Assignment 1

Question 1

(a) Worst-Case Number of Symbols for Encoding in Roman Numerals

To start, the best-case number of symbols for encoding is a one-to-one mapping between the Roman numeral system and the decimal (base-10) system. For example, the decimal number 1 is represented by the Roman numeral I, meaning one symbol is required. This implies that, for a positive integer n with m symbols, the lower bound of the corresponding Roman numerals is also m symbols.

Looking at the upper bound, one can prove that the encoding is bounded by $O(m^2)$. There exists some positive real $F_m \leq c \cdot m^2$ for all m in $\mathbb{N}$. Proof by construction will be used, where some concrete c is proposed and shown that it works.

Trying some values of m, such that the number of symbols needed increases:

Decimal Number	# of Symbols	Roman Numeral	# of Symbols
1	1	I	1
99	2	$XCIX$	4
999	3	$CMXCIX$	6
9999	4	$\overline{IX}CMXCIX$	9
99999	5	$\overline{XCI\overline{X}}CMXCIX$	11

Seeing what c would work:

$$m = 1, F_1 = 1, 1^2 = 1 \rightarrow c = 1 \; works$$
$$m = 2, F_2 = 4, 2^2 = 4 \rightarrow c = 1 \; works$$
$$m = 3, F_3 = 6, 3^2 = 9 \rightarrow c = 1 \; works$$
$$m = 4, F_4 = 9, 4^2 = 16 \rightarrow c = 1 \; works$$
$$m = 5, F_5 = 11, 5^2 = 25 \rightarrow c = 1 \; works$$

It appears that $c = 1$ works. Thus, this chosen c will be adopted and checked with a proof by induction:

Paolo Torres

- Base case: $m = 1, F_1 = 1, 1^2 = 1, 1 \leq 1 \rightarrow True$
- Step: Seek to show $F_m \leq m^2$ given that $F_k \leq k^2$ for all $k = 1, 2, \dots, m-1$
- Assume true for $m = k$, show true for $m = k + 1$
- $F_k = k^2, F_{k+1} = (k+1)^2, F_k \leq F_{k+1} \rightarrow Done$

Thus, the upper bound is $O(m^2)$. Furthermore, analyzing the examples used in the table above, it can be seen that the number of symbols needed for the decimal numbers, squared, are all ≤ the number of symbols needed for their Roman numeral equivalents. Therefore, the tight bound for this encoding is $\Theta(m^2)$.

(b) Worst-Case Number of Symbols for Encoding in Decimal System

The worst-case number of symbols to encode a positive integer in decimal notation is $\Theta(m)$. For every digit in the positive integer n, the decimal system allows each to be represented by a symbol from $1 - 9$. First prove that $F_m = \Omega(m)$:

- By trial and error: It appears that $F(m) \geq m$ for all $m \geq 1$
- To prove: For all positive integers $m \geq 1 \rightarrow F_m \geq m$
- By induction on m. Base case: $m = 1, F_m = 1, 1 \geq 1 \rightarrow True$
- Step, assume: Indeed, true that for all $m = 1, 2, \dots, k \rightarrow F_k \geq k$
- To prove: $F_{k+1} \geq k + 1$
- Indeed: $k + 1 \geq k \rightarrow Done$

Thus, the lower bound is $\Omega(m)$. Now for the upper bound, a similar proof from part (a) can be used to prove a bound of $O(m)$. Using $c = 1$ and a proof by induction on m:

- Base case: $m = 1, F_1 = 1, 1 \leq 1 \rightarrow True$
- Step: Seek to show $F_m \leq m$ given that $F_k \leq k$ for all $k = 1, 2, \dots, m-1$
- Assume true for $m = k$, show true for $m = k + 1$
- $F_k = k, F_{k+1} = k + 1, F_k \leq F_{k+1} \rightarrow Done$

Thus, the upper bound is $O(m)$. Therefore, the tight bound for this encoding is $\Theta(m)$. Comparing the Roman numeral system to the decimal system, the decimal encoding is better than the Roman numeral encoding from part (a). It is more efficient with respect to the number of symbols used, as it can represent these integers with the same or less symbols.

Paolo Torres

Assignment 1

Question 2

The goal for this problem is to prove that the Harmonic Series, namely $\sum_{i=1}^{n} \frac{1}{i}$, has an upper bound of $O(\log n)$. First, let $n = 2^k$ to exponentiate and work with a proof involving log. The Harmonic Series then becomes:

$$\sum_{i=1}^{2^k} \frac{1}{i} = H_{2^k}$$

Consequently, this series can be expanded as follows:

$$H_{2^k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \cdots + \frac{1}{2^{k-1}} + \frac{1}{2^{k-1}+1} + \cdots + \frac{1}{2^k - 1}$$

To carry the proof through, group the terms by powers of two and compare, similar to how divergence is proven for the Harmonic Series [1]. Thus, the grouping is shown as:

$$H_{2^k} = (1) + \left(\frac{1}{2} + \frac{1}{3}\right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right) + \cdots + \left(\frac{1}{2^{k-1}} + \frac{1}{2^{k-1}+1} + \cdots + \frac{1}{2^k - 1}\right)$$

There is also one extra $\frac{1}{2^k}$ term in the series. Then, a valid upper bound can be generated by using this strategy of increasing powers of two and applying it to the grouped terms as follows:

$$U_{2^k} = (1) + \left(\frac{1}{2} + \frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right) + \cdots + \left(\frac{1}{2^{k-1}} + \frac{1}{2^{k-1}} + \cdots + \frac{1}{2^{k-1}}\right)$$

Clearly, $H_{2^k} \leq U_{2^k}$, where U_{2^k} represents the desired upper bound in question. Now, note that each grouped set of terms adds up to 1, therefore all the grouped terms added together equals k. This, along with the extra term, leads to the following inequality:

$$H_{2^k} \leq k + \frac{1}{2^k}$$

Subbing back in $n = 2^k$ leads to the following equation for k:

$$\log n = \log 2^k = k \cdot \log 2$$

$$k = \frac{\log n}{\log 2} = \log_2 n$$

Paolo Torres

Assignment 1

Using the hint of adopting the exact form of $\log_b n + c$, together with the power of two grouping, leads naturally to use $b = 2$ and $c = 1$. Finally, this results in the following inequality:

$$H_n \leq \log_2 n + 1$$

With this result, the value of $c = 1$ is an additive constant, meaning it can be omitted from the big O notation. This simplifies the inequality:

$$H_n \leq \log_2 n$$

Therefore, the final upper bound of the Harmonic Series is $O(\log n)$. In mathematical form, this means that:

$$\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$$

This concludes that the upper bound of the Harmonic Series is indeed $O(\log n)$.

Paolo Torres

Assignment 1

Question 3

- First adopt case-analysis (tackle each case one at a time)
- Use logical deduction for every case?
- More specifically, prove equality ($x = y$) in two parts: $x \leq y$ and $y \leq x$
- For example, prove $\gcd(a, b) = \gcd(a - b, b)$ by proving $\leq$ and $\geq$
- Example: Prove the case that both a, b are even:
    - Let $g = \gcd(a/2, b/2)$
    - Then, g divides both $a/2$ and $b/2$
    - Therefore, $2g$ divides both a and b
    - Therefore, $\gcd(a, b) \geq 2g$
    - Then proceed to prove $\leq$
- To identify what n is, first look at what input(s) the algorithm in question takes
    - And that is a pair of non-negative integers $\langle a, b \rangle$
- So, to encode (i.e., write down these two numbers), it would take $\Theta(\log a + \log b)$ symbols, which is the same as $\Theta(\log(\max\{a, b\}))$, so this is our n

Question 4

- First enunciate a correctness property for $FactTwo$ that assumes only that $lo \leq hi$ and $lo, hi \in \{1, 2, \ldots, n\}$
- But doesn't demand that $lo = 1$ nor that $hi = n$

Question 5

- We can infer things based on properties of mod
- E.g., since:

$$a \times a \bmod p = (a \bmod p) \times (a \bmod p) \bmod p$$

- Then we know that:

$$a^b \bmod p = (a \bmod p)^b \bmod p$$

Paolo Torres

Assignment 1

References

[1] E. W. Weisstein, "Harmonic Series," Wolfram MathWorld, [Online]. Available: https://mathworld.wolfram.com/HarmonicSeries.html. [Accessed 23 January 2020].

Paolo Torres