高级算法设计与分析 Lecture 3

授课时间: 2020 年 3 月 2 日 授课教师: 孙晓明 记录人: 聂均鸿

1 复杂性类回顾

1.1 将多比特输出的问题转化为判定问题

在讨论计算问题时，通常可以用二进制将其进行编码，于是该问题就对应一个函数 $f:\{0,1\}^n \to \{0,1\}^m$。而为了研究问题的计算复杂性，通常可以把问题对应的多比特输出函数 f 转化为若干个单比特输出函数（判定问题），即 $f_1, f_2, \ldots, f_m$，其中 $f_i:\{0,1\}^n \to \{0,1\}, \forall i \in \{1,\ldots,m\}$。

例 1 (Factoring) 设正整数 x 的二进制表示为 $(x_1x_2\ldots x_n)_2$，问题想要求正整数 y 和 z 使得 $x = yz$，即它们是 x 的一个因数分解。这是一个多比特输出问题，但可以通过类似如下二分的方式将它转化为若干个单比特输出问题：首先判定 "x 是否存在因子 $y <= 4$"，再判定 "x 是否存在因子 $y <= 8$"，"x 是否存在因子 $y <= 16$"，⋯⋯，直到找到 y 的可能范围，再在此范围内继续求解类似问题、缩小 y 的可能范围，最终确定因子 y。

例 2 (Graph coloring) 给定图 G，问题想要求解 G 的一个可行的 rgb-3 染色方案。它可以作如下转化：首先求解问题 "G 是否存在顶点 v_1 被染成 r（or g or b）的 3 染色方案"，再求解问题 "在顶点 v_1 已经被染成 r（or g or b）的情况下，G 是否存在顶点 v_2 被染成 r（or g or b）的 3 染色方案"，⋯⋯，直至所有顶点的颜色都被确定下来。

1.2 一个计算问题对应的语言

对于一个单比特输出的问题 $f:\{0,1\}^n \to \{0,1\}$，它对应的语言 L_f 为集合 $\{x|f(x)=1\}$，即全体使得问题回答 "Yes" 的输入串组成的集合。也就是说，f 是在判断 "输入 x 是否属于语言 L_f"。

例 3 问题 "x 是质数吗?" 对应的语言为 $L_{Prime} = \{x|x \text{ is a prime}\} = \{2,3,5,7,11,\ldots\}$。

例 4 合数判定问题对应的语言为 $L_{Composite} = \{4,6,8,9,\ldots\}$。

例 5 图的连通图判定问题对应的语言为 $L_{CONN} = \{G : G \text{ is connected}\}$。

1.3 P, NP, co-NP

复杂性类是在计算上具有特定性质的一类语言的集合。

1. P(Polynomial-time)：具有确定性多项式时间求解算法的语言的集合。例如：L_{CONN}，$L_{\overline{CONN}}$，L_{PM}（PM：Perfect Matching，图的完美匹配），$L_{\overline{PM}}$，这些集合都属于 P。
2. NP(Non-deterministic Polynomial-time)：称语言 L 属于语言类 NP，当且仅当对 L 存在多项式时间算法 A，L 的一个实例 x 是否属于 L 可以借助辅助证据 w 由 A 验证，即 $x \in L \iff \exists w, A(x,w)=1$。例如：$L_{G3C}$。
3. co-NP：co-NP $= \{\overline{L}|L \in \text{NP}\}$。称语言 L 属于语言类 co-NP，当且仅当对 L 存在多项式时间算法 A，$x \in L \iff \forall w, A(x,w)=1$。例如：$L_{\overline{G3C}}$。

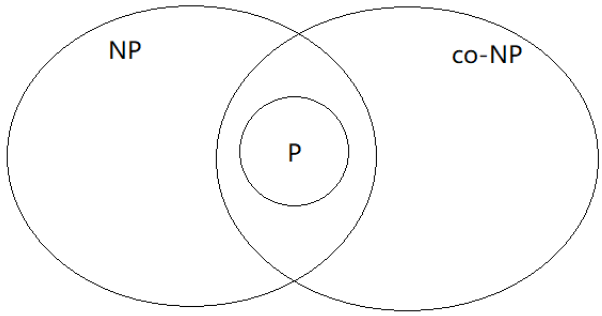

图 1: P、NP、co-NP

2 随机复杂性类

一个随机算法 A 除了接收输入串 x 外，还可以使用一些随机比特 r 来计算输出结果 $A(x,r)$。这些随机比特的引入可能使得算法的结果出错（RP、co-RP、BPP），也可能使得算法的运行时间变成一个随机变量（ZPP）。

2.1 RP、co-RP

2.1.1 RP(Randomized Polynomial-time)

称语言 L 属于 RP，当且仅当对 L 存在多项式时间随机算法 A，A 以 x 为输入，并使用若干随机比特 r，使得

$$x \in L \Longrightarrow \Pr_r(A(x,r)=1) \geq \frac{1}{2},$$
$$x \notin L \Longrightarrow \Pr_r(A(x,r)=0) = 1.$$

可以看出，类 RP 中的问题都存在单边错误（1-sided error）的多项式时间算法，而且这个算法不“轻易”回答“Yes”，比较“保守”。

定理 1. $RP \subseteq NP$。

该定理直接根据定义就可以证明。下面的定理说明了类 RP 定义中的 $\frac{1}{2}$ 可以改成任意小于 1 的正数 $1-\epsilon$，其中 ϵ 即算法的错误率。记由错误率为 ϵ 定义出的复杂性类为 RP_ϵ，

定理 2. $RP_\alpha = RP_\beta, \forall\ 0 < \alpha, \beta < 1$。

证明 以 $\alpha = 0.01, \beta = 0.99$ 为例证明。根据定义，显然 $\mathrm{RP}_{0.01} \subseteq \mathrm{RP}_{0.99}$。下证 $\mathrm{RP}_{0.99} \subseteq \mathrm{RP}_{0.01}$，即证 $\forall L \in \mathrm{RP}_{0.99} \Longrightarrow L \in \mathrm{RP}_{0.01}$。

首先由 $\mathrm{RP}_{0.99}$ 定义，$L \in \mathrm{RP}_{0.99}$ 意味着存在一个多项式时间随机算法 A，使得

$$x \in L \Longrightarrow \Pr_r(A(x,r)=1) \geq 0.01,$$
$$x \notin L \Longrightarrow \Pr_r(A(x,r)=0) = 1.$$

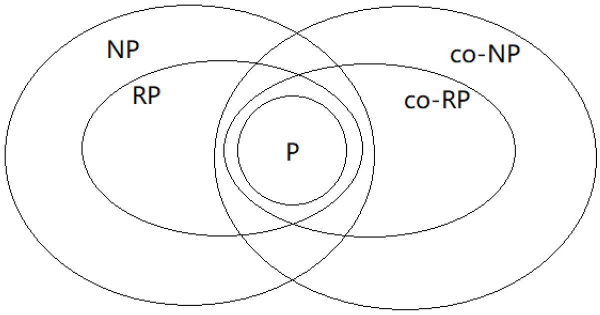

图 2: RP、co-RP

现在构造随机算法 $\tilde{A}$ 如下：$\tilde{A}$ 接收 x 作为输入，并使用 k 个独立随机比特串 $r_1, r_2, \ldots, r_k$（k 的大小待定），重复调用算法 A 得到 $Y_1 = A(x, r_1), Y_2 = A(x, r_2), \ldots, Y_k = A(x, r_k)$。若在输出 $Y_1, \ldots, Y_k$ 中存在 1，则算法输出 1；否则，算法输出 0。

显然，算法 $\tilde{A}$ 是一个多项式时间随机算法，且易知 $x \notin L \Longrightarrow \Pr_{r_1,\ldots,r_k}(\tilde{A}(x, r_1, r_2, \ldots, r_k) = 0) = 1$。下证 k 足够大时，可以使得 $x \in L, \Pr_{r_1,\ldots,r_k}(\tilde{A}(x, r_1, r_2, \ldots, r_k) = 0) \leq 0.01$。

$$\begin{aligned}
&\Pr_{r_1,\ldots,r_k}(\tilde{A}(x, r_1, r_2, \ldots, r_k) = 0)\\
=&\Pr_{r_1,\ldots,r_k}(A(x, r_1) = A(x, r_2) = \cdots = A(x, r_k) = 0)\\
=&\Pr_{r_1}(A(x, r_1) = 0)\Pr_{r_2}(A(x, r_2) = 0)\ldots\Pr_{r_k}(A(x, r_k) = 0)\\
\leq& 0.99^k.
\end{aligned}$$

其中第三行的等号成立是由于 $r_1, r_2, \ldots, r_k$ 的独立性。取 $k \geq 459$，则 $\Pr_{r_1,\ldots,r_k}(\tilde{A}(x, r_1, r_2, \ldots, r_k) = 0) \leq 0.01$。从而语言 L 存在多项式时间随机算法 $\tilde{A}$ 满足 $\mathrm{RP}_{0.01}$ 的条件。 □

2.1.2 co-RP

称语言 L 属于 co-RP，当且仅当对 L 存在多项式时间随机算法 A，A 以 x 为输入，并使用若干位随机比特 r，使得

$$\begin{aligned}
x \in L &\Longrightarrow \Pr_r(A(x, r) = 1) = 1,\\
x \notin L &\Longrightarrow \Pr_r(A(x, r) = 0) \geq \frac{1}{2}.
\end{aligned}$$

对比类 RP 的定义可以看出，co-RP $= \{\overline{L} | L \in RP\}$。另外，co-RP 中的语言 L 对应的算法不“轻易”回答“No”，相对比较“激进”。与定理 1、定理 2相似，对类 co-RP 有如下定理：

定理 3. *co-RP* $\subseteq$ *co-NP*。

定理 4. $co\text{-}RP_\alpha = co\text{-}RP_\beta, \forall 0 < \alpha, \beta < 1$。

2.2 BPP(Bounded-error Probabilistic Polynomial-time)

称语言 L 属于 BPP，当且仅当对 L 存在多项式时间随机算法 A，A 以 x 为输入，并使用若干位随机比特 r，使得

$$x \in L \Longrightarrow \Pr_r(A(x,r)=1) \geq \frac{2}{3},$$
$$x \notin L \Longrightarrow \Pr_r(A(x,r)=0) \geq \frac{2}{3}.$$

可以看出，类 BPP 中的问题都存在双边错误（2-sided error）的多项式时间算法，也就是 Monte Carlo 算法。下面这两个定理是显然的。

定理 5. $RP \subseteq BPP$。

定理 6. $co\text{-}RP \subseteq BPP$。

值得注意的是，一个有效的双面错误算法，无论对于 $x \in L$ 还是 $x \notin L$，错误率 ϵ 都应该严格小于 $\frac{1}{2}$，否则就和掷硬币或者瞎猜没区别了。与类 RP_ϵ 和类 co-RP_ϵ 一样，对于类 BPP 也可以定义错误率为 ϵ 的类 BPP_ϵ，其中 $0 < \epsilon < \frac{1}{2}$。下面的定理说明了 ϵ 的选取对 BPP 的定义来讲无关紧要。

定理 7. $BPP_\alpha = BPP_\beta, \forall 0 < \alpha, \beta < \frac{1}{2}$。

证明 不失一般性，我们证明 $\mathrm{BPP}_{0.01} = \mathrm{BPP}_{0.49}$。显然，$\mathrm{BPP}_{0.01} \subseteq \mathrm{BPP}_{0.49}$；下证 $\mathrm{BPP}_{0.49} \subseteq \mathrm{BPP}_{0.01}$。即证 $\forall L \in \mathrm{BPP}_{0.49} \Longrightarrow L \in \mathrm{BPP}_{0.01}$。

首先由 $\mathrm{BPP}_{0.49}$ 定义，$L \in \mathrm{BPP}_{0.49}$ 意味着存在一个随机多项式时间算法 A，使得

$$x \in L \Longrightarrow \Pr_r(A(x,r)=1) \geq 0.51,$$
$$x \notin L \Longrightarrow \Pr_r(A(x,r)=0) \geq 0.51.$$

现在构造随机算法 $\tilde{A}$ 如下：$\tilde{A}$ 接收 x 作为输入，并使用 $2k+1$ 个独立随机比特串 $r_1, r_2, \ldots, r_{2k+1}$，重复调用算法 A 得到 $Y_1 = A(x,r_1), Y_2 = A(x,r_2), \ldots, Y_{2k+1} = A(x, r_{2k+1})$。算法输出 $2k+1$ 个回答中占多数的结果。也就是说，设 $Y = \sum_{i=1}^{2k+1} Y_i$，若 $Y \geq k+1$，则算法输出 1；否则，算法输出 0。

显然，算法 $\tilde{A}$ 是一个多项式时间随机算法。当输入 $x \in L$ 时，对于任意 i，由于 $\mathbb{E}(Y_i) = \Pr(Y_i = 1) \geq 0.51$、$Var(Y_i) \leq \frac{1}{4}$ 和 Y_i 的独立性，

$$\mathbb{E}(Y) = \mathbb{E}(Y_1 + \cdots + Y_{2k+1}) = \sum_{i=1}^{2k+1} \mathbb{E}(Y_i) = (2k+1)\mathbb{E}(Y_1) \geq 0.51(2k+1),$$
$$Var(Y) = (2k+1)Var(Y_1) \leq \frac{1}{4}(2k+1).$$

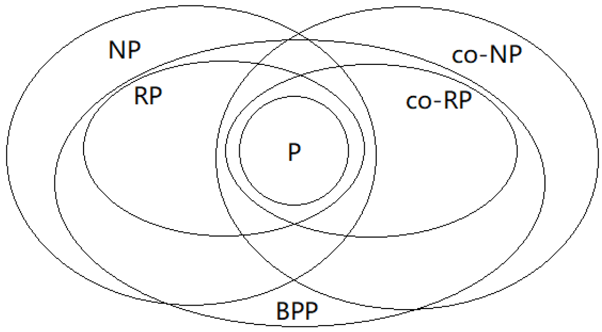

图 3: BPP

从而有

$$
\begin{aligned}
&\Pr_{r_1,\ldots,r_{2k+1}}(\tilde{A}(x,r_1,r_2,\ldots,r_{2k+1})=0) \\
=&\Pr(Y\le k) \\
=&\Pr(Y-\mathbb{E}(Y)\le k-\mathbb{E}(Y)) \\
\le&\Pr(Y-\mathbb{E}(Y)\le -0.02k-0.51) \\
\le&\Pr(|Y-\mathbb{E}(Y)|\ge 0.02k+0.51) \\
\le&\frac{Var(Y)}{(0.02k+0.51)^2} \\
\le&\frac{2k+1}{4(0.02k+0.51)^2}.
\end{aligned}
$$

其中第六行的不等号利用了 Chebyshev 不等式。取 $k\ge 124950$ 就可以使得 $\Pr_{r_1,\ldots,r_{2k+1}}(\tilde{A}(x,r_1,r_2,\ldots,r_{2k+1})=0)\le 0.01$，从而语言 L 存在随机多项式时间算法 $\tilde{A}$ 满足 $\mathrm{BPP}_{0.01}$ 的条件。 □

2.3 ZPP(Zero-error Probabilistic Polynomial-time)

上面介绍的复杂性类 RP、co-RP 和 BPP 共同的特点是其中的语言存在有错误率的算法，但它的运行时间是严格多项式时间，这也就是 Monte Carlo 算法。下面介绍的类 ZPP 对应的是 Las Vegas 算法，也就是算法能以概率 1 输出正确结果，但它只能保证期望运行时间是多项式时间，而对于某些 bad case，可能运行指数时间甚至不停机。

称语言 L 属于 ZPP，当且仅当对 L 存在期望运行时间为多项式时间的随机算法 A，A 以 x 为输入，并使用若干位随机比特 r，使得

$$x\in L \iff A(x,r)=1.$$

下面这个定理表明了类 ZPP 与 RP、co-RP 之间的关系，而它的证明过程也可以看作是提供了一种 ZPP 中语言对应的 Monte Carlo 算法和 Las Vegas 算法之间互相转化的方法。

定理 8. *$ZPP = RP\cap co\text{-}RP$。*

证明 首先证明 $\mathrm{ZPP} \subseteq \mathrm{RP} \cap \text{co-RP}$。下证 $\mathrm{ZPP} \subseteq \mathrm{RP}$，即证 $\forall L \in \mathrm{ZPP} \Longrightarrow L \in \mathrm{RP}$。

首先由 ZPP 定义，$L \in \mathrm{ZPP}$ 意味着存在一个期望运行时间是多项式时间的算法 A，使得

$$x \in L \iff A(x,r) = 1.$$

设算法 A 的期望运行时间为 $T(n)$，现在构造随机算法 $\tilde{A}$ 如下：$\tilde{A}$ 接收 x 作为输入，并使用随机比特串 r，只调用算法 $A(x,r)$ 一次，若 A 在 $10T(n)$ 步之内停机，则算法输出 $A(x,r)$；否则，算法输出 0。

显然，算法 $\tilde{A}$ 是一个随机的多项式时间算法。且易知 $x \notin L \Longrightarrow \Pr_r(\tilde{A}(x,r) = 0) = 1$。下证 $x \in L \Longrightarrow \Pr_r(\tilde{A}(x,r) = 1) \geq \frac{1}{2}$。

设算法 $\tilde{A}$ 某次调用 A 时实际运行时间为 T，则当 $x \in L$ 时，

$$\begin{aligned}
&\Pr_r(\tilde{A}(x,r) = 1) \\
=&1 - \Pr_r(\tilde{A}(x,r) = 0) \\
=&1 - \Pr(T > 10T(n)) \\
\geq&1 - \frac{T(n)}{10T(n)} \\
\geq&\frac{9}{10} > \frac{1}{2}.
\end{aligned}$$

其中第四行的不等号是对非负随机变量 T 使用了 Markov 不等式。

同理可证 $\mathrm{ZPP} \subseteq \text{co-RP}$。从而 $\mathrm{ZPP} \subseteq \mathrm{RP} \cap \text{co-RP}$。

下证 $\mathrm{RP} \cap \text{co-RP} \subseteq \mathrm{ZPP}$, 即证 $\forall L \in \mathrm{RP} \cap \text{co-RP} \Longrightarrow L \in \mathrm{ZPP}$。

根据类 RP 和 co-RP 的定义，$L \in \mathrm{RP} \cap \text{co-RP}$ 意味着存在一个多项式时间随机算法 A_1，A_1 以 x 为输入，并使用若干位随机比特 r，使得

$$\begin{aligned}
x \in L &\Longrightarrow \Pr_r(A_1(x,r) = 1) \geq \frac{1}{2}, \\
x \notin L &\Longrightarrow \Pr_r(A_1(x,r) = 0) = 1.
\end{aligned}$$

同时存在一个多项式时间随机算法 A_2，A_2 以 x 为输入，并使用若干位随机比特 r'，使得

$$\begin{aligned}
x \in L &\Longrightarrow \Pr_{r'}(A(x,r') = 1) = 1, \\
x \notin L &\Longrightarrow \Pr_{r'}(A(x,r') = 0) \geq \frac{1}{2}.
\end{aligned}$$

我们想要构造一个以概率 1 输出正确结果的算法 $\tilde{A}$, 它具有期望多项式运行时间。可以观察到，虽然算法 A_1, A_2 都不保证输出正确结果，但它们都只有 1-sided error。只要 A_1 输出 1，就一定有 $x \in L$（因为 A_1 比较“保守”）；同样地，只要 A_2 输出 0，就一定有 $x \notin L$（因为 A_2 比较“激进”）。通过这个规律，可以构造随机算法 $\tilde{A}$ 如下：

$\tilde{A}$ 接收 x 作为输入，它的第 i 轮产生两个随机比特串 r_i, r'_i、并且分别调用算法 $A_1(x,r_i), A_2(x,r'_i)$，若 $A_1(x,r_i)$ 输出 1，则算法输出 1；若 $A_2(x,r'_i)$ 输出 0，则算法输出 0；否则，算法进入第 $i+1$ 轮。

算法 $\tilde{A}$ 中每轮的分支都不会产生冲突，因为 $A_1(x,r_i) = 1$ 和 $A_2(x,r'_i) = 0$ 不可能同时发生。另外，根据上面的分析，$\tilde{A}$ 只要停机，就会以概率 1 输出正确结果。当然它可能永远不停机，但下面的分析表明它的期望运行时间仍是多项式时间。

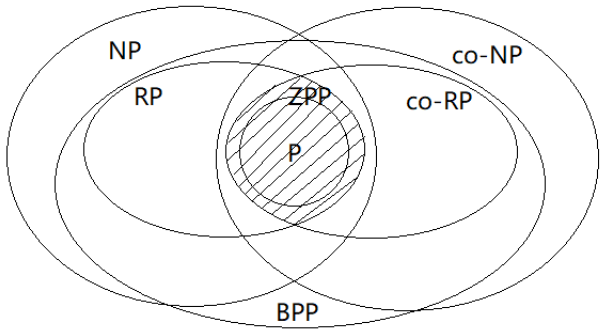

图 4: ZPP

分析样本空间可以知道,事件"$\tilde{A}$ 在第 i 轮停机"的概率 $\Pr(\text{``}\tilde{A}\text{ halts at i-th round''}) = \Pr_{r_i,r_i'}(A_1(x,r_i) = 1 \text{ or } A_2(x,r_i') = 0)$。当 $x \in L$ 时,

$$\begin{aligned}&\Pr_{r_i,r_i'}(A_1(x,r_i) = 1 \text{ or } A_2(x,r_i') = 0)\\ \geq &\Pr_{r_i}(A_1(x,r_i) = 1)\\ \geq &\frac{1}{2}.\end{aligned}$$

同样地，当 $x \notin L$ 时,

$$\begin{aligned}&\Pr_{r_i,r_i'}(A_1(x,r_i) = 1 \text{ or } A_2(x,r_i') = 0)\\ \geq &\Pr_{r_i'}(A_2(x,r_i') = 1)\\ \geq &\frac{1}{2}.\end{aligned}$$

设 $\tilde{A}$ 的运行时间为 T，A_1, A_2 的运行时间分别为 T_1, T_2，则 $\mathbb{E}(T) \leq 2O(T(A_1) + T(A_2))$，所以 $\tilde{A}$ 具有多项式的期望运行时间。 □

3 球盒模型

模型描述的是将 m 个同样的小球均匀随机的放入 n 个编号为 $1, 2, \ldots, n$ 的盒子中，讨论这 n 个盒子中感兴趣的小球分布情形的概率。这个模型有很多实际应用，例如:

- 生日悖论 (birthday paradox),
- 服务器的负载均衡 (load balancing),
- 礼券收集问题 (coupon collector)。