Question 1 (Producing Unusual Probability Distributions, 20 points). Often it is assumed that your random algorithm is given access only to a string of uniform random bits. However, it is sometimes necessary to generate (or approximate) samples from other probability distributions.

(a) Suppose that you want to simulate a biased coin. In particular, you would like to produce a random variable that returns 1 with probability $p$ and 0 otherwise for some specified $0 \leq p \leq 1$. Show how to generate a sample from this distribution using an expected $O(1)$ uniform random bits. [10 points]

(b) Suppose instead that you need to simulate an explicitly given probability distribution on $\{1, 2, 3, \ldots, n\}$ (so you are given the probability that your output should be $i$ for each $i$ as an explicit real number). Show how to simulate a sample from this distribution using an expected $O(\log(n))$ bits (in fact a careful analysis should allow you to use only $\log_2(n) + O(1)$ bits). [10 points]

Solution 1

(a) Idea: Let $p(1) = p$ and $p(0) = 1 - p$. Toss a uniform coin. If the output is $i \in \{0, 1\}$ and $p(i) \geq p(1 - i)$ return $i$. Else repeat the process with the new values of $p(0)$ and $p(1)$ which ensures that the final probability of returning any $i$ is $p(i)$. Now at each step we have a lower bound of $1/2$ on the probability of termination so the expected number of iterations will be less than or equal to 2.

Algorithm $A(p)$:
Generates a random bit $b$

- If $p \geq 1/2$
  - If $b=1$ return 1
  - else return $A(2(p - 1/2))$
- Else
  - If $b=0$ return 0
  - else return $A(2p)$

Correctness: Let the binary representation of $p$ be $(0, p_1p_2 \ldots p_i \ldots)$. We return 1 at the $i^{th}$ iteration if and only if $b = 1$ and $p_i = 1$. So,

$$
P[\text{output is } 1] = \sum_i P[A \text{ terminates at the } i^{th} \text{ iteration with output } 1]
= \sum_i \frac{p_i}{2} \times P[A \text{ doesn’t terminate at any of the first } i - 1 \text{ iterations}]
= \sum_i \frac{p_i}{2^i} = p.
$$
(b) **Algorithm**: W.l.o.g. assume \( n \) to be a power of 2. We will first partition the set into two sets of size \( \frac{n}{2} \) each. Let \( p \) be the combined probability of all the elements of the first partition. We will simulate \( A(p) \) until we get any output. If the output is 1, we will throw the second partition and scale the probabilities of the first partition appropriately to make the sum 1, and then repeat the whole process again. If the output is 2, we will do the opposite. We will iterate until we get a singleton set. This last element left will be the output.

**Correctness**: By using the analysis in part(a) and applying mathematical induction we can prove that any of the \( n-2 \) subsets (that can be created by the above process) is selected with the probability equal to the sum of the probabilities of all its elements. Thus, each element is returned with the appropriate probability.

We perform the partitioning step \( \lceil \log n \rceil \) times. Let \( X_i \) be the number of iterations the algorithm \( A \) takes between the \( i-1 \)th and the \( i \)th partitioning. Using the same argument as in part(a) we can show that \( \forall i \ E[X_i] < 2 \). Let \( X = \sum_i X_i \) be the total number of iterations. By linearity of expectation \( E[X] < 2 \log n + 2 \).

**Question 2** (Expectation and Tail Bounds, 25 points). Let \( U \) be a set of size \( 2n \) (for \( n \) some positive integer). Let \( S \) and \( T \) be two uniformly chosen random subsets of \( U \) each of size \( n \). Let \( X = |S \cap T| \).

(a) Compute the mean \( \mu = E[X] \). [5 points]

(b) Compute \( \text{Var}(X) \). [5 points]

(c) Use the Chebyshev bound to compute an interval \( I \) so that \( X \in I \) with probability at least \( 1 - n^{-1/2} \). [5 points]

(d) Compute the central fourth moment \( E[(X - \mu)^4] \) of \( X \), and use it to find a smaller interval \( I \) so that \( X \in I \) with probability at least \( 1 - n^{-1/2} \). [5 points]

(e) Can a better bound by obtained by applying a Chernoff bound? Why or why not? [5 points]

**Solution 2**

(a) Let \( X_{i,A} \) be the indicator random variable of the event \( i \in A \) (where \( A \in \{S, T\} \)), i.e.

\[
X_{i,A} = \begin{cases} 
1 & i \in A \\
0 & \text{otherwise}
\end{cases}
\]

and let \( X_i \) be the indicator random variable of the event \( i \in S \cap T \), i.e.

\[
X_i = \begin{cases} 
1 & i \in S \cap T \\
0 & \text{otherwise}
\end{cases}
\]

\[
E[X_{i,S}] = E[X_{i,T}] = \left( \frac{2n-i}{2n} \right) = \frac{1}{2} \cdot E[X_i] = E[X_{i,S}X_{i,T}] = E[X_{i,S}]E[X_{i,T}] = 1/4 \quad \text{(as the events of choosing} \ S \ \text{and} \ T \ \text{from} \ U \ \text{are independent). The random variable} \ X = |S \cap T| \ \text{can be expressed by the sum} \ \sum_{i=1}^{2n} X_i. \ \text{Thus, by the linearity of expectation} \ E[X] = n/2.
\]
(b) \[
\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2
\]
\[
= \sum_i \mathbb{E}[X_i^2] + \sum_{i \neq j} \mathbb{E}[X_i X_j] - \frac{n^2}{4}
\]
\[
= \sum_{i \neq j} \mathbb{E}[X_i X_j] + \frac{n}{2} - \frac{n^2}{4} \quad \text{(since } \mathbb{E}[X_i^2] = \mathbb{E}[X_i])
\]
\[
= \sum_{i \neq j} \mathbb{E}[X_{i,S} X_{i,T} X_{j,S} X_{j,T}] + \frac{n}{2} - \frac{n^2}{4}
\]
\[
= (2n)(2n-1) \left[ \left( \frac{n-2}{n} \right)^2 \right] + \frac{n}{2} - \frac{n^2}{4}
\]
\[
= \frac{n^2}{4(2n-1)}
\]
\[
= \frac{n}{8} + O(1)
\]

(c) \(X\) can take values from the interval \([0, n]\) with \(\mathbb{E}[X] = n/2\). We want to find an interval \(I\) of the form \([n/2 - x, n/2 + x]\) such that \(X\) lies in \(I\) with probability at least \(1 - \frac{1}{\sqrt{n}}\). Using \(n/4\) as an upper bound on \(\text{Var}(X)\) and applying the Chebyshev bound, we get

\[
\mathbb{P}[|X - \mu| \leq k\sigma] \geq 1 - \frac{1}{k^2}
\]
\[
\mathbb{P}[|X - \frac{n}{2}| \leq \sqrt{n}\sigma] \geq 1 - \frac{1}{\sqrt{n}}
\]
\[
\mathbb{P}[|X - \frac{n}{2}| \leq \sqrt{n}\sqrt{n/4}] \geq 1 - \frac{1}{\sqrt{n}}
\]
\[
\mathbb{P}[|X - \frac{n}{2}| \leq \frac{n^2}{2}] \geq 1 - \frac{1}{\sqrt{n}}
\]
\[
\mathbb{P}[X \in \{ \frac{n - n^2}{2}, \frac{n + n^2}{2} \}] \geq 1 - \frac{1}{\sqrt{n}}
\]
(d) 
\[ \mathbb{E}[X] = \frac{n}{2} \quad \text{and} \quad \mathbb{E}[X^2] = \frac{n^3}{2(2n-1)} \]

\[ \mathbb{E}[X^3] = \mathbb{E}[X_iX_jX_k] \]
\[ = \sum_{i,j,k} \mathbb{E}[X_i,SX_j,SX_k,S]\mathbb{E}[X_i,TX_j,TX_k,T] \]
\[ = \sum_{i,j,k} \mathbb{E}[X_i,SX_j,SX_k,S]^2 \]
\[ = \sum_{i=j=k} \mathbb{E}[X_i,S]^2 + 3 \sum_{i \neq j \neq k \neq i} \mathbb{E}[X_i,SX_j,SX_k,S]^2 \]
\[ = \frac{n^3(n+1)}{4(2n-1)} \]

\[ \mathbb{E}[X^4] = \sum_{i,j,k,l} \mathbb{E}[X_i,SX_j,SX_k,SX_l,S]^2 \]
\[ = \frac{n^3(n^3 + n^2 - 3n - 1)}{4(2n-1)(2n-3)} \]

\[ \mathbb{E}[(X - \mu)^4] = \mathbb{E}[X^4 - 4X^3\mu + 6X^2\mu^2 - 4X\mu^3 + \mu^4] \]
\[ = \mathbb{E}[X^4] - 4\mu \mathbb{E}[X^3] + 6\mu^2 \mathbb{E}[X^2] - 4\mu^3 \mathbb{E}[X] + \mu^4 \]
\[ = \frac{n^3(3n - 4)}{16(2n-1)(2n-3)} \]
\[ = \frac{3}{64}n^2 + O(n) \]

Using \( n^2/16 \) as an upper bound on \( \mathbb{E}[(X - \mu)^4] \) and applying the Chebyshev bound, we get
\[ \mathbb{P}[|X - \mu| \leq k\sqrt{\mathbb{E}[(X - \mu)^4]}] \geq 1 - \frac{1}{k^4} \]
\[ \mathbb{P}[|X - \frac{n}{2}| \leq \sqrt{n} \sqrt{n^2/16}] \geq 1 - \frac{1}{\sqrt{n}} \]
\[ \mathbb{P}[|X - \frac{n}{2}| \leq \frac{n^3}{2}] \geq 1 - \frac{1}{\sqrt{n}} \]
\[ \mathbb{P}[X \in \left(\frac{n - n^\frac{3}{2}, n + n^\frac{3}{2}}{2}\right)] \geq 1 - \frac{1}{\sqrt{n}} \]

Using the fourth moment we got an interval of size \( n^\frac{3}{2} \), which is smaller than the size of the interval we got using variance (\( n^\frac{3}{4} \)).

(e) Chernoff bound is useful when dealing with sum of random variables. In this case the variables \( X_i,S \) and \( X_j,S \) are not independent and thus we can’t apply Chernoff bound.

**Question 3** (Probability Amplification, 15 points). Suppose that for some decision problem, we have a BPP algorithm, that on any instance computes the correct answer with probability at least 2/3. We wish to improve the probability of error by running the algorithm \( n \) times on the same input using independent randomness between trials and taking the most common result. Using the Chernoff bounds, give an upper bound on the probability that this new algorithm produces the incorrect result.
Solution 3  Let $X_i$ be the indicator random variable of the success of the $i^{th}$ iteration of the algorithm, i.e.

$$X_i = \begin{cases} 1 & \text{$i^{th}$ iteration gives correct answer} \\ 0 & \text{otherwise} \end{cases}$$

and let $X = \sum_{i=1}^{n} X_i$ be the sum of all these $n$ independent random variables. $\mathbb{E}[X_i] = 2/3$, so by the linearity of expectation, we get $\mathbb{E}[X] = 2n/3$. As we choose to go by the majority our failure probability is defined by the event $X \leq n/2$, i.e., majority of the iterations gave the wrong answer. Using Chernoff bound we get,

$$\mathbb{P}[X \leq n/2] = \mathbb{P}[X \leq \mathbb{E}[X](1-1/4)] \leq \left[ e^{\frac{1}{4}} \times \left( \frac{3}{4} \right)^{\frac{3}{4}} \right]^{-\frac{3}{4}n} \leq (0.99)^n,$$

which is exponentially small. Any value of $\mathbb{E}[X]$ greater than $2/3$ will only make this error probability smaller.

Question 4 (Complexity Theory, 20 points). Recall that NP is the class of languages $L$ so that there is a short proof of membership in $L$. In particular, a language $L$ is in NP if and only if there exists a polynomial time, deterministic algorithm $A(x,s)$ so that

$$x \in L \iff \text{There exists } s \text{ so that } A(x,s) \text{ accepts.}$$

(a) Show that $NP \supseteq RP$. [5 points]

(b) Show that if $BPP \supseteq NP$ that $NP = RP$. [15 points]

Solution 4

(a) Any language $L \in RP$ has a randomized polynomial time algorithm $A$ which accepts any input $x \in L$ on more than $1/2$ of its random strings, and rejects any input $x \notin L$ on all its random strings. Treating these random strings as certificates we get an $NP$ algorithm for $L$, proving $RP \subseteq NP$.

(b) For proving $NP = RP$ it’s enough to give an $RP$ algorithm for the $NP$-complete problem $SAT$, i.e., it’s enough to design a randomized algorithm which satisfies the following three properties:

1. Runs in time polynomial in the size of the input (boolean) formula;
2. Rejects with probability 1 if the formula is unsatisfiable; and
3. Accepts with probability greater than $1/2$ if the formula is satisfiable.

If $NP \subseteq BPP$, there will be some $BPP$ algorithm $A$ for $SAT$. W.l.o.g. we can assume that it’s failure probability is at most $\frac{1}{2^n}$, where $n$ is the number of variables of the input formula.

Idea : We will try to construct a satisfying assignment by using the algorithm $A$ as a black-box. If the input formula $\phi(x_1, x_2, \ldots, x_n)$ is satisfiable we know that there is at least one satisfying assignment and that $A$ will accept (w.h.p.). If $A$ accepts, we will pick a variable $x_i$ and set its value to 0 and feed the modified formula to $A$ (for example if $\phi(x_1, x_2, x_3) = (x_1 \lor x_2) \land x_3$, setting $x_2 = 0$ gives $\phi(x_1, 0, x_3) = x_1 \land x_3$ and setting $x_2 = 1$ gives $\phi(x_1, 1, x_3) = x_3$). If $A$ accepts this modified formula we know that (w.h.p.) there is a satisfying assignment with $x_i = 0$, and if $A$ rejects we know that (w.h.p.) there is a satisfying assignment with $x_i = 1$. Repeating this process $n$ times we will get (w.h.p.) a satisfying assignment for $\phi$. Since we want to reject with probability 1 in the case $\phi$ is unsatisfiable, we will do a final check to see if actually $A$ gave us the right answers all $n + 1$ of the times. We will do so by simply computing $\phi$ on this final assignment which we constructed, and then rejecting if $\phi$ is not satisfied.

Algorithm : For any input formula $\phi(x_1, x_2, \ldots, x_n)$ the $RP$ algorithm will run as follows:
→ Run $\mathcal{A}(\phi(x_1, x_2, \ldots, x_n))$, reject if $\mathcal{A}$ rejects, else go to the next step.

→ For $i = n$ to 1, do
  • run $\mathcal{A}(\phi(x_1, x_2, \ldots, x_{i-1}, 0, a_{i+1}, \ldots, a_n))$;
  • if $\mathcal{A}$ accepts set $a_i = 0$, else set $a_i = 1$.

→ Reject if $\phi(a_1, a_2, \ldots, a_n) = 0$, else accept.

**Correctness :** The algorithm clearly satisfies property 1 and 2 since it runs $\mathcal{A}$ only $n + 1$ number of times, and comes up with an actual satisfying assignment before accepting. For property 3 consider the case when the formula is satisfiable and each of the $n + 1$ simulations of the algorithm $\mathcal{A}$ gives us the correct answer. The assignment $(a_1, a_2, \ldots, a_n)$ will actually be a satisfying assignment of $\phi$, and thus we will accept. The probability of this not happening can easily be computed by a simple union bound over the error probabilities of all the $n + 1$ steps, which is upper bounded by $\frac{n+1}{2^n}$. Therefore, our algorithm satisfies all the three properties and thus is an $RP$ algorithm for $SAT$ resulting in $NP = RP$.

**Question 5** (Yao’s Principle for Distribution Testing, 20 points). In distribution testing, you are given $n$ independent samples from a distribution $X$ that is guaranteed to either be in some family $\mathcal{C}_1$ of distributions or some other family $\mathcal{C}_2$ of distributions on the same domain. Your objective is to have a randomized algorithm that for any $X$ determines which family it is a member of with at least $\frac{2}{3}$ probability over the set of samples chosen. Show that this is always possible to do unless there is some ensemble $\mathcal{E}$ of probability distributions taken from $\mathcal{C}_1 \cup \mathcal{C}_2$ (that is a probability distribution over probability distributions) so that if $X$ is taken at random from $\mathcal{E}$ and $n$ independent samples are then chosen from $X$, there is no deterministic algorithm that given these samples can determine whether $X$ was taken from $\mathcal{C}_1$ or $\mathcal{C}_2$ with $\frac{2}{3}$ success rate or better.

**Solution 5** Construct a matrix $M$, each of whose columns represents a deterministic algorithm (which takes $n$ points from the domain $A$ as input, and outputs 1 or 2), and each of whose rows represents a distribution from the set $\mathcal{C}_1 \cup \mathcal{C}_2$. Since for our purpose two different algorithms with the same “input to output mapping” makes no difference, we can assume that the matrix is of finite size. For any $i, j$ fill the entry $M(i, j)$ by the success probability of the algorithm $j$, when the samples are drawn from the distribution $i$.

$\mathcal{E}$ is the set of all distributions on the set $\mathcal{C}_1 \cup \mathcal{C}_2$. Any element in $\mathcal{E}$ can also be seen as a distribution on the elements of the set $A^n$.

Let $\mathcal{D}$ be the set of all deterministic algorithms, and let $\mathcal{R}$ be the set of all possible distributions on the set $\mathcal{D}$. In other words, let $\mathcal{R}$ be the set of all random algorithms.

By using the min-max theorem we get,

$$\min_{x \in \mathcal{E}} \max_{j \in \mathcal{D}} \mathbb{E}_{i \sim x}[M[i, j]] = \max_{y \in \mathcal{R}} \min_{i \in \mathcal{C}_1 \cup \mathcal{C}_2} \mathbb{E}_{j \sim y}[M[i, j]].$$

Let $X$ be “the success probability of the best deterministic algorithm”, when the input (an element of the set $A^n$) is drawn from any distribution from the set $\mathcal{E}$. From the condition given in the question, we know that $X \geq \frac{2}{3}$. Thus, the expression on the left-hand side is at least $\frac{2}{3}$.

The expression on the right-hand side is precisely, “the success probability of the best randomized algorithm”, when the input (an element of the set $A^n$) is drawn from a distribution, chosen adversarially from the set $\mathcal{C}_1 \cup \mathcal{C}_2$. This is the value we wanted to be a constant strictly greater than $\frac{1}{2}$. By the equality we know that this is at least $\frac{2}{3}$. Hence, proved.