CS 505 Spring 2025 — Homework 3 (Sample Solutions)

Alexander R. Block

Due Date: March 20, 2025, no later than 2:00pm Central Time.

1 The Polynomial Hierarchy and Alternations (25 Points)

1.1 Part 1 (5 Points)

Show that if 3SAT $\leq_p \overline{\text{3SAT}}$, then $\mathbf{PH} = \mathbf{NP}$.

Proof of Problem 1 Part 1. Suppose that 3SAT $\leq_p \overline{\text{3SAT}}$. Notice that since 3SAT is **NP**-complete and $\overline{\text{3SAT}}$ is **coNP**-complete, this implies that $\mathbf{NP} = \mathbf{coNP}$. By definition, this tells us that $\Sigma_1^p = \Pi_1^p$. By the Theorem we proved in class, this implies that $\mathbf{PH} = \mathbf{NP} = \mathbf{coNP}$. □

1.2 Part 2 (10 Points)

Prove that $\mathbf{APSPACE} = \mathbf{EXP}$.

Hint: the non-trivial direction is $\mathbf{EXP} \subseteq \mathbf{APSPACE}$. *This proof will use ideas that were explored in the proof that* SAT $\notin \mathbf{TISP}(n^{1.1}, n^{0.1})$.

Proof of Problem 1 Part 2. The easier direction is $\mathbf{APSPACE} \subseteq \mathbf{EXP}$, which we start with. Recall that $\mathbf{APSPACE} = \cup_{c \in \mathbb{N}} \mathbf{APSPACE}(n^c)$, where $\mathbf{APSPACE}(S)$ is the set of all languages L decidable by an alternating Turing machine using at most $O(S)$ extra space and an unlimited number of alternations.

Let $L \in \mathbf{APSPACE}$. In particular, assume that $L \in \mathbf{APSPACE}(n^c)$ for some constant $c \geq 1$. Let M be the ATM deciding L. Let $x \in \{0,1\}^n$ and $S = n^c$. Since M uses at most S space on any input x of length n, we know that the configuration graph $G_{M,x}$ has at most 2^s configurations.

We construct a DTM N which on input x constructs the configuration graph $G_{M,x}$ for the ATM M on input x. Then, given this constructed graph $G_{M,x}$, $N(x)$ then labels the nodes in the graph according to the alternation states and outputs 1 if and only if the root of the configuration graph is labeled 1 after labeling all nodes. This clearly requires $O(2^s) = O(2^{n^c})$ time, so $L \in \mathbf{EXP}$.

Now, the non-trivial direction is showing $\mathbf{EXP} \subseteq \mathbf{APSPACE}$. Let $L \in \mathbf{EXP}$ and assume that $L \in \mathbf{DTIME}(2^{n^c})$ for some constant $c \geq 1$. Let M be the DTM which decides L in $O(2^{n^c})$ time for any $x \in \{0,1\}^n$.

Our goal is to encode the computation of $M(x)$ using an ATM with only polynomial space. Consider the configuration graph of $M(x)$, $G_{M,x}$. Each configuration has size $O(2^{n^c})$ and there are at most $O(2^{n^c})$ configurations. Since M is a DTM, we know that $G_{M,x}$ is actually a line graph/a path.

We cannot directly encode the entire configuration at each time step, as this would require exponential space. Instead, we look back to the Cook-Levin Theorem and use ideas from here to encode the computation in the ATM. Note that the configuration graph $G_{M,x}$ is equivalently represented by a table of size $O(2^{n^c}) \times O(2^{n^c})$. Without loss of generality, assume it is of size $2^{n^c} \times 2^{n^c}$, as in the proof of the Cook-Levin Theorem, where every row of the table is a configuration of the machine $M(x)$ and row $i+1$ follows from row i via the transition function of M (if the table is valid).

Let c_{ij} denote the jth entry of row i in this table. By the proof of the Cook-Levin Theorem, we know that the validity of c_{ij} only depends on a constant number of values from row $i-1$. In particular, the validity of c_{ij} is uniquely determined by $c_{i,k}$ for $k \in \{j-1, j, j+1\}$ and the transition function δ of M.

With this information, we can define a function g_δ such that for any tape symbol $\gamma \in \Gamma$, $c_{ij} = \gamma$ if and only if $g_\delta(\gamma_1, \gamma_2, \gamma_3) = \gamma$, where $\gamma_1 = c_{i-1,j-1}$, $\gamma_2 = c_{i-1,j}$, and $\gamma_3 = c_{i-1,j+1}$.

So we have

$$\forall\gamma \left(c_{ij} = \gamma \iff \begin{array}{l} \gamma_1 = c_{i-1,j-1} \wedge \\ \gamma_2 = c_{i-1,j} \wedge \\ \gamma_3 = c_{i-1,j+1} \wedge \\ \gamma = g_\delta(\gamma_1,\gamma_2,\gamma_3) \end{array} \right).$$

To decide if $c_{ij} = \gamma$, we can use non-determinism to "guess" the values $\gamma_1, \gamma_2, \gamma_3$. That is,

$$\forall\gamma\exists\gamma_1,\gamma_2,\gamma_3 \left(c_{ij} = \gamma \iff \begin{array}{l} \gamma_1 = c_{i-1,j-1} \wedge \\ \gamma_2 = c_{i-1,j} \wedge \\ \gamma_3 = c_{i-1,j+1} \wedge \\ \gamma = g_\delta(\gamma_1,\gamma_2,\gamma_3) \end{array} \right).$$

Now, we need to recursively verify that $\gamma_1 = c_{i-1,j-1}$, $\gamma_2 = c_{i-1,j}$, and $\gamma_3 = c_{i-1,j+1}$, which we can do using a $\forall$. That is, we can check that $\forall i, j \in [2^{n^c}]\forall k \in \{j-1, j, j+1\}$ that $\gamma_{k-j+2} = c_{i-1,k}$. Note that this only requires polynomial space, as do all the above checks. Finally, we can add final check that there exists an accepting state in the final configuration in the table, which again is only polynomial space. Then, we have an ATM N which on input x uses alternations to verify the validity of the table encoding the computation of $M(x)$ using only polynomial space, so $L \in \mathbf{APSPACE}$. □

1.3 Part 3 (10 Points)

Suppose that there exists a language/oracle A such that $\mathbf{P}^A = \mathbf{NP}^A$. Prove that $\mathbf{PH}^A \subseteq \mathbf{P}^A$. (I.e., the proof that $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{PH} = \mathbf{P}$ relativizes).

Proof of Problem 1 Part 3. Let A be a language/oracle such that $\mathbf{P}^A = \mathbf{NP}^A$. We show that for all i, $(\Sigma_i^p)^A, (\Pi_i^p)^A \subseteq \mathbf{P}^A$. First, for $i = 1$, we have that $(\Sigma_i^p)^A = \mathbf{NP}^A$, so the base case of the inductive hypothesis holds; in particular, $\mathbf{P}^A = \mathbf{NP}^A$ implies that $\mathbf{NP}^A = \mathbf{coNP}^A$. Now, assume the induction holds for some level $i > 1$. We show it holds for $i + 1$. Let $L \in (\Sigma_{i+1}^p)^A$. Then, there exists a polynomial-time deterministic oracle Turing machine M such that $x \in L$ if and only if

$$\exists w_1 \forall w_2 \ldots Q_{i+1} w_{i+1} M^A(x, w_1, w_2, \ldots, w_{i+1}) = 1,$$

where each $w_j \in \{0,1\}^{p(|x|)}$ for some polynomial p, $Q_{i+1} = \exists$ if $i + 1$ is odd and $Q_{i+1} = \forall$ if $i + 1$ is even, and M makes at most a $q(|x|)$ oracle queries to A for some polynomial q.

As in the proof from class of the non-oracle version of the theorem, define a new language L' consisting of all pairs (x, w_1) such that

$$\forall w_2 \ldots Q_{i+1} w_{i+1} M^A(x, w_1, w_2, \ldots, w_{i+1}) = 1,$$

where Q_{i+1} is again defined analogously as above. Now, clearly $L' \in (\Pi_i^p)^A$. By our inductive hypothesis, this implies that $L' \in \mathbf{P}^A$. So there exists an oracle Turing machine D running in polynomial time such that $(x, w_1) \in L'$ if and only if $D^A(x, w_1) = 1$. Transforming this back to our original statement for L, we have that $x \in L$ if and only if $\exists w_1 D^A(x, w_1) = 1$. So clearly $L \in \mathbf{NP}^A = \mathbf{P}^A$. This implies that $(\Sigma_{i+1}^p)^A \subseteq \mathbf{P}^A$, and also implies that $(\Pi_{i+1}^p)^A \subseteq \mathbf{P}^A$ since $\mathbf{P}$ is still closed under complement even relative to an oracle. Thus, we have shown $\mathbf{PH}^A \subseteq \mathbf{P}^A$. □

2 Randomized Computations (25 points)

2.1 Part 1 (10 Points)

Prove that $\mathbf{BPL} \subseteq \mathbf{P}$, where $\mathbf{BPL}$ is the set of all languages in $\mathbf{BPP}$ that are decidable in on a PTM using at most $O(\log(n))$ additional space on the read/write tapes.

Hint: Try to compute the probability that the machine ends up in an accepting configuration using either dynamic programming or matrix multiplication.

Proof of Problem 2 Part 1. Let $L \in \mathbf{BPL}$ and suppose M_L is the $\mathbf{BPL}$ machine deciding L. Consider the directed configuration graph $G_{M_L,x}$ for any x given as input to M_L. Since M_L only uses $O(\log(n))$ space for $n = |x|$, $G_{M_L,x}$ as at most $O(n^c)$ nodes for some constant c. For ease of presentation, suppose that the number of nodes in $G_{M_L,x}$ is exactly n^c. Write down the $n^c \times n^c$ adjacency matrix $\mathbf{A}$ of $G_{M_L,x}$, where $\mathbf{A}_{i,j} = 1/2$ if configuration j is reachable from configuration i in one step of the computation (i.e., in configuration i, with probability $1/2$ the machine $M_L(x)$ chooses a transition function with probability $1/2$ and this transition function goes from configuration i to configuration j). Now, if both transition functions take configuration i to configuration j, write $\mathbf{A}_{i,j} = 1$. Otherwise, simply write $\mathbf{A}_{i,j} = 0$.

By definition of $\mathbf{BPP}$ (and thus $\mathbf{BPL}$), each row of the matrix $\mathbf{A}$ sums to exactly 1 (i.e., it is a stochastic matrix). This is because the machine M_L has two transition functions it chooses from during any time-step of the computation. So either any row of $\mathbf{A}$ has exactly two entries with value $1/2$, or one entry with value 1. Since $\mathbf{A}$ is a stochastic matrix, we can repeatedly take powers of $\mathbf{A}$ to determine the probability of reaching configuration i from configuration j. For example, in the matrix $\mathbf{A}^2$, if entry (i, j) is non-zero, it means that configuration j is reachable within 2 time-steps from configuration i.

Repeating this process for $k = n^c$ times (i.e., computing $\mathbf{A}^k$) reveals to us all configurations that are connected to the starting configuration. In particular, if we let configuration 1 be the starting configuration, then for $\mathbf{B} = \mathbf{A}^{n^c}$, row 1 of $\mathbf{B}$ tells us all configurations j reachable from configuration 1 (the starting configuration) in at most n^c steps (the total number of vertices in $G_{M_L,x}$).

This gives us an algorithm in $\mathbf{P}$ to decide L. Let D be this DTM. On input x, $D(x)$ constructs the adjacency matrix $\mathbf{A}$, computes $\mathbf{B} = \mathbf{A}^{n^c}$, then scans the first row of $\mathbf{B}$, outputting accept if and only if there is some accepting configuration j and $B_{1,j} > 0$. All of these steps take polynomial time, so $L \in \mathbf{P}$. $\square$

2.2 Part 2 (10 Points)

Prove that a language L is in $\mathbf{ZPP}$ if and only if there exists a polynomial-time PTM M with outputs $\{0, 1, \perp\}$ such that for every $x \in \{0, 1\}^*$, the following holds:

$$\Pr[M(x) \in \{L(x), \perp\}] = 1 \quad \Pr[M(x) = \perp] \leq \frac{1}{2}.$$

Hint: recall that $\mathbf{ZPP}$ *is defined with respect to expected polynomial time. You will need to argue that if you require strict polynomial time (related to the expected runtime), the probability your computation hasn't halted when you reach the hard polynomial-time cutoff is upper bounded by* $1/2$.

Proof of Problem 2 Part 2. For the first direction, let $L \in \mathbf{ZPP}$. Then there exists a PTM M_L deciding L in expected polynomial-time. Let p be this polynomial. That is, if we let $T_{M_L,x}$ denote the random variable for the runtime of $M_L(x)$, we have $\mathbb{E}[T_{M_L,x}] = p(|x|)$.

Define a new probabilistic Turing machine R which operates as follows. On input x, $R(x)$ does the following.

1. Set $T = 2 \cdot p(|x|)$.
2. Simulate $M_L(x)$ for at most T time-steps.
   - If $M_L(x)$ halts within the time bound T and outputs b, output b.

3. Output $\perp$.

Now, for any input x, clearly $R(x) \in \{L(x), \perp\}$. This is because by definition of **ZPP**, if $M_L(x)$ halts, it always outputs $b = L(x)$. By definition of $R(x)$, $R(x)$ either outputs the result of $M_L(x)$ or $\perp$. Now, we argue that $\Pr[R(x) = \perp] \leq 1/2$. $R(x)$ only outputs $\perp$ if $M_L(x)$ exceeds the time-bound $T = 2 \cdot p(|x|)$. By Markov's inequality, we have

$$\Pr[T_{M_L,x} \geq T] \leq \frac{\mathbb{E}[T_{M_L,x}]}{T} = \frac{p(|x|)}{2p(|x|)} = \frac{1}{2}.$$

For the other direction, let L be a language decidable by a PTM in strict polynomial time such that for every x, $\Pr[M(x) \in \{L(x), \perp\} = 1$ and $\Pr[M(x) = \perp] \leq 1/2$. We construct a new machine N which for any x satisfies $N(x) = L(x)$ and runs in expected polynomial time. The machine N on input x simply runs $M(x)$ until $M(x)$ does not output $\perp$. By definition, we know that $M(x) = L(x)$ when it does not output $\perp$, and that $M(x)$ outputs $\perp$ with probability at most $1/2$. All that is left to show is that $N(x)$ in expectation executes $M(x)$ at most a polynomial number of times since $M(x)$ runs in polynomial time.

Let T denote the random variable for the number of times that $N(x)$ executes $M(x)$. Then,

$$\mathbb{E}[T] = \sum_{i=1}^{\infty} i \cdot \Pr[T = i].$$

Note that $T = i$ if and only if $M(x) = \perp$ the previous $i - 1$ times. Each execution of $M(x)$ is independent of the prior executions. Therefore, we have

$$\mathbb{E}[T] = \sum_{i=1}^{\infty} i \cdot \Pr[T = i] = \sum_{i=1}^{\infty} i \cdot \Pr[M(x) \neq \perp] \cdot \Pr[M(x) = \perp]^{i-1} = \sum_{i=1}^{\infty} i \cdot (1 - p) \cdot p^{i-1},$$

where $p = \Pr[M(x) = \perp] \leq 1/2$. Rewriting the right-hand side, we have

$$\sum_{i=1}^{\infty} i \cdot (1 - p) \cdot p^{i-1} = (1 - p)/p \sum_{i=1}^{\infty} i \cdot p^i = \frac{(1 - p)}{p} \cdot \frac{p}{(p - 1)^2},$$

where the final equality holds for $|p| < 1$. Notice that $(p - 1)^2 = (1 - p)^2$, so we have

$$\mathbb{E}[T] = \frac{1}{1 - p} \leq 2$$

since $1 - p \geq 1/2$. Therefore, the expected number of executions of $M(x)$ for $M(x) \neq \perp$ is 2, so $N(x)$ runs in expected polynomial time. □

2.3 Part 3 (5 Points)

Prove that if **NP** $\subseteq$ **BPP**, then **NP** = **RP**.

Proof of Problem 2 Part 3. Suppose that **NP** $\subset$ **BPP**. Now, recall the definition of **RP**. $L \in$ **RP** if and only if there exists a polynomial-time DTM M such that

$$x \in L \implies \Pr_r[M(x, r) = 1] \geq 2/3$$
$$x \notin L \implies \Pr_r[M(x, r) = 0] = 1,$$

where $r \in \{0, 1\}^{\text{poly}(|x|)}$.

First, notice that regardless of the hypothesis, we have that **RP** $\subseteq$ **NP**, as languages in **RP** can be thought of as **NP** languages with at least 2/3 of the NTM paths leading to an accepting configuration; in other words, the above definition of $M(x, r)$ is clearly also an **NP** verifier for L.

To show that **NP** $\subseteq$ **RP**, we show that 3SAT $\in$ **RP**. Since **NP** $\subseteq$ **BPP**, let B be the **BPP** machine for 3SAT. Then, $\phi \in$ 3SAT if and only if $\exists r$ such that $\phi(r) = 1$. Moreover,

$$\begin{aligned} \phi \in \text{3SAT} &\implies \Pr[B(\phi) = 1] \geq 2/3 \\ \phi \notin \text{3SAT} &\implies \Pr[B(\phi) = 0] \geq 2/3. \end{aligned}$$

Let ϕ be an n-variable 3CNF formula. Define a new machine R as follows. On input ϕ, $R(\phi)$ does the following.

1. Set $\varphi = \phi$.
2. For $i = 1, 2, \ldots, n$:
   (a) Set $\varphi_0 = \varphi(x_i = 0)$ and $\varphi_1 = \varphi(x_i = 1)$.
   (b) Run $B(\varphi_0)$ and $B(\varphi_1)$ each independently some number of times k. Let $b_{0,j}$ be the result of $B(\varphi_0)$ and $b_{1,j}$ be the results of $B(\varphi_1)$ for $j \in [k]$.
   (c) Set $b_0 = \text{majority}(b_{0,1}, \ldots, b_{0,k})$ and $b_1 = \text{majority}(b_{1,1}, \ldots, b_{1,k})$.
   (d) If $b_0 = 0$ and $b_1 = 0$, reject.
   (e) If $b_0 = 1$ and $b_1 = 0$ or $b_1 = 1$, set $\varphi = \varphi_0$.
   (f) If $b_0 = 0$ and $b_1 = 1$, set $\varphi = \varphi_1$.
3. Check that the assignment $x_1, \ldots, x_n$ satisfies ϕ (or that $\varphi = 1$). Output 1 if true and 0 otherwise.

First, note that if $\phi \notin$ 3SAT then $R(\phi)$ will always reject in step (4), regardless of the generated assignment x. Now, suppose that $\phi \in$ 3SAT. The machine $R(\phi)$ tries to generate a satisfying assignment for ϕ. Next, in round i of step (3), $R(\phi)$ uses B to check if φ_0 and φ_1 are satisfiable. It does so by checking each independently some number of times k, and taking the majority.

Let X_i denote the random variable for $b_0 = 0 \wedge b_1 = 0$ in round i. Then, we have

$$\begin{aligned} \Pr[R(\phi) = 0 \mid \phi \in \text{3SAT}] &\leq \sum_{i=1}^{n} \Pr\left[X_i \mid \bigwedge_{j=1}^{i-1} \overline{X}_j\right] \cdot \Pr\left[\bigwedge_{j=1}^{i-1} \overline{X}_j\right] \\ &\leq \sum_{i=1}^{n} \Pr[X_i] \leq n \cdot p, \end{aligned}$$

where p is the probability that $b_0 = 0 \wedge b_1 = 0$ when $\phi \in$ 3SAT. The upper bound follows from upper bounding $\Pr[\wedge_j \overline{X}_j] \leq 1$ and the fact that X_i is independent of $\wedge_j \overline{X}_j$. Therefore, we choose k so that by the Chernoff bound, we have that $p \leq 1/(3n)$. Such a k will be polynomial in n, so overall the machine R runs in polynomial time (since B also runs in polynomial time). Therefore, 3SAT $\in$ **RP** and thus **NP** $\subseteq$ **RP**. $\square$