接下来，我们表明 $\mathrm{SAT}_H \notin \mathsf{P}$。如果它属于 P，那么存在一个图灵机 M 在至多 cn^c 步中解决这个问题。这意味着存在一个整数 $i > c$ 使得 $M = M_i$。由 H_i 的定义，对于任意 $n > 2^{2^i}$，都有 $H(n) \leq i$，和上面类似地，这意味着对于足够长的输入，$\mathrm{SAT}_H = \mathrm{SAT}$。

为了表明它不是 NP-完全的，我们只要表明 H 在 n 区域无穷时趋于无穷。也就是说，对于任意整数 i，只有有限个 n 使得 $H(n) = i$。因为 $\mathrm{SAT}_H \notin \mathsf{P}$，对于任意 i，存在一个输入 $|x|$ 使得给定 i $|x|^i$ 的时间，M_i 不能给出正确的结果。因此，对于任意 $n > 2^{|x|}$，$H(x) \neq i$。

3.4 相对化、强包含和弱包含

接下来，我们希望分析对角线法的限度。首先，我们注意到对角线法都应用了以下两个性质：

- 存在一个通用图灵机可以在不算太大的开销下模拟任意一个图灵机的执行；
- 图灵机可以编号。

实际上，Žák 的一个引理充分地体现了这两个要素，我们在下面稍微看一眼，它有点复杂，可以先深吸一口气：

引理 9（Žák） 考虑一个语言构成的集合 $\mathcal{L} = \{L_i \mid i \in \mathcal{I}\}$，其中 $\mathcal{I}$ 为指标集，L 为某个语言。设 L' 是另一个语言，存在一个映射 $f : L' \to \mathcal{I}$ 使得它的原像是无限集，设 $z : L' \to \mathbb{N}$，使得 $z(x)$ 是满足以下条件的正整数：

- $x1^{z(x)} \in L \Leftrightarrow x \notin L_{f(x)}$
- $\forall k < z(x), x1^k \in L \Leftrightarrow x1^{k+1} \in L_{f(x)}$

则不存在 $i \in \mathcal{I}$ 使得 L 与 L_i 只差有限个字符串。

作为一个联系，请读者用这个结论证明上面的定理 6，这个证明几乎是直接代入。其中，我们取的 $\mathcal{L}$ 就是我们需要用对角线法跳出的复杂度类，而 $\mathcal{I}$ 实际上取的就是图灵机的编号。因此，我们需要注意，这实际上把图灵机当成了一个黑盒，或者说，谕示机（oracle）。

定义 4 一个带语言 L（或复杂度类 C）的谕示机（oracle）的谕示图灵机（oracled Turing machine）是一个带有一个额外纸带以及一个额外状态的图灵机。当图灵机进入状态 q_{query} 时，它可以直接知晓谕示带上写的字符串是否属于语言 L，这当成一个计算步骤。

当然，一个带 SAT 的谕示机的谕示图灵机相当于一个带 NP 的谕示机的谕示图灵机，这很显然，因为 SAT 是 NP-完全的，带了它就可以解决 NP 中的全部问题。同样的，基于此我们可以定义谕示图灵机的复杂度类：P^{C} 表示带复杂度类 C 的确定性谕示图灵机所能在多项式时间内解决的问题，NP^{C} 表示带复杂度类 C 的谕示的非确定性谕示图灵机所能在多项式时间内解决的问题，其他复杂度类的定义同理，不再赘述。这样的复杂度类称为由 C 所相对化（relativized）的复杂度类。

定义 5 如果带语言 A 的谕示机的谕示图灵机能够在复杂度类 P^A 内解决问题 B，那么就称 B 可以被 Cook 归约到 A。

不难发现，Karp 归约无非是一种“只能访问一次谕示机”的归约，而 Cook 归约使得我们可以访问多项式次谕示机。介于两者之间的还有真值表归约（truth table reduction），它表示我们只能并行地访问一次谕示机，让它一次性给我们一些答案。三者对应地记作：

$$\mathsf{C}^{\|A} \subset \mathsf{C}^{A[\log]} \subset \mathsf{C}^A$$

对应 Karp 归约、真值表归约和 Cook 归约。

考虑到这样的想法，我们的对角线方法的局限性就显露出来了：只有对于任意谕示机都能成立的真包含关系才能用对角线方法证明。这种包含被称为强包含（strong inclusion），反之则称为弱包含（weak inclusion）。不幸的是，我们有以下结果：

定理 10 （Baker-Gill-Solovay, 1975） 存在谕示机 A 和 B，使得 $P^A = \mathsf{NP}^A, P^B \neq \mathsf{NP}^B$。

这个结果的证明并不复杂。只要能够构造出两个对应的语言即可。当然，因为构造的复杂性，在此我们不多赘述，有兴趣的读者可以自行查询相关文献。

我们定义以下术语：

定义 6 称语言 L 对于复杂度 C 是低的（low），如果 $\mathsf{C}^L = \mathsf{C}$。如果复杂度 L 中的所有语言 L 对复杂度 C 都是低的，则称 L 对 C 是低的。

那么，这意味着，为了证明弱包含的结论（这并不是没有证明！以后会讲的 $\mathsf{IP} = \mathsf{PSPACE}$ 和 PCP 定理都是经典的例子），对角线方法的目前版本并不充足。它必须涉及某种非相对化的事实，也就是说，某个在图灵机上成立但是在谕示图灵机上不成立的结果。实际上，还有一些工作涉及关于 P 的公理化，例如 Arora-Impagalizzo-Vazirani 公理系统，参见 *Relativizing versus Nonrelativizing Techniques: The Role of Local Checkability*，他们给出了一个正好包含关于 P 的相对化结果而不包含非相对化结果的公理系统。但是，现在并不知道如何扩张这个公理系统使得它能够包含非相对化结果。

3.5 复杂度类的多项式分层 PH

有了谕示机，我们自然会考虑能不能用谕示机推广原来的定义。实际上，这能给出一个非常漂亮的层级图。考虑

$$\Delta_0^{\mathsf{P}} = \Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \mathsf{P}$$
$$\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$$
$$\Delta_{i+1}^{\mathsf{P}} = \mathsf{P}^{\Sigma_i^{\mathsf{P}}}$$
$$\Pi_{i+1}^{\mathsf{P}} = \text{co-}\mathsf{NP}^{(\Sigma_i)^{\mathsf{P}}}$$

不难看出，在我们这里，有 $\Sigma_1^{\mathsf{P}} = \mathsf{NP}, \Pi_1^{\mathsf{P}} = \text{co-}\mathsf{NP}$，而且以下包含关系成立：

$$\Sigma_i^{\mathsf{P}} \subset \Delta_{i+1}^{\mathsf{P}} \subset \Sigma_{i+1}^{\mathsf{P}}, \quad \Pi_i^{\mathsf{P}} \subset \Delta_{i+1}^{\mathsf{P}} \subset \Pi_{i+1}^{\mathsf{P}}, \quad \Sigma_i^{\mathsf{P}} = \text{co-}\Pi_i^{\mathsf{P}}$$

最后，我们定义 $\mathsf{PH} = \cup_{k\geq 1} \Delta_k^P$，下面的事实成立，但我们将不做证明：

- $\mathsf{P} = \mathsf{NP} \Leftrightarrow \mathsf{P} = \mathsf{PH}$
- $\mathsf{NP} = \text{co-}\mathsf{NP} \Rightarrow \mathsf{NP} = \mathsf{PH}$

关于 PH 和 PSPACE 的分层，也有一个 Ladner 型定理，由 Schöning 证明：

定理 11 如果 $\mathsf{PH} \neq \mathsf{PSPACE}$，则存在一个 $\mathsf{PSPACE} \setminus \mathsf{PH}$ 中的非 PSPACE 完全问题。

第四章 随机复杂度类

4.1 随机图灵机

定义 1 （随机图灵机） 一个随机图灵机（PTM）是一个多带图灵机，接受一个随机源 $r \in \{0,1\}^*$ 和一个通常的输入字符串 $x \in \{0,1\}^*$ 作为输入 (x,r)，并且具备一个函数 $R: \mathbb{N} \longrightarrow \mathbb{N}$ 满足对于任意输入 (x,r) 有 $|r| = R(|x|)$。

这里，r 的含义是一个被显式的写出的“黑盒”随机源，可以把他看作对二进制字符串的均匀采样。

PTM 的一个等价的定义是以 $\frac{1}{2}$ 概率选择 2 个转移函数之一的 NTM，由此也可以看出二者之间的紧密联系。PTM 的输出是一个随机变量，因此当我们说 PTM 判定一个语言 L 时，我们其实在讨论 PTM 输出关于随机源 r 的概率 $\mathsf{Pr}_r[M(x,r)=1]$；在上述等价定义下，这个概率可以定义为所有接受输入 x 的分支数量与所有可能的分支数量的比值，在这种情况下，PTM 的运行时间被定义为所有运行分支的最大运行时间。

4.2 RP 和 BPP

我们考虑一族函数 $f(n)$，称一个语言 $L \in \{0,1\}^*$ 是：

- $\mathsf{RTIME}(f(n))$ （randomized time）的，如果对于所有输入规模n，概率图灵机能在 $f(n)$ 步之内停机，且

$$x \in L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] \geq \frac{1}{2}$$
$$x \notin L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] = 0$$

- $\mathsf{BPTIME}(f(n))$ （bounded-error probabilistic time），如果对于所有输入规模 n，概率图灵机能在 $f(n)$ 步之内停机，且

$$x \in L \Rightarrow \Pr_{r \sim \{0,1\}^{R(|x|)}}[M(x,r)=1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow \Pr_{r \sim \{0,1\}^{R(|x|)}}[M(x,r)=1] \leq \frac{1}{3}$$

注意到我们限制了随机源长度 $r \in \{0,1\}^{\mathsf{poly}(|x|)}$。

RTIME 族刻画的是类似 Monto-Carlo 算法的过程，具备单侧错误（one-sided error），语言 L 中的输入以较小的错误概率被接受，不在语言 L 中的输入总是会被拒绝；BPTIME 族刻画的则是双侧错误（two-sided error）。然后我们定义两个相应的复杂度类：

- $\mathsf{RP} = \bigcup_{k \geq 1} \mathsf{RTIME}(n^k)$
- $\mathsf{BPP} = \bigcup_{k \geq 1} \mathsf{BPTIME}(n^k)$

定义 RP 类的单侧错误概率 $\frac{1}{2}$ 看上去有点随便，但是实际上其可以被替换成任意数值。反复运用图灵机，即可减少错误概率，这个过程叫错误归约（error reduction）。

具体来说，我们定义如下复杂度类：

定义 2 （$\mathsf{RP}_{n^{-c}}$与$\mathsf{RP}_{2^{-n^s}}$）

1. 令 $\mathsf{RP}_{n^{-c}}$ 为能被多项式时间 PTM 识别的语言 L 类，满足存在固定的常数 $c>0$，使得

$$x \in L \Rightarrow \Pr_{r\sim\{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] \geq n^{-c}$$
$$x \notin L \Rightarrow \Pr_{r\sim\{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] = 0$$

2. 令 $\mathsf{RP}_{2^{-n^s}}$ 为能被多项式时间 PTM 识别的语言 L 类，满足存在固定的常数 $s>0$，使得

$$x \in L \Rightarrow \Pr_{r\sim\{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] \geq 1-2^{-n^s}$$
$$x \notin L \Rightarrow \Pr_{r\sim\{0,1\}^{\mathsf{poly}(|x|)}}[M(x,r)=1] = 0$$

然后证明如下引理：

引理 1 $\mathsf{RP}_{n^{-c}} = \mathsf{RP}_{2^{-n^s}} = \mathsf{RP}$

证明

首先 $\mathsf{RP}_{2^{-n^s}} \subseteq \mathsf{RP} \subseteq \mathsf{RP}_{n^{-c}}$ 是显然的，我们只需证明 $\mathsf{RP}_{n^{-c}} \subseteq \mathsf{RP}_{2^{-n^s}}$。假设语言 L 可以被概率图灵机 M 在多项式时间内以错误概率 $1-n^{-c}$ 识别，那么我们定义概率图灵机 M_1 为：

$$M_1(x,(r_1,r_2,...r_k)) = M(x,r_1) \vee M(x,r_2) \vee ... \vee M(x,r_k)$$

这里，仍有 $r_i \in \{0,1\}^{\mathsf{poly}(|x|)}$ 并取 $k=n^{c+s}$。显然 M_1 的运行时间仍然是关于输入规模的级别的，同时可知存在常数 $c>0$：

$$\begin{aligned} x \in L \Rightarrow & \Pr_r[M_1(x,r)=1] \\ & = 1 - \Pr_r[M_1(x,r)=0] \\ & \geq 1 - \prod_{i=1}^{k} M(x,r_i) \geq 1-(1-n^{-c})^{n^{c+s}} \\ & \geq 1-2^{-n^s} \end{aligned}$$

$$x \notin L \Rightarrow \Pr_r[M_1(x,r)=1] = \prod_{i=1}^{k} M(x,r_i) = 0$$

∎

类似地，我们可以定义 co-RP 复杂度类以及证明对应的错误归约引理。

关于 BPP 类定义中的双侧错误概率 $\frac{1}{3}$ 只是用于表明我们判定的成功概率有一个 $\frac{1}{2}$ 界的，注意如果进行随机的猜测我们就可以达到 $\frac{1}{2}$ 成功率，但是只要这个概率是远离 $\frac{1}{2}$ 界的，这个双侧错误概率就可以被类似的方法减少。类似地，我们定义：

定义 3 （$\mathsf{BPP}_{n^{-c}}$与$\mathsf{BPP}_{2^{-n^s}}$）

1. 令 $\mathsf{BPP}_{n^{-c}}$ 为能被多项式时间 PTM 识别的语言 L 类，满足存在固定的常数 $c>0$，使得

$$x \in L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathrm{poly}(|x|)}}[M(x,r)=1] \geq \frac{1}{2} + n^{-c}$$

$$x \notin L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathrm{poly}(|x|)}}[M(x,r)=1] \leq \frac{1}{2} - n^{-c}$$

2. 令 $\mathsf{BPP}_{2^{-n^s}}$ 为能被多项式时间 PTM 识别的语言 L 类，满足存在固定的常数 $s > 0$，使得

$$x \in L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathrm{poly}(|x|)}}[M(x,r)=1] \geq 1 - 2^{-n^s}$$

$$x \notin L \Rightarrow \Pr_{r \sim \{0,1\}^{\mathrm{poly}(|x|)}}[M(x,r)=1] \leq 2^{-n^s}$$

定义 M_1 为：

$$M_1(x,(r_1,r_2,...r_k)) = \mathrm{majority}(M(x,r_1),M(x,r_2),...,M(x,r_k))$$

首先，我们不加证明地使用如下引理

引理 2 （Chernoff-Hoeffding 界的一个推论） 令 $X_1, X_2, ...X_k$ 是独立的随机布尔变量，且对 $1 \leq i \leq n$ 有 $\Pr[X_i = 1] = p$。令 $\delta \in (0,1)$，有如下上界：

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} X_i - p\right| > \delta\right] < e^{-\frac{\delta^2}{4}pk}$$

对 $1 \leq i \leq k$，每次调用 M 回答正确的概率 $\Pr_{r_i}[M(x,r_i) = L(x)] \geq \frac{1}{2} + n^{-c}$ 只需要取 $k = 8n^{2s+c}$ 对于上述引理取 $p = \frac{1}{2} + n^{-c}$ 和 $\delta = \frac{1}{2}n^{-c}$，我们的 M_1 的双侧错误概率为：

$$\Pr\left[\frac{1}{k}\sum_{i=1}^{k}[M(x,r) \neq L(x)] \leq \frac{1}{2} + \frac{1}{2}n^{-c}\right] \leq e^{-\frac{1}{4n^{-2c}}\frac{1}{2}8n^{2c+s}} \leq 2^{-n^s}$$

或者，你可以直接估计组合数，对 $1 \leq i \leq k$，每次调用 M 回答正确的概率 $\Pr_{r_i}[M(x,r_i) = L(x)] \geq \frac{1}{2} + n^{-c}$，则我们的 M_1 回答错误当且仅当我们对 M 的 k 次重复测试中至多获得了 $\frac{k}{2}$ 次正确答案，我们的 M_1 的双侧错误概率为：

$$\begin{aligned}
\sum_{i=0}^{\lceil \frac{k}{2} \rceil}\binom{k}{i}\left(\frac{1}{2}+n^{-c}\right)^i\left(\frac{1}{2}-n^{-c}\right)^{k-i} &\leq \sum_{i=0}^{\lceil \frac{k}{2} \rceil}\binom{k}{i}\left(\frac{1}{2}+n^{-c}\right)^{\frac{k}{2}}\left(\frac{1}{2}-n^{-c}\right)^{\frac{k}{2}} \\
&\leq \left(\frac{1}{4}-n^{-2c}\right)^{\frac{k}{2}}\sum_{i=0}^{\lceil \frac{k}{2} \rceil}\binom{k}{i} \\
&\leq \left(\frac{1}{4}-n^{-2c}\right)^{\frac{k}{2}} \cdot 2^k \\
&= \left(1-4n^{-2c}\right)^{\frac{k}{2}} \\
&\leq \delta
\end{aligned}$$

为了使错误概率小于某个 δ，仅需取 $k = \frac{2\log\delta^{-1}}{\log\left(1-4n^{-2c}\right)^{-1}} = s\left(n^{-2c}\right) \cdot \log\left(\frac{1}{\delta}\right)$ 对某个依赖 n 的常数 s

总之，我们得到了如下引理：

引理 3 （BPP 增强引理）

$$\mathsf{BPP}_{n^{-c}} = \mathsf{BPP}_{2^{-n^s}} = \mathsf{BPP}$$

说明上述错误归约引理是为了方便我们判定一些随机算法的复杂度类：

例 1 （Miler-Rabin 素数测试） 我们知道由费马小定理，素数 p 有 $a^p \equiv p(\mathrm{mod}\, p)$，那么反过来，如果一个数 p 满足 $a^p = p(\mathrm{mod}\, p)$，是否可以判定 p 是素数呢（这被成为费马素数测试）？很可惜，答案是否定的，反例的这些合数被成为费马伪素数，最小的费马伪素数为 341。因此我们考虑多选取几个数 a 进行测试以减少错误率，然而仍然存在一些数，例如卡麦克尔数，它们对所有 a 都满足费马小定理，因此我们需要更复杂的算法。

Miler-Rabin 素数测试：对于某个基数 p，我们记 $p-1 = 2^s \cdot d$，如果对某个的 $0 \le r \le s-1$，以下两个条件之一满足：

- $a^d \equiv 1(\mathrm{mod}\, p)$
- $a^{2^r d} \equiv 1(\mathrm{mod}\, p)$

则 p 至多有 $\frac{1}{4}$ 概率是个伪素数。

例 2 （多项式相等判定） 给定域 $\mathbb{F}$ 上的多项式环 $\mathbb{F}[x_1, x_2, ..., x_n]$ 中的两个多项式 $f(x_1, x_2, ..., x_n)$ 和 $g(x_1, x_2, ..., x_n)$ 判定 $f = g$，或者等价地，判定 $f - g = 0$。

这个问题的定义需要一定的严格界定，不敢兴趣的读者可以跳过这部分

多项式是以什么形式被“编码”的：

- 显式的一串系数：$f(x_1, ..., x_n) = \sum_{\alpha_1, ..., \alpha_n \in \mathbb{N}} = c_{\alpha_1, ..., \alpha_n} x_1^{\alpha_1} x_n^{\alpha_n}$，在这种情况下，相等判定是显然的。
- 黑盒：对于任何输入，给出多项式的值
- 数学表达式：一串符号，例如 $x_3(x_5 - x_6) + x_2 x_4(2x_3 + 3x_5)$ 或者形式化地，使用算术电路来描述。虽然我们可以讲多项式完全展开，但是可能会导致指数级的数量的项，因此并不是一个高效的方法。

多项式系数。一般来说，我们会假设多项式系数属于某个域 $\mathbb{F}$ 或者更一般地，一个整环，例如 $\mathbb{Q}$、$\mathbb{Z}$ 和 $\mathbb{F}_p$。

相等。$f = g$ 可以指的是形式相等（多项式展开后每个单项式的系数相同）或者函数相等（对于所有输入，多项式的值相同），在 $\mathbb{Z}$ 上，这两个定义是等价的。然而，对于有限域这并不成立：

$$f(x) = \prod_{\alpha \in \mathbb{F}} (x - \alpha)$$

是一个 $\mathbb{F}$ 上的多项式，但是值恒为 0，因此一般来说我们考虑的是形式相等。

我们给出多项式零判定的一个随机算法。假设我们已知多项式在域 $\mathbb{F}$ 上，次数至多为 d：

- 令 $S \subset \mathbb{F}$ 为任意一个大小为 $2d$ 的子集
- 在 $\mathbb{F}$ 中随机选择 $\alpha_1, \alpha_2, ..., \alpha_n$
- 计算 $f(\alpha_1, \alpha_2, ..., \alpha_n)$，如果为 0，则认为 $f = 0$，否则认为 $f \neq 0$

下面我们不加证明地使用如下引理：

引理 4 （Schwartz–Zippel 引理） 如果 f 是一个在域 $\mathbb{F}$（或整环）上次数为 d 的非零多项式，对于任意的 $S \subset \mathbb{F}$，有：

$$\Pr_{\alpha_1,\alpha_2,...,\alpha_n \in S}[f(\alpha_1,\alpha_2,...,\alpha_n)=0] \leq \frac{d}{|S|}$$

显然，如果 $f=0$，那么这个算法总是能够正确判定；下面我们说明如果 $f \neq 0$，那么这个算法至多 $\frac{1}{2}$ 的概率判定错误。因此，多项式相等判定是一个 RP 复杂类问题。

例 3 （图的完美匹配） 对图 $G=(E,V)$，图 G 的一个匹配是一个边的子集 $E' \subset E$，使得对于任意的边 $e,e' \in E'$，e 和 e' 没有共同的端点。一个完美匹配是一个匹配，使得图的每个顶点都在匹配中。给定一个图 G，判定 G 是否有完美匹配。

注意，对二分图 $G=(V_1,V_2,E)$，存在时间复杂度为 $O\left(|E| \cdot \sqrt{|V_1|+|V_2|}\right)$ 完美匹配的确定性算法（使用 alternating paths 或者归约到 MAX FLOW）。我们这里给出一个利用多项式相等判定的随机化算法，它基于以下引理：

引理 5 （二分图完美匹配的等价问题） 对二分图 $G=(V_1,V_2,E)$，假设其邻接矩阵为 $A^G_{i,j}$，G 有一个完美匹配，当且仅当 $\det(A) \neq 0$

证明 首先我们假设 $|V_1|=|V_2|$（否则不存在完美匹配），其次我们记 $|V_1|=|V_2|=n$ 与 $|E|=m$，然后我们定义 G 邻接矩阵 $A^G_{i,j}$ 为

$$A^G_{i,j}=\begin{cases} x_{i,j} & \text{如果 } (i,j) \in E \\ 0 & \text{否则} \end{cases}$$

其中 $x_{i,j}$ 是一个符号变量。

充分性：如果 G 存在完美匹配，则存在 $\sigma : X \longrightarrow Y$ 的一个双射，于是 σ 是 $\{1,2,...,n\}$ 上的一个排列，根据行列式的定义：

$$\det\left(A^G_{i,j}\right)=\sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_i^n A^G_{i,\sigma(i)} = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_i^n x_{i,\sigma(i)}$$

是一个关于变量 $x_{i,j}$ 的多项式，对于一个完美匹配 $\forall i \in V_1, (i,\sigma(i)) \in E$，所以 $\det\left(A^G_{i,j}\right) \neq 0$

必要性：如果 $\det\left(A^G_{i,j}\right) \neq 0$，则存在某个排列 σ 使得 $\prod_i^n x_{i,\sigma(i)} \neq 0$，于是 $\forall i \in V_1, (i,\sigma(i)) \in E$，因此 σ 对应了一个完美匹配。

∎

因此，$\det(A)$ 的次数至多是 n，我们取子集 $S=\left\{1,2,...,n \cdot 2^{n^2}\right\} \subset \mathbb{Z}$，均匀随机地取 $\alpha_1,\alpha_2,...,\alpha_{n^2} \in S$，由 Schwartz–Zippel 引理：

$$\Pr_{\alpha_1,\alpha_2,...,\alpha^{n^2}}[\det(A)=0] \leq \frac{1}{2^{n^2}}$$

4.3 两两独立性

在增强引理中，我们使用了重复 k 次的方法进行错误规约，但是注意到在运行时间翻了 k 倍同时，所需要的随机源长度也相应翻了 k 倍。 有没有可能在进行错误规约的同时，所需的随机源长度更短呢？ 实际上，上述 k 次所使用的随机源 $r_1, ..., r_k$ 是独立同分布的，我们可以证明，这个条件可以被放宽到两两独立性，从而减少对随机源的需求。

我们来看一些经典案例：

例 4 （异或） 假设 $x_1, ..., x_r$ 从 $\{0,1\}$ 中均匀采样，对非空子集 $S \subseteq \{1, 2, ..., r\}$，定义：

$$Z_S = \bigoplus_{i \in S} x_i$$

则 $\mathcal{Z} = \{Z_S\}_{S \subseteq \{1,2,...,r\}}$ 是一组 $2^r - 1$ 大小两两独立的随机变量。

$|\mathcal{Z}| = 2^r - 1$ 是显然的。 下面我们证明两两独立性，对于 $S, T \subseteq \{1, 2, ..., r\}$ 且 $S \neq T$，和任意的 $a, b \in \{0, 1\}$，我们有：

$$\Pr_{x_1,...,x_r \in \{0,1\}}[Z_S = a \wedge Z_T = b] = \Pr_{x_1,...,x_r \in \{0,1\}}[Z_S = a \mid Z_T = b] \cdot \Pr_{x_1,...,x_r \in \{0,1\}}[Z_T = b]$$

注意到由于 $S \neq T$，存在 $\{x_i\}_{i \in S \setminus T}$ 使得 Z_S 的值等概率地为 0 或者 1，因此：

$$\begin{aligned}\Pr_{x_1,...,x_r \in \{0,1\}}[Z_S = a \wedge Z_T = b] &= \Pr_{x_1,...,x_r \in \{0,1\}}[Z_S = a \mid Z_T = b] \cdot \Pr_{x_1,...,x_r \in \{0,1\}}[Z_T = b] \\ &= \Pr_{x_1,...,x_r \in \{0,1\}}[Z_S = a] \cdot \Pr_{x_1,...,x_r \in \{0,1\}}[Z_T = b]\end{aligned}$$

这个例子说明了，我们为了获取大小为 n 的一组两两独立的随机变量，只需要 $r = \log(n+1)$ 的真随机。 不过注意到，对于上述方法生成的分布 $(Z_1, Z_2, ..., Z_n) \in \{0,1\}^n$，其支撑集只有 $2^r = n + 1$ 大小，指数级别小于 $\{0,1\}^n$ 的状态空间。

例 5 （全域哈希（Universal Hashing）函数族） 称一个类 $\mathcal{H} : \{h : \{1, ..., N\} \longrightarrow \{1, ..., M\}\}$ 是一族全域哈希函数，如果对任意 $x, y \in \{1, ..., N\}$ 和任意 $a, b \in \{1, ..., M\}$，满足：

1. $\Pr_{h \in \mathcal{H}}[h(x) = a] = \frac{1}{M}$
2. $\Pr_{h \in \mathcal{H}}[h(x) = a \wedge h(y) = b] = \frac{1}{M^2}$

我们可以构造出一族全域哈希函数，从而按如下方式构造出一组两两独立的随机变量： 采样 $h \sim \mathcal{H}$，取 $X_i = h(i)$ 即可。 仅需要 $\log|\mathcal{H}|$ 长度的随机源来采样 h

---

构造方法。对某个自然数 $n \in \mathbb{N}$, 取 $N = 2^n$。 采样 $c, d \sim \{1, 2, ..., N\}$，并定义哈希函数：

$$\begin{aligned}h_{c,d} : \{1, ..., N\} &\to \{1, ..., N\} \\ x &\mapsto cx + d\end{aligned}$$

于是 $\mathcal{H} = \left\{h_{c,d}\right\}_{c,d \in \{1,...,N\}}$，采样仅需 $O(\log N)$ 长度的随机源。

证明这是一族全域哈希函数。 对于任意 $x, y \in \{1, ..., N\}$ 和 $a, b \in \{1, ..., N\}$，我们有：

$$\begin{aligned}\Pr_{h\in\mathcal{H}}[h(x)=a\wedge h(y)=b] &= \Pr_{c,d\in\{1,\dots,N\}}\left[h_{c,d}(x)=a\wedge h_{c,d}(y)=b\right]\\ &= \Pr_{c,d\in\{1,\dots,N\}}[cx+d=a\wedge cy+d=b]\\ &= \Pr_{c,d\in\{1,\dots,N\}}\left[c=\frac{a-d}{y-x}\wedge d=\frac{bx-ay}{x-y}\right]\\ &= \Pr_{c,d\in\{1,\dots,N\}}\left[c=\frac{a-d}{y-x}\right]\cdot\Pr_{c,d\in\{1,\dots,N\}}\left[d=\frac{bx-ay}{x-y}\right]\\ &= \frac{1}{N^2}\end{aligned}$$

推论 6（全域哈希函数族的一般情形） 可以使用 $\log N+\log M$ 长度的随机源构造一族全域哈希函数，从而获取一组在 $\{1,...,M\}$ 上的两两独立随机分布。

下面我们证明等待已久的问题，BPP 的错误规约可以使用更少的随机源长度。假设，原本的图灵机 M 需要 $\mathsf{poly}(n)$ 长度的随机源，我们记 $R=2^{\mathsf{poly}(n)}$，这等价于从 $\{1,...,R\}$ 上均匀采样。定义 M_2 为：

$$M_2(x,(r_1,r_2,...,r_k))=\mathrm{majority}(M(x,r_1),M(x,r_2),...,M(x,r_k))$$

不同的是，这次我们只需要一组 $r_1,r_2,...,r_k\in\{1,...,R\}$ 两两独立的随机变量即可，由推论我们仅需要 $\log k+\log R$。不过，这里的 k 应该与之前的情况有所区别，我们记 $Y_i=(M(x,r_i)=L(x))$ 和 $Y=\sum_{i=1}^k Y_i$，然后计算 M_2 的正确率。我们可以发现 Y_i 之间并不是两两独立的，所以 Chernoff-Hoeffding 界并不适用，我们转而使用 Chebyshev 不等式，先估计方差，注意由于 r_i 之间是两两独立，所以 Y_i 也是两两独立，且 Y_i 可以被看作 $p=\frac{2}{3}$ 的伯努利分布：

$$\begin{aligned}\mathrm{Var}(Y)&=\mathrm{Var}\left(\sum_{i=1}^k Y_i\right)\\ &=\sum_{i=1}^k\mathrm{Var}(Y_i)+\sum_{i\neq j}\mathrm{Cov}(Y_i,Y_j)\\ &=\sum_{i=1}^k\mathrm{Var}(Y_i)\\ &\le\frac{2}{9}k\end{aligned}$$

于是 M_2 错误的概率应该有上界：

$$\begin{aligned}\Pr_r[M_2(x,r_1,r_2,...,r_k)\neq L(x)]&\le\Pr_r\left[\left|Y-\frac{2}{3}k\right|>\frac{1}{10}k\right]\\ &\le\frac{100}{k^2}\cdot\frac{2}{9}k=O\left(\frac{1}{k}\right)\end{aligned}$$

因此，仅需取 $k=\Omega\left(\frac{1}{\varepsilon}\right)$，$M_2$ 就具有 ε 的双侧错误概率

4.4 ZPP 与 PP

我们考虑一族函数 $f(n)$，称一个语言 $L\in\{0,1\}^*$ 是：

- ZTIME($f(n)$)（zero-error probabilistic time）的，如果对于所有输入规模n，概率图灵机能在**期望** $f(n)$ 步之内停机，且

$$x \in L \Rightarrow \Pr_{r \sim \{0,1\}^{\text{poly}(|x|)}}[M(x,r)=1]=1$$
$$x \notin L \Rightarrow \Pr_{r \sim \{0,1\}^{\text{poly}(|x|)}}[M(x,r)=1]=0$$

注意到，ZTIME 类总是会返回正确的结果，它的期望运行时间是 $f(n)$，即使有些分支的运行时间可能超过了 $f(n)$，它刻画了类似 Las-Vegas 算法的过程，具备零错误（zero-error）。然后我们定义相应的复杂度类：$\text{ZPP}=\bigcup_{k\geq 1}\text{ZTIME}(n^k)$

ZPP 实际上还有个有趣的等价定义：

定义 4 （ZPP 的等价定义） 语言 $L \in \text{ZPP}$，如果存在一个概率图灵机 M，如果对于所有输入规模为 n，M 能在 poly(n) 步内停机，输出 $\{0,1,\text{unknown}\}$ 之一，满足

$$x \in L \Rightarrow M(x,r) \in \{1,\text{unknown}\} \text{且} \Pr_r[M(x,r)=\text{unknown}] \leq \frac{1}{2}$$
$$x \notin L \Rightarrow M(x,r) \in \{0,\text{unknown}\} \text{且} \Pr_r[M(x,r)=\text{unknown}] \leq \frac{1}{2}$$

定理 7 P $\subseteq$ ZPP

任意 DTM 能在多项式时间内以零错误输出正确答案。

定理 8 ZPP $=$ co-ZPP

对于任意 $L \in \text{ZPP}$，假设零错误概率图灵机 M 识别 L，则 $\overline{L}$ 可以被 对 M 输出取反识别，因此 $\overline{L} \in \text{ZPP}$。故 $L \in \text{co-ZPP}$。

定理 9 ZPP $=$ RP $\cap$ co-RP

ZPP $\subseteq$ RP 和 ZPP $\subseteq$ co-RP 并不是显然的，因为 ZPP 类问题的某些分支上可能由超过 poly(n) 的运行时间，需要进行转化。 假设 $L \in \text{ZPP}$ 能被零错误概率图灵机 M 以期望 poly(n) 步识别，构造一个概率图灵机 N，它模拟 M 的 3 poly(n) 步，如果 $M(x,r)$ 停机则 $N(x,r)=M(x,r)$ 否则 $N(x,r)=0$。

- 如果 $x \in L$，假设 T 是 $M(x,r)$ 关于 r 的运行时间随机变量，由马尔可夫不等式：

$$\Pr_r[N(x,r)=0]=\Pr_r[T>3\ \text{poly}(n)] \leq \frac{E[T]}{3\ \text{poly}(n)} \leq \frac{1}{3}$$

- 如果 $x \notin L$，如果 M 停机了，则会以零错误拒绝 x，因此 N 返回 0；否则 N 默认返回 0。无论何种情况，都有：

$$\Pr_r[N(x,r)=1]=0$$

因此 $L \in \text{RP}$，所以 ZPP $\subseteq$ RP。

接下来我们证明 RP $\cap$ co-RP $\subseteq$ ZPP，假设 $L \in \text{RP} \cap \text{co-RP}$ 能相应地被零错误概率图灵机 M_1 和 M_2 以 poly(n) 步识别，构造一个概率图灵机 N，模拟 M_1 和 M_2：

- 如果 $M_1(x,r)=1$，则 $N(x,r)=1$，否则
- 如果 $M_2(x,r)=0$，则 $N(x,r)=0$，否则
- 重复模拟 M_1 和 M_2 直到上述之一发生

N 如果有输出，一定是零错误的，因此我们考虑 N 的期望运行时间。对于输入 x，假设 T 是 $N(x,r)$ 关于 r 的模拟轮数随机变量：

- 如果 $x\in L$，$\mathsf{Pr}_r[N(x,r)\ \textbf{在 1 轮模拟后停机}]=\mathsf{Pr}_r[M_1(x,r)=1]\geq\frac{2}{3}$
- 如果 $x\notin L$，$\mathsf{Pr}_r[N(x,r)\ \textbf{在 1 轮模拟后停机}]=\mathsf{Pr}_r[M_2(x,r)=0]\geq\frac{2}{3}$

因此：$E[T]=1+\mathsf{Pr}_r[N(x,r)\ \textbf{不在 1 轮模拟后停机}]=1+\frac{1}{3}E[T]\Rightarrow E[T]\leq\frac{3}{2}$。

故 $L\in\mathsf{ZPP}$

定义 5 （PP） 一个语言 $L\in\mathsf{PP}$，如果对于所有输入规模 n，概率图灵机能在 $\mathsf{poly}(n)$ 步之内停机，且

$$x\in L\Rightarrow\Pr_{r\sim\{0,1\}^{R(|x|)}}[M(x,r)=1]>\frac{1}{2}$$

$$x\notin L\Rightarrow\Pr_{r\sim\{0,1\}^{R(|x|)}}[M(x,r)=1]<\frac{1}{2}$$

定理 10 $\mathsf{PP}=\text{co-}\mathsf{PP}$

由定义显然。

4.5 随机复杂度类关系

定理 11 $\mathsf{P}\subseteq\mathsf{ZPP}\subseteq\mathsf{RP}$

显然

定理 12 $\mathsf{RP}\subseteq\mathsf{NP}$ 且 $\text{co-}\mathsf{RP}\subseteq\text{co-}\mathsf{NP}$

对 $L\in\mathsf{RP}$，我们证明 $L\in\mathsf{NP}$。由于 RP 是单侧零错误，$x\notin L$ 的情形是显然的，我们来看 $x\in L$ 的情形。存在判定 L 的一个 PTM，使得 $\mathsf{Pr}_r[M(x,r)=1]\geq\frac{1}{2}$，因此存在某个随机源 $r\in\{0,1\}^{\mathsf{poly}(n)}$，使得对于任意输入 x，$M(x,r)=L(x)$，因此我们可以把 r 作为证明，构造一个 DTM 判定 L。因此 RP 座落于 PH 的第一层级。

这是一个很有趣的结果，某意义上它暗示了随机性“弱于”非确定性，不明白的读者可以对比一下 PTM 和 NTM 的定义。

定理 13 $\mathsf{RP}\subseteq\mathsf{BPP}$ 且 $\text{co-}\mathsf{RP}\subseteq\mathsf{BPP}$ 进而推出 $\mathsf{ZPP}\subseteq\mathsf{BPP}$

由 RP 和 BPP 的定义是显然的。

定理 14 （Sipser-Gacs-Lautemann） $\mathsf{BPP}\subseteq\Sigma_2^p\cap\Pi_2^p$，或者显式地，$\mathsf{BPP}\subseteq\mathsf{NP}^{\mathsf{NP}}$ 且 $\mathsf{BPP}\subseteq\text{co-}\mathsf{NP}^{\mathsf{NP}}$

因此 BPP 坐落于 PH 的第二层级。

BPP 的概率选择只关于随机源，而和输入 x 无关，因此相比 P 它是一个更好的关于“高效”计算的描述。我们显然有 P $\subseteq$ BPP，但是 P $=$ BPP 与否仍然是一个开放性问题。BPP $\subseteq$ PH 表明，如果 P $=$ NP 则有 P $=$ BPP。

另一方面 NP $\subseteq$ BPP 是不太可能的，否则由增强引理可知，所有 NP 完全问题都有多项式时间的随机算法，这意味着 NP 都能被多项式大小的布尔电路族解决（BPP $\subseteq$ P/Poly），这导致 PH 坍缩，也即 NP $=$ RP 且 PH $\subseteq$ BPP。所以目前我们只能猜想 BPP $\subseteq$ NP，但是我们有如下关系

定理 15 NP $\subseteq$ PP 进而得到 co-NP $\subseteq$ PP

只需证明 SAT $\in$ PP，我们构造一个概率图灵机 M：给定一个 n 个变量的 CNF 公式 φ，我们进行随机均匀赋值，如果赋值满足 φ 则输出 1，否则以 $\frac{1}{2}-\frac{1}{2^{n+1}}$ 概率输出 1 以 $\frac{1}{2}+\frac{1}{2^{n+1}}$ 概率输出 0，从而

$$\varphi \in \mathrm{SAT} \Rightarrow \Pr_r[M(x,r)=1] \geq \left(\frac{1}{2}-\frac{1}{2^{n+1}}\right)\cdot\left(1-\frac{1}{2^n}\right)+1\cdot\frac{1}{2^n} > \frac{1}{2}$$

$$\varphi \notin \mathrm{SAT} \Rightarrow \Pr_r[M(x,r)=1] < \frac{1}{2}-\frac{1}{2^{n+1}} < \frac{1}{2}$$

从而，SAT 判定可以被某一个 PP 类的图灵机解决。

定理 16 （Gill, John） MAJSAT 是 PP 完全问题，从而 PP $\subseteq$ PSPACE

定理 17 （Toda） PH $\subseteq \mathsf{P}^{\mathsf{PP}}$

4.6 去随机化

在上面的例子中我们看到，一方面，良好设计的随机算法能在高效地、以较高正确率判定问题；另一方面，类似 P $=$ RP 或者 P $=$ BPP 等仍然是一个开放性问题。 事实上，随机性也是一种计算的资源，在复杂度理论中，我们非常关心并试图比较各种计算资源（例如我们已经谈论过的时间和空间资源）所带来的计算能力，我们想要知道随机性与空间、时间等资源的关系如何。 为此，我们讨论这个问题：每个随机算法都能被转化为（高效的）确定程序吗?

4.6.1 枚举随机源

枚举是一种基本的构造性去随机方法，我们可以通过枚举所有可能的随机源来模拟一个随机算法，当然结果就是指数级的性能下降。

引理 18 BPP $\subset$ EXP

证明

对于任意 $L \in$ BPP，存在一个 PTM 对于输入规模 n 使用多项式随机源长度 $r \in \{0,1\}^{m(n)}$ 和多项式运行时间 $t(n)$ 以两侧错误概率 $\frac{1}{3}$ 判定 L（BPP 的定义）。 我们可以构造一个 DTM 枚举所有 $r \in \{0,1\}^{m(n)}$ 并计算 PTM 的输出的概率

$$\Pr_r[M(x,r)=1] = \frac{1}{2^{m(n)}}\sum_{r\in\{0,1\}^{m(n)}}\Pr_r[M(x,r)=1]$$

因此这个 DTM 可以在 $2^{m(n)}\cdot t(n)$ 时间内判定 L。

∎

这个方法对适用于所有的 BPP 类问题，但是带来的额外开销达到指数级别，这在通常情况下并不实用。 不过，这里有个有趣的观察，如果原 PTM 所需要的随机源长度很短，对应的 DTM 会相对高效

推论 19 如果 $L \in \mathsf{BPP}$ 需要 $t(n) = \mathsf{poly}(n)$ 时间和 $m(n)$ 长度的随机源，则 $L \in \mathsf{DTIME}\big(2^{m(n)} \cdot t(n)\big)$。 特别地，如果 $m(n) = O(\log n)$，$L \in \mathsf{P}$

换句话说，如果我们能说明所有的随机算法都能只使用少量的随机比特，那么就能推出 $\mathsf{BPP} = \mathsf{P}$。 遗憾的是，但是这一点仍然是未知的，这个结果是目前我们所知的最好的针对所有 BPP 问题的上界。

4.6.2 非构造性去随机化

定理 20 （Adleman） $\mathsf{BPP} \subseteq \mathsf{P/Poly}$

对 $L \in \mathsf{BPP}$，首先使用增强引理将双侧错误概率归约到 2^{-n^2}。因此，存在某些随机源 r^* 使得

$$\Pr_{x \in \{0,1\}^n}[M(x, r^*) \neq L(x)] \leq 2^{-n^2}$$

但是实际上，如果存在某个 x^*，使得 $M(x^*, r^*) \neq L(x)$ 则 $\Pr_{x \in \{0,1\}^n}[M(x, r^*) \neq L(x)] \geq 2^{-n}$，这是不可能的，因此

$$\Pr_{x \in \{0,1\}^n}[M(x, r^*) \neq L(x)] = 0$$

所以，对于规模 n 以及所有的 $x \in \{0,1\}^n$ 一定存在一个足够好的随机源 r^* 能让 DTM 判定问题 L，因此 $L \in \mathsf{P/Poly}$。

相比枚举随机源，这个观察是非常有吸引力的，只要我们能找到这个好的随机源，我们就能在多项式时间内解决 BPP 问题，遗憾的是这个方法是非构造性的，我们仍然不知道比指数时间更好的方法。

4.6.3 条件期望准则

上文中提到的“好随机数”虽然是非构造性，但是对于一类搜索问题，我们可以使用条件期望准则高效地确定“好随机数”。

定义 6 （条件期望准则）

- 如果随机变量 Y 是关于独立的随机变量 $X_1, X_2, ..., X_n$ 的纯函数
- 对于任意 $1 \leq i < n$ 部分赋值 $\{X_1 = x_1, X_2 = x_2, ..., X_i = x_i\}$，条件期望 $E[Y \mid X_1 = x_1, X_2 = x_2, ..., X_i = x_i]$ 可以被高效地计算

我们可以找到一组“好”的赋值 $X_1 = x_1, X_2 = x_2, ..., X_n = x_n$ 使得 $E[Y \mid X_1 = x_1, X_2 = x_2, ..., X_n = x_n] \geq E[Y]$

我们通过归纳法证明这个准则：

基础步：对于 $i = 1$，我们还没进行任何赋值，由期望的定义，至少存在一个 $X_1 = x_1$ 使得 $E[Y \mid X_1 = x_1] \geq E[Y]$

归纳步：假设对于 $1 < i < n$，我们有一组部分赋值 $\{X_1 = x_1, X_2 = x_2, ..., X_i = x_i\}$，条件期望 $E[Y \mid X_1 = x_1, X_2 = x_2, ..., X_i = x_i] \geq E[Y]$，由于

$$\begin{aligned} & E[Y \mid X_1 = x_1, ..., X_i = x_i] \\ = & \sum_x \mathsf{Pr}[X_{i+1} = x \mid X_1 = x_1, ..., X_i = x_i] \cdot E[Y \mid X_1 = x_1, ..., X_i = x_i, X_{i+1} = x_{i+1}] \\ \geq & E[Y] \end{aligned}$$

因此存在某个 $X_{i+1} = x_{i+1}$ 使得 $E[Y \mid X_1 = x_1, ..., X_i = x_i, X_{i+1} = x_{i+1}] \geq E[Y]$。

直觉上，这个准则好比在随机源的每位 0 或者 1 构成的一颗二叉树上进行搜索，每次选择期望值更大的分支，最终找到一个“好”的随机源。 我们首先介绍 MAX CUT 问题，给出求解 MAX CUT 的一个随机算法，然后介绍如何去随机化。

定义 7 （MAX CUT 问题） 给定一个图 $G = (V, E)$，图的割 $C = (S, T)$ 是对顶点 V 的一个划分，对应的割集是 $\mathrm{cut}(S, T) = \{(u, v) \in E \mid u \in S, v \in T\}$

MAX CUT 问题是给出图 G 的一个割，使得割集的大小最大

我们有一个简易的随机算法：对每个 $v_i \in V$，我们使用一个随机比特 r_i，如果 $r_i = 1$ 把他划分给 S，如果 $r_i = 0$ 则划分给 T，最后得到割 (S, T)。对于任意的 $(u, v) \in E$，u 和 v 在同一部分的概率是 $\frac{1}{2}$，因此期望的割集大小是 $\frac{1}{2}|E|$。

我们下面说明怎么使用条件期望法去随机化这个算法。 首先我们给所有点编号 $v_1, v_2, ..., v_n \in V$，我们依次决定将 v_i 划分给 S 还是 T。假设我们已经对确定了前 i 个随机比特，记 $S_i = \{v_j \mid j \leq i, r_i = 1\}$，$T_i = \{v_j \mid j \leq i, r_i = 0\}$ 表示当前部分的划分，$U = \{v_{\{i+1\}}, ...v_n\}$ 为未划分的顶点，于是：

$$E[\mathrm{cut}(S, T) \mid r_1, ..., r_i] = |\mathrm{cut}(S_i, T_i)| + \left|\{(u, v) \in E \mid u \in U_i \text{ 或者 } v \in U_i\}\right|$$

等式右边的第一项是已有的割集，第二项则是剩下的边如果至少有一个顶点还没被划分，则有 $\frac{1}{2}$ 的概率在割集中。 下面我们考虑 r_{i+1}，首先注意到，r_{i+1} 对第二项没有影响，因为无论如何 $U_{i+1} = U_i \setminus \{v_{\{i+1\}}\}$，因此对结果的贡献恒定为 $\frac{1}{2}$，于是有影响的只有第一项：

$$\left|\mathrm{cut}(S_{i+1}, T_{i+1})\right| = |\mathrm{cut}(S_i, T_i)| + \begin{cases} \left|\{(v_{i+1}, v) \mid v \in S_i\}\right| \text{ 如果 } r_{i+1} = 1 \\ \left|\{(v_{i+1}, v) \mid v \in T_i\}\right| \text{ 如果 } r_{i+1} = 0 \end{cases}$$

因此，条件期望可以被高效计算，我们能得到一个大小至少为 $\frac{1}{2}|E|$ 的割集。

4.6.4 伪随机生成器和 Nisan-Wigderson 构造

注意到，如果我们能将 BPP 所需的随机源长度减少到 $O(\log n)$，那么我们就能给 BPP 的所有问题去随机化。 于是一个思路是：能不能构造一个特殊的函数 G，我们仅使用长度 $\ell = O(\log n)$ 的真随机比特作为种子，就能生成一个“足够随机”的 $m = \mathsf{poly}(n)$ 长的伪随机源 $G(U_\ell)$?

于是存在一些根本性的界定问题，一个伪随机源什么时候可以被叫作“足够随机”：

- 信息论层面或者统计学测度上的，比如计算伪随机源的熵，比较它和理想的均匀分布之间的差，或者要求两两独立等。
- 柯尔莫戈洛夫复杂度层面上，比如考察伪随机源是否可被压缩性。
- 计算上的不可区分性，比如考察对于高效的算法是否能够区分伪随机源和真的均匀分布。

前两种方式并不适合伪随机生成器场景，我们在这里会采用第三种度量方式，并严格地定义什么不可区分性，并说明伪随机生成器。

定义 8 （伪随机生成器） 令 $\mathcal{C}$ 是一组运行时间至多为 S 的函数 $f:\{0,1\}^m \longrightarrow \{0,1\}$， 伪随机数生成器是一个函数 $G:\{0,1\}^\ell \longrightarrow \{0,1\}^m$，其中 ℓ 是生成器的种子长度。

我们说生成器 G 是一个 (S,ε)-伪随机生成器，如果对于任意的 $f \in \mathcal{C}$，有：

$$\left| \Pr_{s\in\{0,1\}^\ell}[f(G(s))=1] - \Pr_{r\in\{0,1\}^n}[f(r)=1] \right| < \varepsilon$$

函数族 $\mathcal{C}$ 的可能看起来有些奇怪，但是本质上是表征**非均匀**的计算模型，通常使用大小至多为 $O(S)$ 的布尔电路族或者要求长度至多为 S 建议字符串的图灵机。 至于采用非均匀计算模型的原因，我们可以回头看 $\mathsf{BPP} \subseteq \mathsf{P/Poly}$：任意 BPP 类语言 L 和对应的随机算法 $\mathcal{A}(x,r)$，可以对输入 x 和 随机源 r，在多项式时间内回答问题；对于某个固定的输入 x，实际上对应了一组非均匀的计算 $r \mapsto \mathcal{A}(x,r)$，如果 $\mathcal{A}$ 需要 $O(n^c)$ 的运行时间，可以证明 $r \mapsto \mathcal{A}(x,r)$ 可以被大小为 $O(n^c\ \mathsf{polylog}(n))$ 的布尔电路计算。 此外，我们并没有限制生成器 G 的运行时间，即使是指数级别的运行时间也是可以的。

伪随机生成器的定义实际上是在说明，任何高效的判别器 f 都不能区分生成器的输出和真实的均匀分布。 如果 BPP 类的语言 L 和对应算法 $\mathcal{A}(x,r)$ 具有双边错误概率 $\frac{1}{3}$，那么把真随机源 r 换成伪随机源 $G(s)$，我们有：

$$\left| \Pr_{s\in\{0,1\}^l}[\mathcal{A}(x,G(s))=1] - \Pr_{r\in\{0,1\}^n}[\mathcal{A}(x,r)=1] \right| < \varepsilon$$

$$\Longrightarrow \Pr_{s\in\{0,1\}^l}[\mathcal{A}(x,G(s)) \neq L(x)] < \frac{1}{3}+\varepsilon$$

因此，只需要 ε 足够小，算法 $\mathcal{A}$ 只需要长度为 ℓ 的随机源，运行在 $G\left(\{0,1\}^\ell\right)$ 上，就能取得和真随机源一样的效果，从而我们减少了所需的真随机源的长度。

总之，只要我们有 $\ell = O(\log n)$，我们就能有梦寐以求的去随机化：

引理 21 如果有 $(2^{\varepsilon\ell},\varepsilon)$-伪随机数生成器，则 $\mathsf{BPP}=\mathsf{P}$

伪随机生成器的概念和密码学生成器有密切的关系。 实际上，函数族/非均匀计算 $\mathcal{C}$ 表述了“区分 G 和真随机”这个问题是“难以（高效）计算”的，如果存在“计算困难的”函数，我们可以使用这个函数构造出伪随机生成器。 这是**平均复杂度类**和**密码学**里的重要概念。 有一系列定理从假设“计算困难”的函数出发，讨论去随机化的可能性：

定理 22 （Impagliazzo-Wigderson） 对于 $s:\mathbb{N}\longrightarrow\mathbb{N}$，如果存在 $f \in \mathsf{DTIME}\left(2^{O(\ell)}\right)$，使得对于每一个输入长度 $\ell \in \mathbb{N}$， f 对于非均匀计算时间 $s(\ell)$ 在最坏情况下困难的，如果有 $s(\ell)=2^{\Omega(\ell)}$，则 $\mathsf{BPP}=\mathsf{P}$

实际上，这就是在说存在 EXP 类的问题有较高的布尔电路复杂度。 许多 NP 完全问题，例如 SAT 问题，被普遍认为具有 $2^{\Omega(\ell)}$ 复杂度。

定理 23 （Nisan 1992） 如果随机算法使用了 R 长度的随机源与 S 大小的空间，可以显式地给出一个$\left(S, 2^{-S}\right)$ 伪随机生成器 $G:\{0,1\}^{O\left(S \log \left(\frac{R}{S}\right)\right)} \longrightarrow\{0,1\}^{R}$，且 G 可以在 $\mathsf{poly}(R, S)$ 时间和 $O(S \log R)$ 空间内计算。

特别地，在一台对数空间的图灵机上，使用 $O\left(\log ^{2} n\right)$ 长度的随机源，就可以生成多项式长度的伪随机数

定理 24 （Nisan-Wigderson 构造） 定义组合设计（combinatorial design）(ℓ, a) 为集合 $\{0,1, \ldots, d\}$ 的一个子集族 $\left\{S_{1}, \ldots, S_{m}\right\}$：

- $\forall i,\left|S_{i}\right|=\ell$
- $\forall i \neq j,\left|S_{i} \cap S_{j}\right| \leq a$

给定函数 $f:\{0,1\}^{\ell} \longrightarrow\{0,1\}$ 和 (ℓ, a) 设计 $S_{1}, \ldots, S_{m} \subset\{0,1, \ldots, d\}$，定义 Nisan-Wigderson 生成器为

$$
\begin{aligned}
G:\{0,1\}^{d} &\longrightarrow\{0,1\}^{m} \\
x &\mapsto f\left(x|_{S_{1}}\right) \cdot f\left(x|_{S_{2}}\right) \cdots f\left(x|_{S_{m}}\right)
\end{aligned}
$$

其中，$x|_{S_i}$ 表示 x 在下标集合 S_i 上的投影。 如果 f 是一个 $\left(s, \frac{1}{2}-\frac{\varepsilon}{m}\right)$ 平均困难复杂度的函数，那么 G 是一个 $(s-m \cdot a \cdot 2^{a}, \varepsilon)$ 伪随机生成器。

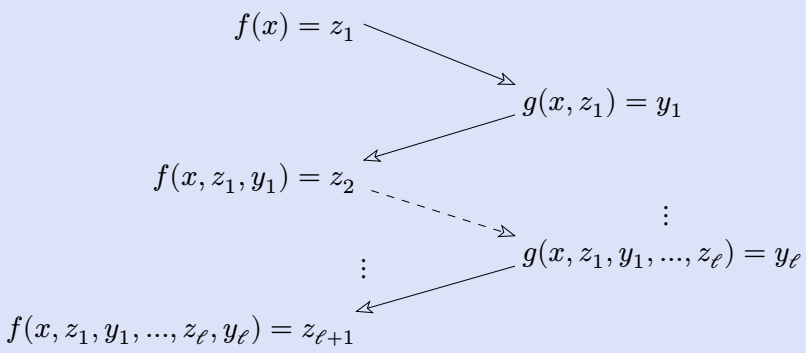
第五章 交互证明系统

NP 复杂度通常表示为一个验证器 V 对于输入 x 和证明 y 进行验证。但是在是实际中，人们通常使用的是更一般的交互式验证系统，也即，一方不断地提供一系列证明（被称为证明者，prover），另一方不断向前者要求给出证明并进行验证（被称为验证者，verifier），直到验证者被说服为止。

5.1 dIP 和 IP

我们首先考虑，把 NP 推广到交互式证明过程：

定义 1 （确定性函数的交互） 对于函数 $f, g : \{0,1\}^* \longrightarrow \{0,1\}^*$ 和整数 $k \geq 0$。输入 $x \in \{0,1\}^*$ 上的 f 与 g 之间 k 轮的交互，是一列如下定义的字符串 $z_1, y_1, ..., y_\ell, z_{\ell+1} \in \{0,1\}^*$，其中 $2\ell - 1 < k$：



我们把这一列字符串记作 $\langle f, g \rangle(x)$。其中，把 f 最后的输出 $f(x, z_1, y_1, ..., z_\ell, y_\ell) = z_{\ell+1}$ 记作 $\mathsf{out}_f\langle f, g \rangle(x)$ 并假设 $\mathsf{out}_f\langle f, g \rangle(x) \in \{0,1\}$

定义 2 （确定性交互证明系统） 一个语言 L 有一个 k **轮的确定性交互证明系统**，如果在输入 $x, z_1, y_1, ..., z_\ell, y_\ell$ 上存在运行时间为 $\mathsf{poly}(|x|)$ 的确定性图灵机 V，对于任意函数 P 都存在一个 k 轮的交互满足：

$$\text{（完备性）}\quad x \in L \Rightarrow \exists P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{out}_f\langle f, g \rangle(x) = 1$$

$$\text{（可靠性）}\quad x \notin L \Rightarrow \forall P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{out}_f\langle f, g \rangle(x) = 0$$

引理 1 $\mathsf{dIP} = \mathsf{NP}$

证明相当简单，$\mathsf{NP} \subseteq \mathsf{dIP}$ 是平凡的，我们只需证明 $\mathsf{dIP} \subseteq \mathsf{NP}$。如果 $L \in \mathsf{dIP}$，对于一个对于输入 x, 我们把确定性交互证明系统的交互序列 $(a_1, a_2, ..., a_k)$ 作为证书 y，构造一个

也即，使用确定性验证者的交互证明系统最多只需要一轮就能确定结果。使用确定性验证者的交互并没有获得更强的表达能力，我们考虑让验证者是**随机的**。形式化地，令 $V(r)$ 是具有一个私有的随机源 r 的验证者。

定义 3（带随机验证者的交互证明系统） 一个语言 L 有一个 k-轮的带随机验证者的交互证明系统，如果在输入 $x, z_1, y_1, \ldots z_\ell, y_\ell$ 上存在运行时间为 $\mathsf{poly}(|x|)$ 的概率图灵机 $V(r)$，且对于任意的函数 P 有：

$$\text{（完备性）}\quad x \in L \Rightarrow \exists P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{Pr}_{r\in\{0,1\}^{|x|^{O(1)}}}\left[\mathsf{out}_{V(r)}\langle V(r), P\rangle(x) = 1\right] \geq \frac{2}{3}$$

$$\text{（可靠性）}\quad x \notin L \Rightarrow \forall P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{Pr}_{r\in\{0,1\}^{|x|^{O(1)}}}\left[\mathsf{out}_{V(r)}\langle V(r), P\rangle(x) = 1\right] \leq \frac{1}{3}$$

和 BPP 复杂度类似（强化引理），这里的 $\frac{2}{3}$ 和 $\frac{1}{3}$ 只是习惯上的数字，可以对应加强到 $1 - 2^{-n^s}$ 和 2^{-n^s}，其中 $s > 0$ 是一个常数。

5.2 公共随机源与 AM 复杂度

在上面的定义中，随机验证者所需要的随机源（投掷硬币，random coins）是私有的，证明者无法看见。如果证明者能够访问验证者的随机源，就导出了带公共随机源的交互证明问题。我们把带公共随机源的交互证明系统称为 **亚瑟-梅林协议**[2]。此类交互证明系统复杂度类是 $\mathsf{IP}[k]$ 的一个子集，我们记作 $\mathsf{AM}[k]$。

那么，如何形式化地定义“公共随机源”？一种直观是，要只能把验证者的输出信息作为随机源，也就是亚瑟和梅林之间来回传递的信息包含了所有亚瑟所使用的随机猜测。我们使用 r_i 表示亚瑟发送的第 i 个随机字符串，

定义 4（AM） 我们定义复杂度类 $\mathsf{AM}[k] \subseteq \mathsf{IP}[k]$：语言 $L \in \mathsf{AM}[k]$，如果对于输入 x 和交互序列 $r_1, y_1, r_2, \ldots, r_\ell, y_\ell$ 且每个 r_i 都是随机均匀选择的，存在运行时间 $\mathsf{poly}(|x|)$ 的确定性的图灵机 A，对于任意函数 M 有：

$$\text{（完备性）}\quad x \in L \Rightarrow \exists P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{Pr}_{r\in\{0,1\}^{|x|^{O(1)}}}[\mathsf{out}_A\langle A, M\rangle(x) = 1] \geq \frac{2}{3}$$

$$\text{（可靠性）}\quad x \notin L \Rightarrow \forall P : \{0,1\}^* \longrightarrow \{0,1\}^*, \mathsf{Pr}_{r\in\{0,1\}^{|x|^{O(1)}}}[\mathsf{out}_A\langle A, M\rangle(x) = 1] \leq \frac{1}{3}$$

特别地，我们记作 $\mathsf{AM} = \mathsf{AM}[2]$。AM 复杂度类问题只有一对询问-回复：亚瑟先投掷硬币，然后把**所有**硬币投掷的结果告诉梅林，梅林给出一个证明，亚瑟只能依赖先前的硬币投掷结果和梅林的回答**确定性**地验证这个证明。类似地，我们还可以定义 $\mathsf{MA}[k]$ 复杂度类。$\mathsf{MA}[k]$ 复杂度类是**以梅林开始**的 k 轮的亚瑟-梅林协议。

AM 复杂度类有一些有趣的性质：

- （Babai-Moran）对于常数 $k \geq 2$，$\mathsf{AM}[k] = \mathsf{AM}[k+1] = \mathsf{MA}[k+1]$。更进一步，对于任意 $q \in \mathsf{poly}(|x|)$，有
$$\mathsf{AM}[2q] \subseteq \mathsf{AM}[q+1]$$
- （Goldwasser-Sipser）对于任意 $q \in \mathsf{poly}(|x|)$ 有
$$\mathsf{IP}[q] \subseteq \mathsf{AM}[q+2]$$

[2]这个名字来源于亚瑟王传说，亚瑟王的法师梅林具有强大的法力。而梅林虽然不能预测亚瑟王投掷硬币的结果，但是亚瑟王也无法向梅林隐藏先前的投掷结果，梅林可以提前模拟硬币投掷所有可能的结果并以最优的方式作出回答。“亚瑟-梅林协议”这个名字刻画了证明者可以具备的无限计算能力（谕示机）和梅林法力之间的类比。然而梅林并不一定是诚实的，所以亚瑟必须分析梅林针对亚瑟询问所提供的信息，并自行决定问题。

- $\mathsf{AM}[2] = \mathsf{BP} \cdot \mathsf{NP}$，由此可知 $\mathsf{AM}[2] \subseteq \Sigma_3^p$
- $\mathsf{MA}[2] = \mathsf{N} \cdot \mathsf{BPP}$
- $\mathsf{NP} \subseteq \mathsf{MA}$ 且 $\mathsf{BPP} \subseteq \mathsf{MA}$
- $\mathsf{MA} \subseteq \mathsf{PP}$

第六章 量子复杂度

6.1 量子力学基本原理

6.2 量子图灵机

量子图灵机的特殊之处在于其状态的表示。如前所述，量子力学的一个核心想法在于，状态是一个 Hilbert 空间[3]，而状态转移函数则是一个酉矩阵。对照图灵机的定义，首先，我们需要改写内部状态 Q 为一个 Hilbert 空间，并改写状态转移函数中的 $Q \to Q$ 的分量为一个酉矩阵；然后，需要两族符号，一族是内部纸带，其字母表为一个 Hilbert 空间 Σ_{int}；一族是输入输出纸带，其字母表依旧为原样，也就是说，仍然是经典的。注意，在这里，我们没有明确讨论何时进行测量的问题，这是一个非常重要的问题，但为了简单起见，我们只允许其在计算的最后进行测量，给出输出。Bernstein 和 Vazirani 表明了这样的方法的可靠性：可以把所有的测量都推迟到最后。

值得一提的是，目前并没有表明，量子图灵机具备某种意义上的泛用性，即不一定存在一个量子图灵机能够模拟所有物理过程。如果黑洞热力学的 Bekenstein 方程成立，即对于一个被面积为 A 的闭曲面所包围的空间，其中所包含的状态数不超过

$$N(A) = e^{\frac{Ac^3}{4\hbar G}}$$

则我们可以用这么多维数的线性空间完成通用图灵机的构造。

6.3 BQP 复杂度类

类似于概率图灵机的情形，我们定义复杂度类 $\mathsf{BQTIME}\ (f(n))$。称一个语言是 $\mathsf{BQTIME}\ (f(n))$ 的，如果对于任意输入规模 n，量子图灵机能够在 $f(n)$ 步内停机，且测量给出 1 和 0 的概率分别为 $\geq \frac{2}{3}$ 和 $\leq \frac{1}{3}$ 的。我们定义 $\mathsf{BQP} = \bigcup_{k\geq 1} \mathsf{BQTIME}\ (f(n))$。正如在概率图灵机中，我们设定随机源长度应该是多项式级别的，我们也设定状态转移函数的酉矩阵中的每个项都是在多项式时间内可计算的。一个很无聊的事实是，如果我们允许任意实数作为酉矩阵的项，则 BQP 就会包含任意语言，这和概率图灵机中允许随机源以任意实数概率产生某个结果一样。更有趣的是，Adleman、DeMarrais、Huang 和 Solovay 与姚期智分别同期证明了一下结论：

引理 1 限制 BQP 类中的酉矩阵中的元素全都是 $\left\{-1, -\frac{4}{5}, -\frac{3}{5}, 0, \frac{3}{5}, \frac{4}{5}, 1\right\}$ 中的某个数的酉矩阵不影响给出的定义。

与之相对的，我们不再能够很好地定义 ZQP 这样的复杂度类，因为对矩阵的近似就会使得我们跳出这个复杂度类；另外，空间上的限制也毫无意义，如果只允许**一次**测量，那么哪怕是抛硬币这样的过程也是无法模拟的，因为硬币不能被重用（量子比特不可复制）。这部分的讨论可见 J. Watrous, *Quantum simulations of classical random walks and undirected graph connectivity*。另外，我们还有：

定理 2 （Bernstein-Vazirani） $\mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$.

关于 BQP 复杂度类的很多讨论都相当复杂。我们借助 AWPP 复杂度类来讨论它的相关结果，其就是对 BQP 取消掉酉矩阵的限制之后的结果。显然，$\mathsf{BQP} \subset \mathsf{AWPP}$，而我们有：

[3]严格意义上来讲，是 rigged Hilbert space 或称 Gelfand triplet，不过我们暂时不会涉及谱的问题，所以将其视作 Hilbert 空间即可。

- $\mathsf{AWPP} \subset \mathsf{PP}$，因此 $\mathsf{BQP} \subset \mathsf{PP}$；
- AWPP 对 PP 是低（low）的，也就是说，对于任意 $L \in \mathsf{AWPP}$，都有 $\mathsf{PP}^L = \mathsf{PP}$；因此，BQP 对 PP 也是低的；
- 如果 $\mathsf{P} = \mathsf{PSPACE}$，那么对于任意谕示机 G 都有 $\mathsf{P}^G = \mathsf{AWPP}^G$。这意味着存在谕示机 A 使得 $\mathsf{P}^A = \mathsf{AWPP}^A$ 且 PH 分层无限；同理，存在谕示机 A 使得 $\mathsf{P}^A = \mathsf{BQP}^A$ 且 PH 分层无限；
- 存在谕示机 G 使得 AWPP^G 没有完全问题，甚至没有 BPP 难问题；同理，存在谕示机 G 使得 BQP^G 没有完全问题，也没有 BPP 难问题。