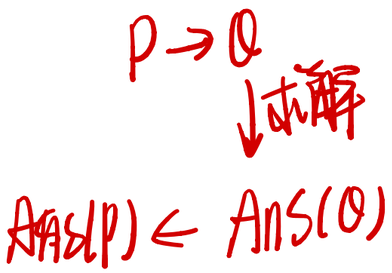
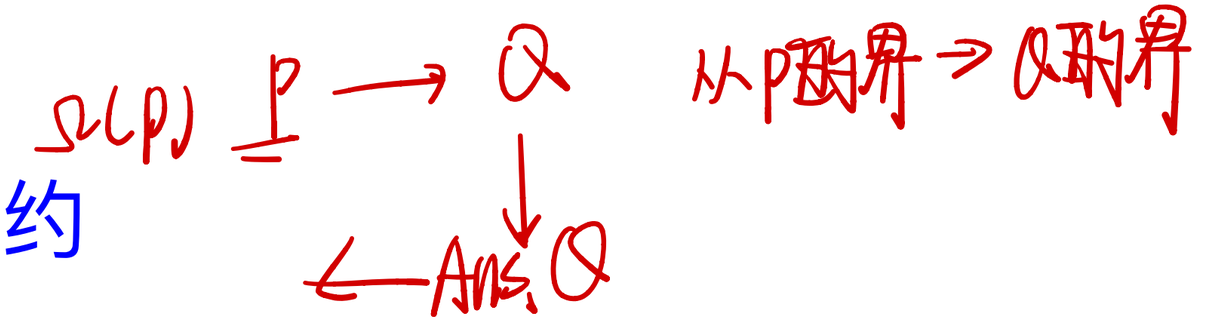
第十章

归约





假设有一个较复杂的问题P，而它看起来与一个已知的问题Q很相似，可以试着在两个问题间找到一个归约（reduction，或者transformation）。

归约是使用解决其它问题的“黑盒”来解决另一个问题。取决于问题解决的先后，归约可以达到两个目标：1) 如果已知Q的算法，那么就可以把使用了Q的黑盒的P的解决方法转化成一个P的算法；2) 如果P是一个已知的难题，或者特别地，如果知道P的下限，那么同样的下限也可能适用于Q。前一个归约是用于获取P的信息；而后者则是用于获取Q的相关信息。

归约让我们理解两个问题间的关系，它是一种可用于寻找解决某问题或其变形的技术

简单字符串匹配问题

A=12345

B=34512

CSM问题：设两个长度为n的字符串$A=a_0a_1...a_{n-1}$及$B=b_0b_1...b_{n-1}$，判断B是否A的循环平移。

方法1：修改6.7节中介绍的Knuth-Morris-Pratt算法来解决CSM KMP算法.

方法2：寻找特定的文本T及特定的模式P使得在T中寻找P等同于判定B是否A的一个循环平移

简单字符串匹配问题

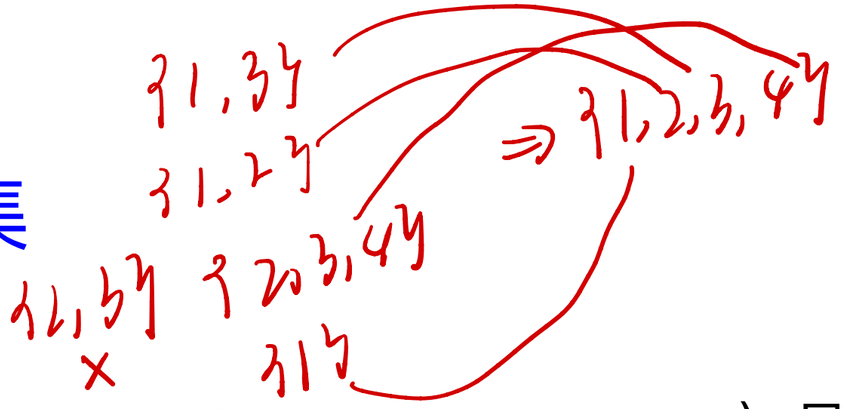
CSM问题：设两个长度为n的字符串$A=a_0a_1...a_{n-1}$及$B=b_0b_1...b_{n-1}$，判断B是否A的循环平移。

方法1：修改6.7节中介绍的Knuth-Morris-Pratt算法来解决CSM

方法2：寻找特定的文本T及特定的模式P使得在T中寻找P等同于判定B是否A的一个循环平移

答案：把T定义为AA（也就是A再连接它本身）。显然，B是A的一个循环平移当且仅当B是AA的一个子串。既然已经知道如何在线性时间内解决SM问题，那么同样有CSM问题的线性时间算法。

不同代表集



设$S_1,S_2\ldots,S_k$是一组集合。不同代表集（System of Distinct Representative，SDR）是一个集合$R=\{r_1,r_2,...r_k\}$，其中 $r_i\in S_i$。由于R是集合，r_i就必须是相异的，即R刚好包含每个集合的一个代表元素。给定一组集合未必能找到一个SDR

问题：给定一由有穷个有穷集合组成的组，找出该组的（任意一个）SDR，或判断SDR是否存在。

设card(S)是S的元素个数。

Hall定理：设$S_1,S_2,\ldots,S_k$是一组集合，其存在SDR当且仅当下面条件成立：$card\left(S_{i_1}\cup S_{i_2}\cup\cdots\cup S_{i_m}\right)\geq m$ 对{1,2,...,k}的任一个子集$\{i_1,i_2,...,i_m\}$，即任意m个集合的并必须至少包含m个不同的元素，对所有的$1\leq m\leq k$。

实用价值不大

不同代表集

Hall定理提出了很简单的条件，但是很不幸，必须检查所有可能的2^k个子集。

把它看成偶图匹配问题：设G=(V,U,E)是一个偶图，使得对每个集合S_i均在V中存在一个顶点v_i，并且对所有可能的元素（即对集合的并中的每个元素）在U中都有一个顶点u_j。每个元素与包含它的所有集合相连，则SDR就是G中规模为k的匹配，可以应用7.10节中讨论的算法。

序列比较

序列比较问题：$A=a_1a_2\ldots a_n$及$B=b_1b_2\ldots b_m$是两个字符串，要逐字符地编辑A直到它变为B。允许插入、删除和替换三种编辑方式，每个方式涉及一个字符，且已知这些方式的代价，目标是使编辑的代价最小。

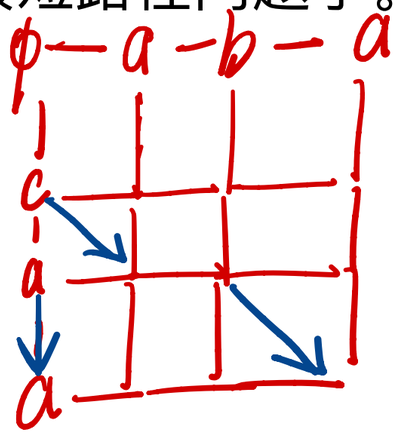
6.8的方法：构造一个$n\times m$的表，其中第ij项对应一个局部编辑，包含了把A的前i个字符编辑成B的前j个字符的代价，最终目标是完成表的右下角项（第nm项）。

序列比较

另一种方法：把表看成一个有向图。表中每一项对应图的一个顶点，这样每个顶点对应一个局部编辑。当v对应的局部编辑能够经过一个以上编辑步骤到w对应的局部编辑时，则存在一条边(v,w)。水平边对应插入，垂直边对应删除，而斜边对应替换。除了对应相同字符的斜边（即无须替换）的代价是0，其它每条边的代价是1。

现在这个问题就变成了一个常规的单源最短路径问题了，每条边都关联一个代价（对应于编辑的代价），而我们正在寻找从顶点[0,0]到顶点[n,m]的最短路径。这样就已经把字符串编辑问题归约成单起点最短路径问题了。

A = caa　　B = aba



序列比较

一般来说寻找最短路径不比直接解决序列比较问题简单，但是归约还是很有用的。

设想有下面各种变形的序列比较问题。编辑的代价不一定是以一个字符进行计算的，在字符串正中插入一块字符的代价与在其它地方逐个插入相同数目的字符的代价可能是不一样的，删除也是如此。换句话说，要对一块字符的操作赋值，无论其大小，而不是对单步操作赋值。又如，我们想为插入一个k字符的块赋予代价，比方说I+ck，其中I是“起步”（start-up）价，c是后来的每个字符的代价。还有许多其它度量方法，但都可以用最短路径形式对其建模，而远比更改原来的问题来得容易，可以在任何需要的地方添加边并对它们赋值，而无须修改问题。

在无向图中寻找三角形

问题：设G=(V,E)是一n个顶点m条边的无向图。设计一个算法判断G是否含有两两连接的三个顶点。

最直截了当的方法是查看所有三个顶点的子集，每个子集可以在常数时间内检查，因而时间复杂性是$O(n^3)$。

设A是G的邻接矩阵。因为G是无向的，所以A对称。记A^2是矩阵A的平方，考虑A^2中各项与图G之间的关系。由矩阵乘法的定义，$A^2[i,j]>0$当且仅当存在k使得A[i,k]和A[k,j]均为1，从图的角度看，当存在一个顶点k，$k\neq i$且$k\neq j$，并且i和j都与k连接的时候$A^2[i,j]>0$ （假设图不包含自回路，这样对所有的i都有A[i,i]=0）。这些意味着存在一个关于i和j的三角形当且仅当i与j相连且$A^2[i,j]>0$。这样，在G中存在一个三角形当且仅当存在i和j使得$A[i,j]=1$且$A^2[i,j]>0$。这样把在图中寻找三角形的问题归约为布尔矩阵相乘问题。

数学建模问题

线性规划

线性规划问题可形式化阐述如下：在满足不等式约束，等式约束和非负约束的条件下使目标函数最大化。

$$\min c(\bar{x}) = \sum_{i=1}^{n} c_i x_i$$

$$s.t. \quad \bar{a}_i \cdot \bar{x} \le b_i$$

$$\bar{e}_j \cdot \bar{x} = d_j$$

$$x_k \ge 0$$

目标函数.

线性的约束条件.

标准形式，松弛变量

$\bar{a}x \le \bar{b} \Rightarrow ax + c = b$

$c \ge 0$ — 松弛变量

网络流量问题归约到线性规划

G=(V,E)是一个有两个特殊顶点的有向图，一个是入度为0的s(源点)，一个是出度为0的t(汇点)。E中的每条边e赋予一个正的权值c(e)，称为e的容量，用来度量经过这条边的最大流量，可为不存在的边赋予容量值0 。这样的一个图称为网络。流量是一个作用在网络的边上的函数f，满足以下两个条件：

1. $0 \le f(e) \le c(e)$：经过一条边的流量不能超过这条边的容量。
2. 对于所有的$v \in V-\{s,t\}$，$\sum_u f(u,v)=\sum_w f(v,w)$：进入一个顶点的流量之和等于离开这个顶点的流量之和(源点和汇点除外)。

网络流量问题归约到线性规划

设变量x_1，x_2，…，x_n表示所有边的流出量（这里n是边的数目）。目标函数是

$$c(\bar{x}) = \sum_{i \in S} x_i$$

其中S是从源出发的边的集合。对应容量约束的约束不等式是：

$$x_i \le c_i \qquad 1 \le i \le n$$

其中c_i是边i的容量。对应守恒约束的约束不等式是：

$$\sum_{\text{离开}v\text{的}x_i} x_i - \sum_{\text{进入}v\text{的}x_j} x_j = 0 \qquad v \in V - \{s,t\}$$

最后，对所有的变量都应用非负约束

静态路由问题归约到线性规划

设G=(V,E)是无向图，表示一个通讯网络。假设网络中每个节点有一个有限缓存，在一个单位时间内只能接收B_i条消息（设所有消息的长度均相同）。再假设任一链路可以传输的消息数目没有限制，每个节点均有无限的消息供应，问题是在单位时间内每条边应该传送多少条消息才能使在网络上的总消息数量最多。用图论表述，问题就是对边赋予权重，连接节点V_i的边的权重≤B_i，而使所有权重的和最大化。

这个图论问题可以很容易表述成一个线性规划问题，对每条边e_i关联一个变量x_i，表示经过e_i的消息数目。目标函数是，

$$c(\bar{x}) = \sum_i x_i$$

约束如下：

$$\sum_{\text{连接}v_j\text{的}e_i} x_i \le B_i \qquad x_i \ge 0$$

慈善家问题归约到线性规划

设有n个组织要向k个计算机系捐助，每个组织i在一年内都有总捐助限度s_i，对部门j的捐助同样也有限度a_{ij}（对某些部门a_{ij}可能是0）。一般来说s_i比$\sum_{j=1}^{k} a_{ij}$要小，因而每个组织都要做一些抉择。此外，设每部门也有收受总金额的限制t_j（虽然这个约束不太现实，不过这是很有趣的）。目标是设计一个算法使总捐助金额最大（不管公平与否）

变量x_{ij}表示组织i对部门j的捐助数目。目标函数是

$$c(\bar{x}) = \sum_{ij} x_{ij}$$

约束如下且所有的变量都是非负的。

$$x_{ij} \le a_{ij} \qquad \sum_{j=1}^{k} x_{ij} \le s_i \qquad \sum_{i=1}^{n} x_{ij} \le t_j$$

分配问题归约到线性规划

对慈善家问题稍作更改：强调每个组织只能向一个部门捐助，而每个部门也仅接收一个组织的捐助。

对每条边(i,j)赋予一个变量x_{ij}，当选择该边时为1，反之为0。目标函数变为：

$$c(\bar{x}) = \sum_{ij} a_{ij} x_{ij}$$

约束如下且所有的变量都非负：

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad\qquad \sum_{i=1}^{n} x_{ij} = 1$$

问题的界

在前面给出的最优算法的定义中，我们提到了问题的下界，如果任何一个求解问题P的算法是$\Omega(f(n))$的，则该问题的下界是$\Omega(f(n))$。实际上对于我们所有遇到的算法，都已经能够找到算法所需计算量的上界，但是找一个特定问题的下界却要困难得多。

问题的界

有一类下界被称为平凡下界，可以在不借助任何计算模型或者进行复杂计算的情况下就能够得到。

对于一个没有排序的序列，为找出其中的最小数，必须检查序列中所有的数，因此在无序序列中进行查找的任何算法必须花费$\Omega(n)$时间。

两个n×n矩阵相乘的任何算法都必须计算并输出恰好n^2个值，因此两个n×n矩阵相乘的任何算法的时间复杂性为$\Omega(n^2)$。

要注意的是，平凡下界虽然不用借助任何计算模型或进行复杂的数学计算就能够推导出来，但是往往会过小而失去意义。例如TSP问题的平凡下界是$\Omega(n^2)$，因为对于n个城市的TSP问题，输入是n(n-1)/2个距离，但是这个平凡下界是没有意义的，因为TSP问题至今还没有找到一个多项式时间算法。

决策树模型

平均情况下基于比较的排序算法的下界

定义：设L是二叉树T的叶节点集合，则T的外路径总长为$EPL(T)=\sum_{x\in L} depth(x)$，其中depth(x)表示节点x的深度

一棵二叉树T被称为有最小外路径总长的二叉树，如果其EPL(T)是所有有同样叶节点个数的二叉树中最小的

在叶节点个数相同的情况下

一棵有最小EPL的二叉树必然是每个内节点恰好有两个子节点，并且所有叶节点必定出现在最底下两层

若具有最小EPL的二叉树T有x个叶节点，则$EPL(T)>x(\lceil \log x \rceil -1)$

定理：若各种输入情况等概率出现，则任何一个基于比较的排序算法平均需要至少logn!-1次比较，其中n为待排序的数字个数

平均比较次数 $C(n)=\frac{EPL(T)}{S} > T\log S\rceil -1$

$C(n) > \log n! -1$

$EPL(T) = \sum_{x \in L} depth(x)$

T → EPL最小，证明 $EPL(T) \geq x(\lceil \log x \rceil - 1)$

若T在h-1层内部节点m个 $0 < m < 2^{h-1}$

h-1层叶节点 $2^{h-1} - m$ h层有2m个叶节点 ⇒ 总叶结点 $S = 2^{h-1} + m$

↳ $2^{h-1} < S \leq 2^h$

↳ $h-1 < \log S \leq h$

↳ $h = \lceil \log S \rceil$

$EPL(T) > (h+1)(2^{h-1} + m) = S(h-1)$

↓

$EPL(T) > S(\lceil \log S \rceil - 1)$

下界的归约

基本原理：如果解决问题Q的算法无需增加太多运行时间就可改为解决问题P的算法，那么问题P的下界也能适用于问题Q。

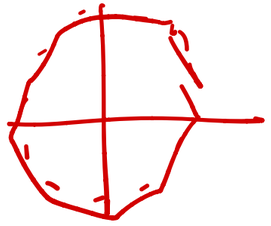
寻找简单多边形算法复杂度的下界

考虑用一个简单闭多边形连接平面上一系列点的问题，我们已经知道如何用排序来解该问题，在某些假设下，这个问题是不可能比排序来得更快。所以，不改进排序方法就无法改进简单闭多边形问题的求解速度（改进意味着速度的提高大于常数因子）。

定理10.1：给定一个在O(T)时间内的简单多边形问题的（黑盒）求解算法，就可以在O(T+n)时间内进行排序。

$T = O(n \log n)$

推论10.2：在决策树模型下，在平面上寻找连接一系列给定点的简单多边形，在最坏情况下需要Ω(nlogn)的比较。



要排序 $x_1 \cdots x_n$

↓线性变换

1) $y_1 \cdots y_n$ $-180° \le y_i \le 180°$ $O(n)$

2) 生成 z_i，位于以圆点为圆心的圆上，与x轴夹角 y_i $O(n)$

3) 简单多边形构造 $O(T)$

4) 相应顺序输出 $O(n)$

得到 $x_1 \cdots x_n$ 的顺序

总共：$O(T+n)$

关于矩阵的简单归约

问题：特殊的矩阵相乘是否会更简单？

定理10.3：如果有算法在O(T(n))时间内计算两个n×n的对称实矩阵相乘，满足T(2n)=O(T(n))，那么同样有算法在O(T(n)+n^2)时间内计算两个n×n的任意实矩阵相乘。

$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}^2 = \begin{pmatrix} AB & 0 \\ 0 & BA \end{pmatrix}$ 与10.3类似

定理10.4：如果有算法在O(T(n))时间内计算n×n实矩阵的平方，满足T(2n)=O(T(n))，那么同样有算法在O(T(n)+n^2)时间内计算两个n×n的任意实矩阵相乘。

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}_{2n\times 2n} \cdot \begin{pmatrix} 0 & B^T \\ B & 0 \end{pmatrix}_{2n\times 2n} = \begin{pmatrix} AB & 0 \\ 0 & A^TB^T \end{pmatrix}_{2n\times 2n}$$

A^T：$O(n^2)$

$T(2n)=O(T(n))$

AB：$O(n^2)$

则：$O(T(n)+n^2)=O(T(n))$

常见的错误

最普遍的问题是以错误的方式使用归约
归约过程中不应该加入明显低效的操作。

必须在$O(f(n))$内完成归约.

Huffman: $\Omega(n\log n)$

排序 → Huffman

$x_1,\dots,x_n$

↓

$y_1,\dots,y_n$ 正整数. → 用2^{y_i}作为频率

$(2^m > \sum_{k<m} 2^i)$

用归约:

① $P \to Q$. $P:\Omega(f(n))$. 变化过程在$O(f(n))$内完成. 那么$Q:\Omega(f(n))$

P的实例是任意的. 而Q是特殊的

排序 ∝ 简单多边形

(P) (Q)