并过程对应于二叉树：

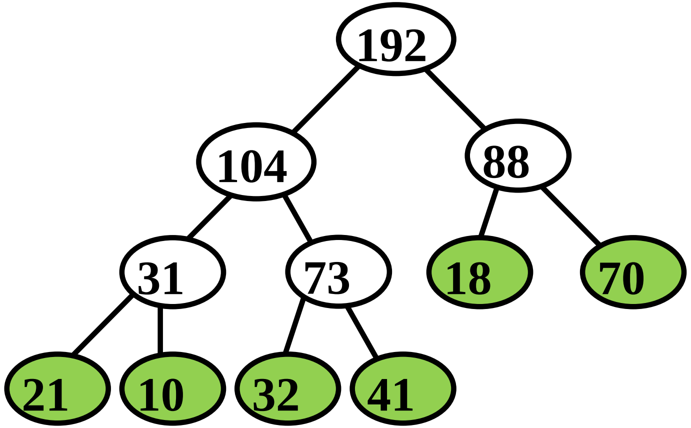
文件为树叶. f_i 与 f_j 归并的文件是它们的父结点.

两两顺序归并

实例：S = { 21,10,32,41,18,70 }



归并代价



(1) (21+10–1)+(32+41–1)+(18+70–1)+

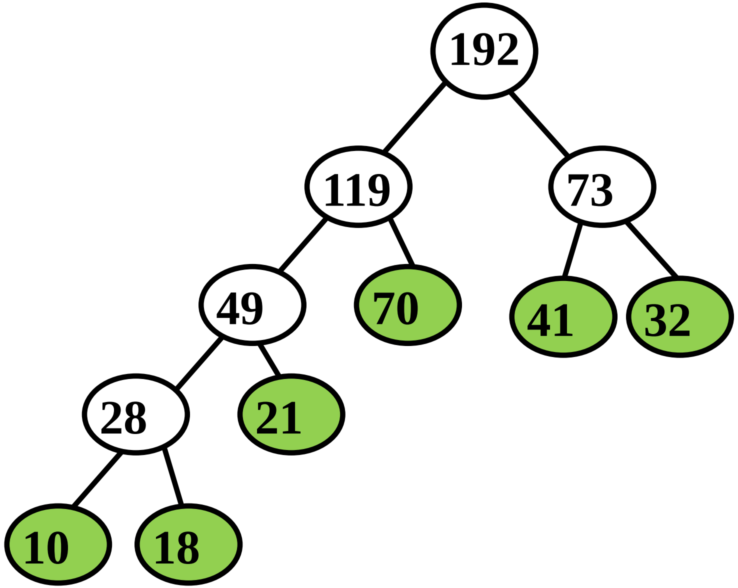
(31+73–1)+(104+88–1) = 483

(2) (21+10+32+41)×3+(18+70)×2–5=483

代价计算公式 $\boxed{\sum_{i\in S} d(i) f_i - (n-1)}$

实例：Huffman树归并

输入：S={21,10,32,41,18,70}



代价: (10+18)×4+21×3+(70+41+32)×2–5=456

小结

- 哈夫曼算法的正确性证明：
  对规模归纳
- 哈夫曼算法的应用：
  文件归并

最小生成树

最小生成树

无向连通带权图

$$G = (V, E, W),$$

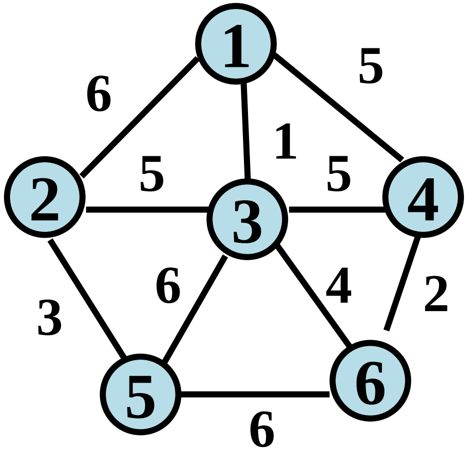
其中 $w(e) \in W$ 是边 e 的权.

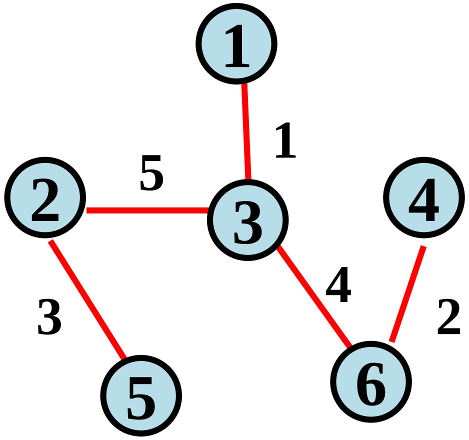
G 的一棵生成树 T 是包含了G 的所有顶点的树，树中各边的权之和 $W(T)$ 称为树的权，具有最小权的生成树称为 G 的最小生成树.

最小生成树的实例

$G=(V,E,W), V=\{1,2,3,4,5,6\}, W$ 如图所示.

$E = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,5\},$
$\{3,4\},\{3,5\},\{3,6\},\{4,6\},\{5,6\}\}$



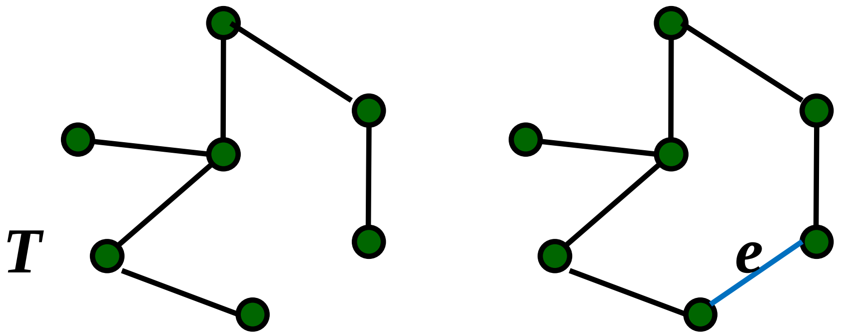


生成树的性质

命题1 设G是 n 阶连通图, 那么

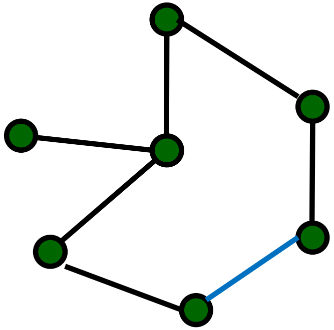
(1) T 是G 的生成树当且仅当T 无圈且有 $n-1$条边.

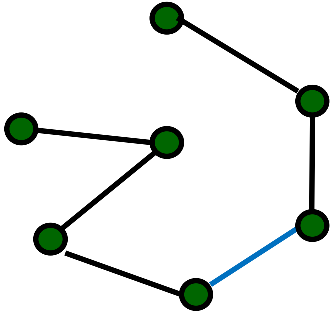
(2) 如果 T 是 G 的生成树， $e \notin T$， 那么 $T \cup \{ e \}$含有一个圈C (回路).



生成树的性质（续）

(3) 去掉圈C的任意一条边，就得到G的另外一棵生成树T'。





T'

生成树性质的应用

- 算法步骤：选择边.
  约束条件：不形成回路
  截止条件：边数达到 n-1.
- 改进生成树 T 的方法
  在 T 中加一条非树边 e , 形成回路 C, 在 C 中去掉一条树边 e_i，形成一棵新的生成树 T'

$$W(T') - W(T) = W(e) - W(e_i)$$

若 $W(e) \leq W(e_i)$，则 $W(T') \leq W(T)$

求最小生成树

问题：

给定连通带权图 $G = (V, E, W)$，$w(e) \in W$是边 e 的权. 求 G 的一棵最小生成树.

贪心法：

Prim 算法，

Kruskal 算法

生成树在网络中有着重要应用

小结

- 生成树与生成树的权
- 最小生成树
- 生成树的性质

Prim算法

设计思想

输入：图 $G=(V,E,W)$, $V=\{1,2,\ldots,n\}$

输出：最小生成树 T

设计思想：

初始 $S=\{1\}$,

选择连接 S 与 $V–S$ 集合的最短边 $e=\{i,j\}$，其中 $i\in S, j\in V–S$. 将 e 加入树 T，j 加入 S.

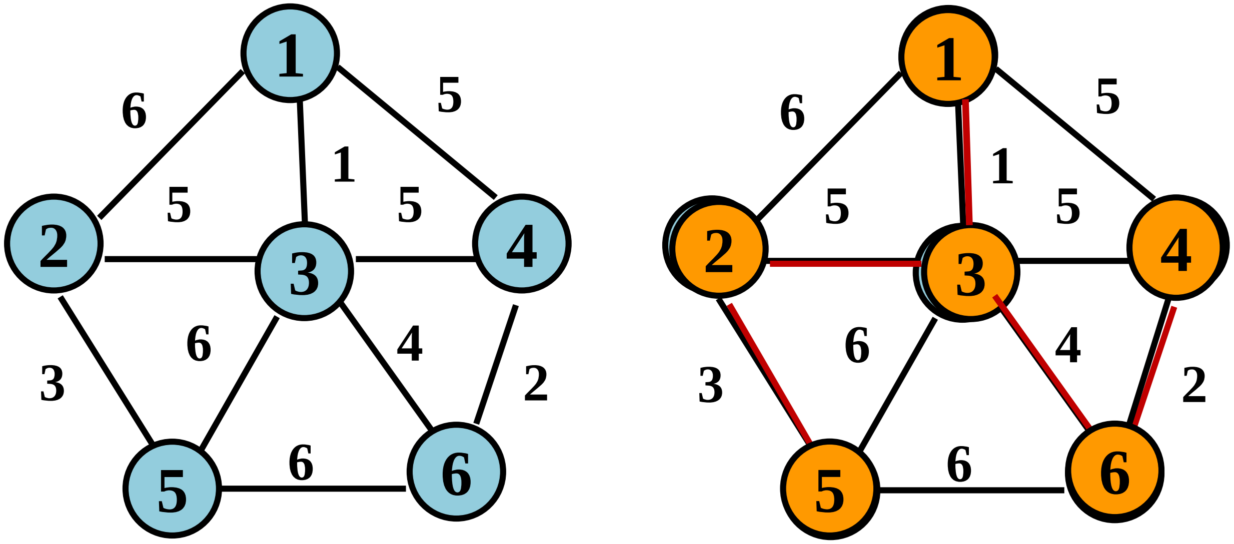
继续执行上述过程，直到 $S=V$ 为止.

伪码

算法 Prim (G, E, W)

1. $S \leftarrow \{ 1 \}$
2. **while** $V - S \neq \varnothing$ **do**
3. 　　从V–S中选择 j 使得 j 到 S 中顶点的边权最小
4. 　　$S \leftarrow S \cup \{ j \}$

实例



正确性证明:归纳法

命题：对于任意 $k < n$, 存在一棵最小生成树包含算法前 k 步选择的边.

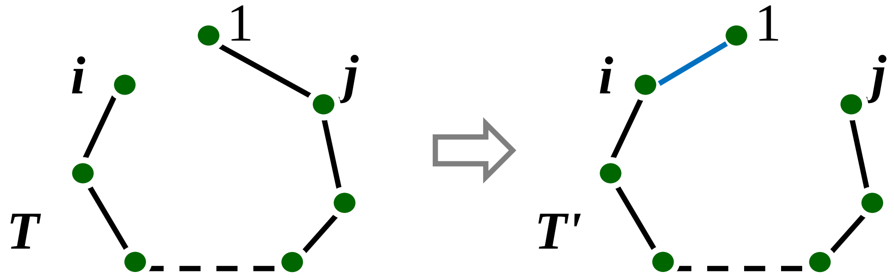
归纳基础：$k = 1$, 存在一棵最小生成树 T 包含边 $e = \{1, i\}$, 其中$\{1, i\}$是所有关联 1 的边中权最小的.

归纳步骤：假设算法前 k 步选择的边构成一棵最小生成树的边，则算法前 $k+1$ 步选择的边也构成一棵最小生成树的边.

归纳基础

证明：存在一棵最小生成树 T 包含关联结点1的最小权的边$e=\{1,i\}$.

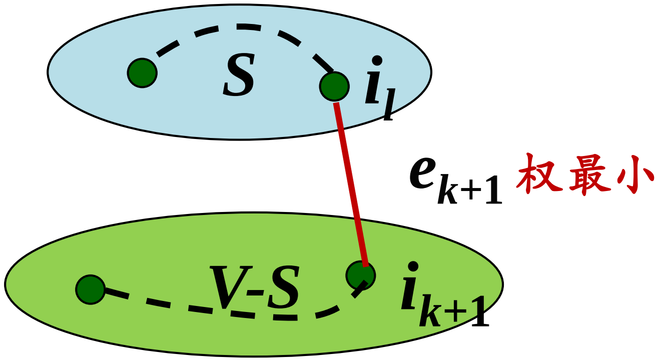
证 设 T 为一棵最小生成树，假设 T 不包含$\{1, i\}$, 则$T\cup\{\{1,i\}\}$ 含有一条回路，回路中关联1的另一条边$\{1, j\}$. 用$\{1, i\}$替换$\{1, j\}$得到树 T'， 则 T' 也是生成树，且 $W(T') \leq W(T)$.



归纳步骤

假设算法进行了k步，生成树的边为$e_1, e_2, \ldots, e_k$，这些边的端点构成集合S. 由归纳假设存在 G 的一棵最小生成树 T 包含这些边.

算法第 $k+1$ 步选择顶点 i_{k+1}, 则 i_{k+1} 到S中顶点边权最小,设此边 $e_{k+1}=\{i_{k+1}, i_l\}$. 若$e_{k+1} \in T$，算法$k+1$步显然正确.



归纳步骤（续）

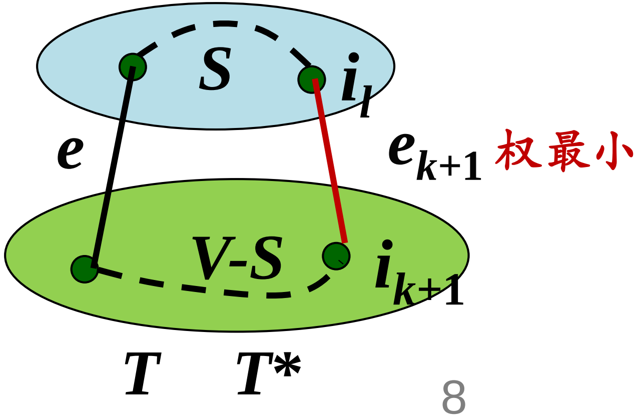
假设 T 不含有 e_{k+1}，则将 e_{k+1} 加到 T 中形成一条回路. 这条回路有另外一条连接 S 与 $V-S$中 顶点的边 e,

令 $T^*= (T-\{e\}) \cup\{e_{k+1}\}$

则 T^*是 G 的一棵生成树,包含$e_1,e_2,\ldots,e_{k+1}$, 且

$$W(T^*) \le W(T)$$

算法到 $k+1$步仍得到最小生成树.



时间复杂度

算法步骤执行$O(n)$次

每次执行$O(n)$时间：

找连接 S与$V\text{-}S$ 的最短边

算法时间：$T(n) = O(n^2)$

小结

- **Prim算法的设计**

  贪心策略：连接S与V-S的最短边

  正确性证明：对步数归纳

  伪码

- 时间复杂度：$O(n^2)$

Kruskal算法

设计思想

输入：图$G=(V,E,W)$, $V=\{1,2,\ldots,n\}$

输出：G的最小生成树 T

设计思想：

(1) 按照长度从小到大对边排序.

(2) 依次考察当前最短边 e，如果 e与 T 的边不构成回路，则把 e 加入树 T，否则跳过 e. 直到选择了n-1条边为止.

伪码

算法 Kruskal

输入：连通图G　　// 顶点数n，边数m

输出：G 的最小生成树

1. 权从小到大排序E的边, $E=\{e_1,e_2,\ldots,e_m\}$

2. $T \leftarrow \varnothing$

3．repeat

4.　　$e \leftarrow E$中的最短边

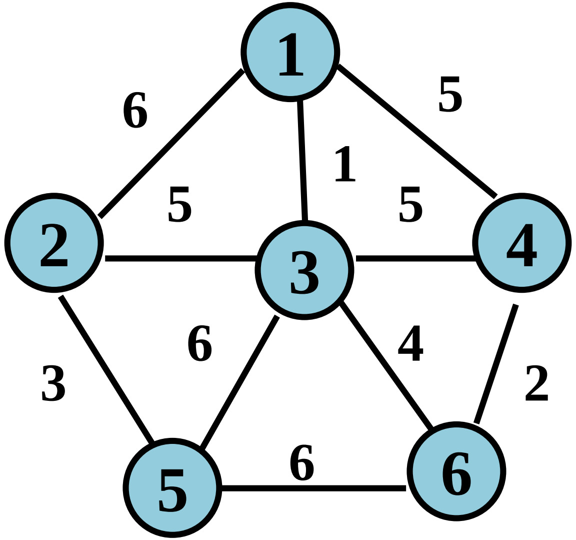
5．　　if e 的两端点不在同一连通分支

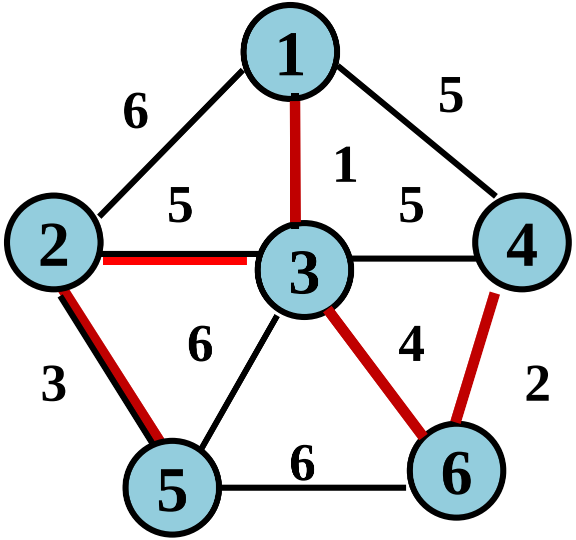
6.　　then $T \leftarrow T \cup \{e\}$

7.　　$E \leftarrow E - \{e\}$

8. until T 包含了$n-1$条边

实例





正确性证明思路

命题：对于任意 n, 算法对 n 阶图找到一棵最小生成树.

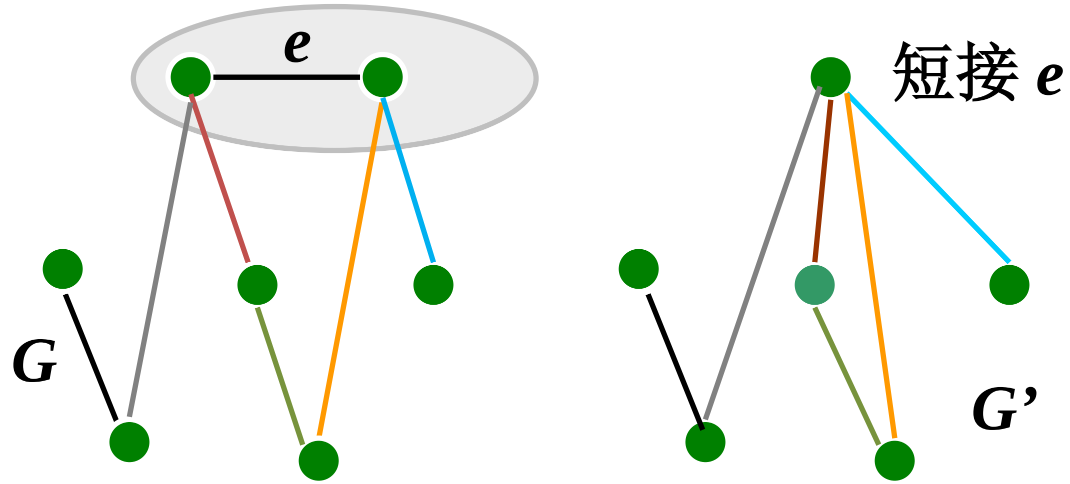
证明思路：

归纳基础　证明：$n = 2$, 算法正确.
G只有一条边，最小生成树就是 G.

归纳步骤　证明：假设算法对于n 阶图是正确的，其中$n>1$，则对于任何$n+1$阶图算法也得到一棵最小生成树.

短接操作

任给 n+1个顶点的图 G，G中最小权边 $e = \{i, j\}$，从G 中短接i 和 j, 得到图 G'.



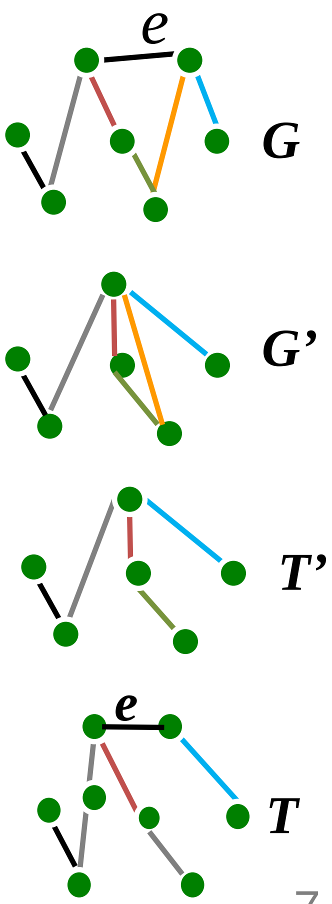
归纳步骤证明

对于任意 n+1阶图G短接最短边e，得到 n 阶图 G'

根据归纳假设算法得到G'的最小生成树 T'

将被短接的边 e “拉伸”回到原来长度，得到树 T

证明 T 是 G 的最小生成树



T是G的最小生成树

$T=T'\cup\{e\}$是关于G 的最小生成树.

否则存在G 的含边e 的最小生成树 $T^*, W(T^*)<W(T)$. (如果 $e\notin T^*$, 在T^* 中加边e，形成回路. 去掉回路中任意别的边所得生成树的权仍旧最小).

在T^*短接e得到G' 的生成树$T^*-\{e\}$,

$$\boxed{\begin{aligned} & W(T^*-\{e\}) = W(T^*) - w(e) \\ & < W(T) - w(e) = W(T') \end{aligned}}$$

与T' 的最优性矛盾.

算法实现与时间复杂度

建立FIND数组，FIND[i] 是结点 i 的连通分支标记.

(1) 初始 FIND[i] = i.

(2) 连通分支合并，较小分支标记更新为较大分支标记

每个结点至多更新 log$n$次，
建立和更新 FIND数组: O (n log$n$)

时间：$O(m \log m)+O(n\log n)+O(m)$
$= O(m \log n)$

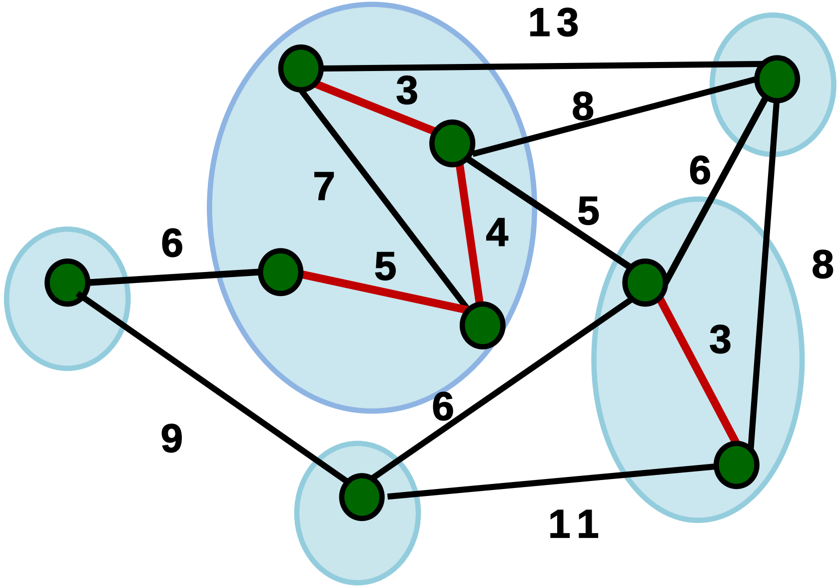
应用：数据分组问题

一组数据(照片, 文件, 生物标本)要把它们按照相关性进行分类.

用相似度函数或“距离”来描述个体之间的差距.

如果分成5类，使得每类内部的个体尽可能相近，不同类之间的个体尽可能地“远离”. 如何划分?

应用：数据分组问题



单链聚类

类似于Kruskal算法

(1) 按照边长从小到大对边排序

(2) 依次考察当前最短边 e，如果 e 与 已经选中的边不构成回路，则把 e 加入集合，否则跳过 e. 计数图的连通分支个数.

(3) 直到保留了k个连通分支为止.

小结

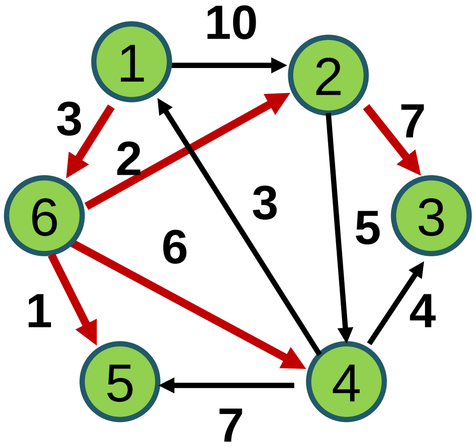
- **Kruskal算法的贪心策略：在不构成回路条件下选当前最短边**
- 正确性证明：对规模归纳
- 时间复杂度：$O(m\log n)$
- 应用：单链聚类

单源最短路径

单源最短路问题

给定带权有向网络 $G=(V,E,W)$，每条边 $e=<i,j>$的权 $w(e)$为非负实数，表示从 i 到 j 的距离. 源点 $s\in V$.

求: 从 s 出发到达其它结点的最短路径.



源点：1

1→6→2：*short* [2] = 5

1→6→2→3：*short* [3] = 12

1→6→4：*short* [4] = 9

1→6→5：*short* [5] = 4

1→6：*short* [6] = 3

Dijkstra算法有关概念

$x \in S \Leftrightarrow x \in V$ 且从 s 到 x 的最短路径已经找到

初始：$S = \{ s \}$， $S = V$ 时算法结束

从s到u相对于S的最短路径：从s到u且仅经过 S 中顶点的最短路径

$dist\,[u]$：从s到u相对S最短路径的长度
$short\,[u]$：从s到u的最短路径的长度
$dist\,[u] \geq short[u]$

算法设计思想

输入：有向图 $G = (V, E, W)$,

$V = \{ 1, 2, \ldots, n \}$，$s = 1$

输出：从s到每个顶点的最短路径

1. 初始 $S=\{1\}$
2. 对于$i \in V-S$，计算1到 i 的相对 S 的最短路，长度 $dist[i]$
3. 选择$V-S$中 $dist$ 值最小的 j，将 j 加入 S，修改$V-S$中顶点的$dist$ 值.
4. 继续上述过程，直到 $S=V$为止.

伪码

算法 Dijkstra

$S \leftarrow \{ s \}$

$dist[s] \leftarrow 0$

for $i \in V - \{ s \}$ do

　　$dist[i] \leftarrow w(s,i)$ //s到i没边, $w(s,i)=\infty$

while $V - S \neq \varnothing$ do

　　从$V - S$取相对S的最短路径顶点 j

　　$S \leftarrow S \cup \{ j \}$

　　for $i \in V - S$ do

　　　　if $dist[j]+w(j,i)<dist[i]$

　　　　then $dist[i] \leftarrow dist[j]+w(j,i)$

更新dist值

运行实例

输入： $G=<V,E,W>$，源点 1

$V=\{1,2,3,4,5,6\}$

$S=\{1\}$,

$dist[1]=0$

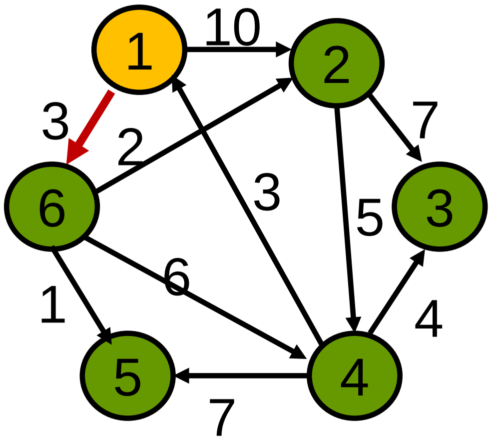
$dist[2]=10$

$dist[6]=3$

$dist[3]=\infty$

$dist[4]=\infty$

$dist[5]=\infty$



实例（续）

***S*={1,6}**

***dist* [1] = 0**

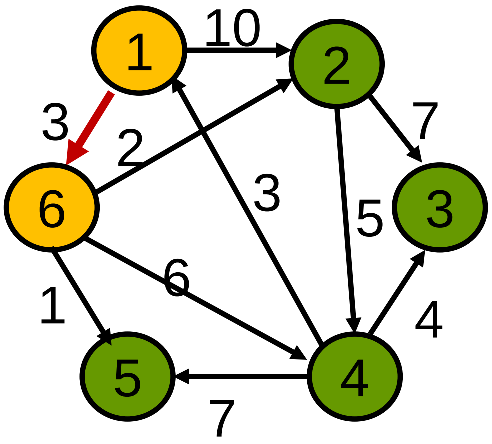
***dist* [6] = 3**

***dist* [2] = 5**

***dist* [4] = 9**

***dist* [5] = 4**

***dist* [3] = ∞**



运行实例（续）

S={1,6,5}

dist [1]=0

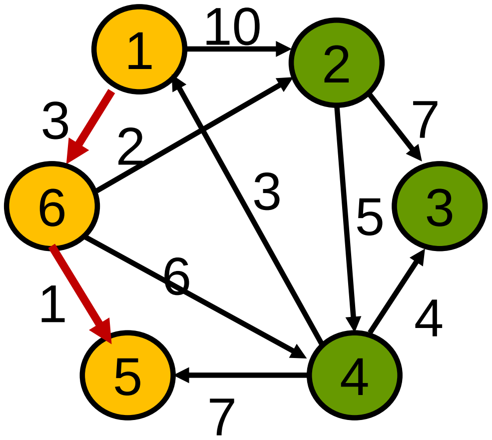
dist [6]=3

dist [5]=4

dist [2]=5

dist [4]=9

dist [3]=∞



运行实例（续）

S **={1,6,5,2}**

dist**[1]=0**

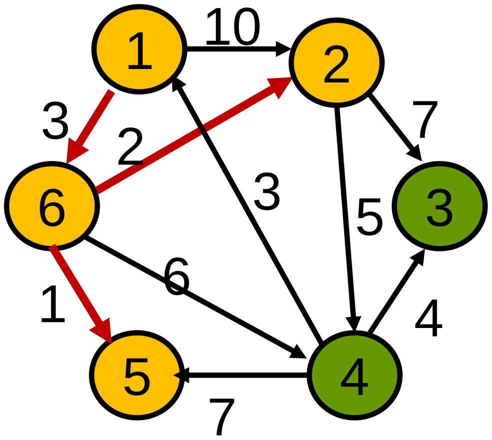
dist**[6]=3**

dist**[5]=4**

dist**[2]=5**

dist**[4]=9**

dist**[3]=12**



运行实例（续）

***S*={1,6,5,2,4}**

***dist*[1]=0**

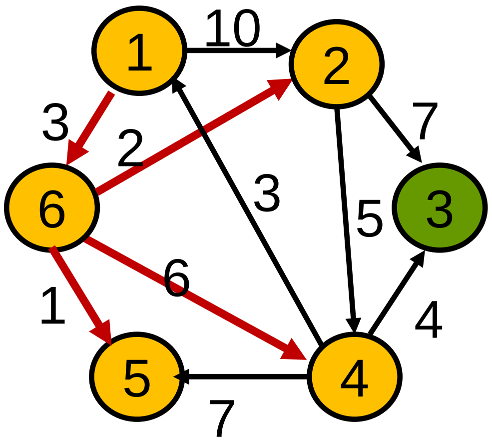
***dist*[6]=3**

***dist*[5]=4**

***dist*[2]=5**

***dist*[4]=9**

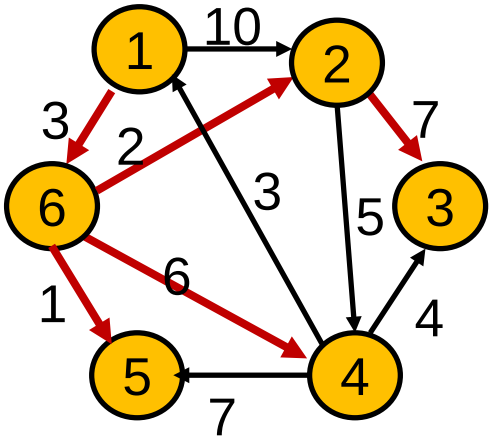
***dist*[3]=12**



运行实例（续）

S={1,6,5,2,4,3}

***dist* [1]=0**
***dist* [6]=3**
***dist* [5]=4**
***dist* [2]=5**
***dist* [4]=9**
***dist* [3]=12**



***short*[1]=0, *short*[2]=5, *short*[3]=12,**
***short*[4]=9, *short*[5]=4, *short*[6]=3.**

小结

- **单源最短路径问题**
- **Dijkstra算法设计思想及伪码**
- **运行实例**

Dijkstra算法的正确性

归纳证明思路

命题：当算法进行到第 k 步时，对于 S 中每个结点 i，

$$dist\,[i] = short\,[i]$$

归纳基础

$k = 1$, $S = \{s\}$, $dist\,[s] = short\,[s] = 0$.

归纳步骤

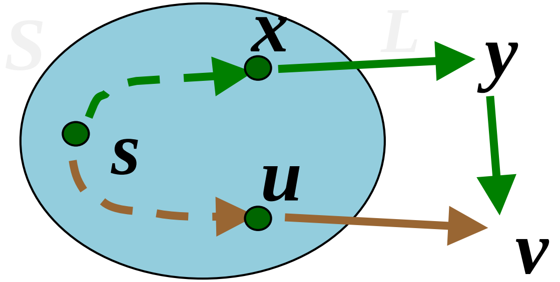
证明：假设命题对 k 为真，则对 k+1 命题也为真.

归纳步骤证明

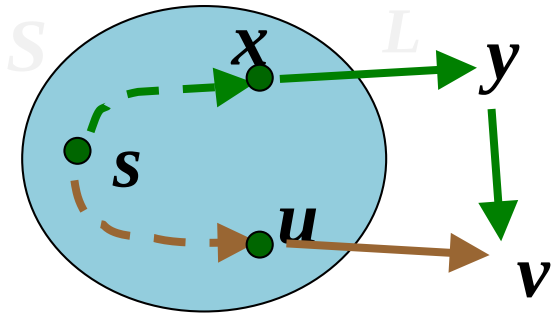
假设命题对*k*为真，考虑 *k*+1步算法选择顶点*v* (边<*u*,*v*>). 需要证明

$$\boxed{dist[v]=short[v]}$$

若存在另一条 *s*-*v* 路径 *L* (绿色)，最后一次出 *S* 的顶点为 *x*, 经过 *V*–*S* 的第一个顶点 *y*，再由 *y* 经过一段在 *V*– *S* 中的路径到达 *v* .



归纳步骤证明（续）



在 k+1步算法选择顶点 v，而不是 y，

$$dist\ [v] \le dist\ [y]$$

令 y 到 v 的路径长度为 $d(y,v)$

$$dist\ [y]+d(y,v) \le L$$

于是 $dist\ [v] \le L$，即 $dist[v]= short\ [v]$

时间复杂度

- 时间复杂度：$O(nm)$

  算法进行n-1步

  每步挑选1个具有最小$dist$函数值的结点进入S，需要 $O(m)$时间

- 选用基于堆实现的优先队列的数据结构，可以将算法时间复杂度降到$O(mlogn)$

贪心法小结

- 贪心法适用于组合优化问题.
- 求解过程是多步判断过程，最终的判断序列对应于问题的最优解.
- 判断依据某种“短视的”贪心选择性质，性质的好坏决定了算法的正确性. 贪心性质的选择往往依赖于直觉或者经验.

贪心法小结（续）

- 贪心法正确性证明方法：

  (1) 直接计算优化函数，贪心法的解恰好取得最优值

  (2) 数学归纳法（对算法步数或者问题规模归纳）

  (3) 交换论证

- 证明贪心策略不对：举反例

贪心法小结（续）

- 对于某些不能保证对所有的实例都得到最优解的贪心算法（近似算法），可做参数化分析或者误差分析.
- 贪心法的优势：算法简单，时间和空间复杂性低

贪心法小结（续）

- 几个著名的贪心算法

  最小生成树的Prim算法

  最小生成树的Kruskal算法

  单源最短路的Dijkstra算法