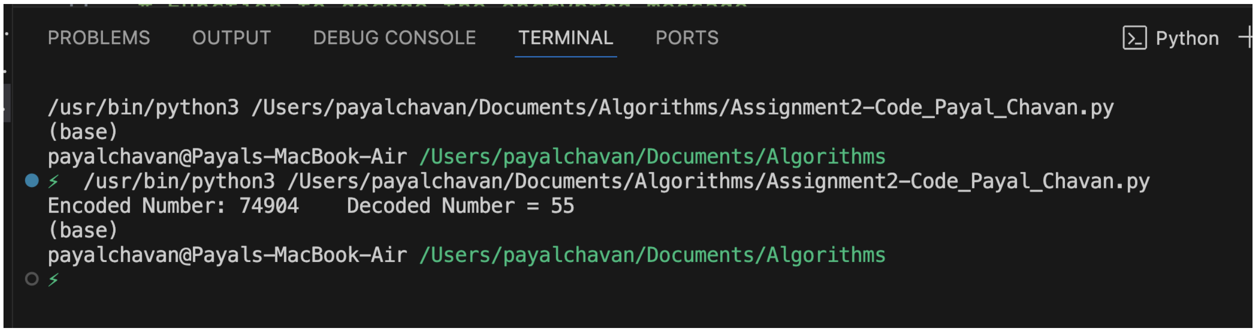
CS5800: Algorithms SEC 05 Summer Full 2024 (Seattle)
Homework 2 - Coding Problem

Payal Chavan

05/22/2024

Coding Break-encryption output:



```
/usr/bin/python3 /Users/payalchavan/Documents/Algorithms/Assignment2-Code_Payal_Chavan.py
(base)
payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms
● ⚡  /usr/bin/python3 /Users/payalchavan/Documents/Algorithms/Assignment2-Code_Payal_Chavan.py
Encoded Number: 74904    Decoded Number = 55
(base)
payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms
○ ⚡
```

Figure 1: Screenshot of Break-Encryption Output

Question 1: What is the big-Oh complexity of your algorithm in terms of the number of bits of the message?

Solution: The **break-encryption** function has a time complexity of $\mathcal{O}(N)$, where N is the input size.
However, for the complexity in terms of the number of bits, b, of the message is $\mathcal{O}(2^b)$.
This is because the number of possible messages is 2^b (since each bit can be either 0 or 1), and the function needs to iterate through all of these possibilities in the worst case.

Question 2: What is the relationship between the message's integer value and the number of bits?

Solution: If you have an integer value N, the number of bits b required to represent it in binary can be calculated using the formula: $b = \log_2(N) + 1$
The $\log_2(N)$ function gives the number of times we can divide N by 2 before we get a value less than or equal to 1. The $+1$ is there because we round down after the division.
For example, if N is 8 (which is 2^3), $\log_2(N)$ is 3. But we need 4 bits to represent the number 8 in binary (1000), so we add 1.

Question 3: What would be the run-time for your algorithm if the message size was 256 bits? Any guesses how long it would take to run?

Solution: The time complexity of the break-encryption function is $\mathcal{O}(2^b)$, where b is the number of bits. So, for a 256-bit message, the time complexity would be $\mathcal{O}(2^{256})$. To understand, 2^{256} is about 10^{77}. The runtime of the algorithm would depend on the specific key length. Assuming, if we could check a billion (10^9) keys per second, it would still take approximately (10^{68}) years to check all possible keys.
This is why RSA encryption is considered secure for practical purposes. The time required to break the encryption using

a brute force approach is excessively large.

Reflection:
Through this coding exercise, I have understood how the RSA Encryption works.

Acknowledgements:
I would like to express my sincere gratitude to the following individuals and resources for their invaluable contributions to this assignment:
1) Prof. Bruce Maxwell: Thank you, Professor Bruce Maxwell for your guidance in this assignment. Your expertise in algorithms greatly influenced my work.
2) Classmates and TA's: I appreciate the discussions of my classmates, and TA's who clarified my doubts.
3) DPV Algorithms Textbook: This textbook was a useful resource for me to understand the basics of cryptography.
4) https://www.geeksforgeeks.org/rsa-algorithm-cryptography/: This website provided clear explanations of RSA Algorithm.