Analysis and Design of Algorithms

April 2019

1 Warm up

Lets modify the classic merge sort algorithm a little bit. What happens if instead of splitting the array in 2 parts we divide it in 3? You can assume that exists a three-way merge subroutine. What is the overall asymptotic running time of this algorithm?

The result would still be the same, talking in broad strokes. The log base would most likely be 3 if the operations had the same running time, but they would more likely be less efficient, and would be more likely to be overkill, since multiples of 3 and 9 are rarer than those of 2 and 4, and therefore would likely be wasted time.

So I have calculated around nlog(3)n.

BONUS: Implement the three-way merge sort algorithm.

2 Competitive programming

Welcome to your first competitive programming problem!!!

- Sign-up in Uva Online Judge (`https://uva.onlinejudge.org`) and in CodeChef if you want (we will use it later).
- Rest easy! This is not a contest, it is just an introductory problem. Your first problem is located in the "Problems Section" and is **100 - The 3n + 1 problem**.

- Once that you finish with that problem continue with **458 - The Decoder**. Again, this problem is just to build your confidence in competitive programming.
- *BONUS:* **10855 - Rotated squares**

3 Simulation

Write a program to find the minimum input size for which the merge sort algorithm always beats the insertion sort.

- Implement the insertion sort algorithm
- Implement the merge sort algorithm
- Just compare them? No !!! Run some simulations or tests and find the average input size for which the merge sort is an asymptotically "better" sorting algorithm.

Note: Include (.tex) and attach(.cpp) your source code and use a dockerfile to interact with python and plot your results.

BONUS: Compare both algorithms against any other sorting algorithm

4 Research

Everybody at this point remembers the quadratic "grade school" algorithm to multiply 2 numbers of k_1 and k_2 digits respectively.

Your assignment now is to compare the number of operations performed by the quadratic grade school algorithm and Karatsuba multiplication.

- Define Karatsuba multiplication

Karatsuba multiplication is an algorithm that uses a divide and conquer approach to multiplying, breaking any given number into pieces and recursively calculating the sums and products between them in a seemingly random way, getting maybe better performance than the traditional approach.

- Implement grade school multiplication
- Implement Karatsuba multiplication
- Compare Karatsuba algorithm against grade school multiplication

Karatsuba works recursively, while grade school works iteraively.Karatsuba will have better complexity than the (n^2) complexity that we will get on grade school, getting a n^{nlog3} complexity.

- Use any of your implemented algorithms to multiply $a * b$ where:

  a: 3141592653589793238462643383279502884197169399375105820974944592

  b: 2718281828459045235360287471352662497757247093699959574966967627

Note: Include(.tex) and attach(.cpp) your source code, of course.

BONUS: How about Schönhage-Strassen algorithm ?

5 Wrapping up

Arrange the following functions in increasing order of growth rate with $g(n)$ following $f(n)$ if $f(n) = \mathcal{O}(g(n))$

1. n^2
2. $n^2 log(n)$
3. $n^{log(n)}$
4. 2^n
5. 2^{2^n}