高级算法设计与分析 Lecture 2

授课时间: 2021 年 3 月 15 日 授课教师: 孙晓明 记录人: 资威

1 Birthday paradox

问题背景 按照鸽笼原理，要保证 n 个人中必定有两个人的生日同月同日，那么 n 至少要比 365 大，但是根据实际观察发现，在远少于 365 个人的时候，就会有两个人生日是同月同日，为什么会产生这样的悖论呢? 在这里将每个人的生日设置成均匀随机，更加通用的，让生日可以为 N 天中的任意一天(可以理解为不是在地球上的一年)，总共有 m 个人，下面的定理2表明只要有 $m \geq \sqrt{2N}+1$，就能让两个人同一天生日的概率大于 0.5，可以想象这里 N 总是比 m 大的。定理2的证明思路是：首先把计算存在两人生日是同一天的概率转化成任意两个人的生日都不在同一天的概率；再回忆上课时的实验，考虑每个人依次报出自己的生日，并且一直没有发现两个人的生日在同一天的概率该如何展开计算。

引理 1. 当 $x \in (0,1)$ 时，有 $e^{-x} \geq 1-x$。

证明

$$
\begin{aligned}
e^{-x} &= 1 - \frac{1}{1!}x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3 \dots \\
&= (1-x) + (\frac{1}{2!}x^2 - \frac{1}{3!}x^3) + (\frac{1}{4!}x^4 - \frac{1}{5!}x^5) \dots
\end{aligned}
$$

每个括号内都是 $(\frac{1}{2k!}x^{2k} - \frac{1}{(2k+1)!}x^{2k+1})$ 的结构，因为 $x \in (0,1)$，所以每个括号内都是正的，从而：

$$
\geq 1-x
$$

□

定理 2. 对于 m 个两两互相独立随机变量 $\Pr(x_i = j) = \frac{1}{N}$，$i = 1,2,\dots,m$; $j = 1,2,\dots N$。若是满足 $m \geq \sqrt{2N}+1$，则有 $\Pr(\exists\ x_i = x_j, i \neq j) > 1 - e^{-1} (> 0.5)$。

证明

$$
\begin{aligned}
&\Pr(\exists\ x_i = x_j, i \neq j) \\
&= 1 - \Pr(\forall i < j, x_i \neq x_j) \\
&= 1 - 1(1 - \frac{1}{N})(1 - \frac{2}{N}) \dots (1 - \frac{m-1}{N})
\end{aligned}
$$

根据引理 1：

$$
\begin{aligned}
&\geq 1 - e^{-(\frac{1}{N} + \frac{2}{N} + \dots + \frac{m-1}{N})} \\
&= 1 - e^{-\frac{(m-1)m}{2N}}
\end{aligned}
$$

令 $m \geq \sqrt{2N}+1$

$$
\geq 1 - e^{-1}
$$

□

2 随机算法减少通信复杂度

问题描述 有两个文本 x 和 y，使用二进制串表示，这两个文本分别在两台计算机中，我们记为 Alice 和 Bob，设这两台计算机都有足够使用的计算能力，现在需要判定两个文本 x，y 是否相等，复杂性的定义为 Alice 和 Bob 所需要传递的 bit 数量，我们将这个量称作通信复杂度。

确定性算法的下界 可以想象若是只使用确定性算法，Alice 和 Bob 之间至少需要交换 n 个 bit，其中 $n = \min\{|x|, |y|\}$，才能确定 x 和 y 是否相等，从阶上来说，确定性算法的通信复杂度有下界 $\Omega(n)$。

随机算法 现在给出一个能解决这个问题的随机算法1，这里不失一般性的假设 $|x| = |y|$，因为可以在最开始互相交换两个文本的长度，这一步只需要 $O(\log n)$ 的通信复杂度，不改变整体算法复杂度的阶。

Algorithm 1: Equality Test

Input: 两个二进制串 x 和 y，并且 $n = |x| = |y|$，两个二进制串分别属于 Alice 和 Bob。

Output: 若是 $x = y$ 则输出 1，反之输出 0。

1 Alice 选取一个素数 $p \in (n^2, 2n^2)$ (见 Chebyshev's theorem)。

2 Alice 和 Bob 都根据自己的文本构建两个多项式 $f_x(z) = \sum_{i=1}^{n} x_i z^i$ 和 $f_y(z) = \sum_{i=1}^{n} y_i z^i$。

3 Alice 随机选择一个正整数 $z_0 \in [1, p-1]$。并且计算 $m = f_x(z_0) \mod p$ 和 z_0 一起送给 Bob。

4 Bob 根据拿到的 z_0 计算 $f_y(z_0) \mod p$ 若是和 m 相等则输出 1 反之输出 0。

正确性与复杂度分析 首先说明为什么这个算法是对的，考虑当 $x = y$ 的时候，这个算法显然以 100% 的概率输出 1，当 $x \neq y$ 的时候，这个算法有一定的概率错误的输出 1，现在我们来计算这个概率是多大。

当 $x \neq y$ 的时候，算法错误的输出 1 代表此时有 $f_x(z_0) = f_y(z_0) \mod p$，我们设 $g(z) = f_x(z) - f_y(z)$，因为 $x \neq y$ 我们有 $g(z) \neq 0$，并且 $\deg g(z) \leq n$。现在我们有 $g(z_0) = 0 \mod p$，由于 $g(z) = 0 \mod p$ 至多有 $\deg g(z)$ 个解 (Lagrange's theorem (number theory))，而我们的 z_0 是从范围至少为 n^2 个整数中随机选取的，因此刚好选到一个根的概率不超过 $\frac{1}{n}$，也就是错误概率不超过 $\frac{1}{n}$。

现在分析 Alice 给 Bob 传递了多少个 bit。Alice 送了 z_0 和 $m = f_x(z_0) \mod p$ 两个数过去，可以看到这两个数都不超过 p，又因为 $p \in (n^2, 2n^2)$，所以使用 $2\log n + 1$ 个 bit 就能表达整数 p，从而通信复杂度有上界 $O(\log n)$。

3 概率基础

符号约定 随机变量 (简写为 $r.v.$) 通常使用如 X 和 Y 等大写英文字母表示，$\mathbb{E}$ 表示期望，Var 表示方差，σ 表示标准差，$c \in \mathbb{R}$ 表示常数。

期望的线性性

$$\mathbb{E}(X + Y) = \mathbb{E}(X) + \mathbb{E}(Y)$$

$$\mathbb{E}(cX) = c\mathbb{E}(X)$$

方差的定义与性质

$$Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2) = \mathbb{E}(X^2) - \mathbb{E}^2(X)$$

$$Var(X + Y) = Var(X) + Var(Y) + 2Cov(X, Y)$$

$$Cov(X, Y) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)$$

注意，当 X 和 Y 互相独立的时候有 $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$。
标准差：$\sigma = \sqrt{Var}$。

4 Markov Inequality

定理 3. 设 $r.v.$ $X \geq 0$，则对 $\forall\, c > 0$，有

$$\Pr(X \geq c) \leq \frac{\mathbb{E}(X)}{c}.$$

证明 假设 X 是连续型随机变量（离散型更易证明）

$$\begin{aligned}
\mathbb{E}(X) &= \int_0^{+\infty} x f(x) dx \\
&\geq \int_c^{+\infty} x f(x) dx \\
&\geq \int_c^{+\infty} c f(x) dx \\
&= c \int_c^{+\infty} f(x) dx = c \Pr(X \geq c)
\end{aligned}$$

□

作业 设随机变量序列 $X_1, X_2, \ldots$ 互相独立，与 X 同分布，$\Pr(X = 1) = p$，$\Pr(X = 0) = 1 - p$。定义 T 为第一次出现 $X_t = 1$ 时的实验次数，即 $T = \min\{t | X_t = 1\}$。试证 $\mathbb{E}(T) = \frac{1}{p}$。

5 Chebyshev Inequality

定理 4. 设 $r.v.$ X，则对 $\forall\, c > 0$，有

$$\Pr(|X - \mathbb{E}(X)| \geq c) \leq \frac{Var(X)}{c^2}.$$

证明

$$\begin{aligned}
\Pr(|X - \mathbb{E}(X)| \geq c) &= \Pr((X - \mathbb{E}(X))^2 \geq c^2) \\
&\leq \frac{\mathbb{E}((X - \mathbb{E}(X))^2)}{c^2} \quad \text{(根据 Markov Inequality)} \\
&= \frac{Var(X)}{c^2}
\end{aligned}$$

□

例 0 设随机变量序列 $Y_1, Y_2, \dots Y_n$ 相互独立，与 Y 同分布，$\Pr(Y=0) = \Pr(Y=1) = 1/2$，定义 $X = Y_1 + Y_2 + \cdots + Y_n$，我们试图估计 $\Pr(|X - \frac{n}{2}| \geq 5\sqrt{n})$ 的值。因为其独立性，我们有：

$$\mathbb{E}(X) = \sum_{i=1}^{n} \mathbb{E}(Y_i) = n/2$$

$$Var(X) = \sum_{i=1}^{n} Var(Y_i) = n/4$$

由 Chebyshev Inequality 有 $\Pr(|X - \frac{n}{2}| \geq c) \leq \frac{n}{4c^2}$。若取 $c = 5\sqrt{n}$, 则有 $\Pr(|X - \frac{n}{2}| \geq 5\sqrt{n}) \leq 1\%$，注意这里取的 c 和 X 的标准差在同一量级。

6 Quicksort

快速排序算法具体是怎么运行的这里默认大家知道。这里只强调一个地方，即选择 pivot 的时候使用均匀随机选择，这样的快排是一个随机算法。现在分析它的时间复杂度，我们以一次比较作为一个计算步骤，时间复杂度定义为算法运行过程中的比较次数，定义长度为 n 的数组，使用快排排序所需的比较次数为 $T(n)$，注意 $T(n)$ 是一个随机变量，设第一个 pivot 为数组中的第 K 个数，则有：

$$T(n) = n - 1 + T(K-1) + T(n-K)$$

这里的 K 依然是个随机变量，式子右边的 $n-1$ 代表按照 pivot 分成两个集合所需要的比较次数。则我们想要得到：

$$\begin{aligned}\mathbb{E}(T(n)) &= n - 1 + \sum_{i=1}^{n} \Pr(K=i)(\mathbb{E}(T(i-1)) + \mathbb{E}(T(n-i))) \\ &= n - 1 + \frac{2}{n}\sum_{i=0}^{n-1} \mathbb{E}(T(i))\end{aligned}$$

两边同乘 n：

$$n\mathbb{E}(T(n)) = n(n-1) + 2\sum_{i=0}^{n-1} \mathbb{E}(T(i))$$

令 n 为 $n-1$：

$$(n-1)\mathbb{E}(T(n-1)) = (n-1)(n-2) + 2\sum_{i=0}^{n-2} \mathbb{E}(T(i))$$

两式相减得到：

$$\begin{aligned}n\mathbb{E}(T(n)) &= (n+1)\mathbb{E}(T(n-1)) + 2n - 2 \\ \frac{\mathbb{E}(T(n))}{n+1} &= \frac{\mathbb{E}(T(n-1))}{n} + \frac{4}{n+1} - \frac{2}{n}\end{aligned}$$

递归的进行类似的展开：

$$
\begin{aligned}
&= \frac{\mathbb{E}(T(n-2))}{n-1} + \frac{4}{n} - \frac{2}{n-1} + \frac{4}{n+1} - \frac{2}{n} \\
&\vdots \\
&= \frac{\mathbb{E}(T(1))}{2} + \sum_{i=2}^{n} \frac{4}{i+1} - \sum_{i=2}^{n} \frac{2}{i} \\
&= 2\sum_{i=2}^{n} \frac{1}{i} + \frac{2-2n}{n+1} \\
&\le 2\int_{1}^{n} \frac{1}{x}\mathrm{d}x + \frac{2-2n}{n+1} \\
&< 2\ln n
\end{aligned}
$$

因此 $\mathbb{E}(T(n)) \approx 1.39n\log_2 n$

7 Matrix Multiplication Verification (MMV)

问题背景与定义 我们知道两个 $n \times n$ 矩阵 A, B 相乘，使用朴素的算法，其复杂度为 $O(n^3)$，这里一步代表一次实数乘法或加法，现有最好的确定性算法能达到 $O(n^{2.37268})$ 的复杂度，这一段改进的历史和最新结果的介绍可以从一篇公众号文章中了解：精炼镭射方法与快速矩阵乘法。我们现在所考虑的矩阵乘积验证问题（Matrix Multiplication Verification, MMV）相对矩阵乘法有些许的差别：

定义 5. *Matrix Multiplication Verification(MMV)*：输入 3 个 $n \times n$ 的矩阵 A, B, C，矩阵元素和其间的运算在素数 p 的同余类 $\mathbb{Z}_p$ 上，判定是否 $AB = C$。

需要注意的是，只要我们 p 取得足够大，那么把计算限定在同余类上并不会让问题变简单；同时研究表明这里加上一个用于比较的答案矩阵 C 之后并不会减少这个问题的难度 (若是只使用确定性算法)，或者说确定性算法想要判定 MMV 问题所需的复杂度也是 $O(n^{2.37268})$，而随机算法2能把复杂度降低到 $O(n^2)$

Algorithm 2: MMV

Input: 3 个 $n \times n$ 的矩阵 A, B, C。

Output: 若是 $AB = C$ 则输出 1，反之输出 0。

1 均匀随机地取向量 $\vec{x} \in \mathbb{Z}_p^n$。

2 计算 $AB\vec{x}$ 和 $C\vec{x}$。

3 若两者相同则输出 1，反之输出 0。

复杂性分析 计算 $AB\vec{x}$ 和 $C\vec{x}$ 需要 $O(n^2)$ 次基础运算，注意这里需要从右往左计算，比较两个结果向量需要 $O(n)$ 次基础运算，所以复杂度有上界 $O(n^2)$。

正确性分析 当 $AB = C$ 时，对任意 $\vec{x}$ 算法总是得到 $AB\vec{x} = C\vec{x}$，故输出 1；当 $AB \neq C$ 时，存在 $\vec{x}$ 满足 $AB\vec{x} = C\vec{x}$，算法有概率随机选择这样的 $\vec{x}$ 最终错误地输出 1，下面分析出现该错误的概率：

设 $D = AB - C$，由于 $D \neq 0$，故 D 中至少有一个元素不为零，不妨设其为 d_{11}，有

$$
\begin{aligned}
&\Pr(AB\vec{x} = C\vec{x}) \\
=&\Pr(D\vec{x} = 0) \\
=&\Pr\left((d_{11}x_1 + d_{12}x_2 + \cdots + d_{1n}x_n = 0) \wedge \ldots \wedge (d_{n1}x_1 + \cdots + d_{nn}x_n = 0)\right) \\
&\text{因为 } d_{11} \text{ 不为 0，我们放缩到下式不是平凡的。} \\
\leq&\Pr(d_{11}x_1 + d_{12}x_2 + \cdots + d_{1n}x_n = 0) \\
=&\Pr(x_1 = -d_{11}^{-1}(d_{12}x_2 + \cdots + d_{1n}x_n)) \\
&\text{只有一个 } x_1 \text{ 的取值能让等式相等。} \\
=&1/p
\end{aligned}
$$

综上 $\Pr(output = 1 | AB \neq C) \leq 1/p$，$\Pr(output = 0 | AB = C) = 0$，该算法是单边错误算法。证明中非 0 元的位置之所以不重要，是因为我们总可以放缩的时候放缩到拥有非 0 元的那一行。同时可以注意到，因为这个算法式单边错误的，所以可以通过重复这个算法 k 次使得 $\Pr(all\ output = 1 | AB \neq C) \leq (1/p)^k$。

8 复杂性

一个判定问题，代表着对于这个问题的任何输入，输出只有两个“是”或者“非”。或者说一个判定问题 $f(x)$，对于任何输入 x(注意所有输入都能被编码成字符串 x)，有 $f(x) \in \{0, 1\}$，把所有令 $f(x) = 1$ 的 x 收集起来放在一个集合 L 里，这个集合 L 就被定义为这个判定问题的语言，举几个例子：

1. 连通图对应的语言：$CONN = \{G | G\text{是连通图}\}$
   非连通图对应的语言：$\overline{CONN} = \{G | G\text{不是连通图}\}$

2. 有完美匹配的图对应的语言：$PerfectMatching = \{G | \text{图 } G \text{ 中有完美匹配}\}$
   无完美匹配的图对应的语言：$\overline{PerfectMatching} = \{G | \text{图 } G \text{ 中没有完美匹配}\}$

3. 可 3 染色图对应的语言：$G3C = \{G | \text{图 G 能够被 3 染色}\}$
   不可 3 染色图对应的语言：$\overline{G3C} = \{G | \text{图 G 不能够被 3 染色}\}$

4. 有 Hamilton 回路的图对应的语言：$HC = \{G | \text{图 } G \text{ 中有 Hamilton 回路}\}$
   无 Hamilton 回路的图对应的语言：$\overline{HC} = \{G | \text{图 } G \text{ 中没有 Hamilton 回路}\}$

5. $3SAT = \{\phi | \phi\text{是合取范式，每个子句中只有三个文字，且}\phi\text{可满足}\}$；
   $\overline{3SAT} = \{\phi | \phi\text{是合取范式，每个子句中只有三个文字，且}\phi\text{不可满足}\}$

注意一个判定问题的语言包含了这个判定问题的所有信息，因此可以把语言理解成判定问题自身。

1. $\mathcal{P}$(Polynomial time)：图灵机可在多项式时间内判定的语言组成的集合。可简单理解为经典计算模型能在多项式时间内解决的判定问题集合。后面的定义中的“算法”严格来说都应该使用“图灵机”表述。

2. $\mathcal{NP}$(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 $\forall\ G \in G3C$，存在 G 的一个 3 染色方案，将其一同输入给算法 A 可以用多项式时间验证该染色方案确实对 G 合法，而对 $\forall G \notin G3C$，则任何 3 染色方案都不合法；再如 $\forall \phi \in 3SAT$，存在 ϕ 的一个成真赋值，将其一同输入给算法 A 可以用多项式时间验证该赋值确实使 ϕ 为真，而对 $\forall \phi \notin 3SAT$，任何赋值都不可使 ϕ 为真。所以 $G3C$ 和 $3SAT$ 都属于 NP。

8.1 随机复杂性类

1. $\mathcal{RP}$(Randomized Polynomial time)：语言 $L \in$ RP 当且仅当存在随机多项式时间算法 A, 对 L 的符合实例 $x \in L$, 随机数 r, $\Pr(A(x,r) = 1) \geq \frac{1}{2}$；对实例 $x \notin L$, 随机数 r, $\Pr(A(x,r) = 1) = 0$。

2. co-$\mathcal{RP}$：$L \in$ co-RP 当且仅当存在随机多项式时间算法 A，对 L 的符合实例 $x \in L$，随机数 r，$\Pr(A(x,r) = 0) = 0$；对实例 $x \notin L$，随机数 r，$\Pr(A(x,r) = 0) \geq \frac{1}{2}$。

3. $\mathcal{BPP}$(Bounded-error Probabilistic Polynomial time)：$L \in$ BPP 当且仅当存在随机多项式时间算法 A，对 L 的符合实例 $x \in L$，随机数 r，$\Pr(A(x,r) = 1) \geq \frac{2}{3}$；对实例 $x \notin L$，随机数 r，$\Pr(A(x,r) = 1) \leq \frac{1}{3}$。

按照定义，$MMV \in$ co-RP，而 $\overline{MMV} \in$ RP。