CS 40: Computational Complexity

Sair Shaikh

October 13, 2025

Problem 12. Prove that $\mathrm{RP} \cap \mathrm{coRP} = \mathrm{ZPP}$ and that $\mathrm{RP} \cup \mathrm{coRP} \subseteq \mathrm{BPP}$.

Solution. We first show $\mathrm{RP} \cap \mathrm{coRP} = \mathrm{ZPP}$ by showing both containments.

Let $L \in \mathrm{RP} \cap \mathrm{coRP}$. Then, using the definitions of RP and coRP, there exist polynomial time bounded PTMs M and M' (corresponding to being in RP and coRP respectively) such that:

$$x \in L \implies \left(\mathbf{Pr}(M(x,r) \neq L(x)) \leq \frac{1}{2}\right) \wedge (\mathbf{Pr}(M'(x,r) \neq L(x)) = 0)$$
$$x \notin L \implies (\mathbf{Pr}(M(x,r) \neq L(x)) = 0) \wedge \left(\mathbf{Pr}(M'(x,r) \neq L(x)) \leq \frac{1}{2}\right)$$

Note that if $x \notin L$, then M is guaranteed to reject. By contrapositive, if M accepts, we know that $x \in L$. Similarly, if M' rejects, we know that $x \notin L$. Based off of these observations, we construct a PTM A as follows:

- Simulate M. If M accepts, then accept. Otherwise, continue.
- Simulate M'. If M' rejects, then reject. Otherwise, continue.
- Repeat the last two steps with a different random seed (use further bits off the random tape).

We claim that A is a zero-error PTM with expected run-time bounded by a polynomial.

The zero-error claim is clear, as per our observation above. We only accept when M accepts, which implies $x \in L$, and similarly reject when M' rejects implying $x \notin L$. Thus, we only need to analyze the time-bound.

First, note that each iteration (running M and M' once each) runs in polynomial time. Concretely, let $i_1, i_2 \in \mathbb{N}$ be such that:

$$\forall r \in \{0,1\}^* : \text{TIMECOST}_M(n, r) \in O(n^{i_1}) \wedge \text{TIMECOST}_{M'}(n, r) \in O(n^{i_2})$$

where n represents the familiar abuse of notation, indicating a max over all inputs of length n. Then, running both in sequence is $O(n^i)$ where $i := \max(i_1, i_2)$ (basic exercise from CS30/31). That is, each iteration, regardless of random seed, runs in $\leq c \cdot n^i$ for some $c > 0$.

Now, let $x \in \Sigma^n$ be some input. Let p be the probability (over the randomness) that the first iteration returns on this input, i.e. accepts or rejects. Since we are using new independent random bits on each iteration, each iteration has the same probability of returning. Thus, the number of iterations until and including a successful answer is a random variable, $X \sim \text{Geom}(p)$ with expectation $\frac{1}{p}$.

Overall, then the runtime for A is bounded as:

$$\text{TIMECOST}_A(n, r) \leq X \cdot (c \cdot n^i)$$

Taking expectation over the randomness, we get:

$$\mathbb{E}_r[\text{TIMECOST}_A(n, r)] \leq \mathbb{E}_r[X] \cdot (c \cdot n^i) = \frac{1}{p} \cdot (c \cdot n^i)$$

Thus, we are only left to show that p is sufficiently large. We have two cases:

1. If $x \in L$, then M' always accepts and we wait till M accepts to return. On each iteration, M accepts with probability $\geq \frac{1}{2}$. Thus, $p \geq \frac{1}{2}$.
2. If $x \notin L$, then M always rejects and we wait for M' to reject to return. On each iteration, M' rejects with probability $\geq \frac{1}{2}$. Thus, $p \geq \frac{1}{2}$.

Thus, overall, $\frac{1}{p} \leq 2$. Thus,

$$\mathbb{E}_r[\text{TIMECOST}_A(n, r)] \leq 2c \cdot n^i$$

Thus, $L \in \text{ZPP}$.

For the other containment, let $L \in \text{ZPP}$ and M be the associated PTM. We will show that $L \in \text{RP}$ and $L \in \text{coRP}$ seperately.

We know that whenever M returns, it returns correctly. Moreover, there exists $c > 0$ and $i \in \mathbb{N}$ such that:

$$\forall x : \mathbb{E}_r[\text{TIMECOST}_M(x, r)] \leq c \cdot |x|^i$$

Using Markov's Inequality, we note that:

$$\begin{aligned} &\mathbf{Pr}[\text{TIMECOST}_M(x, r) \geq 2\,\mathbb{E}_r[\text{TIMECOST}_M(x, r)]] \leq \frac{1}{2} \\ \implies &\mathbf{Pr}[\text{TIMECOST}_M(x, r) \geq 2(c \cdot |x|^i)] \leq \frac{1}{2} \end{aligned}$$

where the 2nd line follows from the bound for the expectation above. Thus, we design two PTMs A and B as follows:

- Simulate M for $2(c \cdot |x|^i)$ steps.
- If M returns in that time, return the same answer. Otherwise return False for A and True for B.

We claim that A and B satisfy the constraints for L to be in RP and coRP respectively. Indeed, since they run for a fixed polynomial number of steps, both of them are polynomial time bounded, regardless of the randomness. Thus, we only need to show the error-bounds.

Note that A returns true if and only if M returns true (within the time limit). Thus, A does not error when $x \notin L$, i.e.

$$x \notin L \implies \mathbf{Pr}[A(x) \neq L(x)] = 0$$

Moreover, we know the probability that M does not return in the given time-bounds is $\leq \frac{1}{2}$ from the last inequality above. The event that A returns False erroneously (i.e. when $x \in L$) is a subset of the event where M does not return within the time-limit. Thus,

$$x \in L \implies \mathbf{Pr}[A(x) \neq L(x)] \leq \frac{1}{2}$$

Thus, $L \in$ RP. Similar analysis on B shows that $L \in$ coRP.

Thus, ZPP $=$ RP $\cap$ coRP.

For the 2nd part, we need to show that RP $\subseteq$ BPP and coRP $\subseteq$ BPP.

Let $L \in$ RP and C be the associated polynomial time-bounded PTM. We construct a new machine C' that executes C twice (using fresh random bits off of the random tape), and rejects if both simulations reject, otherwise it accepts. Using our previous observations, we know if $x \notin L$, then C is guaranteed to reject x, thus, C' is guaranteed to reject x, i.e.

$$x \notin L \implies \mathbf{Pr}(C'(x, r) \neq L(x)) = 0 \leq \frac{1}{3}$$

However, if $x \in L$, to make an error, we want both runs of C to reject. Since we are using fresh random bits from the random tape, these runs are independent. Thus, we have:

$$x \in L \implies \mathbf{Pr}(C'(x,r) \neq L(x)) = \mathbf{Pr}(C(x,r) \neq L(x))^2 \leq \left(\frac{1}{2}\right)^2 = \frac{1}{4} \leq \frac{1}{3}$$

Since we only ran C twice, C' is also polynomial time bounded. Thus, we have shown:

$$L \in \text{BPP}$$

Since L was arbitrary, we have:

$$\text{RP} \in \text{BPP}$$

The same argument works for coRP, where we modify the provided machine to only accept if both runs accept, and reject otherwise.

Problem 13. Give a full formal proof that

$$\mathrm{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)} = \mathrm{BPP} = \mathrm{BPP}_{\gamma(n),\gamma(n)}$$

for all error bounds where $\delta(n)$ is polynomially small and $\gamma(n)$ is exponentially small. That is, there exist constants $c, d > 0$ such that $\delta(n) = 1/O(n^c)$ and $\gamma(n) = 1/2^{O(n^d)}$.

Solution. It is easy to show that $\mathrm{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)} \supseteq \mathrm{BPP} \supseteq \mathrm{BPP}_{\gamma(n),\gamma(n)}$ with at most a few repetitions. For instance, to show $\mathrm{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)} \supseteq \mathrm{BPP}$, we need to boose a machine with at most $\frac{1}{3}$ error guarantee to less than $\frac{1}{2} - \frac{k}{n^c}$ guarantee for some given $k > 0$. This is already true for large enough n, as $\frac{k}{n^c} \in \Omega(1)$, i.e. eventually grows smaller than the difference between $\frac{1}{2}$ and $\frac{1}{3}$. For the fininitely many remaining input lengths, $\delta(n)$ takes on some fixed value, thus, $\frac{1}{2} - \delta(n)$ is some fixed constant (dependent on n). Since there are finitely many such small n, we can take a max over them to get some constant (worse-case) value, and assume this is the error guarantee for all small n. We will show, in the remainder of the proof, that it is possible to boost a constant error guarantee to an exponentially small guarantee, while remaining polynomial bounded in time. Thus, in particular, we can boost it to $\leq \frac{1}{3}$. Similar argument shows that $\mathrm{BPP} \supseteq \mathrm{BPP}_{\gamma(n),\gamma(n)}$.

Thus, for our main proof, we want to show that $\mathrm{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)} \subseteq \mathrm{BPP} \subseteq \mathrm{BPP}_{\gamma(n),\gamma(n)}$, that is, we can "boost" to reduce the error probability from $\frac{1}{2} - \delta(n)$ to $\frac{1}{3}$ to $\gamma(n)$ with only at most a polynomial blowup in time.

Let $L \in \mathrm{BPP}_{1-\delta(n),1-\delta(n)}$ and M be the associated polynomial time bounded PTM. Our algorithm is to run M k times and take a majority vote. Note that each subsequent run reads and uses fresh random bits from the tape. We claim that $k = \frac{3}{4\delta(n)^2}$ suffices.

Let $x \in \Sigma^n$ be some input. Let $X_1, \cdots, X_k$ be the indicator variables for whether the ith returned an erroneous answer, i.e. did not match $L(x)$. Note that these are iid as each run uses fresh random bits. Let

$$p := \mathbf{Pr}[X_i] = \mathbf{Pr}_r[M(x, r) \neq L(x)]$$

From the definition of $\mathrm{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)}$, we know that:

$$\forall i : \mathbb{E}[X_i] = \mathbf{Pr}[X_i = 1] = p \leq \frac{1}{2} - \delta(n)$$

Let $S := \sum_{i=1}^{k} X_i$. Then, by linearity of expectation,

$$\mathbb{E}[S] = \sum_{i=1}^{k} \mathbb{E}[X_i] = k \cdot \mathbb{E}[X_1] \leq k \cdot \left(\frac{1}{2} - \delta(n)\right)$$

Using the iid property, we can also find a bound for the variance:

$$\mathbf{Var}[S] = k \cdot \mathbf{Var}[X_i] = k(p(1-p)) \leq \frac{k}{4}$$

where the last inequality follows from $p(1-p) \leq \frac{1}{4}$ for $p \in [0,1]$.

Note that we only make an error when a majority of the runs are erroneous, i.e. $S \geq \frac{k}{2}$. We can bound the probability of error as follows:

$$\begin{aligned}\mathbf{Pr}\left[S \geq \frac{k}{2}\right] &= \mathbf{Pr}\left[S - \frac{k}{2} + k\delta(n) \geq k\delta(n)\right] \\ &= \mathbf{Pr}\left[S - \mathbb{E}[S] \geq k\delta(n)\right] \\ &\leq \mathbf{Pr}\left[|S - \mathbb{E}[S]| \geq k\delta(n)\right]\end{aligned}$$

Using Chebyshev's inequality, we note that:

$$\mathbf{Pr}\left[|S - \mathbb{E}[S]| \geq k\delta(n)\right] \leq \frac{\mathbf{Var}[S]}{(k\delta(n))^2} \leq \frac{k}{4k^2\delta(n)^2} = \frac{1}{4k\delta(n)^2}$$

Thus, since $k = \frac{3}{4\delta(n)^2}$, we get:

$$\mathbf{Pr}\left[S \geq \frac{k}{2}\right] \leq \frac{1}{3}$$

Since $\delta(n) = \frac{1}{O(n^c)}$, $k = O(n^{2c})$. Thus, we only have to repeat M a polynomial number of times, thus, our algorithm is still polynomial time-bounded (the product of polynomials is a polynomial). Thus, we have shown that $L \in \text{BPP}$ (since our bound works for all inputs x). Since L was arbitrary, this proves $\text{BPP}_{\frac{1}{2}-\delta(n),\frac{1}{2}-\delta(n)}$.

Next, we use the same trick of taking a majority vote again to "boost" our algorithm to $\gamma(n)$ error.

Let A be the machine described above (with at most symmetric $\frac{1}{3}$ error). Let $Y_1, \cdots, Y_m$ be random variables indicating if the ith run of A produced an incorrect result. As before, these variables are iid. Let $q := \mathbb{E}[Y_i] = \mathbf{Pr}[X_i = 1] \leq \frac{1}{3}$ and $Y = \sum_{i=1}^{m} Y_i$ be the sum. Note that by linearity of expectation, $\mathbb{E}[Y] = m \cdot q$. We only accept erroneously if $Y \geq \frac{m}{2}$, thus, we try to bound the probability of that.

$$\mathbf{Pr}\left[Y \geq \frac{m}{2}\right] = \mathbf{Pr}\left[Y \geq mq\left(\frac{1}{2q}\right)\right]$$

Note that as $q \leq \frac{1}{3}$, $\frac{1}{2q} \geq \frac{3}{2}$, thus, we get:

$$\mathbf{Pr}\left[Y \geq mq\left(\frac{1}{2q}\right)\right] = \mathbf{Pr}\left[Y \geq mq\left(1 + \frac{1}{2}\right)\right]$$

Then, using the Chernoff bound (with $\delta = \frac{1}{2}$), as Y is the sum of independent bernoulli's, we get:

$$\mathbf{Pr}\left[Y \geq mq\left(1+\frac{1}{2}\right)\right] \leq \left(\frac{e^{\frac{1}{2}}}{\left(\frac{3}{2}\right)^{\frac{3}{2}}}\right)^{mq}$$

Using Wolfram Alpha to evaluate, we note that:

$$r := \frac{e^{\frac{1}{2}}}{\left(\frac{3}{2}\right)^{\frac{3}{2}}} \approx 0.897 < 1$$

Let $a > 0$ be the constant such that $\gamma(n) \geq \left(\frac{1}{2}\right)^{a \cdot n^d}$ (using basic manipulation of big-oh notation). Thus, it suffices to pick m such that:

$$r^{mq} \leq \left(\frac{1}{2}\right)^{a \cdot n^d}$$

Working this out, we get:

$$\begin{aligned} mq \log_2 r &\leq -an^d \\ m &\geq -\frac{a}{q \log_2 r} \cdot n^d && (\text{as } \log_2 r < \log_2 1 = 0) \\ m &\geq -\frac{3a}{\log_2 r} \cdot n^d && (\text{as } q \leq \frac{1}{3}) \end{aligned}$$

Thus, picking $m = \lceil \frac{-3a}{\log_2 r} \rceil \cdot n^d$ (m here is positive as $\log_2 r < 0$) suffices to get:

$$\mathbf{Pr}\left[Y \geq \frac{m}{2}\right] \leq \gamma(n)$$

Since $m = O(n^d)$ is only polynomially large in n, we only have to run A polynomially many times. Thus, overall, our run-time is a bounded as a polynomial in n. Thus, $L \in \text{BPP}_{\gamma(n),\gamma(n)}$ (since our bound works for all inputs x). Since L was arbitrary, we get $\text{BPP} \subseteq \text{BPP}_{\gamma(n),\gamma(n)}$. That completes the proof.

Problem 14. Let X and Y be finite sets and let Y^X denote the set of all functions from X to Y. A family $\mathcal{H} \subseteq Y^X$ is said to be 2-universal if the following property holds, with $h \in_R \mathcal{H}$ picked uniformly at random:

$$\forall x, x' \in X, \forall y, y' \in Y, \quad (x \neq x') \implies \mathbf{Pr}_h[h(x) = y \wedge h(x') = y'] = \frac{1}{|Y|^2}.$$

Consider the sets $X = \{0,1\}^n$ and $Y = \{0,1\}^k$, with $k \leq n$. Treat the elements of X and Y as column vectors with 0/1 entries. For a matrix $A \in \{0,1\}^{k \times n}$ and vector $b \in \{0,1\}^k$, define

$$h_{A,b}(x) = Ax + b,$$

where all additions and multiplications are performed mod 2.
Now consider the family of functions

$$\mathcal{H} = \{h_{A,b} : A \in \{0,1\}^{k \times n}, b \in \{0,1\}^k\}$$

First, prove that

$$\forall x \in X, \forall y \in Y : \mathbf{Pr}_h[h(x) = y] = \frac{1}{|Y|}$$

Next, prove that $\mathcal{H}$ is a 2-universal family of hash functions.

Solution. Fix some arbitrary $x \in X = \{0,1\}^n$. For a given $A \in \{0,1\}^{k \times n}$ and $b \in \{0,1\}^k$, let $h_{A,b}(x)_i = (Ax + b)_i$ denote the ith bit of the output. With the notation set up, we claim that:

1. $h(x)_i$ is equally likely to be 0 or 1 (over the choice of $h \in \mathcal{H}$), i.e.

$$\forall i \in [k] : \mathbf{Pr}_h[h(x)_i = 0] = \mathbf{Pr}_h[h(x)_i = 1] = \frac{1}{2}$$

2. The value of $h(x)_i$ is independendent of $h(x)_{i'}$ for $i \neq i'$ for all $h \in \mathcal{H}$.

To show these, first note that:

$$(Ax + b)_i = b_i + \sum_j A_{ij} x_j \pmod 2$$

where A_{ij} is the ijth entry of A and b_i is the ith entry of b.

Then note that:

$$(Ax + b)_i = 0 \iff \sum_j A_{ij} x_j \equiv b_i \pmod 2$$

However, for any given A, note that $\sum_j A_{ij} x_j$ has some fixed parity. Moreover, note that $\mathbf{Pr}_b[b_i = 1] = \frac{1}{2}$ as exactly half of our choices for b have 1 in the ith bit (one can set up a

bijection from $b \in \{0,1\}^k$ with $b_i = 0$ to $b \in \{0,1\}^k$ with $b_i = 1$), and this is not dependent on our choice for A. Thus,

$$\forall A : \mathbf{Pr}_b \left[\sum_j A_{ij} x_j \equiv b_i \pmod 2 \right] = \frac{1}{2}$$

as b_i is equally likely to match or not match the parity of $\sum_j A_{ij} x_j$. Thus, we have that:

$$\forall A : \mathbf{Pr}_b[(Ax + b)_i = 0] = \frac{1}{2}$$

Since this is true for all A, we can write:

$$\mathbf{Pr}_{A,b}[(Ax + b)_i = 0] = \mathbf{Pr}_h[h(x)_i = 0] = \frac{1}{2}$$

Since, $(Ax + b)_i \in \{0, 1\}$, we also have:

$$\mathbf{Pr}_h[h(x)_i = 0] = 1 - \frac{1}{2} = \frac{1}{2}$$

That proves (1). For (2), fix some $h \in \mathcal{H}$, i.e. fix some A, b. Now, note that $(Ax + b)_i$ only depends on the entries A_{ij}, for all j, of A and b_i of b. Thus, for $i \neq i'$, $(Ax + b)_i$ and $(Ax + b)_{i'}$ depend on disjoint set of entries of A and b. Similar to our argument for b_i before, for all i, j $\mathbf{Pr}_A[A_{ij} = 1] = \mathbf{Pr}_b[b_i = 1] = \frac{1}{2}$ (we can set up similar bijections to show this), independently of other entries in A and b. These two facts combined imply that the value of $(Ax + b)_i$ is independent of $(Ax + b)_{i'}$ for $i \neq i'$. That proves (2).

Now, fix some arbitrary $y \in \{0,1\}^k$. Note that:

$$\begin{aligned}
\mathbf{Pr}_h[h(x) = y] &= \mathbf{Pr}_{A,b}[Ax + b = y] \\
&= \mathbf{Pr}_{A,b}\left[\bigwedge_i (Ax + b)_i = y_i \right] \\
&= \prod_{i=1}^{k} \mathbf{Pr}_{A,b}\left[(Ax + b)_i = y_i \right] && \text{(by (2))} \\
&= \prod_{i=1}^{k} \frac{1}{2} && \text{(by (1), as } y_i \text{ is fixed)} \\
&= \left(\frac{1}{2} \right)^k \\
&= \frac{1}{|Y|}
\end{aligned}$$

Since x and y were arbitrary, this suffices to show that:

$$\forall x \in X, \forall y \in Y : \mathbf{Pr}_h[h(x) = y] = \frac{1}{|Y|}$$

Next, we need to show that $\mathcal{H}$ is a 2-universal family of hash functions. Thus, let $y, y' \in Y$ and $x, x' \in X$ with $x \neq x'$. Note that:

$$Ax + b = y \wedge Ax' + b = y' \iff b = y - Ax = y' - Ax' \iff A(x' - x) = y' - y$$

where all operations are mod 2. Then, for a fixed A, the last equation is either consistent or not. If it is consistent, then there is only one b that works, as all other variables in the equation are fixed. If it is inconsistent, no such b exists. Thus,

$$\mathbf{Pr}_b[(Ax + b = y \wedge Ax' + b = y')|A] = \begin{cases} \frac{1}{|Y|} & \text{if } A(x' - x) = y' - y \\ 0 & \text{otherwise} \end{cases}$$

Moreover, since $x' - x \neq 0$, we claim that there are an equal number of matrices A such that $A(x' - x) = z$ for any z. One way to see this is as follows:

- Since $x' - x \neq 0$, we can construct a basis for X with $x' - x$ being the first basis vector.
- Then, expressed in this basis, the set of A that send $x' - x$ to z are precisely all the matrices that have z as their first column, with the other entries $((k-1) \times n)$ many)free to take on any values.
- Then, it is easy to see that there are equally as many matrices A such that $A(x' - x) = z$ for every z as there are exactly $2^{(k-1)\times n}$ such matrices for each z.

In particular, averaging over all A, we get:

$$\mathbf{Pr}_A[A(x' - x) = z] = \frac{1}{|Y|}$$

Picking $z = y' - y$, and putting everything together, note that:

$$\begin{aligned} \mathbf{Pr}_h[h(x) = y \wedge h(x') = y'] &= \mathbf{Pr}_{A,b}[Ax + b = y \wedge Ax' + b = y'] \\ &= 0 + \mathbf{Pr}_b[Ax + b = y \wedge Ax' + b = y'|A] \, \mathbf{Pr}_A[A(x' - x) = y' - y] \\ &= \frac{1}{|Y|} \cdot \frac{1}{|Y|} \\ &= \frac{1}{|Y|^2} \end{aligned}$$

This shows that $\mathcal{H}$ is a 2-universal hash family.