Homework 2

Problem 1 Dice

Notes to know: - CryptoMath.py does my calculations for entropy to double check my maths. If i didn't show any work its because i used that program and failed to show the work in a timely manner.

- Problem5.py and Problem4.py solve the answers to the questions in problems 5 and 4 respectively.
- I included a markdown version of this if you need to see what I did before this was transferred to a PDF.

A

Entropy information is calculated by finding the entropy for each event and adding all of those numbers up. This was calculated by finding the Entropy of one die and then adding that 4 times for 4 events.

- For One event: 2.585
- For Four Events: 10.340

B

Following the logic in Part A, I created a list of four values, three of which are 1/6 and the last value is 3/6. This is because 3 of the 6 sides are now 5s meaning there is a 50% chance or 3/6 chance of rolling a five.

- For One Event: 1.792
- For 5 Events: $ 8.962 $

C

The best way to maximize entropy for a die is to leave it as it is, having an equal chance to roll any of the 6 numbers will give the die the highest entropy value. The Dice should be marked `[1, 2, 3, 4, 5, 6]` to achieve a maximum entropy of 2.585

D

To minimize entropy there must be a single number that repeats as much as possible, because of this there are six possible answers for a die. Such as using `[6,6,6,6,6,6]` which should provide the smallest entropy number of -93.0587.

E

Rolling two dice and only monitoring the sum means that the numbers I use will be a little different from a single die. The numbers I will be monitoring is 2-12 (it starts at 2 because you cannot roll a 1 on two dice), because the 11 possible numbers can be made from multiple combinations of dice the probability is actually over 36 possibilities instead of 11. For each die roll if you add up each combination from Die A and Die B, the number of possibilities should be 36 where there are 6 possible combinations for a sum of 7. This makes these fractions for calculating Entropy = `[1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36,1/36]`. Calculating the Entropy from this I get 3.2744

Problem 2. Balls in a Bin

A

In this problem there are 9 balls total, 2 are yellow, 3 are red, and 4 are green. So the set to calculate should be `[2/9,3/9,4/9]` the calculated entropy for a single event is 1.5726

B

The information entropy for n events where n is a positive integer equals the entropy of a single event multiplied by the number of desired events. As determined above the entropy of an event would be 1.5726 so the entropy for n events would be $n \times 1.5726$ or $1.5726n$

C

The total number of balls is increased from nine to eleven. The number of red balls is increased from three to four and yellow is increased from two to three. the way to calculate this event is to modify the list of values:

```
import math
listOfValues = [3/9,2/9,4/9] #initial set of values is 9 the lists sets red, yellow and gre
newListOfValues = [4/11, 3/11, 4/11] # set the same as the above list respectively.\
EntropyValue = 0
for value in List:
    EntropyValue += Value * math.log2(1/Value) # P * log_2(1/P)
print(EntropyValue)
```

The first value will equal **1.792** the new list of values will instead make **1.573**

D

The best way to create the highest entropy value is to ensure that the probability of pulling any ball is as even as possible. So to take the ball set `[4/11, 3/11, 4/11]` where there are 4 red balls, 3 yellow balls, and four green balls respectively. In maximize entropy by adding or removing a single ball of any color, a yellow ball must be added such that the set would equal `[4/12, 4/12, 4/12]`. This makes an entropy value of 1.585

E

Currently the bag contains four red balls, three yellow balls and four green balls. The best way to minimize entropy is to ensure that a single value is higher than any of the other numbers and that none of the numbers have equal values. Because I can only modify a single ball, in this instance the best way to minimize entropy would be to add a red ball. This would create the set `[5/12, 3/12, 4/12]` which gives the entropy 1.5546 which is still similar to the highest entropy answered in D.

F

Though making the probabilities of the balls as even as possible would normally maximize entropy, because a new ball can be added, adding any color would be better. Adding more complexity to the ball bag (adding another color) increases entropy more than equalizing the number of colored balls would.

Problem 3. Proving Modular Arithmetic

A

This proof is relatively simple, plugging in the value for b into the a equation and foiling the answer. The basis of this is that if you take $\mod n$ of $\mod n$ then you will always get the exact same result. Such that $X \mod n == X \mod n \mod n$

$$
\begin{aligned}
a = b(\mod n) and\, b = c(\mod n) \therefore\ & a = c(\mod n) \\
& if\ \ b = c(\mod n) then \\
& a = c(\mod n)(\mod n) \\
& a = c(\mod n)(\mod n) \equiv c(\mod n)
\end{aligned}
$$

B

To prove this I used the quotient remainder theorem to redefine a and b in terms of remainders. Below I set showed that $a = cq_1 + r_1$ and $b = cq_2 + r_2$ therefore a mod $n = r_1$ and $b \mod n = r_2$. By setting and b to this I can prove the theorem using simple substitution once the equations are foiled. The \$ (n(q__1-q__2) + r__1 - r__2)\$ step is completed by removing the n due to the fact that n mod n is 0. That step leaves the remainders which as noted above should be a mod n and b mod n therein proving the Modular Subtraction Rule.

$$
\begin{aligned}
& a = nq_1 + r_1 where 0 \leq r_1 < n \therefore\ a \mod n = r_1 \\
& b = nq_2 + r_2 where 0 \leq r_2 < n \therefore\ b \mod n = r_2 \\
& (a - b) \mod n = (nq_1 + r_1 - nq_2 - r_2) \mod n \\
& (a - b) \mod n = (n(q_1 - q_2) + r_1 - r_2) \mod n \\
& (a - b) \mod n = (r_1 - r_2) \mod n \\
& (a \mod n - b \mod n) \mod n = (r_1 - r_2) \mod n
\end{aligned}
$$

C

For GCD(A,B) the new

$$A = BQ_1 + (B \mod A)$$

and

$$D = GCD(B, A \mod B)$$

So by putting this equation to use below and using the equation for determining the quotient. I determined that the quotient would be 0, this is because n mod n+1 is n and n - n is 0.

$$\begin{aligned}
&GCD(n,n+1) \;\; n = \frac{n-(n \mod (n+1))}{(n+1)} \times (n+1) + (n \mod n+1) \\
&GCD(n,n+1) \;\; n = 0 \times (n+1) + (n) \quad d = GCD(n+1,n) \\
&GCD(n+1,n) \;\; n = \frac{(n+1)-(n+1 \mod n)}{n} \times n + ((n+1) mod\ n) \\
&GCD(n+1,n) \;\; n = \frac{n}{n} \times n + 1 \quad d = GCD(n,1) \\
&GCD(n,1) \;\; n = \frac{n-(n \mod 1)}{1} \times 1 + (n \mod 1) \\
&GCD(n,1) \;\; n = \frac{n}{1} \times 1 + (0) \quad d = GCD(1,0) \\
&GCD(n,n+1) = 1
\end{aligned}$$

Because b is now equal to 0 the Common Divisor can be found as what the new A equals.

Problem 4 Euclidean Algorithm

programmed using Problem4.py

A

First i must prove that r_{n-1} (remainder) divides into both n and q through induction:

$$\begin{aligned}
&n = q_1 d + r_1 \\
&d = r_1 * q_2 + r_2 \\
&r_1 = r_2 * q_3 + r_3 \\
r_{n-1}\ &\text{divides into } r_{n-3} \\
&r_{n-3} = r_{n-2} * q_{n-1} + r_{n-1} = \\
&(r_{n-1} * q_n) * q_{n-1} + r_{n-1} = (q_n * q_{n-1} + 1) * r_{n-1} \\
r_{n-1}\ &\text{divides into } r_{n-4} \\
&r_{n-3} * q_{n-2} + r_{n-2} = (q_n * q_{n-1} + 1) * r_{n-1} * q_{n-2} + r_{n-1} * q_n = \\
&((q_n * q_{n-1} + 1) * q_{n-2} + q_n) * r_{n-1}
\end{aligned}$$

This pattern will continue to go on. This proves that r_{n-1} divides into n and that r_{n-1} divides into d. Now I would need to prove that r_{n-1} is the greatest common denominator. So to prove it, let c be a common divisor for n and d.

$n = ac$ and $d = bc$

$n = q_1 d + r_1$ This then proves that c divides into r_1

$r_1 = n - q_1 d = ac - q_1 bc = (a - q_1 b)c$
which means that c is then divisible by r_{n-1}. It follows that the greatest common divisor g must divide into r_{n-1} we then know that $r_{n-1} \leq g$ and from the proofs I have shown, $g = r_{n-1}$ where g is the greatest common divisor. ### B

GCD(1105, 425) = 85

C

GCD(2078, 9602) = 2

D

GCD(22142, 16762) = 2

Problem 5 Extended Euclidean Algorithm

Programmed using Problem5.py

A

The Extended Euclidean Algorithm finds both the greatest common divisor d but also two additional integers x and y that satisfy

$$ax + by = d\,\mathrm{gcd}(a, b)$$

First Consider the set

$$K = \{ax + by | x, y \in Z\}$$

Let K be the smallest positive elemet of K. since $k \in K$ there would then be $x, y \in Z$ so that

$$k = ax + by$$

Because Z is in the Euclidean Domain:

$$\begin{aligned}
a &= qk + r \text{ with } 0 \leq r < k \\
&\therefore \\
r &= a - qk \\
&= q - q(ax + by) \\
&= a(1 - qx) + b(-qy) \\
&\in K
\end{aligned}$$

Because k is the smallest positive element in K and it is implied that r must be 0, then $a = qk$ therefore k divides a and b. The k is then a common divisor of a and b and therefore $k \leq \mathrm{gcd}(a, b)$

gcd(a, b) divides both a and b and $k = ax + by$, gcd(a, b) divides k. If gcd(a, b) divides k and $k \leq$ gcd(a, b) therefor $k \equiv$ gcd(a, b) and since $k = ax + by$ becomes:

$$\gcd(a, b) = ax + by$$

B

1 * 1 + 8 * 0 = 1 8 * -4 + 33 * 1 = 1 33 * 9 + 74 * -4 = 1 74 * -13 + 107 * 9 = 1 107 * 22 + 181 * -13 = 1 181 * -57 + 469 * 22 = 1 469 * 79 + 650 * -57 = 1 650 * -215 + 1769 * 79 = 1

C

19 * 1 + 133 * 0 = 19 133 * -6 + 817 * 1 = 19 817 * 7 + 950 * -6 = 19 950 * -13 + 1767 * 7 = 19

D

2 * 1 + 18 * 0 = 2 18 * -1 + 20 * 1 = 2 20 * 3 + 58 * -1 = 2 58 * -4 + 78 * 3 = 2 78 * 7 + 136 * -4 = 2 136 * -18 + 350 * 7 = 2 350 * 25 + 486 * -18 = 2 486 * -93 + 1808 * 25 = 2 1808 * 211 + 4102 * -93 = 2 4102 * -515 + 10012 * 211 = 2 10012 * 12056 + 234378 * -515 = 2

Problem 6 Ideal Block Cipher

A

The number of reversible mappings for the ideal block cipher for a block of n bits is \$ 2ˆn! \$. For a mapping to be reversible, each block must be able to map into a unique ciphertext block. In order to enumerate all possible mappings, the block equivalent to 0 can map into any of the 2^n possible blocks. The block with value 1 can map into any of the \$ 2ˆn-1 \$ possible blocks. Thus the total number of reversible mappings of an ideal block is \$ 2ˆn! \$.

B

The reason this discrepancy exists is because the number of usable values depends on the desired output of the cipher. Though there exists $n \times 2^n$ possible mappings, these will include irreversible mappings which cannot be decrypted and therefore do not work as a cipher. By limiting ourselves to reversible mappings the number of different transformations becomes $2^n!$. So even though there exists far more possible mappings for the ideal cipher the amount that can actually be used is limited to $\log_2(2^n!)$ which is why the larger n is the more secure the cipher becomes.