Algorithm Analysis

Abraham J. Reines

April 14, 2024

Task 6.3.1

In the examination of behavior of function $f(n) = 10^{12}n^3 + 10^6n^2 + n + 1$ against polynomial $g(n) = n^4$, we have a theoretical analysis complemented by limit calculations both with paper and pencil and using SymPy to derive the following conclusions:

1. **Is $10^{12}n^3 + 10^6n^2 + n + 1$ not equivalent to $\Theta(n^4)$?**
   Yes. $10^{12}n^3 + 10^6n^2 + n + 1$ is not $\Theta(n^4)$ because the highest order term, n^3, says what the growth rate of the function is. As n appraoches infinity, the n^4 term would grow faster than the n^3 term. $10^{12}n^3 + 10^6n^2 + n + 1$ is $\Theta(n^3)$, not $\Theta(n^4)$. It cannot be bounded above and below by n^4 for a large n.

2. **Is $10^{12}n^3 + 10^6n^2 + n + 1$ equivalent to $o(n^4)$?**
   Yes. $10^{12}n^3 + 10^6n^2 + n + 1$ is $o(n^4)$ because as n grows, the function $10^{12}n^3 + 10^6n^2 + n + 1$ increase at a slower rate than a function proportional to n^4. Thus, it can be bound above by $C \cdot n^4$ for some constant C and sufficiently large n.

3. **Is $10^{12}n^3 + 10^6n^2 + n + 1$ equivalent to $\omega(n^4)$?**
   No. $10^{12}n^3 + 10^6n^2 + n + 1$ is not $\omega(n^4)$ because it does not grow faster than any function proportional to n^4. For $f(n)$ to be $\omega(g(n))$, $f(n)$ must eventually exceed $C \cdot g(n)$ for any constant C, which is not the case here as n^3 is the dominant term in $f(n)$, and it grows slower than n^4.

6.3.2

Application Case:

- 250GB hard disk with 160GB of files.
- CPU with one core, 4GHz clock speed, and 64-bit registers.
- Capable of 4×10^9 8-character comparisons per second.

Search Time Analysis

1. **1KB Signature:**

   The time needed to search for a 1KB signature case is 5 seconds. This is a worst case scenario, which means it is unrealistic and a Boyer-Moore calculartion will be better.

2. **Database of 1 Million 1KB Signatures:**

   For a Database with 1 million 1KB signatures, 57.87 days will approximately be a worst case scenario. This is where the signature searching is idependant and linear. THe Boyyer-Moore algorithm is going to be gaster than this with sublinear performatnce.

3. **Improving Efficiency:**

   To further improve the efficiency of malware/virus detection through string matching, we utilize paralell processing by distributing the search workload across multiple CPU cores or even differant machines; implementing more advanced string matching algorithms such as Aho-Corasick, which is desinged for searching multiple patterns simultaneously, can significantly reduce the time complexitity. Optimizing the algorithm fo preprocessing of the virus signatures and employing hueristic methods to skip unlikely sections of data can also enhance performance.

Screenshots of Hand Written Calculations

1. $\forall\, c > 0,\ \exists\, n$ s.t $\underbrace{10^{12} \cdot n^3}_{} + 10^6 \cdot n^2 + n + 1 > c \cdot n^4$

Since $n^4 > n^3$, the polynomial cannot be $O(n^4)$. n^4 grows faster than n^3. The function does not reach or exceed the growth of n^4!

2. $$\lim_{n \to \infty} \left(\frac{10^{12} \cdot n^3 + 10^6 \cdot n^2 + n + 1}{n^4} \right)$$

$$= \lim_{n \to \infty} \left(\frac{10^{12}}{n} \right) + \lim_{n \to \infty} \left(\frac{1{,}000{,}000}{n^2} \right) + \lim_{n \to \infty} \left(n^{-3} \right) + \lim_{n \to \infty} \left(n^{-4} \right) = 0 + 0 + 0 + 0$$

$$= \boxed{\emptyset}$$

This means the polynomial grows slower than n^4; exactly what $o(n^4)$ describes! The polynomial is $o(n^4)$!

3. For the polynomial to be $\omega(n^4)$ it must be infinite. The limit is not ∞!

Figure 1: Hand written calculations for task 6.3.1

1. $\dfrac{160 \cdot 10^9}{8} \div (4 \cdot 10^9)_s = 5_s$ for 1 KB sig.

2. $$\frac{\frac{\frac{5 \cdot 10^6}{60}}{60}}{24} = 57.87_s$$ for 1,000,000 1KB sig.

Figure 2: Hand written calculations for task 6.3.2

References

1. Wikipedia contributors. (n.d.). Boyer–Moore string-search algorithm. *Wikipedia, The Free Encyclopedia*. Retrieved April 14, 2024, from `https://en.wikipedia.org/wiki/Boyer-Moore_string-search_algorithm`

2. HackerEarth. (n.d.). Boyer-Moore algorithm. *HackerEarth*. Retrieved April 14, 2024, from `https://www.hackerearth.com/practice/algorithms/string-algorithm/string-searching/tutorial/`

3. Programiz. (n.d.). *Asymptotic Analysis: Big-O Notation and More*. Retrieved April 14, 2024, from `https://www.programiz.com/dsa/asymptotic-notations`

4. Wikipedia contributors. (2024, April 13). Asymptotic analysis. In *Wikipedia, The Free Encyclopedia*. Retrieved 15:00, April 14, 2024, from `https://en.wikipedia.org/w/index.php?title=Asymptotic_analysis&oldid=currentID`

Academic Integrity Pledge

"This work complies with the JMU honor code. I did not give or receive unauthorized help on this assignment."