Aaron Thompson
CS-2252-01
21 April 2019

# Categorize 2: Reverse Rot

From Chapter 2, I decided to do the problem Reverse Rot, which can be found at this address: https://open.kattis.com/problems/reverserot. I decided to do this problem because it deals with the history of Information Security in the early ciphers that used to be used to encrypt data. ROT13 is a variant that was used to scramble data so that the average joe looking at the data wouldn't be able to understand it. However, it is still very easy to crack and figure out what it is trying to say.

The category that this belongs to is simple data structures. Because of how the problem is set up, I decided to use 2 arrays. The first array is every possible value that the input could be. We use this array to then create the cipher because we can just start reading from x steps, and put it in the spot of A (so assuming the steps were 3, we would start reading the alphabet at C, and put it in A's spot in the cipher array). In doing this, we complete the 1st major requirement of encoding the string, knowing the cypher to encode it with. The next step is to encrypt the string with our cipher. We complete this by stepping through every letter and replacing it based on the cipher array. We know where we are going to look based on the character it is. If it is A → Z, we are going to just do the character number – 65 to know where to look. Otherwise, there are only 2 other characters, and we predefined where those new values would be in our cipher array (the last 2 indices). That completes the 2nd requirement of encrypting it with our cipher. The last part would be the reversing part, which could be the bane of some people's existence. I coded it in python, and therefore it is easy to do, and we can just do that when we are printing it. And that is the final step to our code.

For the input and output, there are two major things that I would like to test. First I would test the edges of the problem, which is when N (our steps in the alphabet that we shifted) is equal to either 1 or 27. The last sample test case that I have provided is one that I would make more of the same type with different numbers and give as the only sample input, a palindrome. This makes it so that they must read the prompt that it must be reversed at the end, so if you just go straight to the input you will be misinformed.

**Input:**

```
27 ABCD
1 .OOKD
1 TACOCAT
0
```

**Output:**

```
CBA.
ELPPA
UBDPDBU
```

For the code below, the time complexity should be O(N), because there are no nested for loops. As to the point in a competition that I would try to tackle this problem, I would probably end up doing this first. Looking at this problem it peaked my interest because it was working with an old cipher, as well as I was instantly able to see the code for the program to work in my head.

Aaron Thompson
CS-2252-01
21 April 2019

```
alphaArray =
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","
_","."]
cypherArray = []

def encrypter(decryptedString):
        returning = []
        for i in decryptedString:
                if not (i == "_" or i == "."):
                        returning.append(cypherArray[ord(i)-65])
                else:
                        if i == "_":
                                returning.append(cypherArray[26])
                        if i == ".":
                                returning.append(cypherArray[27])
        return returning

def cypherCreator(step):
        global alphaArray, cypherArray
        cypherArray = []
        i = step
        for ii in alphaArray[i:]:
                cypherArray.append(ii)
        for ii in alphaArray[0:i]:
                cypherArray.append(ii)

def main():
        flag = True
        line = input()
        step = int(line.split()[0])
        if step == 0:
                print("REVERSE_ROT")
                flag = True
        decryptedString = line.split()[1]
        while flag:
                cypherCreator(step)
                encryptedArray = encrypter(decryptedString)
                print("".join(encryptedArray[::-1]))
                line = input()
                step = int(line.split()[0])
                if step == 0:
                        break
                decryptedString = line.split()[1]
main()
```