Irving Reyes Bravo
02/26/2025
CS 4920 Spring 2025
HW 2

Problem 1. Dice

In each *event*, you throw a dice and observe the number.

a. What is the information entropy value of one event? What is the entropy value of six events?
A six-sided die has an equal probability of landing on any of its six faces:
$P(x) = \frac{1}{6}$ for $x \in \{1, \quad 2, \quad 3, \quad 4, \quad 5, \quad 6\}$
The entropy of one roll:
$H_1 = -\sum_{i=1}^{6} \frac{1}{6}\log_2\frac{1}{6} = -6 * \frac{1}{6}\log_2\frac{1}{6}$
$H_1 = \log_2 6$
Approximating values:
$H_1 \approx \log_2 6 \approx 2.585$ bits
For six independent dice rolls:
$H_6 = 6H_1 = 6 * 2.585 \approx 15.50$ bits

b. Now suppose you modify the dice so that the sides that originally showed number six becomes five (the numbers 1, 2, 3, 4 occupy one side each, and 5 occupies two sides). What is the information entropy of one event? What is the entropy of six events?
If the modified die has faces $\{1, 2, 3, 4, 5, 5\}$, then the new probability distribution is:
$P(1) = P(2) = P(3) = P(4) = \frac{1}{6}, \ P(5) = \frac{2}{6} = \frac{1}{3}$
The entropy of one roll:
$H_1 = -(4 * \frac{1}{6}\log_2\frac{1}{6} + \frac{2}{6}\log_2\frac{2}{6})$
$H_1 = -(4 * \frac{1}{6} * (-2..585) + \frac{2}{6} * (-1.585))$
Approximating values:
$H_1 \approx 2.459$ bits
For six independent rolls:
$H_6 = 6H_1 \approx 14.75$ bits

c. You can further modify the dice by changing the numbers it shows on the sides. How would you maximize the entropy? What is the information entropy of one event in this case?
Entropy is maximized when all outcomes are quality probable. The uniform distribution over six sides is already the highest entropy possible: $H_{max} = \log_2 6 = 2.585$ bits per roll

d. You can further modify the dice by changing the numbers it shows on the sides. How would you minimize the entropy? What is the information entropy of one event in this case?
Entropy is minimized when the outcome is deterministic. If all six faces show the same number (e.g., all 1s), then: $P(1) = 1, \ P(2) = P(3) = P(4) = P(5) = P(6) = 0$
$H_1 = -1(1\log_2 1 + 0) = 0$ bits
For six rolls: $H_6 = 6 * 0 = 0$ bits

e. You use two typical dice (each showing 1, 2, 3, 4, 5, 6 on the sides). You throw the two dice and observe the sum. What is the information entropy of this event?
Rolling two standard dice, the sum ranges from 2 to 12 (with non-uniform probabilities):

Sum	Probability
2	$\frac{1}{36}$
3	$\frac{2}{36}$
4	$\frac{3}{36}$
5	$\frac{4}{36}$
6	$\frac{5}{36}$
7	$\frac{6}{36}$
8	$\frac{5}{36}$
9	$\frac{4}{36}$
10	$\frac{3}{36}$
11	$\frac{2}{36}$
12	$\frac{1}{36}$

The entropy formula:

$H = -\sum_{x=2}^{12} P(x) \log_2 P(x)$

Approximating the sum:

$H \approx 3.27$ bits

Problem 2. Deck of Cards

There is a standard 52-card deck of cards (e.g., https://en.wikipedia.org/wiki/Standard_52-card_deck). There are a total of 52 cards, e.g., no extra joker cards. In each *event*, you shuffle the deck of cards randomly, pick a card, and observe the suit and the rank.

a. What is the information entropy value of one event?
   In a well-shuffled 52-card deck, each card is equally likely:
   $P(x) = \frac{1}{52}$ for each card.
   The entropy of 1 roll:
   $H_1 = -\sum_{i=1}^{52} \frac{1}{52} \log_2 \frac{1}{52}$
   $H_1 = \log_2 52 \approx 5.7$ bits
b. What is the entropy value of four events?
   $H_4 = 4 * H_1 = 4 * 5.7 = 22.8$ bits
c. Now suppose the face cards (the jack's, queen's, and king's) are considered the same as 10's. What is the information entropy of one event?
   Each suit now has the ranks $\in$ {A, 2, 3, 4, 5, 6, 7, 8, 9, 10}, reducing the number of unique renaissance from 13 to 10. Since there are four suits, the number of distinct outcomes is now:

$$P(x) = \frac{1}{4(10)} = \frac{1}{40}$$

So, each outcome is equally likely:

$H_1 = \log_2 40 \approx 5.32$ bits

d. Now suppose you only consider the suit in the card-picking event (i.e., rank does not matter and gets ignored). What is the information entropy of one event?

Since there are only 4 suits, the number of distinct outcomes is now:

$$P(x) = \frac{1}{4}$$

So, each outcome is equally likely:

$H_1 = \log_2 4 = 2$ bits

e. You can control and change the suit (but not the rank) of all the cards. How would you modify the deck of cards to minimize the information entropy? What is the resulting information entropy of an event after modifying the cards?

To minimize entropy, I make the suit of all 52 cards identical (e.g., all spades). Since there is only one possible outcome:

$P(x) = 1$

$H_1 = 0$ bits

f. You can change both the suit and the rank of the cards. How would you modify the cards to minimize the information entropy? What is the resulting information entropy of an event after modifying the cards?

To minimize entropy, I make all 52 cards identical (e.g., all are Ace of Spades). Since there is only one possible outcome:

$H_1 = 0$ bits

Problem 3.

a. Reformulate Equation (2.1), removing the restriction that *a* is a nonnegative integer.

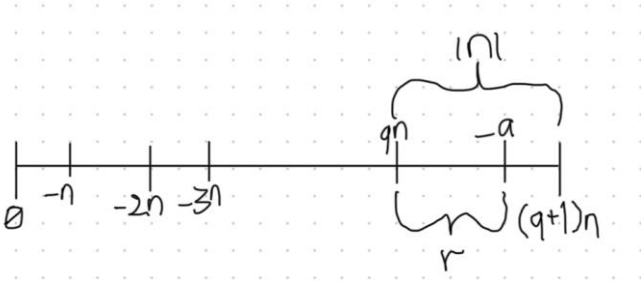
To generalize the given Division Algorithm equation for any integer a, I allow a to be negative while ensuring that the remainder r remains within a valid range.

For any integer a with positive integer n, there exist unique integers q and r such that:

$a = qn + r, \ 0 \leq r < |n|$

Where q is the integer quotient (positive or negative), with r, the remainder, always remaining as a nonnegative integer that is less than the absolute value of n.

b. Draw a figure similar to Figure 2.1 for $a < 0$.



Problem 4.

Prove the following:

a. a ≡ b (mod n) and b ≡ c (mod n) imply a ≡ c (mod n)

PROOF:

By the definition of Modular Congruence:
a ≡ b (mod n) means n divides (a – b), i.e., (a – b) = kn for some integer k.
b ≡ a (mod n) means n divides (b – c), i.e., (b – c) = mn for some integer m.
Now, adding both equations:
$(a - b) + (b - c) = kn + mn$
Simplifying:
$a - c = n(k + m)$
Since (k + m) is an integer, I conclude that n divides (a – c), meaning:
a ≡ c mod n
Thus, Modular Congruence is transitive (Q.E.D).

b. [(a mod n) - (b mod n)] mod n = (a - b) mod n
PROOF:
By the definition of Modulo:
Let $a \pmod n = r_a$, so $a = q_a n + r_a$ where $0 \leq r_a < n$.
Let $b \pmod n = r_b$, so $b = q_b n + r_b$ where $0 \leq r_b < n$.
Now, I solve:
$(a - b) = (q_a n + r_a) - (q_b n + r_b)$
$= (q_a - q_b)n + (r_a - r_b)$
Taking (mod n) on both sides:
$(a - b) \bmod n = (q_a - q_b)n + (r_a - r_b) \bmod n$
Since $(q_a - q_b)$n is a multiple of n, it vanishes under (mod n), leaving:
$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$
But by definition, $r_a = a \bmod n$ and $r_b = b \bmod n$, so:
$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$ (Q.E.D.)

c. For two consecutive integers n and n+1, GCD(n,n+1)=1
PROOF:
I assume for contradiction that $d = GCD(n, n + 1)$ is a common divisor of both n and (n + 1).
This means:
$d \mid n$ and $d \mid (n + 1)$
Since d divides both n and (n + 1), it must also divide their difference:
$d \mid [(n + 1) - n] = 1$
Since the only integer that dividends 1 is 1 itself, I conclude that d =1, meaning:
$GCD(n, n + 1) = 1$
Thus, consecutive integers are always prime (Q.E.D.).

Problem 5.

a. State the Euclidean Algorithm (E.A.).
The E.A. is an efficient method for computing the Greatest Common Divisor (GCD) of two integers (a and b). Its core principle is: GCD(a, b) = GCD(b, a mod b).
The algorithm's stems are as follows:
1. Start with two integers (a and b), where a > b.
2. Replace a with b and b with (a mod b).
3. Repeat Step 2 until b = 0.

The final nonzero value of a is GCD(a, b).

b. State and prove the Extended Euclidean Algorithm.

The E.E.A. finds the GCD of two integers a and b, and computes integers x and y such that:
ax + by = GCD(a, b)
The equation is known as Bezout's Identity, and if GCD(a, b) = 1, then x (mod b) is known as the multiplicative inverse of a modulo b.
PROOF:
The standard E.A. recursively computes GCD(a, b) using:
GCD(a, b) = GCD(b, a mod b) until $r_i = 0$, at which point r_{i-1} is the GCD.
The E.A.A. works by keeping track of additional coefficients x and y that satisfy the equation:

$$x_i = x_{i-2} - q_i x_{i-1}$$

$$y_i = y_{i-2} - q_i x_{i-1}$$

Where q_i is the quotient in each division step.
At the end, when GCD(a, b) = 1, x gives the multiplicative inverse of a modulo b.
Description of a Program:
This Python program implements the Extended Euclidean Algorithm to compute the greatest common divisor (GCD) of two integers a and b while also finding coefficients x and y such that ax + by = GCD(a, b). If the GCD is 1, x represents the multiplicative inverse of a modulo b. The program iterates through the algorithm step-by-step, maintaining a table of intermediate values, including remainders, quotients, and coefficients.
After computing the results, it writes them to an output file ("output_problem_values"), presenting the data in a structured table format that mirrors the stepwise execution of the algorithm in Table 2.4. The program processes multiple tests cases and saves the results separately for each pair of integers.