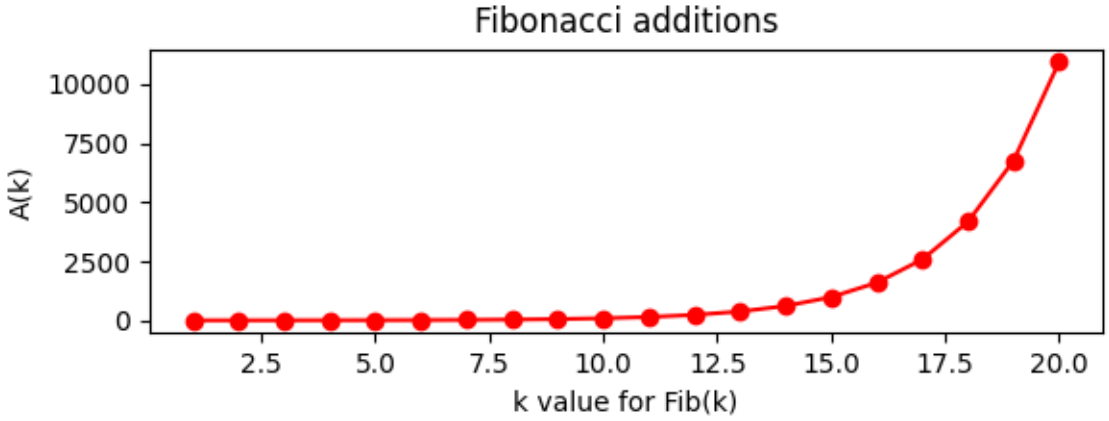
Jordan Edginton

CS-415 SP '21

Project #1 report

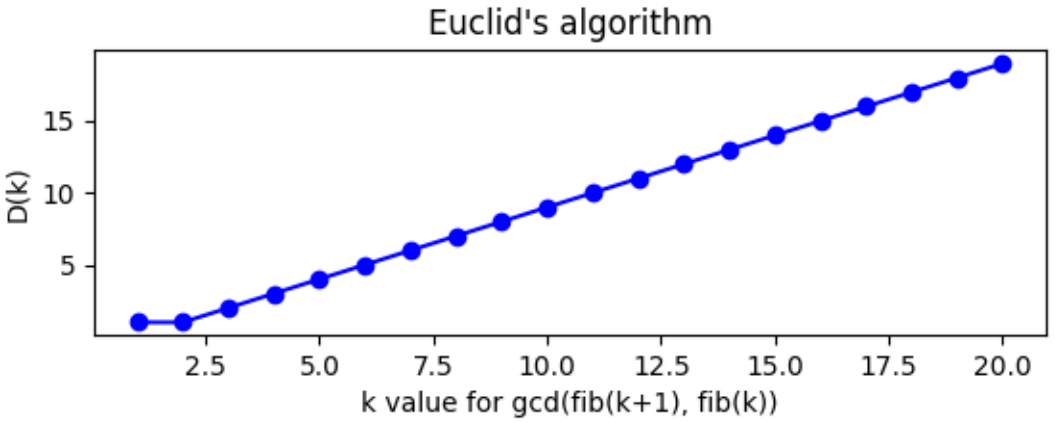
Task 1:

General: The values for k used in the Task 1 graphs are 1 to 20. I thought 10 was a bit too small and went with 20, but after making the scatterplots I see that 20 may have been overkill.



$\Theta(n^2)$

a) Here we see the graph for the additions of a Fibonacci sequence. Looking at it, we can see that A(n) is increasing in an exponential curve. if we think about it in terms of the function itself (A(n) = A(n-1) + A(n-2)) it makes sense that it would be an exponential growth since as k increases the amount we add to A(K) will also increase and not be constant (which would be Θ(n)). I am a bit surprised though at how steep the curve starts to become after each iteration when it gets to 15.

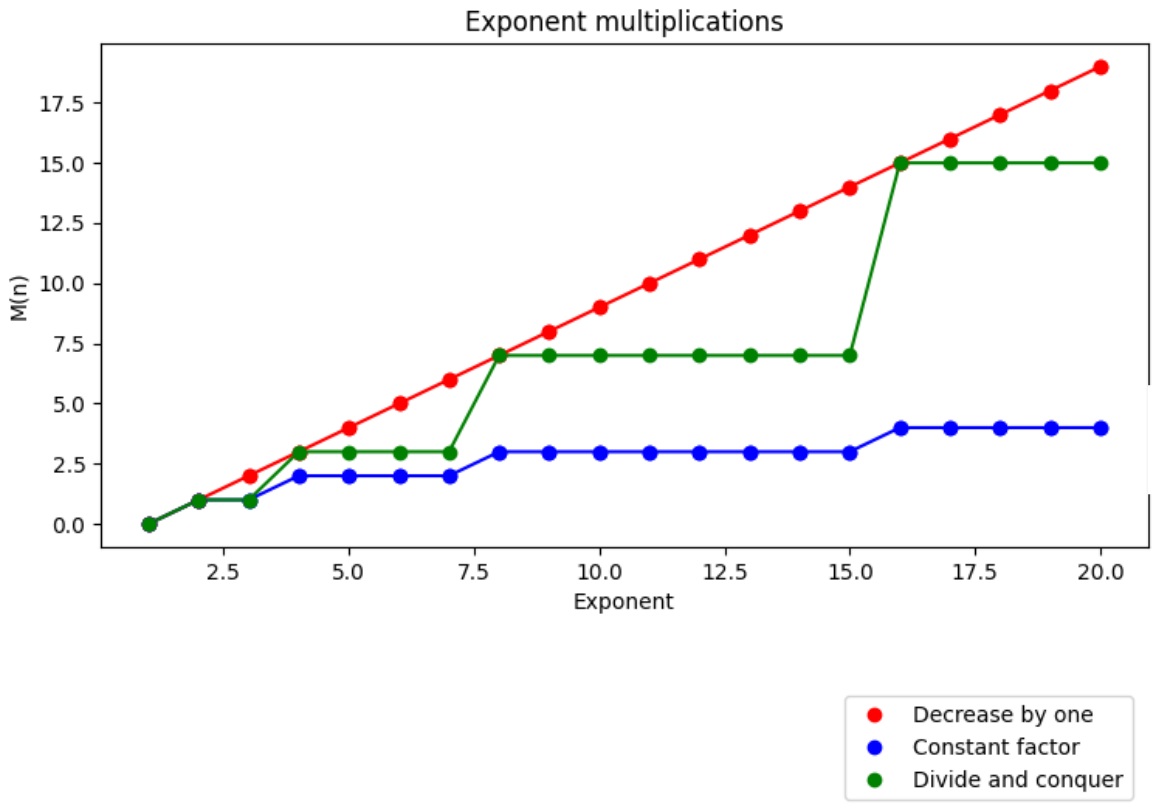


$\Theta(n)$

b) Here we have a graph showing the divisions used by Euclid's algorithm to solve gcd(fib(k+1), fib(k)). Seeing how the graph increases at a constant rate (linear), we can conclude that the algorithm is Θ(n). Honestly, this is a bit surprising to me. Seeing the growth rate of fib(), one would think this would possibly look a little similar, but no. Inputs do not matter for Euclid.

Task 2:

General: I chose the max to be 20 for this graph to make sure I got 16 (2^4) in it and a bit extra.



Θ(n)

Θ(log(n))

Θ(log(n))

This is a bit interesting. I know from class that Decrease By One (DBO) is Θ(n), which we can see from the graph since it's a straight light, and so I was expecting to take off from the other two. Instead, Divide And Conquer (DAC) is equal where the exponent is equivalent to an exponential of 2. I can also imagine that the divide between DBO and DAC will grow as the numbers get larger. As for Decrease by Constant Factor (DCF) though, I think the reason it's less than DAC is that it's calling the function less and therefore not triggering the variable iterator like DAC does. I don't know if that's what the outcome is or if I had my iterator in the wrong place. Either way, you can see that both DCF and DAC are logarithmic curves.

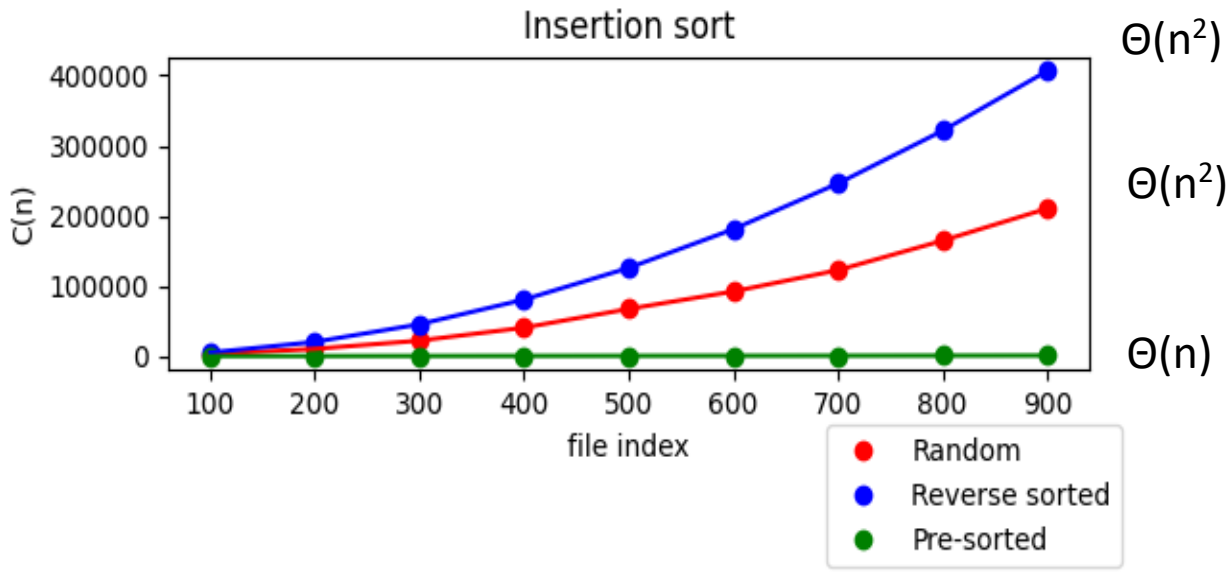
Decrease by one: Θ(n)

Decrease by constant factor: Θ(log(n))

Divide and conquer: Θ(log(n))

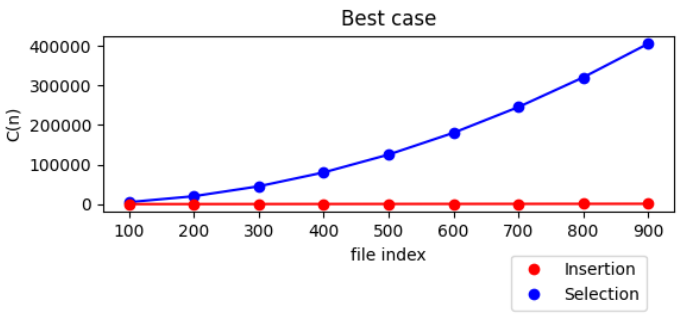
Task 3, Insertion sort:

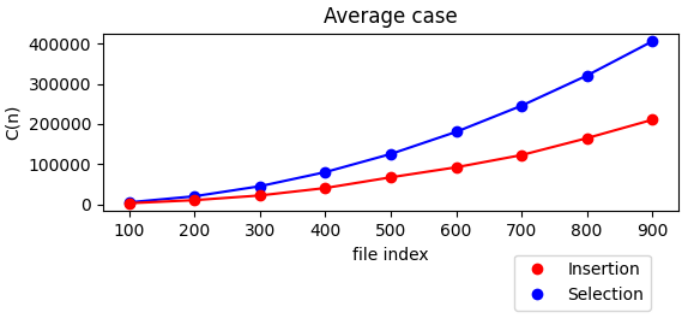
General: I chose the max to be 900 for this graph for run-time reasons. I was a bit surprised to see that it only needed that much anyways.



Here we have our graph for insertion sort based on a random, reverse sorted, and pre-sorted list. Looking at this graph, it makes you realize just how big of a difference $\Theta(n^2)$ is compared to $\Theta(n)$. The pre-sorted list looks as though it's flat when compared to the other two, and I chose the lowest item amounts out of the files too. It's also interesting to see that even though random (which we'll say is our average case) is $\Theta(n^2)$, it's still significantly better than the worst case. I was expecting it to be much closer to the worst case and not practically halfway between best and worst.

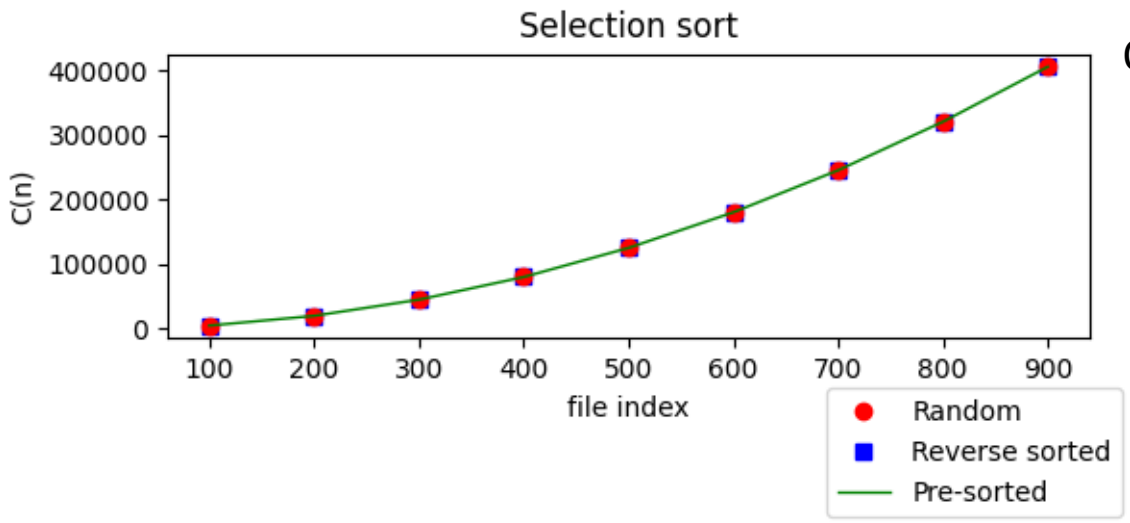
Best case: $\Theta(n)$ average case: $\Theta(n^2)$ worst case: $\Theta(n^2)$





Task 3, Selection sort:

General: I chose the max to be 900 for this graph for run-time reasons.



$\Theta(n^2)$

Lastly, we have our graph for selection sort based on a random, reverse sorted, and pre-sorted list. I tried to make each case a different shape so we can see if they were all overlapping each other, and they are. This is showing us that the best, average, and worst case are all the same. We can also see that the graph has an exponential curve which is $\Theta(n^2)$. Since we are checking comparisons and not swaps, it makes sense that all three lists have the same amount since the algorithm will always check every item in the list multiple times, but swapping only once per item.

Best case: $\Theta(n^2)$ average case: $\Theta(n^2)$ worst case: $\Theta(n^2)$