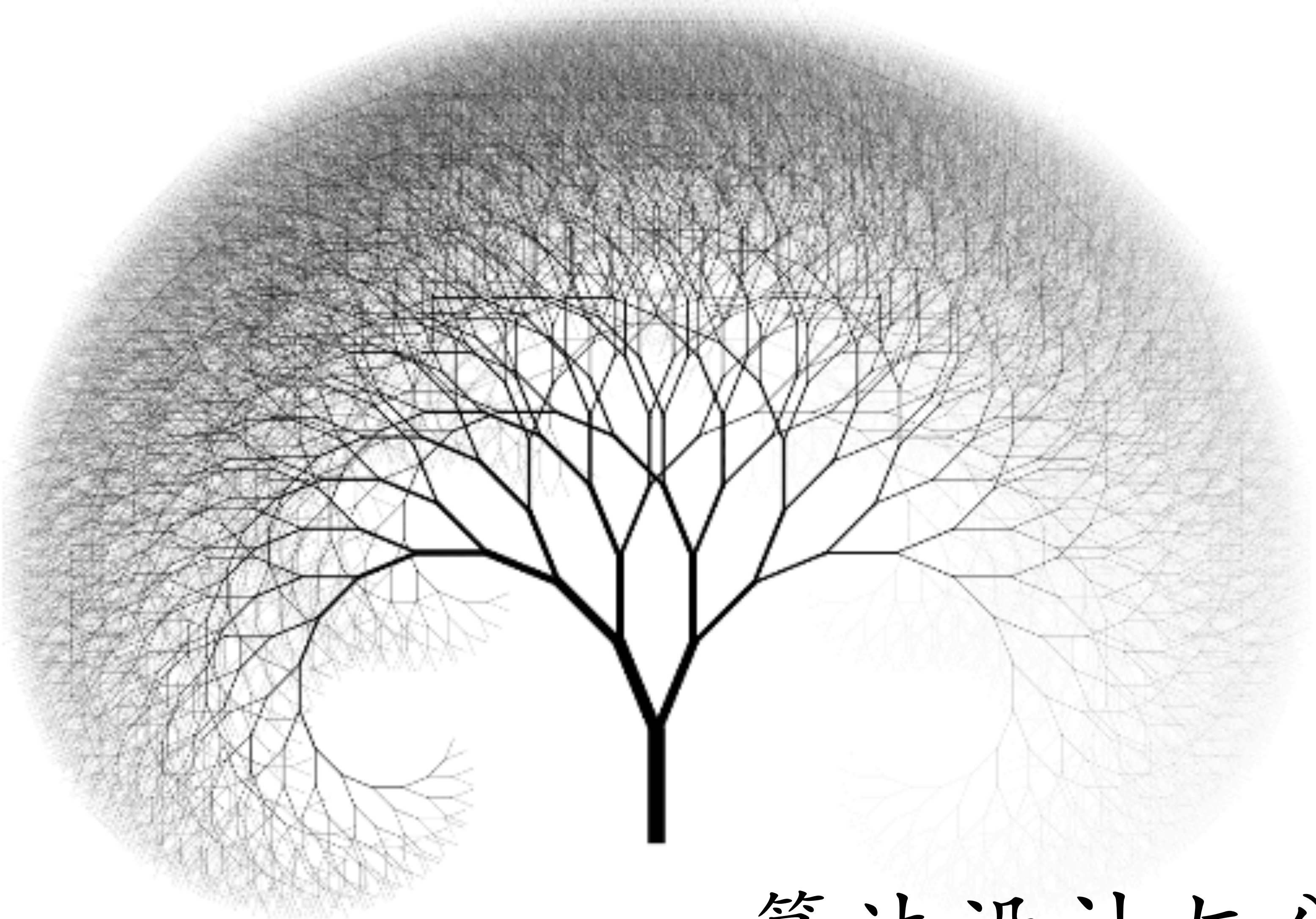


算法设计与分析

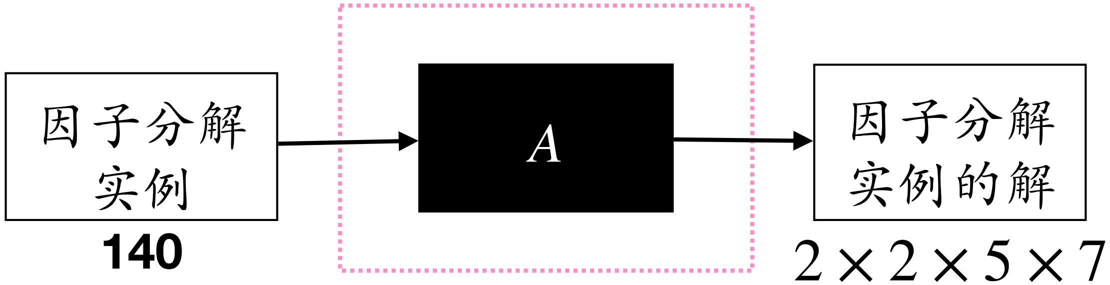
刘 安
苏州大学 计算机科学与技术学院
http://web.suda.edu.cn/anliu/

归约

Reduction

动机

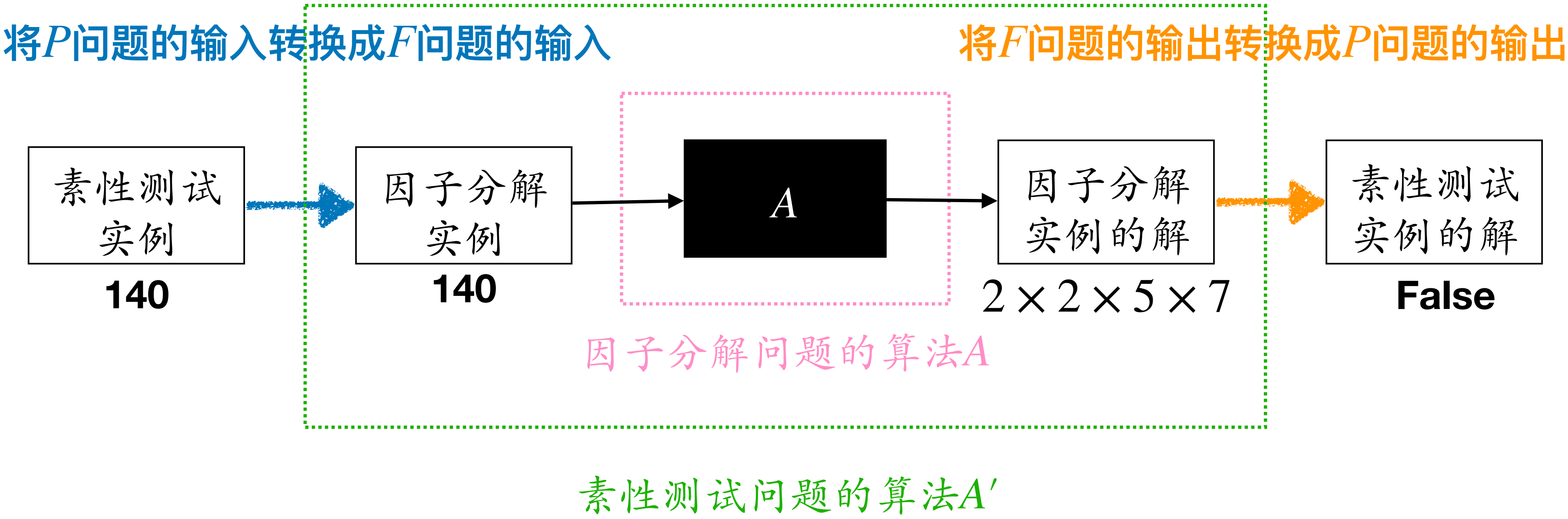
- 因子分解Factoring：给定数字N，将它表示为其素因子的乘积形式
  - $140 = 2 \times 2 \times 5 \times 7$
- 素性测试Primality：给定数字N，判定其是否为素数
  - 140不是素数
- 假设算法A可以求解因子分解问题，可以使用它求解素性测试问题吗？



因子分解问题的算法A

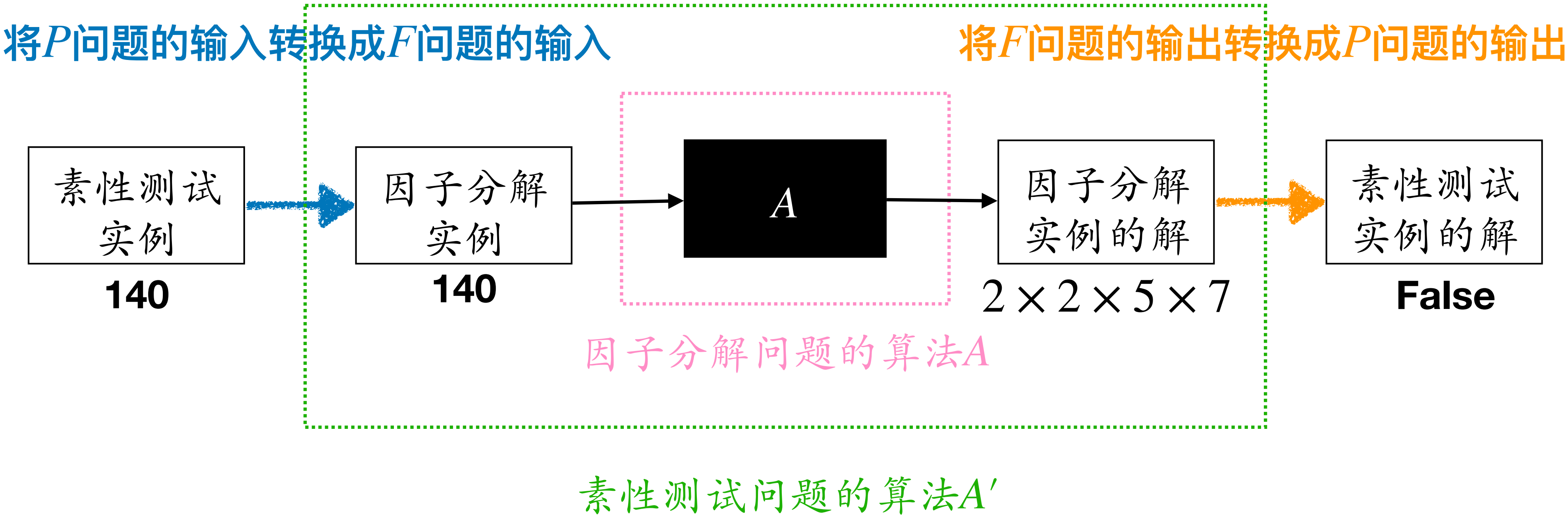
动机

- 因子分解Factoring：给定数字N，将它表示为其素因子的乘积形式
  - $140 = 2 \times 2 \times 5 \times 7$
- 素性测试Primality：给定数字N，判定其是否为素数
  - 140不是素数
- 假设算法A可以求解因子分解问题，可以使用它求解素性测试问题吗？



动机

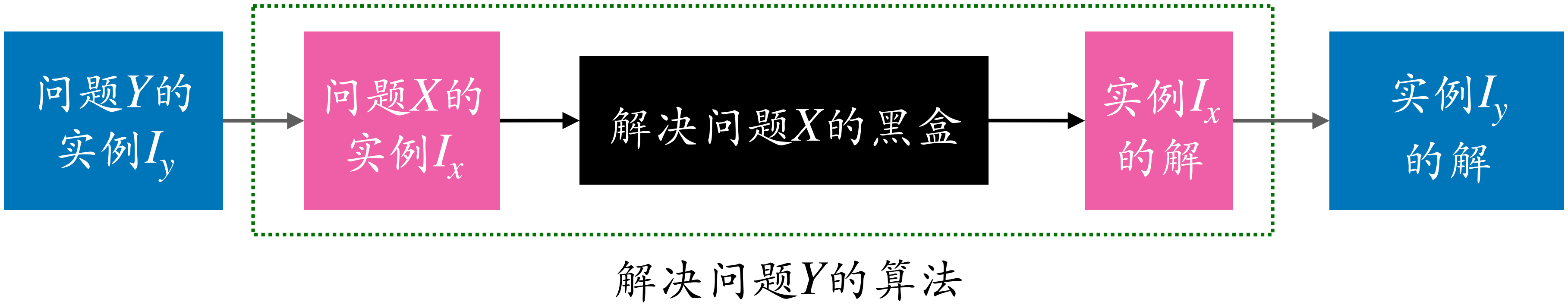
- A_i：将P问题的输入转换成F问题的输入的算法
- A_o：将F问题的输出转换成P问题的输出的算法
- $T(A') \approx T(A_i) + T(A) + T(A_o)$
- 利用因子分解求解素性测试不是一个好的办法
  - 大整数的因子分解是困难的！$\Rightarrow$ RSA加密算法安全性的基石
  - 素性测试有相当高效的算法



多项式时间归约

- 如果问题Y的任意实例可以通过
  - 多项式次数的标准计算步骤，加上
  - 对解决问题X的黑盒的多项式次数调用

来解决，那么称：问题Y可以在多项式时间归约为问题X，记为$Y \leq_p X$



多项式时间归约

- 如果问题Y的任意实例可以通过
  - 多项式次数的标准计算步骤，加上
  - 对解决问题X的黑盒的多项式次数调用

来解决，那么称：问题Y可以在多项式时间归约为问题X，记为$Y \leq_p X$

- 如果X可以在多项式时间内求解，那么
  - Y也可以在多项式时间内求解
  - 用途：设计算法、证明算法的下界
- 如果在多项式时间内无法求解Y，那么
  - X也不能在多项式时间内求解
  - 用途：确定问题之间的相对难度（传播难解性）

对偶问题

- 如果$Y \leq_p X$并且$X \leq_p Y$，那么记$Y \equiv_p X$，问题X和问题Y称为对偶问题
- 最大公约数GCD：给定整数m和n，求它们的最大公约数
- 最小公倍数LCM：给定整数m和n，求它们的最小公倍数
- $LCM \equiv_p GCD$

$$\because GCD(m,n) \times LCM(m,n) = m \times n$$

$$\therefore LCM(m,n) = \frac{m \times n}{GCD(m,n)} \Rightarrow LCM \leq_p GCD$$

$$\therefore GCD(m,n) = \frac{m \times n}{LCM(m,n)} \Rightarrow GCD \leq_p LCM$$

循环移位

Cyclic Shift

循环移位问题

- 令$A = a_1a_2\cdots a_n$和$B = b_1b_2\cdots b_n$是两个长度为n的字符串。确定B能否通过A循环移位得到？
  - 比如当$A = 1234$时，$B = 3412$可以通过循环移位得到，但$C = 1324$不行
- 朴素的归约
  - 将其归约为字符串是否相等的问题：$O(n)$时间内可求解（记为算法A）
  - 最坏情况下调用n次算法A，所以总的时间复杂度是$O(n^2)$
- 聪明的归约
  - KMP算法：找到串W在串S中第一次出现的位置，时间复杂度$O(m+k)$，其中m是串S的长度，k是串W的长度
  - 令串$S = AA = a_1a_2\cdots a_na_1a_2\cdots a_n$，串$W = B$
  - B可以通过A循环移位得到当且仅当串W在串S中出现
  - 时间复杂度$O(n)$

配对问题

Pairing

配对问题

- 输入：两个整数序列$X = (x_1, x_2, \cdots, x_n)$，$Y = (y_1, y_2, \cdots, y_n)$
- 输出：将X中第k小的元素与Y中第k小的元素组成一对，其中$1 \le k \le n$

23	42	17	93	88	12	57	90

48	59	11	89	12	91	64	34

配对$\leq_p$排序

配对问题的输入

23	42	17	93	88	12	57	90

48	59	11	89	12	91	64	34

将配对问题的输入转换成排序问题的输入 $O(1)$

23	42	17	93	88	12	57	90

48	59	11	89	12	91	64	34

排序算法

排序算法

$O(n \log n)$

12	17	23	42	57	88	90	93

11	12	34	48	59	64	89	91

将排序问题的输出转换成配对问题的输出 $O(n)$

配对的下界

- 输入：两个整数序列$X = (x_1, x_2, \cdots, x_n)$，$Y = (y_1, y_2, \cdots, y_n)$
- 输出：将X中第k小的元素与Y中第k小的元素组成一对，其中$1 \le k \le n$

- 配对$\le_p$排序 $\Rightarrow$ 求解配对问题的时间复杂度是$O(n \log n)$
- 配对问题是否存在更高效的算法？比如$O(n)$?

- 排序$\le_p$配对?

排序$\leq_p$配对

排序问题的输入

23	42	17	93	88	12	57	90

将排序问题的输入转换成配对问题的输入 $O(n)$

23	42	17	93	88	12	57	90

1	2	3	4	5	6	7	8

配对算法 $O(?)$

23, 3	42, 4	17, 2	93, 8	88, 6	12, 1	57, 5	90, 7

将配对问题的输出转换成排序问题的输出 桶排序：$O(n)$

配对的下界

- 输入：两个整数序列$X=(x_1,x_2,\cdots,x_n)$，$Y=(y_1,y_2,\cdots,y_n)$
- 输出：将X中第k小的元素与Y中第k小的元素组成一对，其中$1\le k\le n$

- 配对$\le_p$排序 $\Rightarrow$ 求解配对问题的时间复杂度是$O(n\log n)$
- 配对问题是否存在更高效的算法？比如$O(n)$?

- 排序$\le_p$配对
  - 输入和输出都可在$O(n)$时间内完成转换
  - 配对存在比$O(n\log n)$更快的算法 $\Rightarrow$ 排序也有比$O(n\log n)$更快的算法
  - 与基于比较的排序下界是$\Omega(n\log n)$矛盾
  - 所以配对的下界是$\Omega(n\log n)$

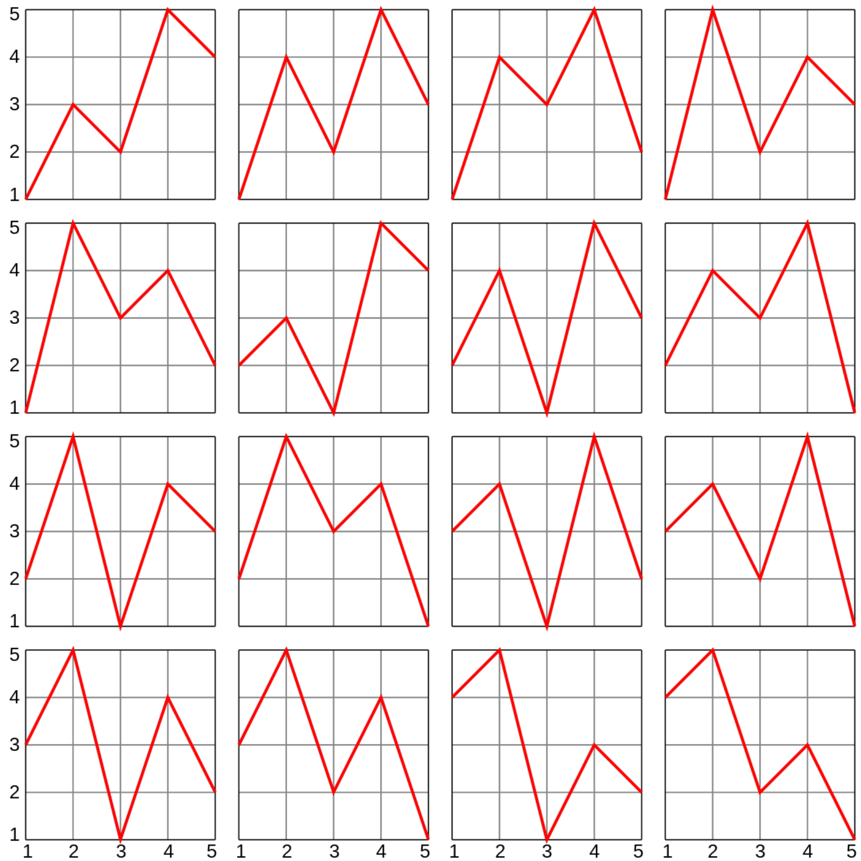
交替排列

Alternating Permutation

交替排列

- 集合$\{1,2,\cdots,n\}$的交替排列是这些数字的排列，其中每个元素交替大于或者小于前一个元素

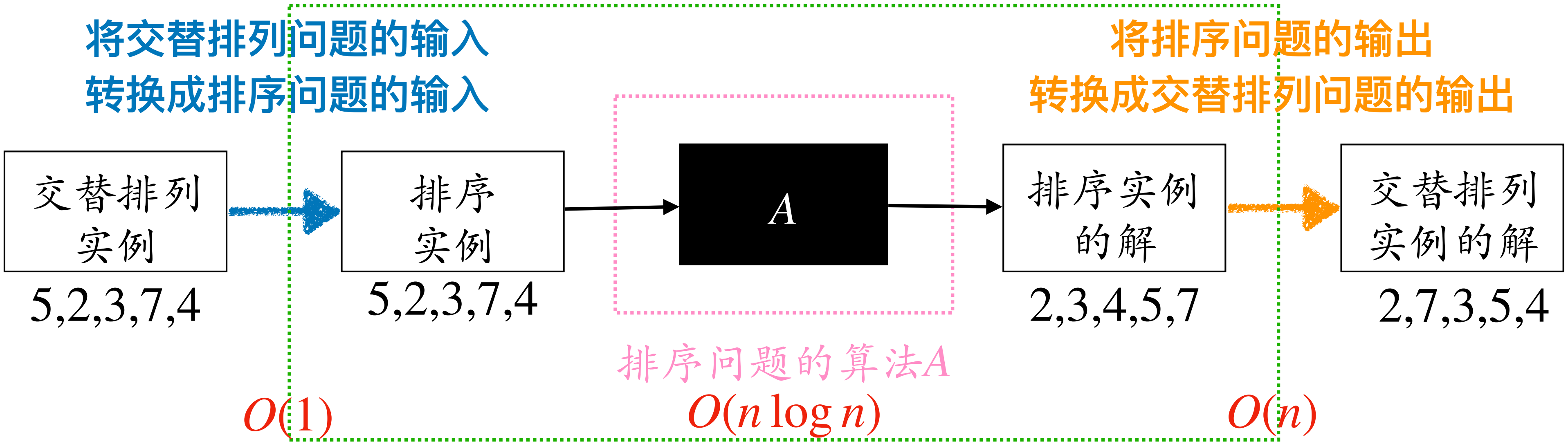
- 集合$\{1,2,3,4\}$有5个交替排列
  - 1,3,2,4 $\Leftarrow$ $1 < 3 > 2 < 4$
  - 1,4,2,3 $\Leftarrow$ $1 < 4 > 2 < 3$
  - 2,3,1,4 $\Leftarrow$ $2 < 3 > 1 < 4$
  - 2,4,1,3 $\Leftarrow$ $2 < 4 > 1 < 3$
  - 3,4,1,2 $\Leftarrow$ $3 < 4 > 1 < 2$



集合$\{1,2,3,4,5\}$的16个交替排列

构造一个交替排列

- 问题P_n：给定n个不同的整数，构造一个交替排列
- 归约为排序问题
- 输出转换：依次选取最小/最大值放入解中，$O(n)$
- 输入转换：无需转换，$O(1)$
- 算法A'的时间复杂度：$O(n\log n)$



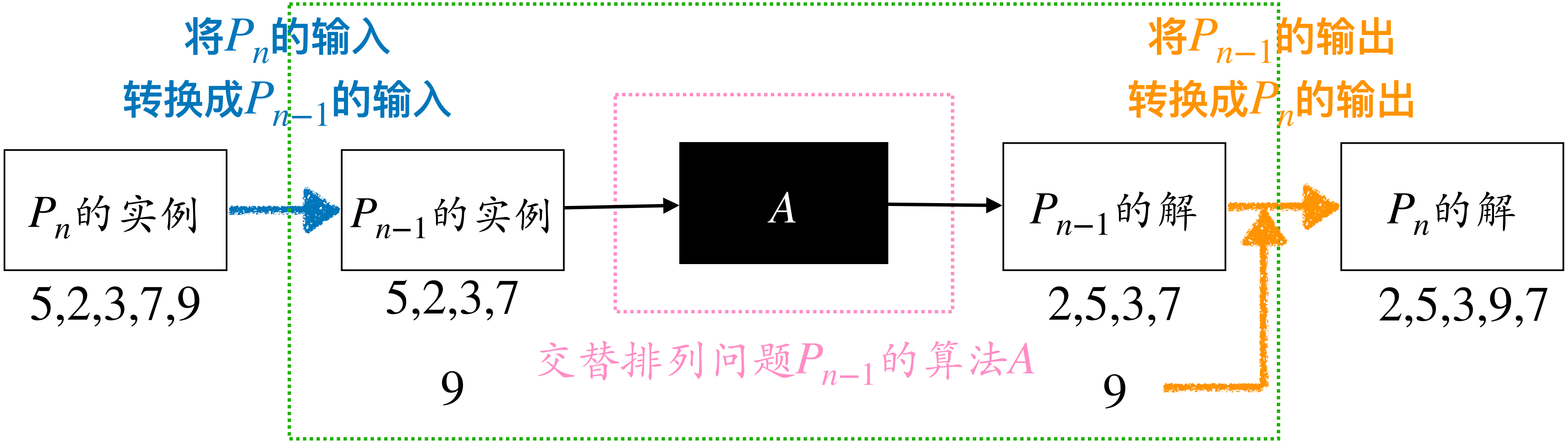
交替排列问题的算法A'

构造一个交替排列

- 问题P_n：给定n个不同的整数，构造一个交替排列

递归
- 归约为交替排列问题P_{n-1}：给定$n-1$个不同的整数，构造一个交替排列
- 输入转换：取前$n-1$个整数，$O(1)$
- 输出转换：比较P_{n-1}解的最后一个元素与第n个整数的大小关系，$O(1)$
- 算法A'的运行时间$T(A') = T(A) + O(1) \Rightarrow T(n) = T(n-1) + O(1)$

$$\Rightarrow T(n) = O(n)$$



交替排列问题P_n的算法A'

二元可满足性问题

The 2-SAT problem

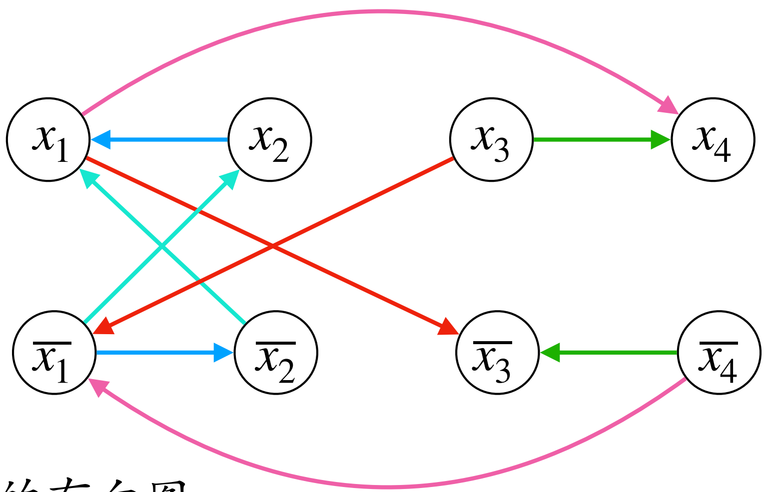
可满足性

- 项 (term)：布尔变量x_i或它的否定$\overline{x_i}$
- 子句 (clause)：不同项的析取 (disjunction)，比如$C_j = x_1 \vee \overline{x_2} \vee x_3$
- 合取范式 (CNF，conjunctive normal form)
  - 不同子句的合取 (conjunction)，比如$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$
- 可满足性问题 (SAT，satisfiability problem)
  - 给定变量集$X = \{x_1, \cdots, x_n\}$上的一个合取范式$\Phi = C_1 \wedge \cdots \wedge C_k$，是否存在满足的真值赋值？
- 二元可满足性问题 (2-SAT)：每个子句的长度为2，即恰好包含2个项
  - $\Phi = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$
    - 上述合取范式是可满足的：$x_1 = \mathrm{T}, x_2 = \mathrm{F}, x_3 = \mathrm{F}, x_4 = \mathrm{T}$
  - 能否构造一个不可满足的合取范式

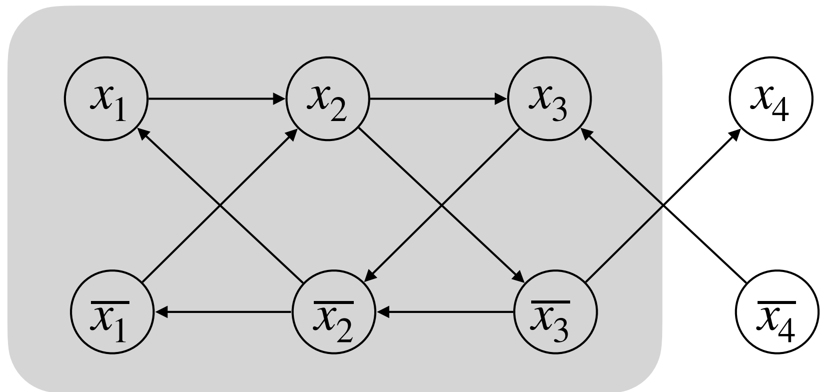
$$\left.\begin{aligned} & x_1 \vee x_2 \\ & \overline{x_1} \vee x_2 \end{aligned}\right\} \Rightarrow x_2 = 1 \qquad \begin{aligned} & \overline{x_2} \vee x_3 \ \Rightarrow x_3 = 1 \\ & \overline{x_2} \vee \overline{x_3} \ \Rightarrow \overline{x_3} = 1 \end{aligned}$$

图的构造

- 令合取范式的变量数为n，子句数为m
  - $\Phi = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$
  - $\Phi' = (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4)$
- 为其构造一个有向图G
  - 为每个变量x_i及其否定$\overline{x_i}$创建一个顶点 — $2n$个顶点
  - 为每个子句$x \vee y$创建两条边$(\overline{x}, y)$和$(\overline{y}, x)$ — $2m$条边
- 命题：Φ是可满足的当且仅当G中不存在同时包含顶点x和$\overline{x}$的强连通分量



Φ对应的有向图



Φ'对应的有向图

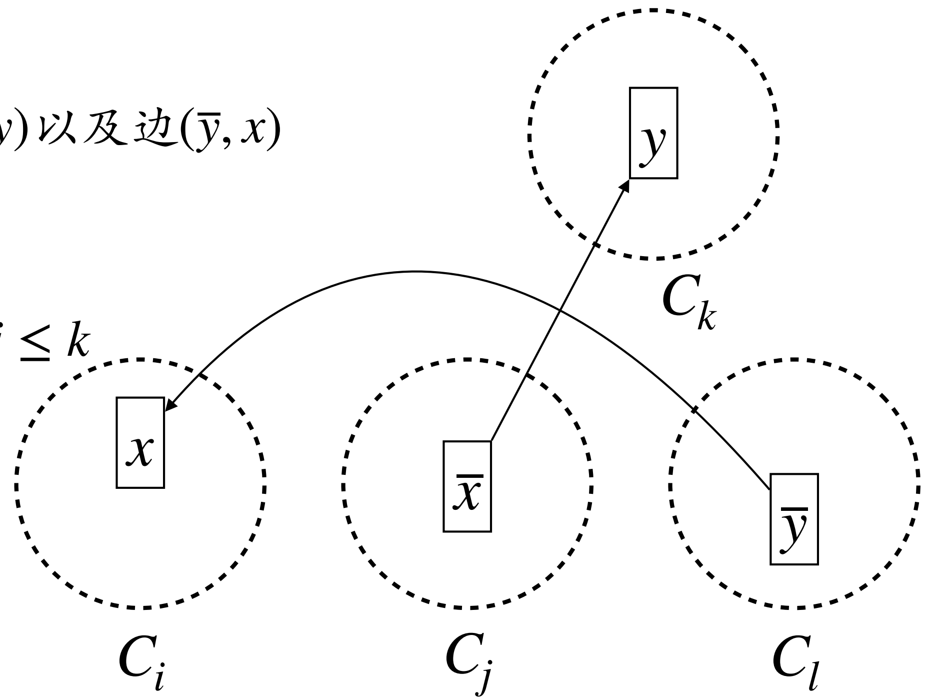
2-SAT与强连通分量

- 命题：Φ是可满足的当且仅当G中不存在同时包含顶点x和$\bar{x}$的强连通分量
- 证明$\Rightarrow$
  - 假设存在一个强连通分量同时包含顶点x和$\bar{x}$
  - 存在一条从x到$\bar{x}$的路径以及一条从$\bar{x}$到x的路径
  - 所以，$x \Rightarrow \bar{x}$且$\bar{x} \Rightarrow x$
  - 因为x与$\bar{x}$不能同时为真，也不能同时为假
    - 要使$x \Rightarrow \bar{x}$成立，x只能为假
    - 要使$\bar{x} \Rightarrow x$成立，$\bar{x}$只能为假，即x为真
  - 矛盾！

x	y	$x \Rightarrow y$
T	T	T
T	F	F
F	T	T
F	F	T

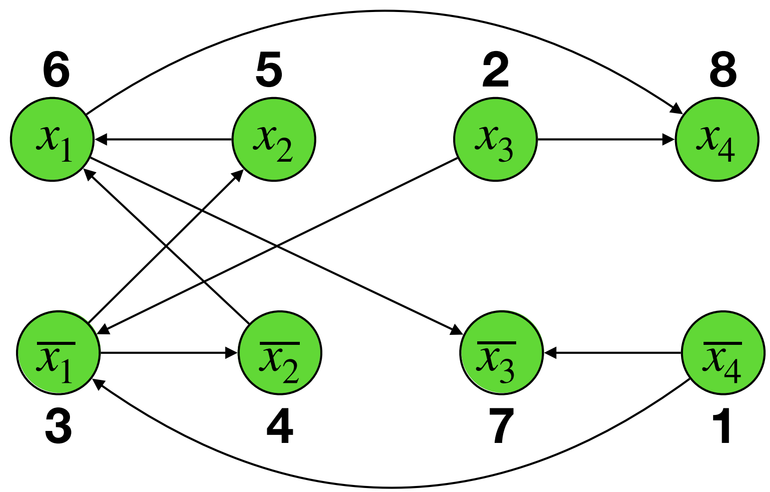
2-SAT与强连通分量

- 命题：Φ是可满足的当且仅当G中不存在同时包含顶点x和$\bar{x}$的强连通分量
- 证明$\Leftarrow$
  - 将G的所有强连通分量拓扑排序，记为$C_1, C_2, \cdots, C_m$
  - 假设x和$\bar{x}$分别在C_i和C_j中，如果$i < j$，令x取值F，如果$i > j$，令x取值T
  - 下面证明这种赋值方案下Φ是可满足的
    - 假设Φ不是可满足的，那么至少存在一个子句$x \vee y$，其中x, y均取值F
    - 下面证明x, y均取值F会导致矛盾
      - 根据构图规则，G中一定存在边$(\bar{x}, y)$以及边$(\bar{y}, x)$
      - x取值F $\Rightarrow$ x在C_i中，$\bar{x}$在C_j中，$i < j$
      - 边$(\bar{x}, y) \Rightarrow y$在$C_k$中，根据拓扑序，$j \leq k$
      - y取值F $\Rightarrow \bar{y}$在C_l中，$k < l$
      - 边$(\bar{y}, x)$的存在与拓扑序矛盾！



运行实例

- $\Phi = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$
  - $scc(x_1) = 6,\ scc(\overline{x_1}) = 3 \Rightarrow x_1 \leftarrow \mathrm{T}$
  - $scc(x_2) = 5,\ scc(\overline{x_2}) = 4 \Rightarrow x_2 \leftarrow \mathrm{T}$
  - $scc(x_3) = 2,\ scc(\overline{x_3}) = 7 \Rightarrow x_3 \leftarrow \mathrm{F}$
  - $scc(x_4) = 8,\ scc(\overline{x_4}) = 1 \Rightarrow x_4 \leftarrow \mathrm{T}$



2-SAT$\leq_p$强连通分量

- 2-SAT问题：给定一个合取范式Φ，是否存在满足的真值赋值?
- 强连通分量问题：给定有向图G，找出它的所有强连通分量
- 输入转换：根据合取范式Φ构造有向图G，$O(m+n)$
- 计算强连通分量，$O(m+n)$
- 输出转换：根据G的强连通分量的拓扑序来决定变量x_i的赋值
  - 对强连通分量进行拓扑排序，$O(m+n)$
  - 确定变量赋值，$O(n)$
- 2-SAT可以在$O(m+n)$时间内解决