Time limit: 90 minutes.
Maximum score: 150 points.
Instructions: For this test, you work in teams of eight to solve a multi-part, proof-oriented question.

Problems that use the words "compute", "construct", "list", or "determine" only call for an answer; no explanation or proof is needed. Answers for these problems, unless otherwise stated, should go on the provided answer sheet. Unless otherwise stated, all other questions require explanation or proof. Answers for these problems should be written on sheets of scratch paper, clearly labeled, with every problem *on its own sheet*. If you have multiple pages for a problem, number them and write the total number of pages for the problem (e.g. 1/2, 2/2).

Place a team ID sticker on every submitted page. If you do not have your stickers, you should write your team ID number clearly on each sheet. Only submit one set of solutions for the team. Do not turn in any scratch work. After the test, put the sheets you want graded into your packet. If you do not have your packet, ensure your sheets are labeled *extremely clearly* and stack the loose sheets neatly.

In your solution for a given problem, you may cite the statements of earlier problems (but not later ones) without additional justification, even if you haven't solved them.

The problems are ordered by content, NOT DIFFICULTY. It is to your advantage to attempt problems from throughout the test.
No calculators.

Introduction

This Power Round is an exploration of information theory, a branch of applied mathematics that is relevant to electrical engineering and computer science. Information theory was developed initially to determine fundamental limits in the problems of data compression and reliable communication of data, but the applications of information theory extend into many other fields like neurobiology and cryptography.

A Brief Review of Probability Notation

We review some notation that will be used throughout the power round. Throughout this section, A and B are events.

Definition: $P(A)$ is the probability that event A happens.

Definition: $P(A \cap B)$ is the probability that both A and B happen.

Definition: $P(A \cup B)$ is the probability that at least one of A and B happens.

Definition: A **random variable** is a variable X that can take on a variety of different values with various probabilities. For example, we can define a random variable X that has value 1 with probability $\frac{1}{4}$ and value 0 with probability $\frac{3}{4}$. This is alternatively written as $P(X = 1) = \frac{1}{4}$ and $P(X = 0) = \frac{3}{4}$.

We will use $\sum_x h(x)$ to refer to the sum of $h(x)$ over all possible values of x. For example, using X as defined in the previous paragraph, we have that $\sum_x P(X = x) = P(X = 0) + P(X = 1) = \frac{3}{4} + \frac{1}{4} = 1$. Note that x can take on different values, but for this random variable, unless $x = 0$ or $x = 1$, then $P(X = x) = 0$.

We will now give a few simple probability problems to warm you up for the power round.

1. (a) You flip a fair coin twice. Let A be the event "the first coin flip was heads" and B be the event "the second coin flip was heads."
   i. [**1**] Compute $P(A)$.
   ii. [**1**] Compute $P(B)$.
   iii. [**1**] Compute $P(A \cap B)$.
   iv. [**1**] Compute $P(A \cup B)$.

   (b) You flip a fair coin three times. Let X be a random variable which has value equal to the number of times the coin landed heads.
   i. [**1**] Compute $P(X = 0)$.
   ii. [**1**] Compute $P(X = 1)$.
   iii. [**1**] Compute $P(X = 2)$.
   iv. [**1**] Compute $P(X = 3)$.

   (c) You have two four-sided dice. Let A be the event "the sum of the two numbers rolled is even" and B be the event "the product of the two numbers rolled is even."
   i. [**2**] Compute $P(A)$.
   ii. [**2**] Compute $P(B)$.
   iii. [**2**] Compute $P(A \cap B)$.
   iv. [**2**] Compute $P(A \cup B)$.

Entropy

Information theory is fundamentally concerned with entropy. We start by defining entropy and proving some basic results about entropy.

Definition: The **entropy** of a random variable X is $H(X) = -\sum_x P(X = x)\log_2 P(X = x)$. In the event where $P(X = x) = 0$, we define $P(X = x)\log_2 P(X = x) = 0$.

Example: Given a random variable X where $P(X = 0) = \frac{1}{3}$ and $P(X = 1) = \frac{2}{3}$, we have that $H(X) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = \log_2 3 - \frac{2}{3}$.

2. (a) [**2**] Given a fair coin, we flip the coin once and define a random variable X on the result of the coin flip. If the coin flip is heads, then $X = 1$. Otherwise, $X = 0$. Compute $H(X)$.

   (b) [**3**] Given a fair coin, we flip the coin twice and define a random variable Y on the results of the coin flips. The random variable is defined as follows:

   - If the first flip is heads and the second is heads, $Y = 0$.
   - If the first flip is heads and the second is tails, $Y = 1$.
   - If the first flip is tails and the second is heads, $Y = 2$.
   - If the first flip is tails and the second is tails, $Y = 3$.

   Compute $H(Y)$.

   (c) [**3**] Given a fair coin, we flip the coin twice and define a random variable Z on the results of the coin flips. The value of Z is equal to the number of times the coin lands heads. Compute $H(Z)$.

   (d) [**4**] Given a random variable X that can only take on the values 0 and 1, compute the maximum possible value of $H(X)$.

   (e) [**3**] Prove that for all random variables X, $H(X) \geq 0$.

Coding Theory

Before we move on to the problem of data compression, we will briefly introduce coding theory.

Computers send messages to each other using only the digits 0 and 1. If we want to send a message through a computer, the computer needs a way to translate our message into zeros and ones. For the purposes of the power round, we assume that we will only be sending integers and we will only translate them into sequences of zeros and ones.

Definition: A **codeword** is a sequence of zeros and ones.

Definition: A **source code** C is a function that converts integers to codewords. We give an example of a source code C as follows:

x	$C(x)$
1	`0`
2	`10`
3	`110`
4	`111`

In the above source code, the integer 1 is converted to the codeword `0`. The integer 4 is converted to the codeword `111`. The integer 12 cannot be converted by the above source code.

Definition: An integer x is **encodable** by a source code C if x can be converted by C into a codeword.

Definition: The **size** of a source code C is the number of integers which are encodable by C. The size of a source code must be at least 1.

For the remainder of the power round, if a source code C is of size N, then the integers 1 through N must be encodable by C.

Therefore, the above source code has size 4 because the integers 1, 2, 3, and 4 are encodable by it.

Definition: The **length** of a codeword $C(x)$ is the number of zeros and ones needed to write out the codeword. When we want the length of the codeword corresponding to the integer x, we write $l(x)$. The length of a codeword must be at least 1.

Therefore, in the above source code, $l(1) = 1$, $l(2) = 2$, and $l(3) = l(4) = 3$.

Definition: Two source codes C_1 and C_2 are **distinct** if they are different sizes or if, for some integer x that is encodable by both C_1 and C_2, $C_1(x) \neq C_2(x)$.

Example: If $C_1(1) = \texttt{0}$ and $C_1(2) = \texttt{1}$, and $C_2(1) = \texttt{1}$ and $C_2(2) = \texttt{0}$, then C_1 and C_2 are distinct.

Definition: A source code C is **nonsingular** if, for every pair of encodable integers x and y, $C(x) = C(y)$ if and only if $x = y$. Note that the source code above is nonsingular.

3. (a) [**2**] Compute the maximum size of a nonsingular source code where every codeword has length exactly 3.

   (b) [**3**] Compute the maximum size of a nonsingular source code where every codeword has length at most 3.

   (c) [**4**] Compute the number of distinct nonsingular source codes of size 4 where every codeword has length exactly 2.

   (d) [**5**] Compute the number of distinct nonsingular source codes of any size, where every codeword has length exactly 2.

Now, we usually want to send fairly long messages, so we want to be able to extend source codes to let us send arbitrarily long messages.

Definition: The **extension** C^* of a source code C takes in a sequence of positive integers $x_1, \ldots, x_n$, and returns the concatenation of the codewords $C(x_1), \ldots, C(x_n)$. In the source code in the beginning of the section, $C^*(1, 2, 4) = \texttt{010111}$.

Definition: An extension is **nonsingular** if, for every pair of sequences s_1 and s_2 consisting of encodable integers, $C^*(s_1) = C^*(s_2)$ if and only if s_1 and s_2 are identical.

Definition: A source code is **uniquely decodable** if its extension is nonsingular.

4. (a) [**1**] Given a source code C where $C(1) = \texttt{00}$, $C(2) = \texttt{01}$, and $C(3) = \texttt{10}$, compute $C^*(1, 2, 3, 1)$.

   (b) [**1**] Given a source code C where $C(1) = \texttt{01}$, $C(2) = \texttt{10}$, and $C(3) = \texttt{00}$, compute the sequence of integers s such that $C^*(s) = \texttt{100001}$.

   (c) [**2**] Given a source code C where $C(1) = \texttt{0}$, $C(2) = \texttt{00}$, and $C(3) = \texttt{000}$, determine all sequences of integers s where $C^*(s) = \texttt{000}$.

   (d) [**2**] Construct a source code of size 2 which is nonsingular, but not uniquely decodable. The codewords must have length at most 3.

   (e) [**3**] Compute the number of source codes of size 2 where the source code is uniquely decodable and every codeword has length at most 2.

   (f) [**4**] Compute the number of source codes of size 3 where the source code is uniquely decodable and every codeword has length at most 2.

Definition: A source code is **instantaneous** if no codeword is a prefix of another codeword. Note that the source code at the beginning of the section is instantaneous.

5. (a) **[3]** Construct a source code that is **uniquely decodable** but not an instantaneous code. The source code must be of size at most 4, and no codeword can have length more than 3.

   (b) **[5]** Prove that all instantaneous codes are uniquely decodable.

   (c) **[7]** The Kraft inequality states that for any instantaneous source code C, $\sum_{x \in C} 2^{-l(x)} \leq 1$. Prove the Kraft inequality.

Data Compression

With coding theory out of the way, we can now move on to the problem of data compression.

Definition: The **expected length** of the source code C corresponding to random variable X is $L(C) = \sum_{x} P(X = x)l(x)$.

6. (a) **[2]** Given a random variable X where $P(X = 1) = \frac{1}{2}$, $P(x = 2) = \frac{1}{3}$, and $P(X = 3) = \frac{1}{6}$, compute the expected length of the following source code.

x	$C(x)$
1	`00`
2	`11`
3	`1010`

   (b) **[3]** Given a random variable X that takes on the values 1, 2, 3, and 4 each with probability $\frac{1}{4}$, construct an instantaneous source code with minimal expected length.

   (c) **[3]** Given a random variable X where $P(X = 1) = \frac{1}{2}$, $P(X = 2) = \frac{1}{4}$, and $P(X = 3) = \frac{1}{4}$, construct an instantaneous source code with minimal expected length.

We will now cover one specific type of codes, Shannon codes.

A Shannon code is constructed as follows: Given a random variable X that takes on all positive integers from 1 to m where $P(X = i) \geq P(X = i + 1)$ for $1 \leq i < m$, define $f(i) = \sum_{k=1}^{i-1} P(X = k)$. $C(i)$ is the binary representation of $f(i)$, rounded off to l_i bits, where $l_i = \left\lceil \log_2 \frac{1}{P(X=i)} \right\rceil$. As an example, the Shannon code for a random variable where $P(X = 1) = \frac{1}{2}$, $P(X = 2) = \frac{1}{4}$, and $P(X = 3) = P(X = 4) = \frac{1}{8}$, the Shannon code generated is:

x	$C(x)$
1	`0`
2	`10`
3	`110`
4	`111`

More rigorously, we have that $f(1) = 0$ and $l_1 = 1$. $f(1)$ written in binary is `0.0`, so $C(1) =$ `0`. $f(2) = \frac{1}{2}$ and $l_2 = 2$. $f(2)$ written in binary is `0.10`, so $C(2) =$ `10`. $f(3) = \frac{3}{4}$ and $l_3 = 3$. $f(3)$ written in binary is `0.110`, so $C(3) =$ `110`. $f(4) = \frac{7}{8}$ and $l(3) = 3$. $f(4)$ written in binary is `0.111`, so $C(4) =$ `111`.

7. (a) **[3]** Construct the Shannon code corresponding to the random variable X where $P(X = 1) = P(X = 2) = P(X = 3) = \frac{1}{4}$ and $P(X = 4) = P(X = 5) = \frac{1}{8}$.

   (b) **[5]** Prove that all Shannon codes are instantaneous.

   (c) **[5]** If L_X is the expected length of the Shannon code for random variable X, prove that $H(X) \leq L_X < H(X) + 1$.

Gibbs' Inequality

There is a useful tool in information theory known as Gibbs' Inequality. We will use it to prove some interesting identities.

Theorem: Given nonnegative numbers $p_1, \dots, p_n$ and positive numbers $q_1, \dots, q_n$ such that $\sum_{i=1}^{n} p_i = 1$ and $\sum_{i=1}^{n} q_i \leq 1$, then $\sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i} \geq 0$.

8. (a) **[5]** Prove Gibbs' Inequality. You may assume without proof that $\ln x \leq x - 1$.

   (b) **[5]** Suppose an information source X has n possible outcomes $x_1, x_2, \dots, x_n$. Intuitively, our uncertainty about the actual value of X should be maximal when all the outcomes are equally likely, i.e. when

   $$p(x_1) = p(x_2) = \cdots = p(x_n) = \frac{1}{n}.$$

   Prove that for any random variable X with n outcomes, $H(X) \leq \log_2 n$ (with equality only in the case above).

At this point, we are now able to state one of the most powerful theorems in information theory, Shannon's source coding theorem. We will be working with a slightly simpler variation of the source coding theorem, which is as follows:

Theorem: The minimum expected length of a uniquely decodable source code C corresponding to random variable X is $H(X)$.

Intuitively, if we retrieve a random value from X, we gain $H(X)$ bits of information. Shannon's source coding theorem says that we cannot compress those $H(X)$ bits of information any further. This is a fundamental limit on how much we can compress data - we can do no better than the natural entropy of the random variable.

We will ask you to prove that this theorem is true for instantaneous source codes. Proving Shannon's source coding theorem for uniquely decodable source codes is beyond the scope of this power round, but proving the theorem for instantaneous source codes is substantially simpler and can be done with Gibbs' inequality and the Kraft inequality, which you may use without proof.

9. (a) **[2]** Given a random variable X where $P(X = 1) = \frac{1}{2}$, $P(X = 2) = \frac{1}{4}$, and $P(X = 3) = P(X = 4) = \frac{1}{8}$, compute $H(X)$.

   (b) **[3]** Given the random variable above, construct a corresponding instantaneous source code with minimum expected length.

   (c) **[10]** Prove that, for a random variable X, it is impossible for an instantaneous source code C corresponding to random variable X to have an expected length less than $H(X)$.

Huffman Codes

Definition: A source code is **optimal** if it is instantaneous and no other instantaneous source code has a shorter expected length than the specified source code.

We will now give an algorithm that will generate optimal source codes. Consider a random variable X with the following probability distribution:

x	$P(X = x)$
1	0.18
2	0.19
3	0.20
4	0.21
5	0.22

$$\begin{tikzcd}
(1, 0.18) & (2, 0.19) & (3, 0.20) & (4, 0.21) & (5, 0.22)
\end{tikzcd}$$

Here is how we will construct an optimal source code. Note that the values $x = 1$ and $x = 2$ are the values that occur least frequently. Therefore, intuitively, we expect that the codewords for $x = 1$ and $x = 2$ will be the longest code words. Furthermore, the two longest codewords should be the same length. If w_1 is the codeword for $x = 1$ and w_2 is the codeword for $x = 2$, then we can have w_1 and w_2 be identical except for the last digit, where w_1 ends with a `0` and w_2 ends with a `1`.

Because $x = 1$ and $x = 2$ will share a common prefix, constructing an optimal source code for the given random variable is the same as constructing an optimal source code for a slightly simpler random variable, where we delete the values associated with $x = 1$ and $x = 2$ and add another value that appears with probability $P(X = 1) + P(X = 2) = 0.37$.

$$\begin{tikzcd}
(3, 0.20) & (4, 0.21) & (5, 0.22) & & (X_1, 0.37) \arrow[dl] \arrow[dr] & \\
& & & (1, 0.18) & & (2, 0.19)
\end{tikzcd}$$

We use the letter X to notate values which do not map to original values, and will number each of these values for clarity. By convention, we have the value appearing less frequently on the left and the value appearing more frequently on the right.

At this point, the values $x = 3$ and $x = 4$ are the least frequently occurring values, so we will combine those values as we did above.

$$\begin{tikzcd}
(5, 0.22) & & (X_1, 0.37) \arrow[dl] \arrow[dr] & & & (X_2, 0.41) \arrow[dl] \arrow[dr] & \\
& (1, 0.18) & & (2, 0.19) & (3, 0.20) & & (4, 0.21)
\end{tikzcd}$$

We have only three values left now. $x = 5$ and $x = X_1$ are the least frequently occurring, so those values get combined.

$$\begin{tikzcd}
& (X_3, 0.59) \arrow[dl] \arrow[dr] & & & (X_2, 0.41) \arrow[dl] \arrow[dr] & \\
(5, 0.22) & & (X_1, 0.37) \arrow[dl] \arrow[dr] & (3, 0.20) & & (4, 0.21) \\
& (1, 0.18) & & (2, 0.19) & &
\end{tikzcd}$$

We only have two values left, so we now combine them.

$$\begin{tikzcd}
 & & & (X_4, 1) \arrow[dll] \arrow[dr] & & & \\
 & (X_2, 0.41) \arrow[dl] \arrow[dr] & & & (X_3, 0.59) \arrow[dl] \arrow[dr] & & \\
(3, 0.20) & & (4, 0.21) & (5, 0.22) & & (X_1, 0.37) \arrow[dl] \arrow[dr] & \\
 & & & & (1, 0.18) & & (2, 0.19)
\end{tikzcd}$$

From this diagram, we now obtain the following source code:

x	$C(x)$
1	`110`
2	`111`
3	`00`
4	`01`
5	`10`

How did we construct this source code? Consider the value $x = 1$. From the top of the tree, we get to the leaf of the tree containing the value 1 by going right to X_3, right to X_1, and left to 1. In terms of directions, we went right, right, left. Replacing left with 0 and right with 1 gives us `110`. Repeating this process for the remaining values completes the source code as given above.

This algorithm generates a **Huffman code**, which is optimal.

10. (a) [**4**] Compute the Huffman code for a random variable with the following probability distribution:

x	$P(X = x)$
1	0.2
2	0.01
3	0.37
4	0.07
5	0.35

Recall that when you take two values and merge them, the value with the lower probability goes on the left.

(b) [**4**] Compute a probability distribution on a random variable X that takes on exactly five values that maximizes the expected length of the resulting Huffman code.

(c) [**4**] Prove that in a Huffman code for a random variable, if there exists some k such that $P(X = k) \geq \frac{1}{2}$, then $l(k) = 1$.

We conclude this power round by asking you to prove that Huffman codes are optimal. This is a fairly involved proof, so we ask you to prove some facts that may be helpful in ultimately proving that Huffman codes are optimal.

11. (a) [**2**] Prove that in an optimal code of X, if $P(X = x) > P(X = y)$, then the codeword associated with x is no longer than the codeword associated with y.

(b) [**2**] Prove that in an optimal code of X, if l is the length of the longest codeword in X, then at least two codewords have length l.

(c) [**3**] Prove that in an optimal code of X, there are two longest codewords which differ only in the last bit.

(d) [**12**] Prove that Huffman codes are optimal.