Solutions to Exercises

1. Show that if $\mathbf{P} = \mathbf{NP}$ for decision problems, then every $\mathbf{NP}$ search problem can be solved in polynomial time.

   Solution Sketch. This is similar to something done in Lecture 7. Let R be the relation that defines an $\mathbf{NP}$ search problem. Define the language L that contains all the pairs (x, z) such that z is the prefix of a solution zz' such that $(x, zz') \in R$. Under the assumption that $\mathbf{P} = \mathbf{NP}$, there is a polynomial time algorithm A that decides L. Now, given x, we can construct a solution y such that $(x, y) \in R$, if such a solution exists, one bit a time using A. (See notes of lecture 7 for more details.)

2. Generalize Theorem 3 in the Notes for Lecture 1. Say that a monotone non-decreasing function $t : \mathbb{N} \rightarrow \mathbb{N}$ is time-constructible if, given n, we can compute $t(n)$ in $O(t(n))$ time. Show that if $t(n)$ is a time-constructible functions then $\mathbf{DTIME}(o(t(n))) \not\subseteq \mathbf{DTIME}(O(t(n) \log t(n)))$.

   Solution Sketch. Let U be the efficient universal Turing machine of Lecture 1. Define the language L that contains all pairs $(\langle M \rangle, x)$ such that U rejects $(\langle M \rangle, (\langle M \rangle, x))$ within $t(n)$ steps, where n is the length of $(\langle M \rangle, x)$. Then L is solvable in time $O(t(n) \log t(n))$. Suppose towards a contradiction that L were solvable in time $o(t(n))$ by a machine T. Then, $U(\langle T \rangle, (\langle T \rangle, x))$ also runs in time $o(t(n))$, where n is the length of $(\langle M \rangle, x)$. For every sufficiently long x, the running time of $U(\langle T \rangle, (\langle T \rangle, x))$ is less than $t(n)$, and so $(\langle T \rangle, x) \in L$ if and only if $U(\langle T \rangle, (\langle T \rangle, x))$ rejects, which happens if and only if $T(\langle T \rangle, x)$ rejects, which is supposed to happen if and only if $(\langle T \rangle, x) \not\in L$

3. Define the class $\mathbf{BPL}$ (for *bounded-error probabilistic log-space*) as follows. A decision problem L is in $\mathbf{BPL}$ if there is a log-space probabilistic Turing machine M such that

   - For every r and every x, $M(r, x)$ halts;
   - If $x \in L$ then $\mathbf{Pr}_r[M(r, x)$ accepts $] \geq 2/3$;
   - If $x \not\in L$ then $\mathbf{Pr}_r[M(r, x)$ accepts $] \leq 1/3$.

   Then

   (a) Prove that $\mathbf{RL} \subseteq \mathbf{BPL}$.

   Solution Sketch. We only need to reduce the error probability from 1/2 to below 1/3, for example by doing two independent repetitions of the algorithm and accepting if and only if both repetitions accept.

   (b) Prove that $\mathbf{BPL} \subseteq \mathbf{SPACE}(O((\log n)^2)$.

Solution Sketch. This is similar to Savitch's algorithm. Let L be a **BPL** language and M the machine required by the definition of **BPL**. Let $p(n)$ be a polynomial upper bound on the running time of M on inputs of length n. We now describe a deterministic $O(\log^2 n)$-space algorithm for L. On input x of length n, we consider the *configuration graph* of $M(\cdot, x)$. A vertex in the graph is a description of a configuration of M on input x: state of the machine, content of the work tape, position of the head on the input tape, position of the head on the work tape. Each vertex has two outgoing edges, corresponding to the two possible next configurations of the machine depending on the bit read from the random tape. The accepting and rejecting configurations have no outgoing edges.

We define a recursive procedure $prob(c_1, c_2, k)$ that given two configurations c_1 and c_2 computes the probability that the machine reaches configuration c_2 starting from configuration c_1 in exactly k steps. The base case when $k = 1$ are treated separately. When $k \geq 2$, then we compute

$$\sum_c prob(c_1, c, \lceil k/2 \rceil) \cdot prob(c, c_2, \lfloor k/2 \rfloor)$$

where the summation is taken over all configurations c. It is enough to represent the probabilities as truncated decimals using $O(\log p(n))$ digits. The truncation introduces errors, but the error in $proc(\cdot, \cdot, p(n))$ is small provided that we use $O(\log p(n))$ digits. The analysis of the recursion is as in Savitch's algorithm.

(c) This last question requires a somewhat different approach: prove that **BPL** $\subseteq$ **P**.

Solution Sketch. Let L be a **BPL** language and M the machine required by the definition of **BPL**. On input x of length n, let C be the number of configurations of $M(\cdot, x)$. Construct a $C \times C$ matrix P such that $P[c_1, c_2] = 1/2$ if c_2 is reachable from c_1 in one step, and $P[c_1, c_2] = 0$ otherwise. For every t, $P^t[c_1, c_2]$ is the probability of reaching configuration c_2 from configuration c_1 in t steps, where P^t is the matrix obtained by multiplying P with itself t times. By computing all powers of P up to the running time of $M(\cdot, x)$ we can compute the accepting probability of $M(r, x)$, and decide if $x \in L$. This time the calculations can be exact: each probability is an integer multiple of $1/2^{p(n)}$, and so it can be represented using a polynomial number of digits.

4. Show that **SIZE**$(n^{O(1)}) \not\subseteq$ **P**.

Solution Sketch. Fix an enumeration $M_1, M_2, \ldots, M_n, \ldots$ of Turing machines. Consider the language L that contains all strings x such that M_n halts on an empty input, where n is the length of x. Then L is undecidable, and, in particular, it is not in **P**. On the other hand, on each input length n, there is a very simple circuit of size $O(1)$ (that either accepts all inputs or rejects all inputs) that solves L on inputs of length n.

5. Show that there is a language in **SPACE**$(2^{n^{O(1)}})$ that does not belong to **SIZE**$(2^{o(n)})$.

Solution Sketch. We know that for every sufficiently large n there is a function $f : \{0,1\}^n \to \{0,1\}$ such that, say, there is no circuit of size $\leq 2^{n/2}$ that computes f. Let h_n be the lexicographically first such function. (The lexicographic order is normally defined over strings, but it is easy to adapt it to functions.) Then define L to be the language that contains all strings x such that $h_n(x) = 1$, where n is the length of x. Then L cannot clearly be solved by circuits of size $\leq 2^{n/2}$ on inputs of length n, but it can be solved in exponential space as follows: given x of length n, enumerate all functions $f : \{0,1\}^n \to \{0,1\}$ in lexicographic order until we find a function that cannot be solved by circuits of size $\leq 2^{n/2}$. The first such function that we encounter is $h_n()$, and then we simply evaluate $h_n(x)$. We can enumerate all functions by enumerating all possible truth-table, which can be done using $O(2^n)$ space (after we are done considering a function, we re-use the same space to write down the truth-table of the next function). In order to check if a given functionc can be solved by a circuit of size $2^{n/2}$, it is enough to enumerate all circuits of size $\leq 2^{n/2}$, using $O(n \cdot 2^{n/2})$ space, and verify if any of the circuits agrees with the function on all inputs. This computation can certainly be done in $2^{O(n)}$ space.

6. In the MAX SAT problem we are given a formula φ in conjunctive normal form and we want to find the assignment of values to the variables that maximizes the number of satisfied clauses. (For example, if φ is satisfiable, the optimal solution satisfies all the clauses and the MAX SAT problem reduces to finding a satisfying assignment.) Consider the following decision problem: given a formula φ in conjunctive normal form and an integer k, determine if k is the number of clauses of φ satisfied by an optimal assignment.

   - Prove that this problem is in **NP** if and only if **NP** = *co***NP**.

     Solution Sketch. Suppose that MAX SAT is in **NP**, and let $V(\cdot,\cdot)$ be the verifier for MAX SAT. Then we can deduce that the co**NP**-complete problem UNSAT (where, given a CNF formula φ we want to decide if it is unsatisfiable) is also in **NP**: a witness that φ is unsatisfiable is a pair (k, y), where k is an integer smaller than the number of clauses of φ and y is such that $V((\varphi, k), y)$ accepts. But if a co**NP**-complete problem belongs to **NP**, it follows that *co***NP** = **NP**.

     Suppose now that **NP** = *co***NP**, and consider the language L that contains pairs (φ, k) such that at least k clauses of φ can be satisfied, and the language L' that contains the pairs (φ, k) such that it is impossible to satisfy k or more clauses of φ. By definition, L is in **NP**, and let $V()$ be its verifier; also L' is in co**NP**, and, by the assumption, also in **NP**, and let $V'()$ be its verifier. We can now put MAX SAT in **NP** by noting that a witness for $(\varphi, k) \in$MAX SAT is a pair (y, y') such that $V((\varphi, k), y)$ and $V'((\varphi, k+1), y')$ both accept.

   - Prove that this problem is in Σ_2.

     Solution Sketch. By definition, $(\varphi, k) \in$MAX SAT if and only if

     $$\exists a.a \text{ satisfies } k \text{ clauses of } \varphi \wedge \forall a'.a' \text{ satisfies } \leq k \text{ clauses of } \varphi$$

which is logically equivalent to the Σ_2 formulation

$$\exists a.\forall a'.(a \text{ satisfies } k \text{ clauses of } \varphi \text{ and } a' \text{ satisfies } \leq k \text{ clauses of } \varphi)$$

7. Define $\mathbf{EXP} = \mathbf{DTIME}(2^{n^{O(1)}})$. Prove that if $\mathbf{EXP} \subseteq \mathbf{SIZE}(n^{O(1)})$ then $\mathbf{EXP} = \Sigma_2$.

   Solution Sketch. Let $L \in \mathbf{EXP}$ and let M be the Turing machine that solves L in time $\leq 2^{p(n)}$ on inputs of length n, where n is the length of the input. Fix a representation of the configurations of $M(x)$; each configuration can be written using $c \cdot 2^{p(n)})$ bits for some constant c. There is a machine M' that, given a string x of length n, and integers $t \leq 2^{p(n)}$ and $i \leq c \cdot 2^{p(n)}$, outputs the i-th bit of the configuration reached by $M(x)$ after t steps. Furthermore, we can implement M' so that it runs in time $2^{O(p(n))}$ on input (x, t, i), where n is the length of x. Since M' solves a decision problem in $\mathbf{EXP}$, there is a family of polynomial size circuits that simulate M'. Let $q(n)$ be a polynomial upper bound to the size of these circuits.

   Our Σ_2 simulation of M, on input x, will "guess" a circuit C of size $q(n)$, then it will verify that for every i and t, the value of $C(x,t,i)$ is consistent with the (constant number of) values $C(x, t-1, \cdot)$ that it depends on. Finally, it will accept if and only if $C()$ predicts that after $2^{p(n)}$ steps $M(x)$ accepts.

$$\begin{aligned} x \in L \quad \text{iff} \quad & \exists C.|C| \leq q(|x|) \\ & \forall t \leq 2^{p(|x|)}, i \leq c \cdot 2^{p(x)} \\ & C(x,t,i) \text{ is consistent with } C(x,t-1,\cdot) \text{ and} \\ & C(x, 2^{p(|x|)}, \cdot) \text{ describes an accepting configuration} \end{aligned}$$

   (A few details are missing, for example one needs to treat the case $t = 0$ separately.)

8. Prove that **ZPP**=**RP**$\cap$co**RP**.

   Solution Sketch. We gave the proof in class.

9. Show that if $\mathbf{NP} \subseteq \mathbf{BPP}$ then $\mathbf{NP} = \mathbf{RP}$.

   Solution Sketch. It is enough to show that if $\mathbf{NP} \subseteq \mathbf{BPP}$ then 3SAT$\in$ **RP**. Let A be a **BPP** algorithm for 3SAT. Given a formula φ with n variables, we first run A on φ. If A rejects, we reject. Otherwise, we try to construct a satisfying assignment for φ one variable at a time. That is, we try instantiating x_1 to 0, and then use A to decide if the resulting formula is satisfiable: if so, then we permanently set x_1 to 0 and proceed with x_2; otherwise we set x_1 to 1 and proceed with x_2. If we manage to construct a satisfying assignment we accept, otherwise we reject, and so on. If φ is unsatisfiable, then we always reject. If φ is satisfiable, then we construct a satisfying assignment and accept provided that the $n+1$ invocations of A that we make are all correct. We can ensure that this happens with high probability by replacing each invocation of A with $O(\log n)$ independent ones and taking the majority answer.

10. Prove that $\mathbf{SPACE}(O(n^{\log n})) \not\subseteq \mathbf{BPP}$.

    Solution Sketch. Consider the language L that contains all pairs $(\langle M \rangle, x)$, where M is a probabilistic machine, such that

    $$\mathbf{Pr}[M(\langle M \rangle, x) \text{ rejects within } n^{(\log n)/3} \text{ steps }] \leq \frac{1}{2}$$

    We see that $L \in \mathbf{SPACE}(O(n^{\log n}))$ and we reach a contradiction if we assume $L \in \mathbf{BPP}$.

11. Change the assumption of Theorem 12 in the notes of Lecture 8 to having a *probabilistic* polynomial time algorithm that on input a formula with exactly one satisfying assignment finds that assignment with probability at least 1/2. Prove that it still follows that $\mathbf{NP} = \mathbf{RP}$.

    Solution Sketch. Use the same proof: the probability of finding an assignment becomes 1/16 instead of 1/8, and it is enough to set $t = 11$.

12. Let $\{X_n\}_{n\geq 1}$ and $\{Y_n\}_{n\geq 1}$ be ensembles (sets) of random variables, where X_n and Y_n take values over $\{0,1\}^n$. Say that $\{X_n\}$ and $\{Y_n\}$ are *indistinguishable* if for every two polynomials p and q and for every large enough n we have that X_n and Y_n are $(p(n), 1/q(n))$-indistinguishable.

    Prove that if $\{X_n\}$ and $\{Y_n\}$ are computationally indistinguishable, and f is a length-preserving (meaning that the length of the output is always equal to the length of the input) polynomial time computable function, then $\{f_n(X_n)\}$ and $\{f_n(Y_n)\}$ are also computationally indistinguishable.

    Solution Sketch. Let $t(n)$ be a polynomial upper bound to the size of a circuit that computes f_n. Suppose that $\{f_n(X_n)\}$ and $\{f_n(Y_n)\}$ are not computationally indistinguishable. Then there are polynomials $p(n)$ and $q(n)$ such that, for infinitely many n, $f_n(X_n)$ and $f_n(Y_n)$ are not $(p(n), 1/q(n))$-indistinguishable, that is, there is a circuit C_n of size $\leq p(n)$ such that

    $$|\mathbf{Pr}[C_n(f_n(X_n)) = 1] - \mathbf{Pr}[C_n(f_n(Y_n))]| \geq 1/q(n) \ .$$

    Considering that $C_n(f_n(\cdot))$ is computable by a circuit of size $t(n) + p(n)$, we deduce that there are infinitely many n such that X_n and Y_n are not $(t(n) + p(n), 1/q(n))$-indistinguishable, and so $\{X_n\}_{n\geq 1}$ and $\{Y_n\}_{n\geq 1}$ are not indistinguishable.

13. Prove that there is an ensemble $\{X_n\}$ that is computationally indistinguishable from the ensamble of uniform distributions $\{U_n\}$, even though only $n^{\log n}$ elements of $\{0,1\}^n$ have non-zero probability in X_n.

Solution Sketch. For every sufficiently large n we show that there is a multi-set S_n of size $n^{\log n}$ such that the uniform disribution over S is $(n^{(\log n)/4}, n^{-(\log n)/4})$-indistinguishable from uniform.

Fix a circuit C, pick at random a multiset S by picking (with replacement) $n^{\log n}$ from $\{0,1\}^n$. We want to compute

$$\mathbf{Pr}_S[|\mathbf{Pr}_{x\sim S}[C(x)=1]-\mathbf{Pr}[C(U_n)=1]|\geq \epsilon]$$

Define $p := \mathbf{Pr}[C(U_n)=1]$ and $N = n^{\log n}$, then, equivalently, we want to compute

$$\mathbf{Pr}_x[||\{x\in S : C(x)=1\}|-p\cdot N|\geq \epsilon\cdot N]\ .$$

Define random 0/1 variables $X_1,\ldots,X_n$ where $X_i=1$ iff $C(x_i)=1$, where x_i is the i-th element that we select to be in S. Then the X_i are independent, and each of them has a probability p of being 1. If then follows from Chernoff bounds that

$$\mathbf{Pr}_x[||\{x\in S : C(x)=1\}|-p\cdot N|\geq \epsilon\cdot N]\leq e^{-\Omega(\epsilon^2 N)}.$$

In particular, if we fix $\epsilon = n^{-(\log n)/4}$, we have

$$\mathbf{Pr}_S[|\mathbf{Pr}_{x\sim S}[C(x)=1]-\mathbf{Pr}[C(U_n)=1]|\geq n^{-(\log n)/4}]\leq e^{-\Omega(n^{(\log n)/2})}$$

The number of circuits of size $n^{(\log n)/4}$ is $e^{O((\log n)^2\cdot n^{(\log n)/4})}$ which is much smaller than $e^{\Omega(n^{(\log n)/2})}$, so even after taking a union bound we have

$$\mathbf{Pr}_S[\exists C.size(C)\leq n^{(\log n)/4}\wedge|\mathbf{Pr}_{x\sim S}[C(x)=1]-\mathbf{Pr}[C(U_n)=1]|\geq n^{-(\log n)/4}]<1$$

In particular, there exists a set S such that

$$\forall C.size(C)\leq n^{(\log n)/4}.|\mathbf{Pr}_{x\sim S}[C(x)=1]-\mathbf{Pr}[C(U_n)=1]|\leq n^{-(\log n)/4}$$

and we let X_n be the uniform distribution over such a set S.

14. Prove that if pseudorandom generators of stretch $2n$ exist, then one-way functions exist.

Solution Sketch. Let G be a pseudorandom generator of stretch $2n$, we prove that it is a one-way function. If it were not a one-way function, then there would a family of polynomial size circuits C_n and a polynomial $q(n)$ such that for infinitely many n

$$\mathbf{Pr}_x[C_n(G(x))=x' : G(x')=G(x)]\geq 1/q(n)$$

Construct now a family of circuits C'_n, still of polynomial size, such that $C'_n(y)=1$ if and only if $G(C_n(y))=y$. Then we have $\mathbf{Pr}[C'_n(G(x))=1]\geq 1/q(n)$, while $\mathbf{Pr}[C'_n(y)=1]\leq 1/2^n$, contradicting the assumption that G is a pseudorandom generator.

15. Prove that if a permutation f has a hard-core predicate B, then f is a one-way permutation.

Solution Sketch. Let f be a permutation and B be a hard-core predicate for f. Suppose that f is not one-way, then there are polynomials $p()$ and $q()$ such that for infinitely many n there is a circuit C_n such that

$$\mathbf{Pr}[C_n(f(x)) = x] \geq 1/q(n)$$

Consider now the following probabilistic process: on input y of length n, we compute $x' = C_n(y)$. If $f(x') = y$ then we output $B(x')$, otherwise we output a random bit. The output of this process, with probability at least $1/2 + 1/2q(n)$, correctly equals $B(f^{(-1)}(y))$, and there is a fixed choice for the final random choice such that the correctness probability is at least as good. For that fixed choice, the above described process can be realized by a polynomial size circuit. This proves that B is not a hard-core predicate for f, and we reach a contradiction.

16. Prove that if $\mathbf{P} = \mathbf{NP}$ then there cannot be any pseudorandom generators, even of stretch $n+1$.

Solution Sketch. Suppose $\mathbf{P} = \mathbf{NP}$ and let $\{G_n\}$ be a family of polynomial time computable functions mapping n bits into $n+1$ bits. Let C_n be a circuit that on input y outputs 1 if $y = G_n(x)$ for some x and 0 otherwise. Since the family $\{C_n\}$ solves a problem in $\mathbf{NP}$, that, by assumption, can be solved in polynomial time, C_n can be realized as a family of polynomial size circuits.

We have $\mathbf{Pr}_x[C_n(G(x)) = 1] = 1$, but $\mathbf{Pr}_r[C_n(r) = 1] \leq 1/2$ because each string r has probability $1/2^{n+1}$ but only 2^n of them are possible outputs of G_n. So we have

$$|\mathbf{Pr}[C_n(G_n(x)) = 1] - \mathbf{Pr}[C_n(r) = 1]| \geq 1/2$$

where the C_n have polynomial size, and $\{G_n\}$ cannot be a pseudorandom generator.