第5讲 贪心算法 (2/2)

罗国杰

gluo@pku.edu.cn

2025年春季学期

算法设计与分析『实验班』

PKU-04833060

贪心法（Greedy Approach）

- 贪心法的设计思想
- 贪心法的正确性证明
- 对贪心法得不到最优解情况的处理
- 贪心法的典型应用
  - 最优前缀码
  - 最小生成树
  - 单源最短路径
- 拟阵相关的贪心法

"Greed, for lack of a better word, is good.
Greed is right. Greed works."

-- Gordon Gekko
(cast: Michael Douglas)
in Wall Street (1987)

得不到最优解的处理方法

- 讨论对于哪些输入贪心法能得到最优解：输入应该满足的条件
  - 例：找零钱问题

- 讨论贪心法的解最坏情况下与最优解的误差（见近似算法）

找零钱问题

问题描述：

设有 n 种硬币，

重量分别为：$w_1, w_2, \ldots, w_n$,

面值分别为：$v_1=1, v_2, \ldots, v_n$.

付的总钱数是：y

问：如何付钱使得所付硬币的总重最轻?

$$\min\{w_1x_1 + w_2x_2 + \ldots + w_nx_n\}$$

$$v_1x_1 + v_2x_2 + \ldots + v_nx_n = y$$

$$x_i \in \mathrm{N}, \quad i = 1,2,\ldots,n$$

找零钱问题：动态规划算法

属于整数规划问题，动态规划算法可以得到最优解

设 $F_k(y)$ 表示用前 k 种硬币，总钱数为 y 的最小重量

递推方程

$$F_{k+1}(y) = \min_{0 \le x_{k+1} \le \left\lfloor \frac{y}{v_{k+1}} \right\rfloor} \{F_k(y - v_{k+1}x_{k+1}) + w_{k+1}x_{k+1}\}$$

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

找零钱问题：贪心算法

假设

$$\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \ldots \geq \frac{w_n}{v_n} \text{且仍有} v_1 = 1$$

使用前 k 种硬币，总钱数为 y

贪心法的总重为 $G_n(y)$，则有如下递推方程

（从后往前选硬币）

$$G_{k+1}(y) = w_{k+1}\left\lfloor \frac{y}{v_{k+1}} \right\rfloor + G_k(y \bmod v_{k+1})$$

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

找零钱问题：n=1, 2 时得到最优解

$n = 1$，只有一种硬币，$F_1(y) = G_1(y)$

$n = 2$，使用单位价值重量小的硬币越多越轻，$F_2(y) = G_2(y)$

记第2种硬币使用x_2个时，总价值为y的硬币总重量

为$H(x_2)=F_1(y-v_2x_2)+w_2x_2$

$$\begin{aligned}
&H(x_2+\delta)-H(x_2)\\
&=[F_1(y-v_2(x_2+\delta))+w_2(x_2+\delta)]\\
&\quad-[F_1(y-v_2x_2)+w_2x_2]\\
&=[w_1(y-v_2x_2-v_2\delta)+w_2x_2+w_2\delta]\\
&\quad-[w_1(y-v_2x_2)+w_2x_2]\\
&=-w_1v_2\delta+w_2\delta=\delta(-w_1v_2+w_2)\le 0
\end{aligned}$$

找零钱问题：n>2时得到最优解的判定条件

定理　假定对所有非负整数y，有$G_k(y)=F_k(y)$，且$v_{k+1}>v_k$，则以下命题等价.

(1) $G_{k+1}(y) \le G_k(y)$

(2) $G_{k+1}(y) = F_{k+1}(y)$

(3) $G_{k+1}(pv_k) = F_{k+1}(pv_k)$

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

其中 $(p-1)v_k < v_{k+1} \le pv_k$，$p \in Z^+$，

$\delta = pv_k - v_{k+1} \quad (0 \le \delta < v_k)$

用条件(4)需 $O(k)$ 时间验证 $G_{k+1}(y)=F_{k+1}(y)$?

对 n 种硬币作出验证, 可在 $O(n^2)$ 时间内完成

找零钱问题：实例

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

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

验证　$G_3(y) = F_3(y)$

$v_3=pv_2-\delta \Rightarrow 14=3\times5-1$ $(p=3, \delta=1)$

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

$pw_2=3\times1=3$

$w_3+G_2(\delta)\le p\, w_2$

找零钱问题：实例

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

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

$v_4= pv_3-\delta \Rightarrow 18=2\times 14-10$ ($p=2$, $\delta=10$)

$w_4+G_3(\delta)=1+G_3(10)=1+2=3$

$pw_3=2\times 1=2$

$w_4+G_3(\delta) > pw_3$, $G_4(y)$不是最优解.

$G_4(pv_3) > F_4(pv_3)$. 即

$G_4(28)=\lfloor 28/18\rfloor+\lfloor 10/5\rfloor = 1+2 = 3$

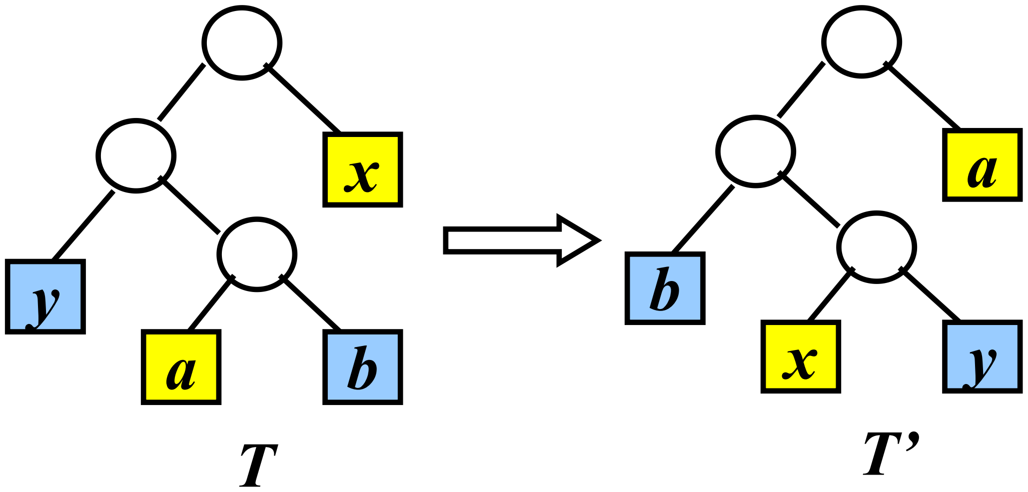
$F_4(28)=28/14 = 2.$

贪心法的典型应用

- 最优前缀码：Huffman树与Huffman编码
- 最小生成树：Prim算法与Kruskal算法
- 单源最短路径：Dijkstra算法

Huffman算法正确性证明：引理

- 引理1 设C是字符集，∀c∈C，f(c)为频率。x, y∈C 且 f(x) 和 f(y) 频率最小，那么存在最优二元前缀码使得 x, y 的码字等长，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，∀ x, y∈T，x 和 y 是树叶兄弟，z 是 x 和 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})∪{z} 的二叉树，那么 B(T)=B(T′)+f(x)+f(y)。

Huffman算法证明：归纳法

定理 Huffman 算法对任意规模为 n ($n\geq2$) 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 n=2，字符集 $C=\{x_1, x_2\}$，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

$$C' = (C - \{x_1,x_2\})\cup\{z\}, \quad f(z) = f(x_1)+f(x_2)$$

根据归纳假设，Huffman算法得到一棵关于字符集 C ′、频率 f(z) 和 $f(x_i)$ ($i=3,4,\ldots,k+1$) 的最优前缀码的二叉树 T′.

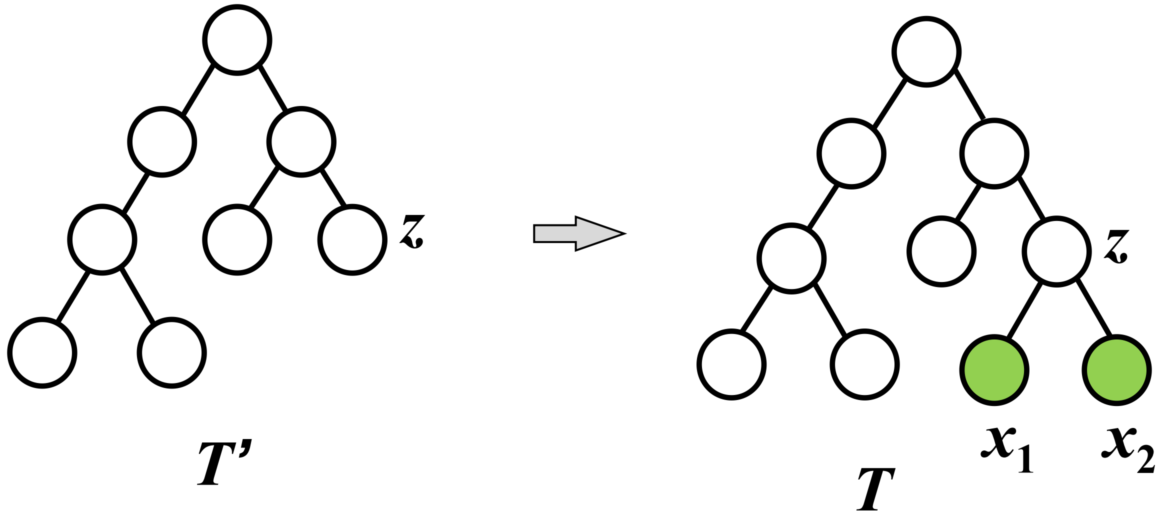
Huffman算法证明：归纳法（续）

把 x_1 和 x_2 作为 z 的儿子附加到 T’ 上，得到树 T，那么 T 是关于字符集 C=(C’ - {z})∪{x_1,x_2} 的最优前缀码的二叉树。

如若不然，存在更优的树T*。根据引理1，其最深层树叶是 x1, x2，且B(T*)<B(T). 去掉T*中的x1和x2，根据引理2，所得二叉树T*’满足

$$B(T^{*\prime}) = B(T^*) - (f(x1)+f(x2)) < B(T) - (f(x1)+f(x2)) = B(T')$$

与T’是一棵关于C’的最优前缀码的二叉树矛盾.



应用：最小生成树

无向连通带权图G=(V,E,W)，$w(e)\in W$是边e的权. G的一棵生成树是包含了G的所有顶点的树, 树中各边的权之和称为树的权，具有最小权的生成树称为G的最小生成树.

命题 设G是 n阶连通图，那么

(1) T是G 的生成树当且仅当 T 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 (回路).

问题：给定连通带权图G，求G的一棵最小生成树.

算法：Prim算法和Kruskal算法

Prim算法

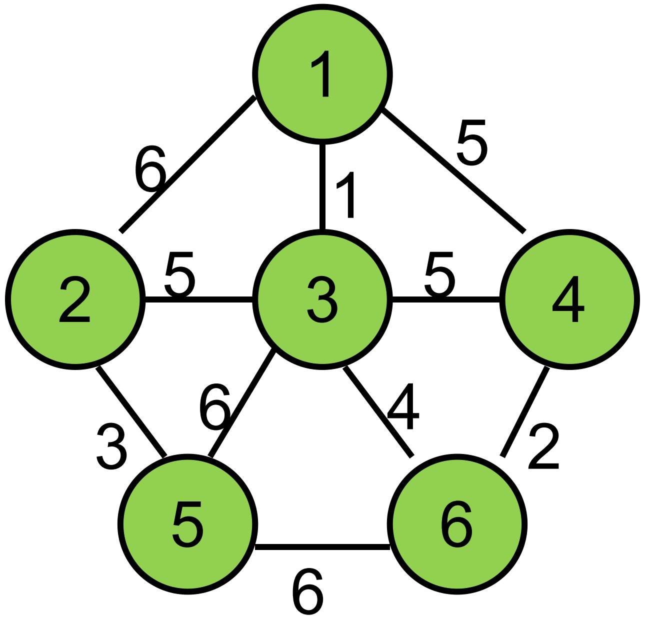
算法 Prim(G,E,W)

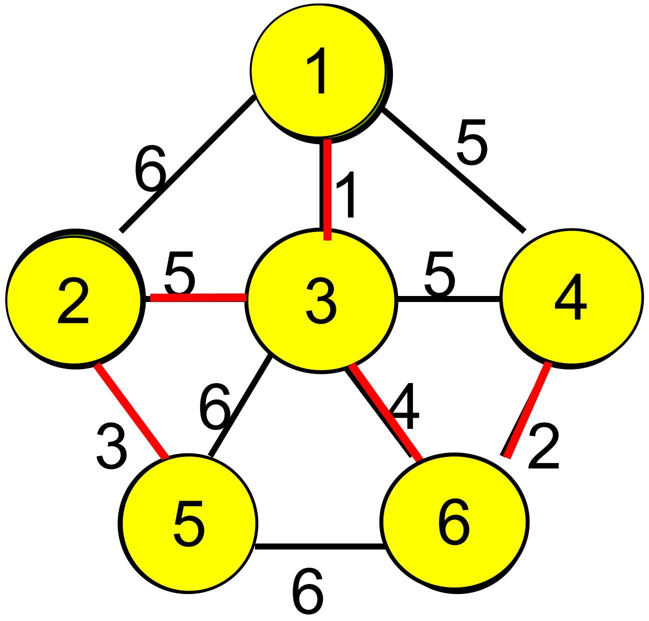
1．$S \leftarrow \{1\}$

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． $S \leftarrow S \cup \{j\}$





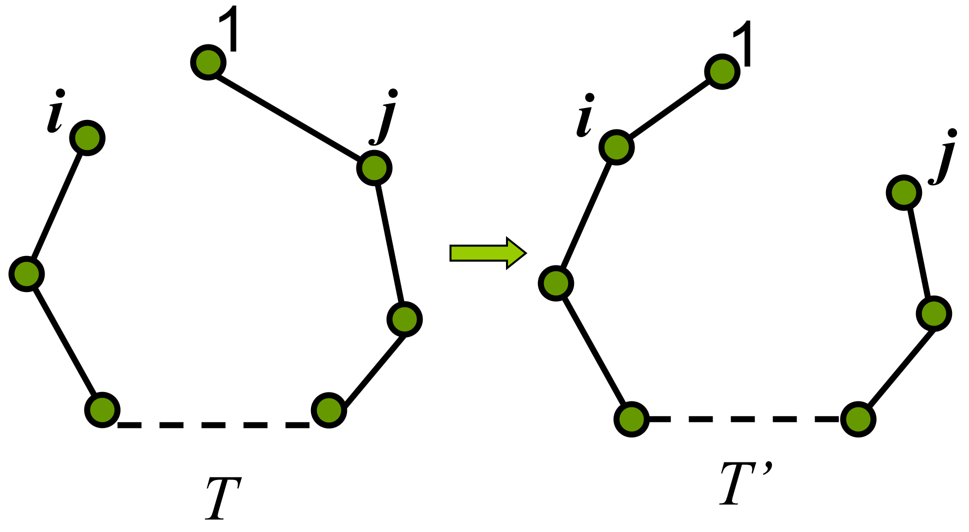
正确性证明

对步数归纳

定理：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.

设T 为一棵最小生成树，假设T 不包含(1,i), 则T∪{(1,i)}含有一条回路，回路中关联1的另一条边为(1,j)，令 T'=(T-{(1,j)})∪{(1,i)}，则T'也是生成树，且W(T')≤W(T).



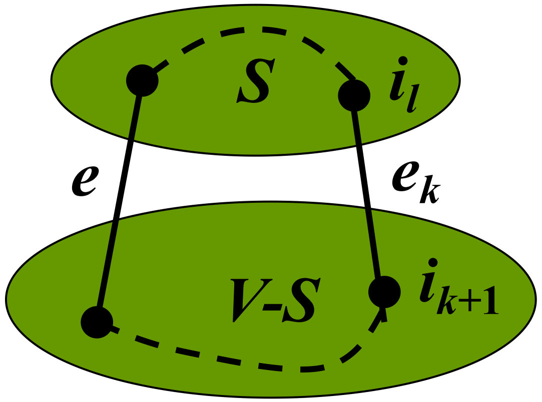
正确性证明(续)

归纳步骤：

假设算法进行了$k-1$步，生成树的边为$e_1,e_2,\ldots,e_{k-1}$，这些边的k个端点构成集合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步选择了顶点i_{k+1}, 则i_{k+1}到S中顶点的边权最小, 设这条边为$e_k=(i_{k+1},i_l)$. 假设T不含有e_k，则将e_k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

$$T^*=(T-\{e\})\cup\{e_k\},$$



则T^*是G的一棵生成树，包含$e_1,e_2,\ldots,e_k$, $W(T^*)\le W(T)$.

算法时间：$T(n)=O(n^2)$

Kruskal算法

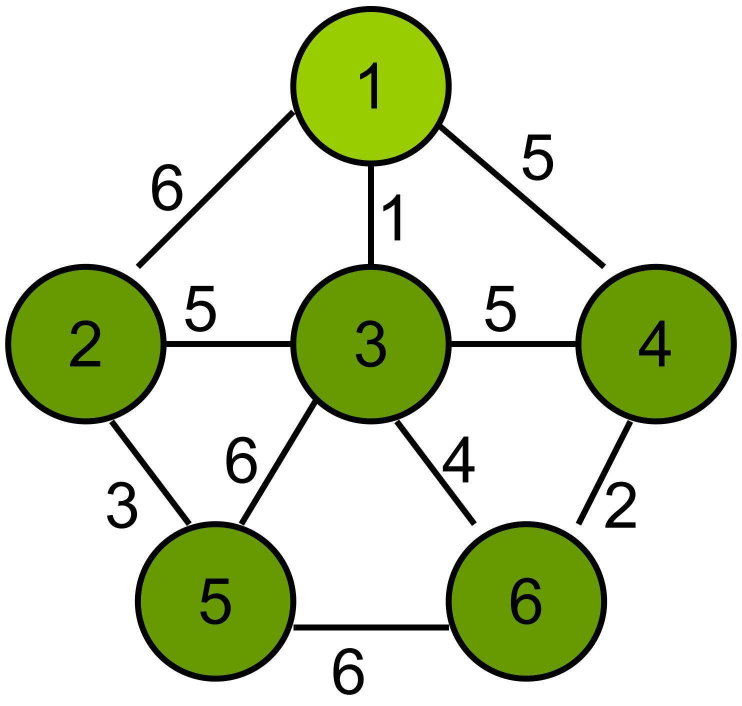
算法4.6 Kruskal

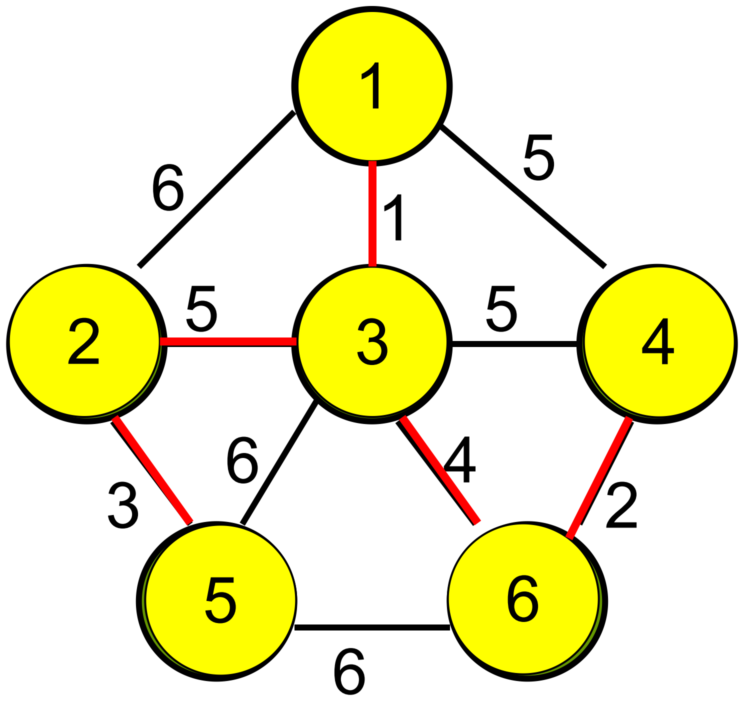
输入：连通图G　　　// 顶点数n，边数m

输出：G的最小生成树

1. 按权从小到大排序G中的边，使得$E=\{e_1, e_2, \ldots, e_m\}$
2. $T \leftarrow \varnothing$
3. repeat
4. 　　$e \leftarrow E$中的最短边
5. 　　if e的两端点不在同一个连通分支
6. 　　then $T \leftarrow T \cup \{e\}$
7. 　　$E \leftarrow E - \{e\}$
8. until T包含了$n-1$条边

实例





Kruskal算法正确性证明

命题：对于任意 $n>1$, 算法对 n 阶图得到一棵最小生成树.

证明 $n=2$, 只有一条边，命题显然为真.
假设对于n个顶点的图算法正确，考虑 $n+1$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'. 根据归纳假设，由算法存在G'的最小生成树T'.令$T=T'\cup\{e\}$, 则T 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^*中短接 e 得到G'的生成树 $T^*-\{e\}$, 且

$$W(T^*-\{e\})=W(T^*)-w(e)<W(T)-w(e)=W(T'),$$

与T'的最优性矛盾.

算法的实现与时间复杂度

数据结构：

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) 两个连通分支合并，则将较小分支结点的FIND值更新为较大分支的标记

时间复杂度：

(1) 每个结点至多更新log$n$次，建立和更新FIND数组的总时间为$O(n\log n)$

(2) 算法时间为

$$\underbrace{O(m\log m)}_{\text{边排序}} + \underbrace{O(n\log n)}_{\text{FIND数组}} + \underbrace{O(m)}_{\text{其他}} = O(m\log n)$$

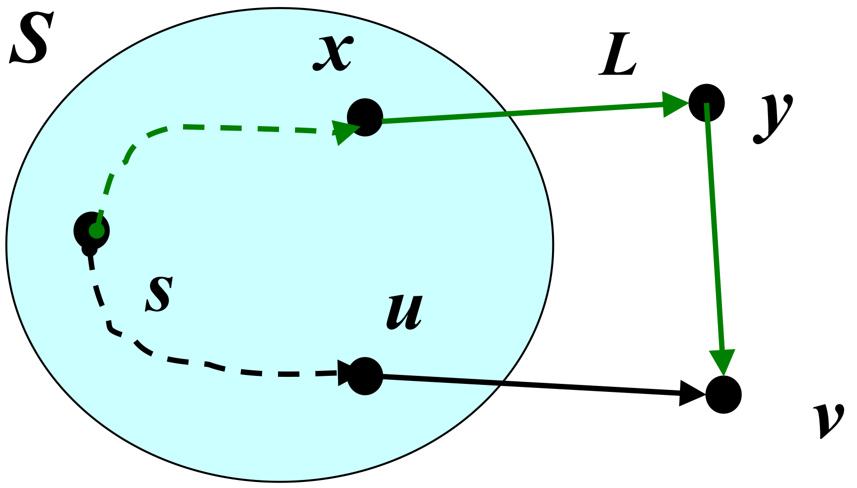
应用：Dijkstra算法（正确性证明）

定理 当算法进行到第 k 步时，对于S 中每个结点 i，

dist[i] = shortest[i]

归纳基础 k=1, S={s}, dist[s]=shortest[s]=0, 命题为真.

归纳步骤 假设命题对于k 为真。考虑 k+1步，选择顶点v (边{u,v})。假若存在另一条 s-v 路径 L（绿色），最后一次出S 的顶点为 x，在这次从S 中出来后经过V–S 的第一个顶点为 y；路径 L 比 s-u-v 路径短，则第 k+1 步肯定不会选 v（至少 y 更优先），矛盾。



Dijkstra算法：使用哪种优先队列?

- 总体性能取决于优先队列性能
  - n = |V|， m = |E|
  - n次 Insert、n次 Delete-Min、≤m次 Decrease-Key
  - 稠密图用数组实现：$O(n^2)$ 条边
  - 稀疏图适合二叉堆：O(n) 条边

优先队列	Insert	Delete-Min	Decrease-Key	总体
数组	O(1)	O(n)	O(1)	$O(n^2)$
二叉堆	O(log n)	O(log n)	O(log n)	O(m log n)
d路堆	$O(d \log_d n)$	$O(d \log_d n)$	$O(\log_d n)$	$O(m \log_{m/n} n)$
斐波那契堆	O(1)	O(log n)	平摊 O(1)	O(m + n log n)
整数优先队列	O(1)	O(log log n)	O(1)	O(m + n log log n)

Lecture slides by Wayne, “Greedy Algorithms II,” for Kleinberg and Tardo’s “Algorithm Design.”

贪心法小结

1. 适用于组合优化问题， 求解过程是多步判断. 判断的依据是局部最优策略，使目标值达到最大(或最小)，与前面的子问题计算结果无关.
2. 局部最优策略的选择是算法正确性的关键.
3. 正确性证明方法：数学归纳法、交换论证. 使用数学归纳法主要通过对算法步数或者问题规模进行归纳. 如果要证明贪心策略是错误的，只需举出反例.
4. 自顶向下求解，通过选择将问题归约为小的子问题.
5. 如果贪心法得不到最优解，可以对问题的输入进行分析或者估计算法的近似比.
6. 如果对原始数据排序之后，贪心法往往是一轮处理，时间复杂度和空间复杂度低.

拟阵理论

拟阵（matroid）

- 关于一类贪心算法的一种理论
- 用于判断一个问题能否应用贪心策略求解
- 可涵盖很多贪心法求解的问题
- 但不是所有的贪心算法都符合拟阵理论

Chapter 16 in CLRS, Introduction to Algorithms (3rd ed.), 2009.
removed (rarely taught) in CLRS, Introduction to Algorithms (4th ed.), 2022.

拟阵的例子

矩阵的所有行组成集合 S

集合 ℓ 的每个元素都是 S 的一个非空子集合，包含若干线性无关（独立）的行

$M = (S, \ell)$ 是拟阵，由于它满足

~~1) S 是一个有穷集合~~

2) 遗传性：ℓ 的每个元素是矩阵的若干行，满足如果 $B \in \ell$ 且 $A \subseteq B$，那么 $A \in \ell$。

// 线性无关行的子集仍是线性无关

3) 交换性：如果 $A \in \ell$，$B \in \ell$，且 $|A|<|B|$，则存在某个元素 $x \in B-A$ 使得 $A \cup \{x\} \in \ell$。

拟阵的定义

称一个有序对 $M = (S, \ell)$ 为**拟阵**，如果它满足以下条件

1) **有穷性**：S 是一个有穷集合

2) **遗传性**：ℓ 为 S 的若干**子集**组成的非空集合（$\ell \subseteq 2^S$ 且 $\ell \neq \emptyset$），满足如果 $B \in \ell$ 且 $A \subseteq B$，那么$A \in \ell$。可称 ℓ 的元素是**独立子集**。显然空集 $\emptyset \in \ell$

3) **交换性**：如果 $A \in \ell$，$B \in \ell$，且 $|A| < |B|$，则存在某个元素 $x \in B-A$ 使得 $A \cup \{x\} \in \ell$。

"拟阵"这个词由 Hassler Whiteney 最早开始使用的，他曾研究**矩阵拟阵**。其中S的元素是一个给定矩阵的各个行，如果某些行线性无关，则它们是独立的，以所有独立行集合构成ℓ，可以证明$M=(S, \ell)$是拟阵。

定义在无向图上的一个拟阵

对一个无向图$G=(V, E)$，定义有序对$M_G=(S_G, \ell_G)$如下：

- 集合S_G定义为E，即图G的边集。
- 如果A是E的子集，则$A\in\ell_G$当且仅当A中无回路。即一组边A是独立的，当且仅当子图$G_A=(V, A)$是一个森林。

定理16.5 对于无向图$G=(V, E)$，$M_G=(S_G, \ell_G)$是拟阵。

证明： 显然，$S_G=E$是有穷集合；

- 并且，ℓ_G是遗传的，因为森林的子集还是森林。
- 现在，只需要证明M_G满足交换性。设$G_A=(V, A)$和$G_B=(V, B)$都是森林，且$|A|<|B|$。也就是说，B比A有更多的边。

定义在无向图上的一个拟阵

- 一个有k条边的森林恰好有$|V|-k$棵树。因为每增加一条边能且只能把森林中的两棵树合成一棵树。
- 于是森林G_A中有$|V|-|A|$棵树，森林G_B中有$|V|-|B|$棵树。
- 可见森林G_B中的树比森林G_A中少，因此森林G_B中存在一棵树T，T的顶点属于森林G_A中的两棵不同的树。而树T显然是连通图，因此T中存在一条边(u,v)，使得结点u,v分别属于森林G_A中的两棵不同的树。
- 把边(u,v)加入到森林G_A，不会产生环，即$A+\{(u,v)\}\in\ell_G$
- 即M_G满足交换性。
- 由此根据拟阵的定义，$M_G=(S_G, \ell_G)$是拟阵。 ■

极大独立子集

- **扩张**：对于拟阵$M = (S, \ell)$，集合$A \in \ell$，元素$x \notin A$，如果能保证把x加入到A中并保持A的独立性，即$A \cup \{x\} \in \ell$，则称x是A的一个扩张。
  - 例如，对于图的拟阵M_G，如果A是一个独立的边集，则边e是A的一个扩张，当且仅当e不在A中，且将e加入到A中后不产生环。
- **极大独立子集**：如果A是拟阵M的一个独立子集，且它没有任何扩张，则称A是极大独立子集。也就是说，A不被M中更大的独立子集所包含。

极大独立子集定理的证明

定理16.6 拟阵中所有极大的独立子集的大小都相同。

证明： 反证法。

- 设A是拟阵M的一个极大独立子集，假设存在M的另一个更大的独立子集B，则根据交换性，A可以扩张到一个更大的独立子集$A \cup \{x\} \in \ell$， $x \in B-A$。这与A是M的极大独立子集矛盾。证毕。 ■

例如对于图G的拟阵M_G， M_G的每个极大独立子集都是一棵包含$|V|-1$条边且恰好连接了G中的所有顶点的自由的树。其实这就是一棵G的生成树。

加权拟阵

- 拟阵$M = (S, \ell)$是加权的，如果有加权函数w，

  $w(x) \to R^+$， $x \in S$

  加权函数w对与子集$A \subseteq S$有和式

  $w(A) = \sum_{x \in A} w(x)$

- 对于某个$A \in \ell$，如果$w(A)$最大，称A为拟阵M的**最优子集**

  最优子集一定是极大独立子集（因为$w(x)>0$，$x \in S$）

- 最小生成树到拟阵最优子集问题的映射

- 设$w(e)$，$e \in E$是图$G=(V, E)$上边的长度函数，定义拟阵M_G上的权函数$w'(e)=w_0-w(e)$，其中$w_0>\max\{w(e), e \in E\}$。则最小生成树$T=(V,A)$对应于最优子集$A$。

加权拟阵上的贪心算法

- **GREEDY(M, w)**

  1 $A \leftarrow \emptyset$

  2 对$S[M]$ 按权函数w降序排列

  3 **for each** $x \in S[M]$ ，按$w(x)$ 单调降序依次取出

  4 　　**do if** $A \cup \{x\} \in \ell[M]$

  5 　　　　**then** $A \leftarrow A \cup \{x\}$

  6 **return** A

- 其中$S[M]$ 和$\ell[M]$表示M的组成，w表示权函数
- 算法的执行时间分析，步骤2排序为$\Theta(n\lg n)$，步骤3~5对每个x做一次，设检验$A \cup \{x\}$独立性需要$f(n)$时间，则GREEDY算法的执行时间为$\Theta(n\lg n+nf(n))$

拟阵具有贪心选择性质

引理16.7 具有加权函数w的加权拟阵$M = (S, \ell)$，设S按权值的单调递减顺序排序。设x是S中第一个使$\{x\}$独立的元素。如果x存在，则存在一个包含x的最优独立子集A

证明：如果这样的x不存在，则唯一的独立集合为空集，证明结束。否则，设B为任意非空最优独立子集，
并假设$x \notin B$（否则，让$A=B$，证明结束）。

- 可以证明，B中不存在权值大于$w(x)$的元素，
  对任意$y \in B$，有$w(x) \geq w(y)$。
- 根据交换性，我们从$\{x\}$开始，通过B一步一步构造最大独立子集A，最终$|A|=|B|$且$A=B-\{y\}+\{x\}$，其中$y \in B$，于是$w(A) = w(B)-w(y)+w(x) \geq w(B)$，故$A$是包含$x$的最优独立子集。 ■

贪心选择的过程不会遗漏

引理16.8 对拟阵$M = (S, \ell)$，对于任意的$x \in S$，如果x是S的独立子集A的一个扩张，则x也是$\emptyset$的一个扩张。

证明：如果x是独立子集A的一个扩张，则$A \cup \{x\}$是独立的，根据遗传性，$\{x\}$也是独立的，所以x也是$\emptyset$的一个扩张 ■

推论16.9 对拟阵$M = (S, \ell)$，如果某个$x \in S$不是$\emptyset$的一个扩张，则x也不会是S的任意独立子集A的一个扩张。

- 这个结论告诉我们，在拟阵中的贪心选择过程，在一开始选择时丢弃的元素，在后面的选择过程中也不再会用到。

拟阵具有最优子结构的性质

引理16.10 对加权拟阵$M = (S, \ell)$，设$x \in S$为在GREEDY算法中第一个选择的元素，则找一个包含x的最优子集问题可以归约为找出加权拟阵$M' = (S', \ell')$的最优子集问题，这里$S'=\{y \in S:\{x,y\} \in \ell\}$，$\ell'=\{B \subseteq S-\{x\}: B \cup \{x\} \in \ell\}$，其中$M'$的权函数为（受限于$S'$）$M$的权函数（称$M'$为$M$由$x$引起的收缩）

证明：如果A是包含x的M的独立子集，那么$A'=A-\{x\}$就是M'的一个独立子集。反之，由M'的独立子集A'可得M的一个独立子集$A=A' \cup \{x\}$。而两种情形中都有$w(A)=w(A')+w(x)$，因此由M中包含x的一个最大权值解可以得M'中的一个最大权值解，反之亦然。■

拟阵上贪心算法的正确性

定理16.7 GREEDY(M,w)返回$M=(S,\ell)$一个最优子集。

证明：分析算法的整个过程：

- 根据推论16.9，一开始被略去的那些不是∅的扩张的元素可以不考虑；
- 一旦选择了第一个元素x，由引理16.7可知，将x加入A是正确的，因为存在包含x的最优子集。
- 最后，由引理16.10，隐含着余下的问题就是一个在M的由x引起的收缩M'中寻找一个最优子集的问题（B在M'中独立等价于$B\cup\{x\}$在M中独立，其中$B\in\ell'$），剩余步骤可找出M'中一个最优子集。
- 整个步骤的结果就是M的最优子集。■

拟阵证明练习

1. 证明：(S, ℓ_k)是拟阵，其中S为任意有穷集合，ℓ_k为S的所有阶最多为k的子集构成的集合，$k \le |S|$。

2. 证明：关于矩阵T的(S, ℓ)是拟阵，其中S为T的所有的列构成的集合，且$A \in \ell$当且仅当A中的各列是线性无关的。

练习题解

问题1的证明：

- S本身就是有穷集合；
- 对于任意$B\in\ell_k$和任意$A\subseteq B$，都有$|A|\leq|B|\leq k$，所以$A\in\ell_k$，满足遗传性；
- 又对于任意的$A,B\in\ell_k$且$|A|<|B|$，任取$x\in B-A$，则$|A\cup\{x\}| = |A|+1 \leq |B| \leq k$，所以$A\cup\{x\}\in\ell_k$，满足交换性。
- 故(S, ℓ_k)是拟阵。■

练习题解

问题2的证明：

- S本身就是有穷集合；
- 对于任意$A\in\ell$，A中各列线性无关，则A的任意子集中的各列也线性无关，所以满足遗传性；
- 对于阶更大的独立子集B，假设B中的任意一列都和A中的列线性相关，因为$|A|<|B|$，可以得出B中的各列线性相关，这与B是独立子集矛盾；
- 故必定存在$x\in B\text{-}A$，x与A中的各列和在一起仍线性无关，即$A\cup\{x\}\in\ell$，故满足交换性。
- 于是(S, ℓ)是拟阵。 ■

一个任务调度问题

——拟阵的应用

一个任务调度问题

在单处理器上对若干单位时间任务进行最优调度，其中每个任务都有一个截止时间和超时惩罚。

- 包含n个单位时间任务的集合$S=\{a_1, a_2, ..., a_n\}$
- n个整数值的期限$d_1, d_2, ..., d_n$，每个d_i满足$1 \le d_i \le n$且任务a_i要求在d_i前完成
- n个非负的权（或惩罚）$w_1, w_2, ..., w_n$，如果任务a_i没有在时间d_i前结束，则导致惩罚w_i，而如果任务在期限之前完成，则没有惩罚。

问题：

找出S的一个调度，使之最优化因延期调度而导致的总惩罚。

调度的规范形式：早任务优先

- 对于一个给定的调度，如果一个任务在规定期限之后完成，这个任务在调度中迟了；否则这个任务在调度中是早的。
- 任何一个调度总可以安排成**早任务优先**的形式，即早任务总是安排在迟任务之前。因为：对处于迟任务a_j后的早任务a_i，交换a_i和a_j不影响a_i是早的和a_j是迟的。
- 早任务优先调度的规范形式：
  早任务先于迟任务，且按期限的递增序对早任务进行调度。
- 对某调度，首先将其安排成按早任务优先的形式，然后只要有两个分别完成于时间k和$k+1$的早任务a_i和a_j使得$d_j < d_i$，则交换a_i和a_j。
- 容易看出，交换后a_j显然仍是早的，因为我们提前了a_j。
- 而对于a_i，因为$k+1 \le d_j$（因为交换前a_j是早的），于是$k+1 < d_i$，所以交换后a_i也是早的。

归约为早任务集合问题

- 根据早任务优先调度的规范形式，寻找最优化调度问题可归约为寻找最优调度中早任务构成的集合A的问题。
- 一旦A确定，可按期限单调递增的顺序列出A中的所有任务，然后按任意顺序输出迟任务$(S-A)$，就可产生出最优调度的一个规范形式。
- 如果存在关于A中任务的一个调度，使得A中的任务都不迟，称一个任务集合A是独立的。
- 某一个调度中的早任务集合就构成了一个独立的任务集。
- 设ℓ是所有独立的任务集构成的集合。
- 如何判断任务集A是否是**独立的**?
- 设$N_t(A)$表示A中期限为t或更早的任务的个数（t早任务个数），$t = 0, 1, ..., n$，其中$N_0(A)=0$。

关于任务集A是否独立的引理

引理16.12 对于任意的任务集合A，下列命题等价：

1) 集合A是独立的；
2) 对于$t = 0, 1, \ldots, n$，有$N_t(A) \le t$；
3) 如果对A的任务按期限的单调递增的顺序进行调度，则没有一个任务是迟的。

证明：假设存在t使得$N_t(A)>t$，则不存在A的无迟任务的调度，因为有多于t个任务要在时间t之前完成，与A独立矛盾，故**1**到**2**成立。

2到**3**是显然的，因为按期限单调递增进行调度不可能出现迟任务。

最后，**3**到**1**可根据独立任务集的定义直接得出。■

- 最小化迟任务的惩罚之和等价于最大化早任务惩罚之和
- 只要找出一个拟阵，即可按加权拟阵的贪心算法解此问题

单位时间调度问题的拟阵

定理16.13 如果S是一个带期限的单位时间任务的集合，且ℓ为所有独立的任务集构成的集合，则(S, ℓ)是拟阵。

证明：一个独立的任务集的每个子集肯定是独立的。

为证明交换性，设B和A为独立任务集，且$|B|>|A|$，设k为使$N_t(B) \leq N_t(A)$成立的最大的t（这样的t一定存在，因为$N_0(B)=N_0(A)=0$）。

又有$N_n(B)=|B|>|A|=N_n(A)$，故有$k<n$且对$k+1\leq j\leq n$中的所有j，$N_j(B)>N_j(A)$，所以B中比A中包含了更多的期限为$k+1$的任务。

设a_i为$B-A$中具有期限$k+1$的一个任务，并令$A'=A\cup\{a_i\}$。

根据定理16.12中的2)，当$0\leq t\leq k$时，$N_t(A')\leq N_t(A)\leq t$；

当$k+1\leq t\leq n$时，$N_t(A')\leq N_t(B)\leq t$；所以A'也是独立的，即$A' \in \ell$。■

用GREEDY算法求解

- 可以先用**GREEDY**算法找出具有最大权值的独立任务集A
- 然后再以A的任务作为早任务得到一个最优调度
- 此算法的运行时间为$O(n^2)$，因为算法中$O(n)$次的独立性检查，每次开销为$O(n)$时间。
- 一个例子：

a_i	1	2	3	4	5	6	7
d_i	4	2	4	4	1	4	7
w_i	70	60	50	40	30	20	10

其中总的罚值为$w_5 + w_6 = 50$.

拟阵理论与贪心法的思考

- 并非所有的贪心问题都满足拟阵理论
  - 如活动安排问题、霍夫曼编码问题等
- 一个问题往往并不直接对应与拟阵
  - 需要对问题进行一定的变换
- 拟阵的GREEDY算法至少得到最初步的贪心算法
  - 针对问题特性的优化会提高贪心算法的性能