# Exercise VII, Computational Complexity 2024

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

## Randomised Complexity

**1** Characterisation of ZPP.

(a) Prove that $L \in \mathsf{ZPP}$ if and only if there is a polynomial-time probabilistic TM $M$ that outputs values in $\{0, 1, ?\}$ such that for every input $x$, we have $M(x) \in \{L(x), ?\}$ and $\Pr[M(x) = ?] \leq 1/2$. *(Hint: Suppose we have a PTM $M$ with expected runtime $t(n)$. Use Markov's inequality to show that $M$ halts within $2t(n)$ steps with probability at least $1/2$.)*

(b) Show that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$. *(Hint: How to combine an* RP*-style algorithm and an* coRP*-style algorithm into a single* ZPP*-algorithm?)*

**Solution:**

(a) Suppose that $L \in \mathsf{ZPP}$ and let $M$ be a PTM computing $L$ with zero error and expected run-time $t(n)$. Consider the PTM machine $M'$ that runs $M$ for $2 \cdot t(n)$ steps and outputs whatever $M'$ outputs or ? if $M'$ wasn't finished. $M'$ runs in worst-case $2t(n)$ and if it doesn't output ?, then it is correct. On the other hand using Markov's inequality:

$$\Pr(M'(x) = ?) = \Pr(M \text{ runs for more than } 2t(n) \text{ steps}) \leq \frac{t(n)}{2t(n)} = \frac{1}{2}$$

Regarding the other direction let $M$ be a PTM for $L$ which runs in worst-case time $t(n)$ and is always correct but might output ?. Consider the PTM $M'$ that simply runs $M$ in turns until it gives a non-? answer. Clearly, $M'$ is always correct. On the other hand, we can bound its expected run-time:

$$\mathbb{E}\left[\text{time spent by } M'\right] \leq t(n) \cdot \mathbb{E}\left[\# \text{ of runs of } M\right] = t(n) \cdot \sum_{i=1}^{\infty} i \cdot 2^{-i} \leq O(t(n))$$

thus, $M'$ witnesses that $L \in \mathsf{ZPP}$

(b) We first show that $\mathsf{ZPP} \subseteq \mathsf{RP}$. Let $M$ be a zero-error expected time $t(n)$ PTM for $L$. Just as in part (a), define $M'$ that runs $M'$ for $3t(n)$ steps. If $M$ had the time to finish in that many steps, return the answer, else reject. $M'$ runs in worst-case polynomial time and note that if $x \notin L$, $\Pr(M'(x) = 1) = 0$. On the other hand using Markov's inequality, we have that if $x \in L$, $\Pr(M'(x) = 1) \geq 2/3$ so that $L \in \mathsf{RP}$ too. Using the same trick, it can be shown that $\mathsf{ZPP} \subseteq \mathsf{coRP}$.

Regarding the other direction, let $L \in \mathsf{RP} \cap \mathsf{coRP}$ and let $M_0$ be a $\mathsf{coRP}$ machine for $L$ and $M_1$ a $\mathsf{RP}$ machine for $L$. Consider the PTM $M$ that runs $M_0$ and $M_1$ in turns until either $M_0$ reject (in which case we know for sure that $x \notin L$) or $M_1$ accept (in which case we know for sure that $x \in L$). $M$ is correct on all inputs, and its expected time is polynomially bounded using a calculation similar to the second part of (a).

**2** Prove that $\mathsf{NP} \subseteq \mathsf{BPP}$ implies $\mathsf{NP} = \mathsf{RP}$.

**Solution:** Observe first that $\mathsf{RP} \subseteq \mathsf{NP}$. Indeed, let $L \in \mathsf{RP}$ and $M$ be a poly-time Turing and $p : \mathbb{N} \to \mathbb{N}$ be a polynomial such that for any $x \in \{0, 1\}^*$:

$$x \in L \implies \Pr_{r \in_U \{0,1\}^{p(|x|)}} (M(x,r) = 1) \geq 2/3$$
$$x \notin L \implies \Pr_{r \in_U \{0,1\}^{p(|x|)}} (M(x,r) = 1) = 0$$

The above is taking the view that a PTM is nothing but a TM with a random input. But this means that $r$ can act as a certificate and shows that $L \in \mathsf{NP}$:

$$x \in L \iff \exists r \in \{0, 1\}^{p(|x|)} : M(x,r) = 1$$

Regarding the other direction, it is sufficient to show that if $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\textsc{Sat} \in \mathsf{RP}$. To this end, let $M$ be a PTM for $\textsc{Sat}$ which is boosted so that for any formula $\varphi$ :

$$\varphi \in \textsc{Sat} \implies \Pr(M(\varphi) = 1) \geq 1 - n^{-2}$$
$$\varphi \notin \textsc{Sat} \implies \Pr(M(\varphi) = 1) \leq n^{-2}$$

Where $n$ is the number of variables in $\varphi$. We design a PTM $M'$ that tries to build a satisfying assignment $x^\star$ for $\varphi$. First, let $\varphi_0$ (respectively $\varphi_1$) be the formula $\varphi$ where $x_1$ is set to be 0 (respectively 1). If $M(\varphi_0) = 1$ we have found a value for $x_1$ (i.e. 0) and continue with the smaller formula $\varphi_0$. Else, if $M(\varphi_1) = 1$, we have found a value for $x_1$ and continue with the smaller formula $\varphi_1$. If both runs reject, we reject $\varphi$. At the end of the process, we get an assignment $x^\star$ which is supposedly satisfying $\varphi$. Finally we output 1 if and only if $\varphi(x^\star) = 1$.

Observe that $M'$ runs in poly-time and that if $M'$ accepts $\varphi$, then it must be that $\varphi \in \textsc{Sat}$ so that if $\varphi \notin \textsc{Sat}$, then $\Pr(M'(\varphi) = 1) = 0$. We still need to prove that if $\varphi \in \textsc{Sat}$ then $\Pr(M'(\varphi) = 1) \geq 2/3$, hence let $\varphi \in \textsc{Sat}$. Using a union bound, we have:

$$\Pr(M'(\varphi) = 0) \leq \Pr(M \text{ fails for } x_1 \text{ or } x_2 \text{ or } \ldots \text{ or } x_n) \leq \sum_{i=0}^{n} n^{-2} \leq 1/3$$

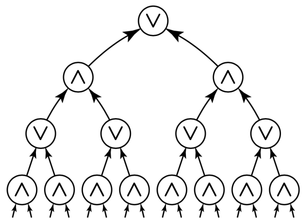
As a technicality, any sub-formula that gets investigated on needs to be padded with trivially satisfiable clauses like $x_1 \vee \neg x_1$. This is to make sure that their size remains $\geq n$ and that we can use the $n^{-2}$ faillure probability bound.

**3** Consider the 4-bit function $f(x) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Suppose we want to evaluate $f$ using a simple query algorithm (a decision tree) that queries, in arbitrary adaptive order, individual input variables $x_i \in \{0,1\}$ until it learns enough information about $x$ in order to output $f(x) \in \{0,1\}$.

(a) Show that any deterministic query algorithm for $f$ needs to query all variables in the worst case. That is, for any deterministic query strategy, there is some input $x$ such that the strategy makes 4 queries.

(b) Show that there is a randomised 0-error strategy (i.e., ZPP style) that, for every input $x$, makes 3 queries *in expectation* and always outputs $f(x)$ correctly.

(c) (*) For $n = 2^k$, define $g\colon \{0,1\}^n \to \{0,1\}$ as the function computed by a depth-$k$ binary tree that has $\vee$- and $\wedge$-gates on alternating levels, and with $n$ variables at the leaves:



Show that every deterministic query algorithm requires $n$ queries to compute $g$ but that there exists an 0-error randomised algorithm with expected query cost

$$3^{k/2} \;=\; n^{\log_2 \sqrt{3}} \;=\; n^{0.79248\ldots}.$$

**Solution:**

(a) Let $t$ be a deterministic query algorithm for $f$. We build a strategy for answering queries that leaves the function value undetermined until the very last query. Suppose without loss of generality that $t$ first queries $x_1$, then we respond 1. Now $t$ can either query $x_2$ or $x_3$ (the case $x_4$ is symmetric). If $x_2$ is queried, then we respond 0 so that $t$ must solve $x_3 \wedge x_4$, further taking two queries. On the other hand if $t$ queries $x_3$, we respond by 1 so that $t$ must now solve $x_2 \vee x_4$, taking two other queries.

(b) A random strategy is as follows. Choose randomly to solve $x_1 \wedge x_2$ or $x_3 \wedge x_4$ (stopping at the first which evaluates to '1'). To solve a problem of type $x_i \vee x_j$, choose the query order at random and stop at the first '0' found. This algorithm is always correct, let us now evaluate its expected cost.

Suppose first that $x$ is such that $f(x) = 1$. It means that at least one and-block evaluates to 1. With probability $\geq 1/2$ we find it in the first try, resulting in only two queries. With the remaining probability, we need 4 queries to evaluate the function. On the other hand, if $f(x) = 0$ it means that both and-blocks evaluates to zero and so we expect to spend 1.5 queries on both. Thus, the overall expected query cost is 3.

(c) To get the upper bound, the idea is to define an adversarial strategy to respond to queries. This strategy is built recursively and the base case is part (a). The same recursive trick applies to evaluate the function in ZPP style.

**4** Suppose Alice holds an $n$-bit string $x \in \{0,1\}^n$ and Bob holds an $n$-bit string $y \in \{0,1\}^n$. Alice and Bob want to decide whether $x = y$ using a *one-way communication protocol* where Alice sends a single message $m = m(x)$ to Bob, and Bob outputs one bit indicating whether or not $x = y$. Their goal is to minimise the bit-length $|m|$ of the message.

(a) Show that any deterministic protocol requires message length $|m| \geq n$.

(*Hint: Exercise V-4.*)

(b) Suppose Alice and Bob now have access to a shared random string $r \in \{0,1\}^\ell$ and that Alice's message $m = m(x,r)$ can depend on $r$. Show that Alice and Bob can succeed with probability $\geq 2/3$ (i.e., BPP style) by sending only $|m| \leq O(1)$ bits.

(*Hint: Suppose* $\ell = n$. *What can you say about the mod-2 inner products* $\langle x, r\rangle \bmod 2$ *and* $\langle y, r\rangle \bmod 2$? *Here* $\langle x, r\rangle = \sum_{i=1}^{n} x_i r_i$.)

**Solution:**

(a) Suppose toward contradiction that Alice has a way to send only $n-1$ bits and solve the problem. This implies that Alice can send at most $2^{n-1}$ different messages, since her input space is $\{0, 1\}^n$, there exists two distinct $x_1, x_2 \in \{0, 1\}^n$ that map to the same message $m \in \{0, 1\}^{n-1}$. Now, since the protocol is correct it must be that if Bob holds $x_1$, it accepts. But since Alice's message is the same for $x_1$ and $x_2$, the protocol also accepts the pair $(x_2, x_1)$: a contradiction.

(b) Let $r_1, \ldots, r_n \overset{\text{iid}}{\sim} U\{0,1\}$ be iid 0/1 random variables. Then, for any two different $x, y \in \{0, 1\}^n$, $\Pr(\langle x, r\rangle = \langle z, r\rangle) = 1/2$. Indeed, supposing without loss of generality that $x_1 \neq y_1$:

$$\begin{aligned}
\Pr(\langle x, r\rangle = \langle z, r\rangle) &= \Pr\left(r_1(x_1+y_1) = \sum_{i=2}^{n} r_i(x_i+y_i)\right) \\
&= \Pr\left(\sum_{i=2}^{n} r_i(x_i+y_i) = 0\right) \cdot \Pr\left(r_1 = 0\right) \\
&\quad + \Pr\left(\sum_{i=2}^{n} r_i(x_i+y_i) = 1\right) \cdot \Pr\left(r_1 = 1\right) \\
&= \frac{1}{2} \cdot \left(\Pr\left(\sum_{i=2}^{n} r_i(x_i+y_i) = 0\right) + \Pr\left(\sum_{i=2}^{n} r_i(x_i+y_i) = 1\right)\right) \\
&= 1/2
\end{aligned}$$

Thus Alice simply sends two bits $a_1 := \langle x, r^1\rangle$ and $a_2 := \langle x, r^2\rangle$. Bob computes $b_1 := \langle y, r^1\rangle$ and $b_2 := \langle y, r^2\rangle$. Bob accepts $a_1 = b_1$ and $a_2 = b_2$ and else rejects.

If $x = y$, then Bob always accepts. On the other hand, if $x \neq y$, then Bob rejects with probability $3/4$. So that overall the protocol only communicates 3 bits and has success probability $3/4$.