CS 170 HW 14

Due **2020-12-07, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 Nostalgia

What's been your favorite homework problem this semester? Tell us the HW number and problem name/number, and briefly explain (a sentence or two) why you liked it.

3 Super-Universal Hashing

Let $\mathcal{H}$ be a class of hash functions in which each $h \in \mathcal{H}$ maps the universe $\mathcal{U}$ of keys to $\{0, 1, \ldots, m-1\}$. Recall that $\mathcal{H}$ is *universal* if for any $x \neq y \in \mathcal{U}$, $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq 1/m$.

We say that $\mathcal{H}$ is super-universal if, for every fixed pair (x, y) of keys where $x \neq y$, and for any h chosen uniformly at random from $\mathcal{H}$, the pair $(h(x), h(y))$ is equally likely to be any of the m^2 pairs of elements from $\{0, 1, \ldots, m-1\}$. (The probability is taken only over the random choice of the hash function.)

(a) Show that, if $\mathcal{H}$ is super-universal, then it is universal.

(b) Suppose that you choose a hash function $h \in \mathcal{H}$ uniformly at random. Your friend, who knows $\mathcal{H}$ but does not know which hash function you picked, tells you a key x, and you tell her $h(x)$. Can your friend tell you $y \neq x$ such that $h(x) = h(y)$ with probability greater than $1/m$ (over your choice of h) if:

(i) $\mathcal{H}$ is universal?

(ii) $\mathcal{H}$ is super-universal?

In each case, either give a choice of $\mathcal{H}$ which allows your friend to find a collision, or prove that they cannot for any choice of $\mathcal{H}$.

Solution:

(a) If $\mathcal{H}$ is super-universal, then the tuple $(h(x), h(y))$ takes on all m^2 possible values with equal probability, and in m of these values we have $h(x) = h(y)$. So $\Pr[h(x) = h(y)] = m/m^2 = 1/m$ as desired.

(b) (i) We can construct a scenario where the adversary can force a collision. On a universe $\mathcal{U} = \{x, y, z\}$, consider the following family $\mathcal{H}$:

	x	y	z
h_1	0	0	1
h_2	1	0	1

$\mathcal{H}$ is a universal hash family: x and y collide with probability $1/2$, x and z collide with probability $1/2$, and y and z collide with probability $0 < 1/2$.

The adversary can determine whether we have selected h_1 or h_2 by giving us x to hash. If $h(x) = 0$, then we have chosen h_1, and the adversary then gives us y. Otherwise, if $h(x) = 1$, we have chosen h_2 and the adversary gives us z.

(ii) Since the pair $(h(x), h(y))$ is equally likely to be any of the m^2 possibilities, the marginal distributions of any two $h(x), h(y)$ are (pairwise) independent and uniformly distributed, so for any y your friend guesses, the chance $h(x) = h(y)$ is $1/m$.

4 Streaming Integers

In this problem, we assume we are given an infinite stream of integers x_1,x_2, ..., and have to perform some computation after each new integer is given. Since we may see many integers, we want to limit the amount of memory we have to use in total. For all of the parts below, give a brief description of your algorithm and a brief justification of its correctness.

(a) Show that using only a single bit of memory, we can compute whether the sum of all integers seen so far is even or odd.

(b) Show that we can compute whether the sum of all integers seen so far is divisible by some fixed integer N using $O(\log N)$ bits of memory.

(c) Assume N is prime. Give an algorithm to check if N divides the product of all integers seen so far, using as few bits of memory as possible.

(d) Now let N be an arbitrary integer, and suppose we are given its prime factorization: $N = p_1^{k_1} p_2^{k_2} \dots p_r^{k_r}$. Give an algorithm to check whether N divides the product of all integers seen so far, using as few bits of memory as possible. Write down the number of bits your algorithm uses in terms of $k_1, \dots, k_r$.

Solution:

(a) We set our single bit to 1 if and only if the sum of all integers seen so far is odd. This is sufficient since we don't need to store any other information about the integers we've seen so far.

(b) Set $y_0 = 0$. After each new integer x_i, we set $y_i = y_{i-1} + x_i \mod N$. The sum of all seen integers at step i is divisible by N if and only if $y_i \equiv 0 \mod N$. Since each y_i is between 0 and $N - 1$, it only takes $\log N$ bits to represent y_i.

(c) We can do this with a single bit b. Initially set $b = 0$. Since N is prime, N can only divide the product of all x_is if there is a specific i such that N divides x_i. After each new x_i, check if N divides x_i. If it does, set $b = 1$. b will equal 1 if and only if N divides the product of all seen integers.

(d) We can do this with $\lceil \log_2(k_1 + 1) \rceil + \lceil \log_2(k_2 + 1) \rceil + \cdots + \lceil \log_2(k_r + 1) \rceil$ bits. For each i between 1 and r, we track the largest value $t_i \leq k_i$ such that p^{t_i} divides the product of all seen numbers. We start with $t_i = 0$ for all i. When a new number m is seen, we find the largest t_i' such that $p^{t_i'}$ divides m and set $t_i = \min\{t_i' + t_i, k_i\}$ We stop once $t_i = k_i$ for all i as this implies that N divides the product of all seen numbers.

5 Era of Ravens

(a) Design an algorithm that takes in a stream $z_1, \ldots, z_M$ of M integers in $[n]$ and at any time t can output a uniformly random element in $z_1, \ldots, z_t$. Your algorithm may use at most polynomial in $\log n$ and $\log M$ space. Prove the correctness and analyze the space complexity of your algorithm. Your algorithm may only take a single pass of the stream. *Hint:* $\frac{1}{t} = 1 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdots \frac{t-1}{t}$.

(b) For a stream $S = z_1, \ldots, z_{2n}$ of $2n$ integers in $[n]$, we call $j \in [n]$ a *duplicate element* if it occurs more than once.

Design an algorithm that takes in S as input and with probability at least $1 - \frac{1}{n}$ outputs a duplicate element. Your algorithm may use at most polynomial in $\log n$ space. Prove the correctness and analyze the space complexity of your algorithm. Your algorithm may only take a single pass of the stream.

Hint: Use $\log n$ copies of the algorithm from part a to keep track of a random subset of the elements seen so far. For proof of correctness, note that there are at most n indices t such that $z_t \neq z_{t'}$ for any $t' > t$, i.e. element z_t never occurs after index t.

Solution:

(a) **Main idea:** Maintain a counter n_1, initially 0, to keep track of how many elements have arrived so far and maintain a "current element" x, also initially 0. When an element z arrives, increment n_1 by 1 and replace x with z with probability $1/n_1$. When queried at any time, output x.

Proof of correctness: To analyze the probability that z_t is outputted at time $t' > t$, observe that z_t must be chosen and then never replaced. This happens with probability

$$\left(\frac{1}{t}\right) \cdot \left(\frac{t}{t+1} \cdot \frac{t+1}{t+2} \cdots \frac{t'-1}{t'}\right).$$

Space complexity: The counter and current element never exceed M and n respective, so we just need $O(\log M + \log n)$ bits of memory.

(b) **Main idea:** The algorithm is to maintain, $\log n$ independent instantiations of the sampling algorithm from part (a). When a new element z arrives at time t, first query all the instantiations and if any of them outputs z, ignore the rest of the stream and output z as the 'duplicative element' at the end of the stream. Otherwise, stream z as input to each of the independent instantiations of the sampling algorithm and continue. If the stream ends without the algorithm ever outputting a value, output 'failed' at the end.

Proof of correctness: Note that if the algorithm returns an element, it is certainly a duplicative element. We now turn our attention to upper bounding the probability that the algorithm returns 'failed'. Suppose our algorithm returned 'failed', and for each instance of part a, let t be the index of the element it sampled. Note that if z_t is a duplicate of element $z_{t'}$ where $t' > t$, then we would have output z_t when we processed $z_{t'}$. So in order for us to output 'failed' z_t not be a duplicate of any element appearing afterwards in the stream. By part a, t is distributed uniformly at random, so this happens with probability at most $n/2n = 1/2$ per the hint. In turn, the $\log n$ instances collectively give us at most $(1/2)^{\log n} = 1/n$ chance of outputting 'failed'.

Space complexity: We use $\log n$ copies of the data structure from part a, each using $O(\log n)$ bits, so we use $O(\log^2 n)$ bits of memory.

6 (Optional) Hadamard Gate

Please note that quantum computing is not in scope for the final.

Recall the Hadamard gate from the most recent homework, defined as follows:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

(a) Suppose we pass a 0 qubit through a Hadamard gate, and observe the output on the other end. With what probability will the observed output be 0? With what probability will the observed output be 1?

(b) What if we had passed in a 1 qubit instead?

(c) Briefly state how we might think to simulate this input/output behavior without the use of quantum gates.

(d) Verify that the Hadamard gate acts as its own inverse. What happens if we pass a 0 or 1 qubit through two Hadamard gates, and then observe the output?

(e) Why might the result to part (d) be surprising given the answer to part (c)?

Solution:

(a) If we multiply the hadamard matrix with the vector [1, 0], we get the resultant vector $\frac{1}{\sqrt{2}} *$ [1, 1]. If we observe this quantum state, we'll get the outputs 0 and 1 with equal probability.

(b) Multiplying the Hadamard matrix with the vector [0, 1] gives us the vector $\frac{1}{\sqrt{2}} *$ [1, -1]. This again corresponds to observing 0 and 1 with equal probability.

(c) We might think to simulate a single Hadamard gate with a coin flip, outputing 0 for heads and 1 for tails.

(d) Multiplying the Hadamard matrix by itself, we get

$$\frac{1}{2}\begin{bmatrix} 1*1+1*1 & 1*1+1*-1 \\ 1*1+-1*1 & 1*1+-1*-1 \end{bmatrix}$$

Which is equal to

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

This implies that if we pass a 0 qubit through two Hadamard gates, we will always observe a 0 qubit as the output. A 1 qubit will also yield an observation of a 1 qubit.

(e) The fact that the Hadamard gate acts as its own inverse throws a wrench in the original coin flip analogy. We can see that two applications of a Hadamard gate would be like taking a fair coin that is heads up, flipping it twice, and always getting heads at the end. Intuitively, this is why the Hadamard gate has no classical analogue.

7 (Optional) Quantum Gates

Please note that quantum computing is not in scope for the final.

(a) The Hadamard Gate acts on a single qubit and is represented by the following matrix:

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Verify that this gate maps the basis states $|0\rangle$ and $|1\rangle$ to a superposition state that will yield 0 and 1 with equal probability, when measured. In other words, explicitly represent the bases as vectors, apply the gate as a matrix multiplication, and explain why the resulting vector will yield 0 and 1 with probabilities 1/2 each, when measured.

(b) Give a matrix representing a *NOT* gate. As in the previous part, explicitly show that applying your gate to the basis state $|0\rangle$ will yield the state $|1\rangle$ (and vice-versa).

(c) Give a matrix representing a gate that swaps two qubits. Explicitly show that applying this matrix to the basis state $|01\rangle$ will yield the state $|10\rangle$. Verify that this matrix is its own inverse.

Solution:

(a) The state $|1\rangle$ can be interpreted as the vector $[0,1]^T$. The results of applying the Hadamard gate to $|1\rangle$ is

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{bmatrix}$$

When measured, this yields 0 with probability $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ and yields 1 with probability $(\frac{-1}{\sqrt{2}})^2 = \frac{1}{2}$.

Likewise, $|0\rangle$ can be interpreted as the vector $[1,0]^T$, so applying the Hadamard gate gives

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}$$

When measured, this yields 0 and 1 with probabilities $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$, each.

(b) The following matrix represents a NOT gate:

$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

To see this, we apply it to $|1\rangle$ to get:

$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$$

Likewise, applying it to $|0\rangle$ yields

$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

Thus, the gate is, in fact, a NOT gate.

(c) As in the book, We represent a two qubit state by $|\alpha\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. This is the same as the vector $[\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}]^T$. Swapping the two qubits is equivalent to swapping the middle two values, since swapping two of the same values is unnecessary. This can be done by the following matrix:

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Applying this matrix to the basis state $|01\rangle$ yields

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

Simple matrix multiplication will show that the matrix is its own inverse.

8 (Extra Credit) The Power of Entanglement

Alice and Bob are playing the following game: A third party sends Alice x, which is 0 with probability 1/2 and 1 with probability 1/2. The third party also sends Bob y with the same distribution. Alice and Bob each choose a bit a, b. They win the game if $a \oplus b = x \wedge y$ ($a \oplus b$ is 0 if $a = b$ and 1 otherwise).

Alice and Bob are not allowed to communicate after receiving x, y, but they may come up with a shared strategy beforehand.

(a) Give a strategy for Alice and Bob that wins the game with probability 3/4.

(b) It turns out that without using quantum information, Alice and Bob can't come up with a strategy that wins with probability better than 3/4.

However, they can do better using quantum information. Consider the following strategy:

- Alice and Bob prepare two qubits in the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Alice takes the first qubit, and Bob takes the second qubit. After this point, Alice and Bob won't communicate.
- Alice receives x and rotates her qubit by an angle of $\pi/8$ if $x = 1$. That is, she maps the state $|0b\rangle$ to the superposition of states $(\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle)|b\rangle$, and the state $|1b\rangle$ to the superposition of states $(-\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle)|b\rangle$.
- Bob receives y and rotates his qubit by an angle of $-\pi/8$ if $y = 1$. That is, he maps the state $|a0\rangle$ to the superposition of states $|a\rangle(\cos(-\pi/8)|0\rangle + \sin(-\pi/8)|1\rangle)$, and the state $|a1\rangle$ to the superposition of states $|a\rangle(-\sin(-\pi/8)|0\rangle + \cos(-\pi/8)|1\rangle)$.
- Alice measures her qubit and chooses the bit she measures.
- Bob measures his qubit and chooses the bit he measures.

In the case that $x = 0, y = 0$, what is the probability Alice and Bob win using this strategy?

(c) In the case where $x = 1, y = 0$, what is the probability Alice and Bob win? This is also the probability they win if $x = 0, y = 1$, but you don't need to show this. You can give your answer rounded to three decimal places.

(d) Show that in the case where $x = 1, y = 1$, after rotating the qubits but before measuring, the qubits are in the superposition:

$$\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle).$$

This implies Alice and Bob win in this case with probability 1/2.

(The following equalities will be useful: $\cos(x) = \cos(-x), \sin(x) = -\sin(-x)$, and $\cos^2(\pi/8) - \sin^2(\pi/8) = 2\sin(\pi/8)\cos(\pi/8) = 1/\sqrt{2}$).

(e) Combining the previous three parts, what are Alice and Bob's overall chances of winning?

Solution:

(a) Alice and Bob both always just choose 0. They only lose if $x \wedge y = 1$, which happens with probability 1/4.

(There are other strategies that get this winning probability, this is just the simplest).

(b) Since Alice and Bob apply no rotations, the qubits are still in the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ when measured. So Alice and Bob will always choose the same bit, i.e. they win in this case with probability 1.

(c) The resulting superposition of states after Alice rotates her qubit is:

$$\frac{\cos(\pi/8)}{\sqrt{2}}(|00\rangle + |11\rangle) - \frac{\sin(\pi/8)}{\sqrt{2}}|01\rangle + \frac{\sin(\pi/8)}{\sqrt{2}}|10\rangle)$$

The probability the qubits collapse into the state 00 or 11 when measured (and thus the probability Alice and Bob win the game in this case) is thus $\cos^2(\pi/8) \approx 0.854$.

(d) The resulting superposition of states after Alice and Bob rotate their qubits is:

$$(\frac{\cos(\pi/8)}{\sqrt{2}}|0\rangle + \frac{\sin(\pi/8)}{\sqrt{2}}|1\rangle)(\frac{\cos(-\pi/8)}{\sqrt{2}}|0\rangle + \frac{\sin(-\pi/8)}{\sqrt{2}}|1\rangle) +$$

$$(\frac{-\sin(\pi/8)}{\sqrt{2}}|0\rangle + \frac{\cos(\pi/8)}{\sqrt{2}}|1\rangle)(-\frac{\sin(-\pi/8)}{\sqrt{2}}|0\rangle + \frac{\cos(-\pi/8)}{\sqrt{2}}|1\rangle) =$$

$$(\frac{\cos^2(\pi/8) - \sin^2(\pi/8)}{\sqrt{2}}(|00\rangle + |11\rangle) + \frac{2\sin(\pi/8)\cos(\pi/8)}{\sqrt{2}}(|10\rangle - |01\rangle) =$$

$$\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle).$$

(e) Their overall winning probability is (roughly) $\frac{1}{4} \cdot 1 + \frac{1}{2} \cdot 0.854 + \frac{1}{4} \cdot \frac{1}{2} = .802$.