Ling Li, ling@cs.caltech.edu
February 2, 2001

11.1 Pseudo-random generator

For any integer $k \geq 1$, define

$$A^k = \underbrace{A \circ A \circ \cdots \circ A}_{\text{the number of } A\text{'s is } k}$$

Since $A : \{0,1\}^m \to \{0,1\}^{m+1}$ is a pseudo-random generator, the complexity of A is bounded by a polynomial $|x|^t$. Note here we always use m as the length of the input. Thus A^k maps $\{0,1\}^m$ to $\{0,1\}^{m+k}$, and the complexity of A^k (for $k \leq m^c - m$) is bounded by

$$m^t + (m+1)^t + \cdots + (m+k-1)^t \leq k \cdot (m+k)^t \leq m^{c(t+1)}.$$

Thus $A^k : \{0,1\}^m \to \{0,1\}^{m+k}$, especially $A^{m^c-m} : \{0,1\}^m \to \{0,1\}^{m^c}$ is poly-time computable.

For any PPT T, we know $T \circ A^k$ for $k \leq m^c - m$ is also a PPT. Here for convenience, we define $T \circ A^k = T$ for $k = 0$. Thus for any $k \leq m^c - m$ and $T \circ A^k$, since A is a pseudo-random generator, there exists a negligible function v_k such that

$$\left| P_{x \leftarrow U_m}(T \circ A^k(A(x)) = 1) - P_{x \leftarrow U_{m+1}}(T \circ A^k(x) = 1) \right| < \nu_k(m).$$

Thus we have

$$\begin{aligned}
&\left| P_{x \leftarrow U_m}(T(A^{m^c-m}(x)) = 1) - P_{x \leftarrow U_{m^c}}(T(x) = 1) \right| = \\
&\qquad \left| \sum_{k=0}^{m^c-m-1} \Big(P_{x \leftarrow U_{m^c-k-1}}(T \circ A^k(A(x)) = 1) - P_{x \leftarrow U_{m^c-k}}(T \circ A^k(x) = 1) \Big) \right| \leq \\
&\qquad \sum_{k=0}^{m^c-m-1} \left| P_{x \leftarrow U_{m^c-k-1}}(T \circ A^k(A(x)) = 1) - P_{x \leftarrow U_{m^c-k}}(T \circ A^k(x) = 1) \right| \leq \\
&\qquad\qquad \sum_{k=0}^{m^c-m-1} \nu_k(m^c - k - 1).
\end{aligned}$$

It is easy to see that $\sum_{k=0}^{m^c-m-1} \nu_k(m^c-k-1)$ is also a negligible function of m. Thus for $x \in \{0,1\}^m$, $A^{m^c-m}(x)$ is indistinguishable U_{m^c}. Together with A^{m^c-m} is poly-time computable, we proved $A^{m^c-m} : \{0,1\}^m \to \{0,1\}^{m^c}$ is a pseudo-random generator.

Ling Li, ling@cs.caltech.edu

11.2 Pairwise independence

Without loss of generality, to prove that $X_1, \ldots, X_n$ are pairwise independent, we can just prove that X_1 and X_2 are independent. Since $X_1, \ldots, X_n$ are independent, we have

$$P\left(\bigwedge_{i=1}^{n} X_i = x_i\right) = \prod_{i=1}^{n} P(X_i = x_i).$$

Thus for any possible value x_1 of X_1 and x_2 of X_2,

$$\begin{aligned}
P(X_1 = x_1 \wedge X_2 = x_2) &= \sum_{x_3,\ldots,x_n} P\left(\bigwedge_{i=1}^{n} X_i = x_i\right) && (1)\\
&= \sum_{x_3,\ldots,x_n} \prod_{i=1}^{n} P(X_i = x_i) && (2)\\
&= P(X_1 = x_1) P(X_2 = x_2) \prod_{i=3}^{n} \sum_{x_i} P(X_i = x_i) && (3)\\
&= P(X_1 = x_1) P(X_2 = x_2). && (4)
\end{aligned}$$

Here the summation in (1) is over all possible values of $X_3, \ldots, X_n$; (2) is due to the independence of $X_1, \ldots, X_n$; we get (3) by distributivity; since the summation of $P(X_i = x_i)$ is over all possible values of X_i, we have $\sum_{x_i} P(X_i = x_i) = 1$. Thus we finally have (4), implying X_1 and X_2 are independent.

So $X_1, \ldots, X_n$ are pairwise independent.

11.3 Number of empty bins

Here is a more general version of the problem. Let k $(1 \le k \le n)$ balls be independently and uniformly tossed into n bins. They might occupy m bins, where $1 \le m \le k$. Let $p_n(k,m)$ denote the possibility that the number of bins occupied by k balls is m. Obviously, $p_n(k,0) = 0$ and for $m > k$, $p_n(k,m) = 0$. And, $\int_{m=1}^{k} p_n(k,m) = 1$.

Consider an event that $(k+1)$ balls occupy m bins, $m \le k+1 \le n$. For the $(k+1)th$ ball, it may be tossed into an occupied bin, or an empty bin. If it is tossed into an occupied bin, then the first k balls must have already occupied m bins, and the probability of such case is $\frac{m}{n} p_n(k,m)$. Otherwise, it is tolled into an empty bin, then the first k balls only occupied $(m-1)$ bins, and the probability for this case is $\left(1 - \frac{m-1}{n}\right) p_n(k,m-1)$. So we get

$$p_n(k+1,m) = \frac{m}{n} p_n(k,m) + \left(1 - \frac{m-1}{n}\right) p_n(k,m-1). \tag{5}$$

We are interested in the expectation of the number of bins that *occupied* by k balls. Let $F_n(k)$ denote the expectation, i.e.,

$$F_n(k) = \sum_{m=1}^{k} m \cdot p_n(k,m).$$

By (5), we have

$$\begin{aligned}
F_n(k+1) &= \sum_{m=1}^{k+1} m \cdot p_n(k+1,m) \\
&= \sum_{m=1}^{k+1} m \cdot \left[\frac{m}{n} p_n(k,m) + \left(1 - \frac{m-1}{n}\right) p_n(k,m-1)\right] \\
&= \sum_{m=1}^{k+1} \left[\frac{m^2}{n} p_n(k,m) - \frac{(m-1)^2}{n} p_n(k,m-1)\right] + \sum_{m=1}^{k+1} \left(m - \frac{m-1}{n}\right) p_n(k,m-1) \\
&= 0 + \sum_{m=1}^{k} p_n(k,m) + \left(1 - \frac{1}{n}\right) \sum_{m=1}^{k} m \cdot p_n(k,m) \\
&= 1 + \left(1 - \frac{1}{n}\right) F_n(k).
\end{aligned}$$

It is easy to see that for one ball, $F_n(1) = 1$. Thus

$$F_n(k) = 1 + \left(1 - \frac{1}{n}\right) + \cdots + \left(1 - \frac{1}{n}\right)^{k-1} = \frac{1 - \left(1 - \frac{1}{n}\right)^k}{1 - \left(1 - \frac{1}{n}\right)} = n - n\left(1 - \frac{1}{n}\right)^k.$$

So for k balls, the expectation of the number of *empty* bins is

$$E_n(k) = n - F_n(k) = n\left(1 - \frac{1}{n}\right)^k.$$

Especially, for n balls, the expectation is $n\left(1 - \frac{1}{n}\right)^n$, which approaches n/e in the limit of large n, since we have

$$\lim_{n \to +\infty} \left(1 - \frac{1}{n}\right)^n = e^{-1}.$$

Ling Li, ling@cs.caltech.edu

11.4 NP $\subseteq$ BPP? NP = RP?

Assume **NP** $\subseteq$ **BPP**. For any language $L \in$ **NP**, the corresponding poly-time verification procedure V accepts the pair (x, y) for at least one "witness" y, if $x \in L$. Thus for any x, we can define a language

$$L_x = \{y' : y' \text{ is a prefix of } y \text{ and } V \text{ accepts } (x, y)\}.$$

Thus $L_x \in$ **NP**, since if $y' \in L_x$, then the corresponding y is such a "witness". So, by our assumption, $L_x \in$ **BPP**.

From homework 10.4, we know there exists a PPT A' for L_x such that for $y' \in L_x$, $P(A'(y') = 1) \geq 1 - e^{-k}$, and for $y' \notin L_x$, $P(A'(y') = 1) < e^{-k}$, where $k = |y'|$ is the length of the input. And further we can design another PPT A by running A' for polynomial times of $|x|$ and using the majority vote to decide the output, such that the probability of error is less than e^{-n}, where n is also a polynomial of $|x|$.

For $x \in L$, since V is poly-time computable, the length of y is bounded by a polynomial $n = |x|^c$. We want to use A to guess the (at most n) bits of y. At first, we try 0 and 1 as the first bit of y. That is, we run $A(y_1 = 0)$ and $A(y_1 = 1)$. If either run gives 1, take the 'right' bit (for which A returns 1) as y_1 and we can continue to guess y_2. If both runs give 0, we then check whether we have already obtained the whole y, by running the verification procedure V on the obtained bits of y. Of course, when guessing y_1 we have no obtained bits of y and V will definitely reject such input. However, this is useful when the other bits of y is under guessing. If the pair of x and the obtained bits of y is accepted, then x is accepted as $x \in L$. Otherwise reject x as $x \notin L$. This procedure continues until x is rejected or accepted or y_{n+1} is reached (thus we reject x).

For $x \in L$, the above procedure runs in poly-time of $|x|$. And the probability that it declares $x \in L$ is at least

$$(1 - e^{-n})^n > 1 - ne^{-n} > \frac{1}{2}.$$

For $x \notin L$, since we can not find a y that can be accepted by V, x will definitely be rejected by our procedure. Thus $L \in$ **RP**, and **NP** $\subseteq$ **RP**.

We have already known that **RP** $\subseteq$ **NP**. Thus from assuming **NP** $\subseteq$ **BPP** we get **NP** = **RP**.