高级算法设计与分析-第三部分

计数问题的算法与复杂性

夏盟佶
Xia, Mingji

中科院软件所
计算机科学国家重点实验室

2021

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

- 图灵机:读字符，写字符，变状态，向右或左移动。

$$\Sigma \times Q \to \Sigma \times Q \times \{\to, \leftarrow\}$$

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

- 图灵机:读字符，写字符，变状态，向右或左移动。

$$\Sigma \times Q \to \Sigma \times Q \times \{\to, \leftarrow\}$$

- 计算时间：输入长度为n时，图灵机在$T(n)$步内完成计算。

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

- 图灵机:读字符，写字符，变状态，向右或左移动。

$$\Sigma \times Q \to \Sigma \times Q \times \{\to, \leftarrow\}$$

- 计算时间：输入长度为n时，图灵机在$T(n)$步内完成计算。
- 多项式时间算法，指计算时间是n的某个多项式，

$$T(n) = poly(n)$$

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

- 图灵机:读字符，写字符，变状态，向右或左移动。

$$\Sigma \times Q \to \Sigma \times Q \times \{\to, \leftarrow\}$$

- 计算时间：输入长度为n时，图灵机在$T(n)$步内完成计算。
- 多项式时间算法，指计算时间是n的某个多项式，

$$T(n) = poly(n)$$

- 计算表格：$T(n) \times T(n)$，第i行记录着第i步时，带子的内容、读写头位置、图灵机的状态。

自动机和图灵机

- 自动机:读字符，变状态。

$$\Sigma \times Q \to Q$$

- 图灵机:读字符，写字符，变状态，向右或左移动。

$$\Sigma \times Q \to \Sigma \times Q \times \{\to, \leftarrow\}$$

- 计算时间：输入长度为n时，图灵机在$T(n)$步内完成计算。
- 多项式时间算法，指计算时间是n的某个多项式，

$$T(n) = poly(n)$$

- 计算表格：$T(n) \times T(n)$，第i行记录着第i步时，
  带子的内容、读写头位置、图灵机的状态。
- 每行每格的内容可用$\Sigma \times (Q \cup \{o\})$的一个字符表示。
  o表示读写头不在此格。
  $\Sigma \times Q$的字符可编码成0-1串。

计算问题：判定、优化、计数

Σ^*表示Σ上所有字符串的全体。

- 判定问题
  $F : \Sigma^* \to \{0, 1\}$
  可用状态集合中的两个停机状态q_0和q_1，来定义输出。
- 优化问题
- 计数问题
  $F : \Sigma^* \to \mathbf{N}$
- 可统一用进入停机状态后，输入带上的符号串定义输出。

Σ^*表示Σ上所有字符串的全体。
只有有限种输入的函数F'，不能作为此研究范围中的计算问题。
例如：数独、围棋、九连环、很多应用与工程问题、……
常规的算法和计算复杂性分析，关心所需的计算资源，是输入长度的什么函数，主要是自变量趋向无穷时，这个函数的增长率。

布尔线路

- 与门 $\wedge$ 或门 $\vee$ 非门 $\neg$

布尔线路

- 与门 $\wedge$ 或门 $\vee$ 非门 $\neg$
- 布尔线路可以表示所有布尔函数$f : X \in \{0,1\}^n \rightarrow \{0,1\}$。

布尔线路

- 与门 $\wedge$　或门 $\vee$　非门 $\neg$
- 布尔线路可以表示所有布尔函数$f : X \in \{0,1\}^n \to \{0,1\}$。
- $f(X) = 1$,当且仅当$X = S_1$或者$X = S_2$或者……。（用或门）

布尔线路

- 与门 $\wedge$　或门 $\vee$　非门 $\neg$
- 布尔线路可以表示所有布尔函数$f : X \in \{0,1\}^n \rightarrow \{0,1\}$。
- $f(X) = 1$,当且仅当$X = S_1$或者$X = S_2$或者……。
  （用或门）
- $X = S_1$，当且仅当它们第一位相同，并且第二位相同，……，第n位相同。
  （用与门）

布尔线路

- 与门 $\wedge$ 或门 $\vee$ 非门 $\neg$
- 布尔线路可以表示所有布尔函数$f : X \in \{0,1\}^n \to \{0,1\}$。
- $f(X) = 1$,当且仅当$X = S_1$或者$X = S_2$或者……。
  （用或门）
- $X = S_1$，当且仅当它们第一位相同，并且第二位相同，……，第n位相同。
  （用与门）
- 只用非门、或门也可以。

布尔逻辑：3CNF公式

- 3CNF公式

  $$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

  其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$，是3元或函数，作用在3个文字上。
  每个文字，是某个变量x_j 或者$\neg x_j$。

3SAT问题

- SAT=satisfiability
- 3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，问是否存在一个变量的赋值$\pi : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$，使得$\pi \models \varphi$，即$\varphi(\pi) = 1$。

NP

- $F \in NP$
  （当且仅当存在多项式时间不确定型图灵机M，使得$M(x)$的接受当且仅当$F(x) = 1$，）
  当且仅当存在多项式时间算法R，R的两个输入x和y总满足$|y| = |x|^k$，
  使得：任意x，$F(x) = 1$当且仅当存在y，$R(x, y) = 1$。

NP

- $F \in NP$
（当且仅当存在多项式时间不确定型图灵机M，使得$M(x)$的接受当且仅当$F(x) = 1$，）
当且仅当存在多项式时间算法R，R的两个输入x和y总满足$|y| = |x|^k$，
使得：任意x，$F(x) = 1$当且仅当存在y，$R(x, y) = 1$。
- 计算R的图灵机，对于长n的输入x和长n^k的输入y，有一个$poly(n) \times poly(n)$的计算表格。

Cook-Levin定理

- Ψ是F到H的一个多项式时间归约，
  如果对任意的X，$F(X) = H(\Psi(X))$。

Cook-Levin定理

- Ψ是F到H的一个多项式时间归约，
  如果对任意的X，$F(X) = H(\Psi(X))$。
- 如果任意$F \in NP$，F可以被多项式时间归约到一个计算问题H，H是NP难的。

Cook-Levin定理

- Ψ是F到H的一个多项式时间归约，
  如果对任意的X，$F(X) = H(\Psi(X))$。
- 如果任意$F \in NP$，F可以被多项式时间归约到一个计算问题H，H是NP难的。
- 3SAT是NP难的。

Cook-Levin定理

- Ψ是F到H的一个多项式时间归约，如果对任意的X，$F(X) = H(\Psi(X))$。
- 如果任意$F \in NP$，F可以被多项式时间归约到一个计算问题H，H是NP难的。
- 3SAT是NP难的。
- 任取一个NP问题F，及其任意一个输入X，考虑$R(X, Y)$的计算表格，把它转化成一个关于Y等的3CNF公式。

归约Ψ用3CNF公式描述计算表格

- 3CNF公式

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。

归约Ψ用3CNF公式描述计算表格

- 3CNF公式
$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$
其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
- 计算表格中除了第一行的X是Ψ的输入，被作为Ψ已知布尔值使用，所有其他$\Sigma \times Q$中的字符被表示成布尔变量。

归约Ψ用3CNF公式描述计算表格

- 3CNF公式
$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$
其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
- 计算表格中除了第一行的X是Ψ的输入，被作为Ψ已知布尔值使用，所有其他$\Sigma \times Q$中的字符被表示成布尔变量。
- 第i+1行的布尔变量，是第i行的常数个变量的布尔函数f，仅依赖计算F的图灵机。

归约Ψ用3CNF公式描述计算表格

- 3CNF公式

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

  其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
- 计算表格中除了第一行的X是Ψ的输入，被作为Ψ已知布尔值使用，所有其他$\Sigma \times Q$中的字符被表示成布尔变量。
- 第i+1行的布尔变量，是第i行的常数个变量的布尔函数f，仅依赖计算F的图灵机。
- 把函数f写成布尔线路。

归约Ψ用3CNF公式描述计算表格

- 3CNF公式
$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$
其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
- 计算表格中除了第一行的X是Ψ的输入，被作为Ψ已知布尔值使用，所有其他$\Sigma \times Q$中的字符被表示成布尔变量。
- 第i+1行的布尔变量，是第i行的常数个变量的布尔函数f，仅依赖计算F的图灵机。
- 把函数f写成布尔线路。
- 用3CNF公式描述非门的效果$[x = \neg y]$：

$$(x \vee y) \wedge (\neg x \vee \neg y)$$

归约Ψ用3CNF公式描述计算表格

- 3CNF公式
$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$
其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
- 计算表格中除了第一行的X是Ψ的输入，被作为Ψ已知布尔值使用，所有其他$\Sigma \times Q$中的字符被表示成布尔变量。
- 第i+1行的布尔变量，是第i行的常数个变量的布尔函数f，仅依赖计算F的图灵机。
- 把函数f写成布尔线路。
- 用3CNF公式描述非门的效果$[x = \neg y]$：
$$(x \vee y) \wedge (\neg x \vee \neg y)$$
- 用3CNF公式描述或门的效果$[x = (y \vee z)]$：
$$(\neg x \vee y \vee z) \wedge (x \vee \neg y) \wedge (x \vee \neg z)$$

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，问是否存在一个变量的赋值满足$\pi$。

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，
  问是否存在一个变量的赋值满足π。
- 3CNF公式$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
  其中每个子句$C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$。
  每个文字，是某个变量x_j 或者$\neg x_j$。

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，
  问是否存在一个变量的赋值满足π。
- 3CNF公式$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- 每个子句就是一个约束，要满足所有约束。

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，
  问是否存在一个变量的赋值满足π。
- 3CNF公式$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- 每个子句就是一个约束，要满足所有约束。
- 每个约束是关于文字的3元或函数。
  对变量而言，有8种约束形式，从$x \vee y \vee z$，到$\neg x \vee \neg y \vee \neg z$。

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，
  问是否存在一个变量的赋值满足π。
- 3CNF公式$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- 每个子句就是一个约束，要满足所有约束。
- 每个约束是关于文字的3元或函数。
  对变量而言，有8种约束形式，从$x \vee y \vee z$，到$\neg x \vee \neg y \vee \neg z$。
- 3SAT问题就是被这八个三元函数所以定义的CSP问题。

3SAT作为CSP的特例

- 约束可满足性问题（类）：
  CSP（Constraint Satisfaction Problem）
- 回顾3SAT问题：
  输入是n个布尔变量$X = \{x_1, x_2, \ldots, x_n\}$上的3CNF公式$\varphi$，问是否存在一个变量的赋值满足$\pi$。
- 3CNF公式$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- 每个子句就是一个约束，要满足所有约束。
- 每个约束是关于文字的3元或函数。
  对变量而言，有8种约束形式，从$x \vee y \vee z$，到$\neg x \vee \neg y \vee \neg z$。
- 3SAT问题就是被这八个三元函数所以定义的CSP问题。
- CSP问题，就是通过规定可用作变量的约束的函数集合$\mathcal{F}$，定义的对应可满足性问题CSP$(\mathcal{F})$。

CSP问题例子

- 图的顶点覆盖:
  顶点集$X=\{x_1,x_2,\ldots,x_n\}$的一个子集，包含每条边的至少一个端点。

CSP问题例子

- 图的顶点覆盖:
  顶点集$X = \{x_1, x_2, \ldots, x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，

CSP问题例子

- 图的顶点覆盖:
顶点集$X=\{x_1,x_2,\ldots,x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，
- 就得到“CSP(二元或关系)”的一个输入$\phi(x_1,x_2,\ldots,x_n)$。

CSP问题例子

- 图的顶点覆盖:
  顶点集$X = \{x_1, x_2, \ldots, x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，
- 就得到“CSP(二元或关系)”的一个输入$\phi(x_1, x_2, \ldots, x_n)$。
- $\phi(\pi) = 1$当且仅当π作为顶点子集是个顶点覆盖。

CSP问题例子

- 图的顶点覆盖:
  顶点集$X = \{x_1, x_2, \ldots, x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，
- 就得到“CSP(二元或关系)”的一个输入$\phi(x_1, x_2, \ldots, x_n)$。
- $\phi(\pi) = 1$当且仅当π作为顶点子集是个顶点覆盖。
- 问一个图G是否存在顶点覆盖，等价于问ϕ是否可被满足，等价于问，所有的边所对应的二元约束可否被满足。

CSP问题例子

- 图的顶点覆盖:
  顶点集$X = \{x_1, x_2, \ldots, x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，
- 就得到“CSP(二元或关系)”的一个输入$\phi(x_1, x_2, \ldots, x_n)$。
- $\phi(\pi) = 1$当且仅当π作为顶点子集是个顶点覆盖。
- 问一个图G是否存在顶点覆盖，等价于问ϕ是否可被满足，等价于问，所有的边所对应的二元约束可否被满足。
- 2SAT

CSP问题例子

- 图的顶点覆盖:
  顶点集$X=\{x_1,x_2,\ldots,x_n\}$的一个子集，包含每条边的至少一个端点。
- 把G的点作为布尔变量，每条边作为二元约束，
- 就得到“CSP(二元或关系)”的一个输入$\phi(x_1,x_2,\ldots,x_n)$。
- $\phi(\pi)=1$当且仅当π作为顶点子集是个顶点覆盖。
- 问一个图G是否存在顶点覆盖，等价于问ϕ是否可被满足，等价于问，所有的边所对应的二元约束可否被满足。
- 2SAT
- Horn-SAT

难和容易之间的问题

Theorem (Ladner定理)

如果P不等于NP，存在NP中的问题，它不在P中，也不是NP难的。

- 证明是采用延迟对角线方法的构造性证明。
- 一个人造的问题。

难和容易之间的问题

Theorem (Ladner定理)

如果P不等于NP，存在NP中的问题，它不在P中，也不是NP难的。

- 证明是采用延迟对角线方法的构造性证明。
- 一个人造的问题。

复杂性二分定理：一定范围内的问题要么难，要么容易

- 绝大多数研究过的NP中的问题，或者是NP难的，或者在P里。
- 一个问题集合中的问题要么是容易的（P），要么是难的（NP难），这种结果称为复杂性二分定理。
- 一种常见的问题集合，CSP（约束满足）问题。

Dichotomy theorem of CSP

$\mathcal{F}$ is a set of relations in Boolean variables.

Theorem (Schaefer, STOC 1978)

Given a constraint set $\mathcal{F}$, the problem CSP($\mathcal{F}$) is in P, if $\mathcal{F}$ satisfies one of the conditions below, and CSP($\mathcal{F}$) is other wise NP-complete.

- *$\mathcal{F}$ is 0-valid (1-valid).*
- *$\mathcal{F}$ is weakly positive (weakly negative). (Horn SAT)*
- *$\mathcal{F}$ is affine. (A system of linear equations)*
- *$\mathcal{F}$ is bijunctive. (2SAT)*

计数问题例子

- 例

  完美匹配数目问题（$\#PerfectMatching$）
  输入：图G（的编码）
  输出：G的完美匹配数目

计数问题例子

- 例
  完美匹配数目问题（$\#PerfectMatching$）
  输入：图G（的编码）
  输出：G的完美匹配数目
- 例
  统计物理中$Ising$模型（$Gibbs$分布）的配分函数。

计数问题例子

- 例
  完美匹配数目问题（#*PerfectMatching*）
  输入：图*G*（的编码）
  输出：*G*的完美匹配数目
- 例
  统计物理中*Ising*模型（*Gibbs*分布）的配分函数。
- 例
  一个贝叶斯网络，*n*个独立隐藏布尔变量$x_1, \ldots, x_n$，*m*个可观测的变量$y_1, \ldots, y_n$。每个y_i是三个隐藏变量的或函数。
  $Pr[x_1 = 1 | y_1 = y_2 = \cdots = y_m = 1]$?
- 例
  #*3SAT*
  输入：*3CNF*公式
  输出：多少赋值满足这个公式

计数问题例子

- 例
  完美匹配数目问题（*#PerfectMatching*）
  输入：图*G*（的编码）
  输出：*G*的完美匹配数目
- 例
  统计物理中*Ising*模型（*Gibbs*分布）的配分函数。
- 例
  一个贝叶斯网络，*n*个独立隐藏布尔变量$x_1, \ldots, x_n$，*m*个可观测的变量$y_1, \ldots, y_n$。每个y_i是三个隐藏变量的或函数。
  $Pr[x_1 = 1 | y_1 = y_2 = \cdots = y_m = 1]$?
- 例
  #3SAT
  输入：*3CNF*公式
  输出：多少赋值满足这个公式
  （后两个例子可以互相归约）

用NP模糊说明环数目的计算难度

- 一个有向图有没有环可以多项式时间计算。
  一个有向图有没有哈密尔顿环是NP难的。
- 一个有向图有多少个环的计算难度？

用NP模糊说明环数目的计算难度

- 一个有向图有没有环可以多项式时间计算。
  一个有向图有没有哈密尔顿环是NP难的。
- 一个有向图有多少个环的计算难度?
- n表示输入图G的顶点数。
  构造图G'，原图G的每条边，被替换成一个$O(m)$大小但有2^m通过方式的构件。

用NP模糊说明环数目的计算难度

- 一个有向图有没有环可以多项式时间计算。
  一个有向图有没有哈密尔顿环是NP难的。
- 一个有向图有多少个环的计算难度？
- n表示输入图G的顶点数。
  构造图G'，原图G的每条边，被替换成一个$O(m)$大小但有2^m通过方式的构件。
- 如果G有长为n的环，G'至少有$(2^m)^n$个环。

用NP模糊说明环数目的计算难度

- 一个有向图有没有环可以多项式时间计算。
  一个有向图有没有哈密尔顿环是NP难的。
- 一个有向图有多少个环的计算难度?
- n表示输入图G的顶点数。
  构造图G'，原图G的每条边，被替换成一个$O(m)$大小但有2^m通过方式的构件。
- 如果G有长为n的环，G'至少有$(2^m)^n$个环。
- 如果G没有长为n的环，至多有$(n+1)^{n-1}$个环。
  （长$n-1$的环至多n^{n-1}）
  每个环对应G'中至多$(2^m)^{n-1}$个环。
  G'至多有$(n+1)^{n-1}(2^m)^{n-1}$个环。

用NP模糊说明环数目的计算难度

- 一个有向图有没有环可以多项式时间计算。
  一个有向图有没有哈密尔顿环是NP难的。
- 一个有向图有多少个环的计算难度？
- n表示输入图G的顶点数。
  构造图G'，原图G的每条边，被替换成一个$O(m)$大小但有2^m通过方式的构件。
- 如果G有长为n的环，G'至少有$(2^m)^n$个环。
- 如果G没有长为n的环，至多有$(n+1)^{n-1}$个环。
  （长$n-1$的环至多n^{n-1}）
  每个环对应G'中至多$(2^m)^{n-1}$个环。
  G'至多有$(n+1)^{n-1}(2^m)^{n-1}$个环。
- 取一个合适的多项式大小的m，区分以上两种情况。
- 完成了从哈密尔顿环存在性问题到环数目的归约。

#P

- $F : \Sigma^* \to \mathbf{N}$ (e.g. $\Sigma = \{0, 1\}$)

例

完美匹配数目问题（*#PerfectMatching*）
输入：图G（的编码）
输出：G的完美匹配数目

#P

- $F : \Sigma^* \to \mathbf{N}$ (e.g. $\Sigma = \{0, 1\}$)

例

完美匹配数目问题（*#PerfectMatching*）
输入：图*G*（的编码）
输出：*G*的完美匹配数目

- 定义

$F \in \#P$
当且仅当存在多项式时间不确定型图灵机M，使得$M(x)$的接受计算路径数目等于$F(x)$，
当且仅当存在多项式时间算法R，R的两个输入x和y总满足$|y| = |x|^k$，使得$F(x) = |\{y|R(x, y) = 1\}|$。

#P

- $F : \Sigma^* \to \mathbf{N}$ (e.g. $\Sigma = \{0, 1\}$)

例

完美匹配数目问题（*#PerfectMatching*）
输入：图*G*（的编码）
输出：*G*的完美匹配数目

- 定义

$F \in \#P$
当且仅当存在多项式时间不确定型图灵机M，使得$M(x)$的接受计算路径数目等于$F(x)$，
当且仅当存在多项式时间算法R，R的两个输入x和y总满足$|y| = |x|^k$，使得$F(x) = |\{y | R(x, y) = 1\}|$。

  - NP里的判定问题问有没有证据y，#P里的计数问题问有多少证据。

#P

- $F : \Sigma^* \to \mathbf{N}$ (e.g. $\Sigma = \{0, 1\}$)

例

完美匹配数目问题（*#PerfectMatching*）
输入：图*G*（的编码）
输出：*G*的完美匹配数目

- 定义

$F \in \#P$
当且仅当存在多项式时间不确定型图灵机M，使得$M(x)$的接受计算路径数目等于$F(x)$，
当且仅当存在多项式时间算法R，R的两个输入x和y总满足$|y| = |x|^k$，使得$F(x) = |\{y|R(x, y) = 1\}|$。

  - NP里的判定问题问有没有证据y，#P里的计数问题问有多少证据。
  - 这个类由L. Valiant于1979年在文章“The complexity of computing the permanent”, Theoretical Computer Science，中首次提出。

#P难

定义

*#P*难问题：
如果*#P*里的所有问题都可以多项式时间图灵归约到F，那么F就是*#P*难问题。

复杂性类联系：

- 如果一个问题是#P难的，那么也是NP难的。

#P难

定义

*#P*难问题：

如果*#P*里的所有问题都可以多项式时间图灵归约到F，那么F就是*#P*难问题。

复杂性类联系：

- 如果一个问题是#P难的，那么也是NP难的。
- Toda定理：

$$PH \subseteq P^{\#P}。$$

#P难问题

- #SAT，Permanent，哈密尔顿回路数目问题，……

#P难问题

- #SAT，Permanent，哈密尔顿回路数目问题，……

Theorem

#SAT是#P难的。

#P难问题

- #SAT，Permanent，哈密尔顿回路数目问题，……

Theorem

#SAT是#P难的。

- 证明类似cook定理。

#P难问题

- #SAT，Permanent，哈密尔顿回路数目问题，……

Theorem

*#SAT*是*#P*难的。

- 证明类似cook定理。

Theorem

$0,1-$*Permanent*是*#P*难的。

#P难问题

- #SAT，Permanent，哈密尔顿回路数目问题，……

Theorem

#SAT是#P难的。

- 证明类似cook定理。

Theorem

$0, 1-$*Permanent*是*#P*难的。

- 因为#SAT可以归约到Permanent，并且归约有传递性。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$
  有向图$G(V, E, W)$中，有向边(j, k)的权重$W(j, k) = A_{j,k}$。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$
  有向图$G(V, E, W)$中，有向边(j, k)的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是G的所有圈覆盖权重之和。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$
  有向图$G(V, E, W)$中，有向边(j, k)的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是G的所有圈覆盖权重之和。
- $V = \{1, 2, \ldots, n\}$，$U = \{1', 2', \ldots, n'\}$。
  无向偶图$H(V, U, E, W)$中，边(j, k')的权重$W(j, k') = A_{j,k}$。

Permanent：偶图的带权完美匹配数目

- A是$n \times n$矩阵。

定义

$$\mathit{Permanent}(A) = \sum_{\pi \in S_n} \prod_{j=1}^{n} A_{j,\pi(j)}$$

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$
  有向图$G(V, E, W)$中，有向边(j, k)的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是G的所有圈覆盖权重之和。
- $V = \{1, 2, \ldots, n\}$，$U = \{1', 2', \ldots, n'\}$。
  无向偶图$H(V, U, E, W)$中，边(j, k')的权重$W(j, k') = A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。

计数版本与判定版本

一个二元函数R定义的计数问题是否是#P难的，和同一个R定义的判定问题是否是NP难的，这两种命题没有联系。

- #SAT是#P难的，其判定版本SAT是NP难的。
- 偶图的完美匹配数目问题是#P难的，其判定版本偶图是否存在完美匹配，是有多项式时间算法的。
  用图的最大匹配算法即可。
- #2SAT是#P难的，其判定版本2SAT有多项式时间算法。
- 线性方程组解的存在性与解的数目都是P的。

#CSP问题类

每一个#CSP问题的实例（输入）和答案（输出）形式是一样的。

- 实例：
  作用于变量集合$X = \{x_1, x_2, \ldots, x_n\}$的一些函数：
  $R_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, R_m(x_{m,1}, \ldots, x_{m,r_m})$，其中$x_{i,j} \in X$。

#CSP问题类

每一个#CSP问题的实例（输入）和答案（输出）形式是一样的。

- 实例：
  作用于变量集合$X = \{x_1, x_2, \ldots, x_n\}$的一些函数：
  $R_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, R_m(x_{m,1}, \ldots, x_{m,r_m})$，其中$x_{i,j} \in X$。
- 答案：

$$\sum_{\sigma: X \to D} \prod_{j=1}^{m} R_j(\sigma(x_{j,1}), \ldots, \sigma(x_{j,r_j}))$$

（D表示一个变量的定义域，例如$\{0, 1\}$。）

#CSP问题类

每一个#CSP问题的实例（输入）和答案（输出）形式是一样的。

- 实例：
  作用于变量集合$X=\{x_1,x_2,\ldots,x_n\}$的一些函数：
  $R_1(x_{1,1},\ldots,x_{1,r_1}),\ldots,R_m(x_{m,1},\ldots,x_{m,r_m})$，其中$x_{i,j}\in X$。
- 答案：

$$\sum_{\sigma:X\to D}\prod_{j=1}^{m}R_j(\sigma(x_{j,1}),\ldots,\sigma(x_{j,r_j}))$$

（D表示一个变量的定义域，例如$\{0,1\}$。）

给定一个函数集合$\mathcal{F}$，就定义了一个#CSP问题#CSP($\mathcal{F}$)，它的实例所用的函数R_j必须来自$\mathcal{F}$。

#CSP问题类

每一个#CSP问题的实例（输入）和答案（输出）形式是一样的。

- 实例：
  作用于变量集合$X=\{x_1,x_2,\ldots,x_n\}$的一些函数：
  $R_1(x_{1,1},\ldots,x_{1,r_1}),\ldots,R_m(x_{m,1},\ldots,x_{m,r_m})$，其中$x_{i,j}\in X$。
- 答案：

$$\sum_{\sigma:X\to D}\prod_{j=1}^{m}R_j(\sigma(x_{j,1}),\ldots,\sigma(x_{j,r_j}))$$

（D表示一个变量的定义域，例如$\{0,1\}$。）

给定一个函数集合$\mathcal{F}$，就定义了一个#CSP问题#CSP($\mathcal{F}$)，
它的实例所用的函数R_j必须来自$\mathcal{F}$。
#2SAT=#CSP($\{F|F(y,z)=y\vee z$或者$\bar{y}\vee z$或者$y\vee\bar{z}$或者$\bar{y}\vee\bar{z}\}$)

#CSP的二分定理

布尔定义域的#CSP问题

- $\{0,1\}$值域
  只有一类问题有多项式时间算法，其他都是#P难的。
  一类：仿射关系。

#CSP的二分定理

布尔定义域的#CSP问题

- $\{0,1\}$值域
  只有一类问题有多项式时间算法，其他都是#P难的。
  一类：仿射关系。
- 非负实数值域
  两类：pure affine和product type

#CSP的二分定理

布尔定义域的#CSP问题

- $\{0, 1\}$值域
  只有一类问题有多项式时间算法，其他都是#P难的。
  一类：仿射关系。
- 非负实数值域
  两类：pure affine和product type
- 复数值域
  两类： $\mathcal{A}$ 和 $\mathcal{P}$(即product type的定义自然推广到复数值域)

第一易解类：$\mathcal{A}$

- $X = (x_1, x_2, \ldots, x_n)$。$x_j \in D = \{0, 1\}$。

第一易解类：$\mathcal{A}$

- $X = (x_1, x_2, \ldots, x_n)$。$x_j \in D = \{0, 1\}$。
- 定义仿射关系函数：$\chi_{(AX=C)}$，即D上n维空间的仿射子空间的指示函数。（D作为大小2的有限域。）

第一易解类：$\mathcal{A}$

- $X = (x_1, x_2, \ldots, x_n)$。$x_j \in D = \{0, 1\}$。
- 定义仿射关系函数：$\chi_{(AX=C)}$，即D上n维空间的仿射子空间的指示函数。（D作为大小2的有限域。）
- $P(x_1, x_2, \ldots, x_n)$是一个整系数多项式，$x_j \in \{0, 1\}$，使用整数加法乘法运算。例如，$x_j^2 = x_j$。

第一易解类：$\mathcal{A}$

- $X = (x_1, x_2, \ldots, x_n)$。$x_j \in D = \{0, 1\}$。
- 定义仿射关系函数：$\chi_{(AX=C)}$，即D上n维空间的仿射子空间的指示函数。（D作为大小2的有限域。）
- $P(x_1, x_2, \ldots, x_n)$是一个整系数多项式，$x_j \in \{0, 1\}$，使用整数加法乘法运算。例如，$x_j^2 = x_j$。
- 要求P的最高次数是2；要求交错二次项的系数是偶数。例如：$3x_1 + 2x_2 + 2x_1x_3 + 4x_2x_3$。

第一易解类：$\mathcal{A}$

- $X = (x_1, x_2, \ldots, x_n)$。$x_j \in D = \{0, 1\}$。
- 定义仿射关系函数：$\chi_{(AX=C)}$，即D上n维空间的仿射子空间的指示函数。（D作为大小2的有限域。）
- $P(x_1, x_2, \ldots, x_n)$是一个整系数多项式，$x_j \in \{0, 1\}$，使用整数加法乘法运算。例如，$x_j^2 = x_j$。
- 要求P的最高次数是2；要求交错二次项的系数是偶数。例如：$3x_1 + 2x_2 + 2x_1x_3 + 4x_2x_3$。
- $F \in \mathcal{A}$，当且仅当有形式$\chi_{(AX=C)} \cdot i^{P(x_1, x_2, \ldots, x_n)}$。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\ldots,x_r$，
解是$x_{r+1}=L_{r+1}(X')$,...$x_n=L_n(X') \mod 2$。$X'=(x_1,\ldots,x_r,1)$。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\dots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\dots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\dots,x_r$，
  解是$x_{r+1}=L_{r+1}(X')$,…$x_n=L_n(X') \mod 2$。$X'=(x_1,\dots,x_r,1)$。
- 例如，方程组$x_3=x_1+x_2+1 \mod 2$
  的解表达式中$L_3(X')=x_1+x_2+1$。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\ldots,x_r$，
  解是$x_{r+1}=L_{r+1}(X'),\ldots x_n=L_n(X') \mod 2$。$X'=(x_1,\ldots,x_r,1)$。
- 例如，方程组$x_3=x_1+x_2+1 \mod 2$
  的解表达式中$L_3(X')=x_1+x_2+1$。
- $$\sum_{x_j\in\{0,1\},j=1,\ldots,r} i^{P(x_1,x_2,\ldots,x_r,L_{r+1}(X'),\ldots,L_n(X'))} \quad ?$$

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\dots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\dots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\dots,x_r$，
  解是$x_{r+1}=L_{r+1}(X')$,...$x_n=L_n(X')$ mod 2。$X'=(x_1,\dots,x_r,1)$。
- 例如，方程组$x_3=x_1+x_2+1 \mod 2$
  的解表达式中$L_3(X')=x_1+x_2+1$。
- $$\sum_{x_j\in\{0,1\},j=1,\dots,r} i^{P(x_1,x_2,\dots,x_r,L_{r+1}(X'),\dots,L_n(X'))} \quad ?$$
- 线性函数L_j在模2之后才是正确x_j值。
  而P采用整数（实际上可以是模4）运算。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\ldots,x_r$，
  解是$x_{r+1}=L_{r+1}(X'),\ldots x_n=L_n(X') \mod 2$。$X'=(x_1,\ldots,x_r,1)$。
- 例如，方程组$x_3=x_1+x_2+1 \mod 2$
  的解表达式中$L_3(X')=x_1+x_2+1$。
- $$\sum_{x_j\in\{0,1\},j=1,\ldots,r} i^{P(x_1,x_2,\ldots,x_r,L_{r+1}(X'),\ldots,L_n(X'))} \quad ?$$
- 线性函数L_j在模2之后才是正确x_j值。
  而P采用整数（实际上可以是模4）运算。
- 下面把解表达式的模2运算融入模4运算。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j\in\{0,1\},j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$
- 假设$AX=C$的自由变量是$x_1,\ldots,x_r$，
  解是$x_{r+1}=L_{r+1}(X')$,...$x_n=L_n(X') \mod 2$。$X'=(x_1,\ldots,x_r,1)$。
- 例如，方程组$x_3=x_1+x_2+1 \mod 2$
  的解表达式中$L_3(X')=x_1+x_2+1$。
- $$\sum_{x_j\in\{0,1\},j=1,\ldots,r} i^{P(x_1,x_2,\ldots,x_r,L_{r+1}(X'),\ldots,L_n(X'))} \quad ?$$
- 线性函数L_j在模2之后才是正确x_j值。
  而P采用整数（实际上可以是模4）运算。
- 下面把解表达式的模2运算融入模4运算。
  - 如果P里有一个一次项x_j，替换成$L_j(X')^2$。
    因为$L^2 \mod 4$等于$L \mod 2$。

$\#CSP(\mathcal{A})$的算法

- $$\sum_{x_j \in \{0,1\}, j=1,\ldots,n} \chi_{(AX=C)} i^{P(x_1,x_2,\ldots,x_n)}$$
- 假设$AX = C$的自由变量是$x_1, \ldots, x_r$，
  解是$x_{r+1} = L_{r+1}(X')$,...$x_n = L_n(X') \mod 2$。$X' = (x_1, \ldots, x_r, 1)$。
- 例如，方程组$x_3 = x_1 + x_2 + 1 \mod 2$
  的解表达式中$L_3(X') = x_1 + x_2 + 1$。
- $$\sum_{x_j \in \{0,1\}, j=1,\ldots,r} i^{P(x_1,x_2,\ldots,x_r,L_{r+1}(X'),\ldots,L_n(X'))} \quad ?$$
- 线性函数L_j在模2之后才是正确x_j值。
  而P采用整数（实际上可以是模4）运算。
- 下面把解表达式的模2运算融入模4运算。
  - 如果P里有一个一次项x_j，替换成$L_j(X')^2$。
    因为$L^2 \mod 4$等于$L \mod 2$。
  - 如果有交错项$2x_jx_{j'}$，替换成$2L_j(X')L_{j'}(X')$即可。

$\sum_{x_j\in\{0,1\}} i^{P(x_1,x_2,\ldots,x_r)}$

- x_1的一次项系数是偶数。提取含$2x_1$的项的公因子。

$$\sum_{x_2,\ldots,x_r}\sum_{x_1} i^{2x_1L(X')+P'(x_2,\ldots,x_r)} = \sum_{x_2,\ldots,x_r}(i^{P'(x_2,\ldots,x_r)}\sum_{x_1}(-1)^{x_1L(X')})$$

$$\sum_{x_1}(-1)^{x_1L(X')} = 2\chi_{(L(X')=0)}，\text{其中} X' = (x_2,\ldots,x_r,1)$$

$$\sum_{x_j\in\{0,1\}} i^{P(x_1,x_2,\ldots,x_r)}$$

- x_1的一次项系数是偶数。提取含$2x_1$的项的公因子。

$$\sum_{x_2,\ldots,x_r}\sum_{x_1} i^{2x_1L(X')+P'(x_2,\ldots,x_r)} = \sum_{x_2,\ldots,x_r}(i^{P'(x_2,\ldots,x_r)}\sum_{x_1}(-1)^{x_1L(X')})$$

$$\sum_{x_1}(-1)^{x_1L(X')} = 2\chi_{(L(X')=0)}，其中X' = (x_2,\ldots,x_r,1)$$

- x_1的一次项系数是奇数。保留一个x_1，提取公因子。

$$= \sum_{x_2,\ldots,x_r}(i^{P'(x_2,\ldots,x_r)}\sum_{x_1}(-1)^{x_1L(X')}i^{x_1})$$

当$L(X') = 0 \mod 2$时，$F(1,x_2,\ldots,x_r) = iF(0,x_2,\ldots,x_r)$；
当$L(X') = 1 \mod 2$时，$F(1,x_2,\ldots,x_r) = -iF(0,x_2,\ldots,x_r)$。

$$\sum_{x_1}(-1)^{x_1L(X')}i^{x_1} = (1+i)i^{3L^2(X')}$$

原本是关于$x_1, x_2, \ldots, x_n$的2^n个赋值对$\chi \cdot i^P$求和，已经看到了如何消除χ中的非自由变量和i^P中的一个变量，代价是函数表达式$\chi \cdot i^P$的幅度受控的变化。
消除一个变量x_i，即从对两个大小2^{n-1}的超平面的点的函数值求和，转化成对其中一个超平面的点的函数值求和。

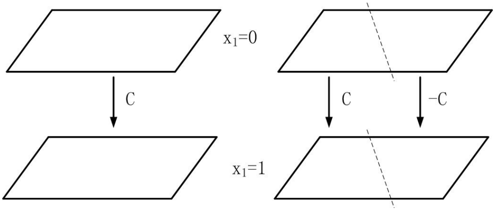


Figure: 虚线右侧区域表示仿射子空间$L(X) = 1$

第一种情况，$C = 1$。第二种情况，$C = \pm i$。
$(1 + i) = i(1 - i)$，妙哉。

$\mathcal{A}$中的二元函数例子

- $$F=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$
- #CSP($\{F\}$)问题的一个实例，是一些F应用到变量$x_1,\ldots,x_n$。
- 这个实例对应图G，$V_G=\{x_1,\ldots,x_n\}$，$(x_j,x_k)\in E_G$当且仅当实例中有约束$F(x_j,x_k)$。
- 考虑被赋值1的顶点形成的子图H。如果H有奇（偶）数条边，所有约束的乘积是-1（resp. 1）。
- #CSP($\{F\}$)(G)=图G的偶数条边的这种子图数目$-$奇数条边的子图数目。
- 推论：图G的偶数条边的子图数目是多项式时间可以计算的。

第二易解类$\mathcal{P}$（product type）

- 先看一种简单情形：
  取一些一元函数F_i，
  定义$F(x_1, x_2, \ldots, x_n) = F_1(x_1)F_2(x_2)\cdots F_n(x_n)$。

第二易解类$\mathcal{P}$（product type）

- 先看一种简单情形：
  取一些一元函数F_i，
  定义$F(x_1, x_2, \ldots, x_n) = F_1(x_1)F_2(x_2)\cdots F_n(x_n)$。
- #CSP($\{F\}$)的算法？

第二易解类$\mathcal{P}$（product type）

- 先看一种简单情形：
  取一些一元函数F_i，
  定义$F(x_1, x_2, \ldots, x_n) = F_1(x_1)F_2(x_2)\cdots F_n(x_n)$。
- #CSP($\{F\}$)的算法？

$$\sum_{x_1,x_2,\ldots,x_n\in\{0,1\}} F(x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n}(F_j(0) + F_j(1))$$

集合$\mathcal{E}$

- 一个n元函数F在集合$\mathcal{E}$，当且仅当它在除某互补串对α和$\bar{\alpha}$之外的输入上值都是0。
  严格定义：$F \in \mathcal{E}$当且仅当存在$\alpha \in \{0,1\}^n$，
  $\forall X, X \notin \{\alpha, \bar{\alpha}\} \Rightarrow F(X) = 0$。

集合$\mathcal{E}$

- 一个n元函数F在集合$\mathcal{E}$，当且仅当它在除某互补串对α和$\bar{\alpha}$之外的输入上值都是0。
  严格定义：$F \in \mathcal{E}$当且仅当存在$\alpha \in \{0,1\}^n$，
  $\forall X, X \notin \{\alpha, \bar{\alpha}\} \Rightarrow F(X) = 0$。
- $\{\alpha, \bar{\alpha}\}$是一个一维仿射子空间。

集合$\mathcal{E}$

- 一个n元函数F在集合$\mathcal{E}$，当且仅当它在除某互补串对α和$\bar{\alpha}$之外的输入上值都是0。
  严格定义：$F \in \mathcal{E}$当且仅当存在$\alpha \in \{0,1\}^n$，
  $\forall X, X \notin \{\alpha, \bar{\alpha}\} \Rightarrow F(X) = 0$。
- $\{\alpha, \bar{\alpha}\}$是一个一维仿射子空间。
- 一个变量赋值若要有非0贡献，它对$\mathcal{E}$中函数F的一个自变量的赋值，决定了它对其他F的自变量的赋值。
  即一个自由变量的值，决定其他非自由变量的值。

集合$\mathcal{E}$

- 一个n元函数F在集合$\mathcal{E}$，当且仅当它在除某互补串对α和$\bar{\alpha}$之外的输入上值都是0。
  严格定义：$F \in \mathcal{E}$当且仅当存在$\alpha \in \{0,1\}^n$，
  $\forall X, X \notin \{\alpha, \bar{\alpha}\} \Rightarrow F(X) = 0$。
- $\{\alpha, \bar{\alpha}\}$是一个一维仿射子空间。
- 一个变量赋值若要有非0贡献，它对$\mathcal{E}$中函数F的一个自变量的赋值，决定了它对其他F的自变量的赋值。
  即一个自由变量的值，决定其他非自由变量的值。
- 易算

$$\sum_{x_1,x_2,\ldots,x_n \in \{0,1\}} F(x_1, x_2, \ldots, x_n)$$

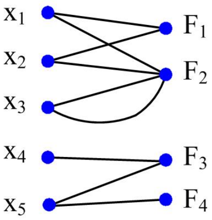
（练习题。提示：结合前面的例子，和#CSP($\mathcal{A}$)中非自由变量的处理方法。直接想也很简单。）

第二易解类$\mathcal{P}$ （product type）

- 一个函数$\mathcal{P}$在中当且仅当能表示成$\mathcal{E}$中函数的乘积。

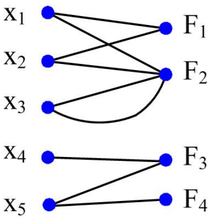
第二易解类$\mathcal{P}$（product type）

- 一个函数$\mathcal{P}$在中当且仅当能表示成$\mathcal{E}$中函数的乘积。
- #CSP问题的实例可以画成偶图。



第二易解类$\mathcal{P}$（product type）

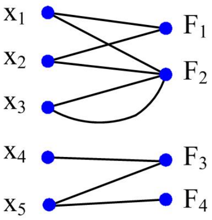
- 一个函数$\mathcal{P}$在中当且仅当能表示成$\mathcal{E}$中函数的乘积。
- #CSP问题的实例可以画成偶图。



- 连通的$\mathcal{E}$函数乘在一起，还是$\mathcal{E}$函数。

第二易解类$\mathcal{P}$（product type）

- 一个函数$\mathcal{P}$在中当且仅当能表示成$\mathcal{E}$中函数的乘积。
- #CSP问题的实例可以画成偶图。



- 连通的$\mathcal{E}$函数乘在一起，还是$\mathcal{E}$函数。
- 一个图的值是它的连通分支的值的乘积。（练习题）

二分定理的#P困难性结论

- 如果$\mathcal{F} \not\subseteq \mathcal{A}$ 并且$\mathcal{F} \not\subseteq \mathcal{P}$，那么$\#CSP(\mathcal{F})$是#P难的。
- 证明思路：
  对不在$\mathcal{A}$的函数降元，得到新的不在$\mathcal{A}$的函数。
  对不在$\mathcal{P}$的函数降元，得到新的不在$\mathcal{P}$的函数。
  对二、三元函数证明二分定理的困难性部分。

参考文献

- Nadia Creignou, Miki Hermann:
  Complexity of Generalized Satisfiability Counting Problems. Inf. Comput. 125(1): 1-12 (1996)
- Martin E. Dyer, Leslie Ann Goldberg, Mark Jerrum:
  The Complexity of Weighted Boolean CSP. SIAM J. Comput. 38(5): 1970-1986 (2009)
- Jin-Yi Cai, Pinyan Lu, Mingji Xia:
  The complexity of complex weighted Boolean #CSP. J. Comput. Syst. Sci. 80(1): 217-236 (2014) 两个易解类的原始证明。
- Complexity Dichotomies for Counting Problems: Volume 1, Boolean Domain, 2017
  Jin-Yi Cai and Xi Chen 计数复杂性二分定理专著。
- Complexity Classifications of Boolean Constraint Satisfaction Problems, 2001
  Nadia Creignou, Sanjeev Khanna, Madhu Sudan 包含了早期的判定问题、计数问题和优化问题的复杂性二分定理。