

1. **BPP = Almost-P.** Let $A : \{0,1\}^* \rightarrow \{0,1\}$ be a function from the set of binary strings of any length to $\{0,1\}$. The class P^A is then the class of languages computable in polynomial time with oracle access to A . That is a function that can be computed in polynomial time by a program that can evaluate A on any given string in constant time. The class **Almost-P** is the class of languages L so that for a random A we have that $L \in P^A$ with probability 1. Prove that **BPP = Almost-P**.

- (a) Show that **BPP** \subset **Almost-P**. Hint: Use the oracle to produce random inputs. You might need an argument along the lines of the one we say in the proof of Adelman's Theorem to ensure that you get the correct answer on *all* inputs. [15 points]

Solution: Consider an arbitrary language $L \in \mathbf{BPP}$ with a **BPP** algorithm π . By the standard error reduction techniques we have seen in class, we may assume without loss of generality that π errs with probability at most 2^{-2n} (on inputs of length n).

Let A be a randomly chosen function, and consider feeding the bits $r_1 = A(0), r_2 = A(00), r_3 = A(000), \dots$ into π for its randomness. By a union bound, the probability that π produces the wrong answer on any input of length n is at most $2^n \cdot 2^{-2n} \leq 2^{-n}$. Thus, the expected number of input lengths for which this P^A algorithm fails is at most

$$\sum_{i=0}^{\infty} 2^{-i} = 2 < \infty.$$

In other words, with probability 1 (over the choice of A) the algorithm only errs on a finite number of input lengths. One can then hard code specific "random" strings on which π is always correct for these remaining input lengths to obtain an algorithm that (with probability 1) is correct on all input lengths.

- (b) Show that **Almost-P** \subset **BPP**. Hint: If $L \in \mathbf{Almost-P}$ there must be some algorithm $B(x, A)$ that with positive probability over A correctly determines whether $x \in L$ for all x . A basic fact about measure theory implies that there is some finite set of strings s_i and values $v_i \in \{0,1\}$ so that if we condition on $A(s_i) = v_i$ for all i , then this probability is at least $2/3$. Given this, build a **BPP** algorithm by simulating this program on a random A with this conditioning. [15 points]

Solution: We will first show the solution contingent on the hint, and afterwards provide a proof of the hint. (This latter part is not required for your homework.)

Consider an arbitrary language $L \in \mathbf{Almost-P}$, and let $B(x, A)$ be the algorithm guaranteed by the hint. For our **BPP** algorithm, we simply run B on x , and simulate the oracle calls to A as follows. Whenever B queries A with some string S , we check if S is one of the strings in $\{s_i\}_i$ (from the hint). If it

is, we output the prescribed value v_i . Otherwise, we simply return a random answer if this is the first time S is queried, or return whatever the previous random assignment was if S has already been queried. This has the effect of sampling A conditioned on the values from the hint, so the probability this algorithm succeeds is at least $2/3$.

We now prove the hint. Again, let $L \in \mathbf{Almost-P}$. By the definition of $\mathbf{Almost-P}$, we have that with probability 1 over A , at least one of the only countably many programs computes L . By basic measure theory, there must exist at least one program which correctly computes L with positive probability. Fix that program $B(x, A)$ in what follows.

We want to condition on a finite number of values in order to boost the success probability of B to at least $2/3$. Toward that end, we claim the event that B computes L with oracle A is the complement of a union of subcubes. Indeed, B computes L unless there is some input x where it produces the wrong answer, but the output of B on x only depends on finitely many values of A . Hence we can express the set

$$W := \{A : \exists x \text{ such that } B(x, A) \text{ is wrong}\}$$

as $\{0, 1\}^* \setminus \bigcup_i C_i$, where every C_i is a subcube. Moreover, we may assume the subcubes are disjoint, since $C_i \setminus \bigcup_{j < i} C_j$ is also a union of subcubes.

Recall that B successfully computes L with positive probability, say ε , so $\sum_i \mu(C_i) = 1 - \varepsilon$, where $\mu(C_i)$ is the measure of C_i . By another basic measure theoretic result, there exists a value N such that $\sum_{i > N} \mu(C_i) \leq \varepsilon/2$. If we now condition on the event E of avoiding these first N subcubes (which occurs with probability ε), B only errs with probability at most $\varepsilon/2$. That is, under this conditioning, the algorithm B succeeds with probability at least $2/3$. We conclude by observing that since E is itself a union of subcubes, there must be some subcube in E on which the conditional success probability of B is at least $2/3$.

2. **Local Maximum Concentration.** For $n \geq 2$ let X_n be the number of local maxima of a random permutation of $\{1, 2, \dots, n\}$. Recall that we showed in class that $\mu_n := \mathbb{E}[X_n] = (n + 1)/3$.

- (a) Compute $\text{Var}(X_n)$. [20 points]

Solution: Let Y_1, Y_2, \dots, Y_n be indicator random variables, where Y_i is 1 if i is a local maxima and 0 otherwise. Note that $X_n = \sum_i Y_i$, so we can express

$$\text{Var}(X_n) = \sum_{i,j} \text{Cov}(Y_i, Y_j),$$

where we recall the covariance

$$\begin{aligned}\text{Cov}(Y_i, Y_j) &= \mathbb{E}[Y_i Y_j] - \mathbb{E}[Y_i] \mathbb{E}[Y_j] \\ &= \Pr[i, j \text{ are both local maxima}] \\ &\quad - \Pr[i \text{ is local maxima}] \Pr[j \text{ is local maxima}].\end{aligned}$$

From here, we proceed by cases according to the boundaries and how far apart i and j are. For the moment, suppose $n \geq 4$; we will handle the remaining cases afterwards.

- If $i = 1$ or n , then $\text{Cov}(Y_i, Y_i) = \frac{1}{2} - \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.
- If $i = 1$ and $j = 2$, or if $i = n - 1$ and $j = n$, then $\text{Cov}(Y_i, Y_j) = 0 - \frac{1}{2} \cdot \frac{1}{3} = -\frac{1}{6}$. (Two adjacent points cannot both be relative maxima.)
- If $i = 1$ and $j = 3$, or if $i = n - 2$ and $j = n$, then $\text{Cov}(Y_i, Y_j) = \frac{5}{4!} - \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{24}$. (Consider the four points that determine whether 1 and 3 are local maxima. There are $4!$ equally likely relative orders, and a straightforward counting argument, or even brute force, gives that five of these arrangements satisfy the relative maxima condition.)
- If $1 < i < n$, then $\text{Cov}(Y_i, Y_i) = \frac{1}{3} - \frac{1}{3} \cdot \frac{1}{3} = \frac{2}{9}$.
- If $1 < i < n - 1$, then $\text{Cov}(Y_i, Y_{i+1}) = 0 - \frac{1}{3} \cdot \frac{1}{3} = -\frac{1}{9}$.
- If $1 < i < n - 2$, then $\text{Cov}(Y_i, Y_{i+2}) = \frac{16}{5!} - \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{45}$.
- If $1 \leq i < j \leq n$ and $i \leq j - 3$, then $\text{Cov}(Y_i, Y_j) = 0$. (For points at distance at least 3, the events that they are local maxima are independent.)

Combining everything together (doubling counts where appropriate to compensate for the cases when $j < i$), we find that

$$\text{Var}(X_n) = \frac{2}{4} + \frac{2(n-2)}{9} + 2 \left(\frac{-2}{6} + \frac{2}{24} + \frac{-(n-3)}{9} + \frac{n-4}{45} \right) = \frac{2(n+1)}{45}$$

whenever $n \geq 4$. For the smaller values, we can similarly compute that

$$\text{Var}(X_2) = 0 \quad \text{and} \quad \text{Var}(X_3) = \frac{2}{9}.$$

- (b) Using Chebyshev's Inequality, find a number T_n so that $X_n - \mu_n < T_n$ with probability at least 99.99%. [5 points]

Solution: We assume $n \geq 3$, since one can set T_2 to be any positive value. By Chebyshev's Inequality, we have that any $t > 0$ satisfies

$$\Pr[X_n - \mu_n \geq t] \leq \Pr[|X_n - \mu_n| \geq t] \leq \frac{\text{Var}(X_n)}{t^2},$$

so we can set $T_n = 100\sqrt{\text{Var}(X_n)}$ to ensure this probability is at most 0.0001.

By the previous part, this is

$$T_3 = 100\sqrt{2/9} \approx 47.14$$

$$T_n = 100\sqrt{2(n+1)/45} \approx 21.08(n+1) \text{ for } n \geq 4.$$

- (c) While we cannot use Chernoff bounds directly, we can write X_n as the sum of three random variables each of which is a sum of bounded, independent random variables. Applying Chernoff bounds to this, find a T'_n so that $X_n - \mu_n < T'_n$ with probability at least 99.99%. [20 points]

Solution: We assume n is sufficiently large for simplicity; a similar, but more involved, analysis can handle the case of all $n \geq 2$.

As in part (a), let Y_1, Y_2, \dots, Y_n be indicator random variables, where Y_i denotes whether or not $i \in [n]$ is a local maxima in a random permutation of $[n]$. Recall that Y_i and Y_j are independent for any i, j of distance $|i - j| \geq 3$. Thus, we separately consider

$$X_n^{(0)} = \sum_{i: i \bmod 3=0} Y_i, \quad X_n^{(1)} = \sum_{i: i \bmod 3=1} Y_i, \quad X_n^{(2)} = \sum_{i: i \bmod 3=2} Y_i,$$

where $X_n^{(0)}, X_n^{(1)}, X_n^{(2)}$ are each sums of independent random variables in the range $[0, 1]$. We denote their expectations by $\mu_n^{(0)}, \mu_n^{(1)}, \mu_n^{(2)}$, respectively. Our goal is to find T'_n such that $\Pr[X_n - \mu_n \geq T'_n] \leq 0.0001$. Note that

$$\begin{aligned} \Pr[X_n - \mu_n \geq T'_n] &= \Pr \left[\sum_i (X_n^{(i)} - \mu_n^{(i)}) \geq T'_n \right] \\ &\leq \Pr \left[\bigvee_i \left(X_n^{(i)} - \mu_n^{(i)} \geq \frac{T'_n}{3} \right) \right] \\ &\leq \sum_i \Pr \left[X_n^{(i)} - \mu_n^{(i)} \geq \frac{T'_n}{3} \right], \end{aligned}$$

where the final inequality follows from the union bound. Hence, it suffices to find a T'_n such that $\Pr \left[X_n^{(i)} - \mu_n^{(i)} \geq \frac{T'_n}{3} \right] \leq 1/30000$ for each $i \in \{0, 1, 2\}$.

By Chernoff's Inequality, we have for any such i and $\delta \in [0, 1]$ that

$$\Pr[X_n^{(i)} - \mu_n^{(i)} \geq \delta \mu_n^{(i)}] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^{\mu_n^{(i)}} \leq \exp \left(\frac{-\mu_n^{(i)} \cdot \delta^2}{3} \right).$$

Crudely lower bounding each $\mu_n^{(i)}$ by $\frac{1}{3}(\frac{n}{3} - 1) \geq \frac{n}{10}$ (for large n), we find that setting $\delta = \sqrt{30 \ln(30000)/n}$ yields the desired upper bound.

In other words, we can choose T'_n to be $3\delta \max_i \mu_n^{(i)}$. By the crude upper

bound $\frac{1}{2}(\frac{n}{3} + 1) \leq \frac{n}{5}$, we conclude by setting

$$T'_n = 3 \cdot \left(\sqrt{\frac{30 \ln(30000)}{n}} \right) \cdot \frac{n}{5} \approx 10.55\sqrt{n}.$$

Note that this is noticeably smaller than the bound obtained in the previous part via Chebyshev's Inequality.

3. **Yao's Minimax Principle for BPP.** We've discussed Yao's Minimax Principle for the runtime of an algorithm, but there is also a version for the probability of success. Consider a query problem where one is trying to find the best possible randomized algorithm that makes at most a bounded number (N) queries. We say that such an algorithm A succeeds with probability p in the worst case if for all inputs x , A makes at most N queries and with probability at least p returns the correct answer for that input. On the other hand, if we fix a distribution D over inputs, the average case probability of success of such an algorithm A with respect to this distribution is the probability that if an input x is selected according to D and A is run on this input that A computes the correct answer for that input.

Prove that for any such problem, the maximum possible worst case probability of success of a randomized algorithm A is the same as the minimum over distributions D over inputs of the maximum possible average case probability of success of a deterministic algorithm A .

Solution: Consider the following zero-sum game interpretation of the problem. The algorithm tries to choose its outputs to maximize the chance of getting the correct answer, while "nature" attempts to thwart the algorithm. Whichever player succeeds receives a point. We can construct a pay-off matrix $M(A, x)$ to be 1 if the deterministic algorithm A is correct on input x , and 0 otherwise.

Recall that a randomized algorithm is simply a distribution over deterministic algorithms. Letting α denote such a distribution, we can express the maximum possible worst case probability of success of a randomized algorithm with a bounded number of queries by

$$\max_{\alpha} \min_x \mathbb{E}_{A \sim \alpha} [M(A, x)],$$

where A is a deterministic algorithm. Similarly, we can express the minimum over distributions D over inputs of the maximum possible average case probability of success of a deterministic algorithm A with a bounded number of queries by

$$\min_D \max_A \mathbb{E}_{x \sim D} [M(A, x)].$$

We conclude by observing that Yao's minimax principle guarantees these quantities

are the same. That is,

$$\max_{\alpha} \min_x \mathbb{E}_{A \sim \alpha} [M(A, x)] = \min_D \max_A \mathbb{E}_{x \sim D} [M(A, x)].$$