Q1 Team Name
0 Points

Pasta_Sandwich

Q2 Commands
5 Points

List the commands used in the game to reach the ciphertext.

enter, enter, go, go, go, give, read

Q3 Analysis
30 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Use Latex wherever required. If your solution is not readable, you will lose marks. If necessary the file upload option in this question must be used TO SHARE IMAGES ONLY.)

We came to the cipher text screen by entering the above commands.
We then see the following 32 space separated numbers: 24 109 76 35 22 94 83 25 106 104 73 87 56 38 56 50 10 92 58 84 44 88 24 112 125 121 125 43 122 55 106 54
This is mentioned to be the hash value of our password.

Then the password contains only the element between 'f' to 'u' and password is imagined as sequence of numbers over field F_127. Also, the letters are in alphabetic order. We could think of two possible mappings for 'f' to 'u' characters, either 'f' = 0 , 'g' = 1, 'h'=2,.... and so on or replace characters with their ASCII values. Firstly, we tried the first way of mapping but got no result. Then, we tried the second way of mapping (i.e. replace each character with its ASCII value) and we got our password.

The formula for hashing was given that, (ith value of hash) = $x_1^{i-1} + \ldots x_m^{i-1}$ where $x_1, x_2, ..., x_m$ are sequence of numbers in which the password letters are viewed as.
We substitute i = 1 for the above formula and we get the following equation
$24 = x_1^0 + x_2^0 + .. + x_m^0$
I.e m = 24.
Thus we find that the length of our password = 24.

Similarly, if we substitute i = 2, we can find that sum of the elements of password in F_127 field = 109, and by substituting i=3, we find that sum of squares of elements of password in F_127 field is 76 and so on.

Our first step was to analyze whether or not brute force solution is feasible to perform.
We know that sequence of password are in alphabetical order, length of password is 24, and number of alphabets available is 16.
Thus using formula for combinations with repetitions, total number of possibilities of password = (16 + 24 - 1) C (24) = 25140840660. { i.e (n + r - 1) C (r) }
REF: https://math.stackexchange.com/questions/884370/find-the-number-of-increasing-words-of-length-n-formed-by-an-alphabet-of-m-l

And on average, for one simple statement, python is able to run over loop of 10^5 elements in 0.01 seconds.
Thus, average time to loop over 25140840660 sequences would be (25140840660/10^5) * 0.01 which is approx, 50 minutes. Thus brute force seemed to be a feasible solution. The main goal is breaking the cryptographic system by any means available. Security which is compromised even by brute force is a valid concern.
Analysis of the time required to run is in the main python notebook.

Once the analysis was done, we proceed with actual brute force attack.
We generated all sequence of possible permutation using python itertools library's function combinations_with_replacement().

Then we used list comprehension of python to iterate through each possibilities and checked cases of sum of squares, sum of cubes, etc till 12th power and found out possible candidates for password. We only found 1 possible candidate which satisfied case till 12th power hence we didn't go till 31st power check. This one possible candidate was: (102, 103, 105, 106, 107, 107, 107, 107, 108, 108, 110, 110, 111, 112, 113, 113, 113, 114, 115, 115, 115, 115, 116, 117) .
We converted these values to their ASCII value and got the string : fgijkkkkllnnopqqqrsssstu. After inputting this string on console, we found that it was the correct password.
Hence our attack was successful.

No files uploaded

Assignment 7 ● GRADED

GROUP
Sharanya Saha
Ayush Sahni
Manjyot Singh Nanra
View or edit group

TOTAL POINTS
50 / 50 pts

QUESTION 1	
Team Name	**0** / 0 pts
QUESTION 2	
Commands	**5** / 5 pts
QUESTION 3	
Analysis	**30** / 30 pts
QUESTION 4	
Password	**15** / 15 pts
QUESTION 5	
Codes	**0** / 0 pts

Q4 Password
15 Points

What was the final command used to clear this level?

fgijkkkkllnnopqqqrsssstu

Q5 Codes
0 Points

It is MANDATORY that you upload the codes used in the cryptanalysis. If you fail to do so, you will be given 0 for the entire assignment.

▾ pasta_sandwicha7.ipynb   Download

Analysing whether Brute force is feasible

We will try to simulate a loop of length 10^5 and note down the time required

In [1]:
```
import time
start = time.time()
for i in range(1, 10**5):
    if i % 10000000 == 0:
        print(i)
end = time.time()
total_time = end-start
print("Time required to run the loop", total_time)
```

```
Time required to run the loop 0.012001752853393555
```

Now we calculate time required to run over 25140840660 iteration loop, because we have 25140840660 sequences to test in brute force

In [2]:
```
(25140840660 / 10**5) * total_time / 60
```

Out [2]:
```
50.28902602131128
```

Thus for a simple statement, it would around 50 minutes, this means that we can go with brute force approach, and it would at least give results within 2-3 hours

We have two approach, mapping alphabets to ascii or mapping it to numbers starting from 0. Former approach gave us answers and latter didnt

In [3]:
```
from itertools import combinations_with_replacement

length_of_password = 24 # first element of hash is 24.


alphabets_to_field_mapping= [ord("f"), ord("g"), ord("h"), ord("i"),
ord("j"), ord("k"), ord("l"), ord("m"), ord("n"), ord("o"), ord("p"),
ord("q"), ord("r"), ord("s"), ord("t"), ord("u"),]
#alphabets_to_field_mapping = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
#

iterator = combinations_with_replacement (alphabets_to_field_mapping,
length_of_password)
```

Visualizing that Combinations with replacements library gives correct sequence

In [4]:
```
for idx, val in enumerate(iterator):
    print(val)
    if idx == 10:
        break
```

```
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
(102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102
```

In [5]:
```
def end_of_loop(i):
    print("A possible value for password we get is", i)
    print("Breaking the list comprehension by using interrupt")
    raise StopIteration

def condition_check(sequence):
    if (
    (sum(sequence)%127 == 109) and
    (sum(map(lambda x: pow(x, 2, 127), sequence))%127==76) and
    (sum(map(lambda x: pow(x, 3, 127), sequence))%127==35) and
    (sum(map(lambda x: pow(x, 4, 127), sequence))%127==22) and
    (sum(map(lambda x: pow(x, 5, 127), sequence))%127==94) and
    (sum(map(lambda x: pow(x, 6, 127), sequence))%127==83) and
    (sum(map(lambda x: pow(x, 7, 127), sequence))%127==25) and
    (sum(map(lambda x: pow(x, 8, 127), sequence))%127==106) and
    (sum(map(lambda x: pow(x, 9, 127), sequence))%127==104) and
    (sum(map(lambda x: pow(x, 10, 127), sequence))%127==73) and
    (sum(map(lambda x: pow(x, 11, 127), sequence))%127==87) and
    (sum(map(lambda x: pow(x, 12, 127), sequence))%127==56)
    ):
        return True
    else:
        return False
```

In [6]:

```
lookup = [
    end_of_loop(sequence) for sequence in iterator
    if condition_check(sequence)
    ]
```

```
A possible value for password we get is (102, 103, 105, 106, 107, 107, 107
Breaking the list comprehension by using interrupt


---------------------------------------------------------------------------
------StopIteration                             Traceback (most
recent call last)~\AppData\Local\Temp/ipykernel_15512/210033711.py in
<module>
----> 1 lookup = [
      2     end_of_loop(sequence) for sequence in iterator
      3     if condition_check(sequence)
      4     ]
~\AppData\Local\Temp/ipykernel_15512/210033711.py in <listcomp>(.0)
      1 lookup = [
----> 2     end_of_loop(sequence) for sequence in iterator
      3     if condition_check(sequence)
      4     ]
~\AppData\Local\Temp/ipykernel_15512/2801710018.py in end_of_loop(i)
      2     print("A possible value for password we get is", i)
      3     print("Breaking the list comprehension by using
interrupt")
----> 4     raise StopIteration
      5
      6 def condition_check(sequence):
StopIteration:
```

Note that the exception raised above is by design and not an error

In [3]:

```
# Printing final password
final_seq = (102, 103, 105, 106, 107, 107, 107, 107, 108, 108, 110,
110, 111, 112, 113, 113, 113, 114, 115, 115, 115, 115, 116, 117)
print("Final password is")
for i in final_seq:
    print(chr(i),end="")
```

```
Final password is
fgijkkkkllnnopqqqrsssstu
```

In [ ]:

Select a question.

Group Members   Submission History   Next Question