CS35, Lab 04: QuickSort and Big-O
icourtn1, sbegum1

Part II: Written Assignment

Big-O Proofs

1. $5n^3 + n^2 + 4$ is $O(n^3)$

   We know $n > 0$ as it's the size of the problem and n is an integer.

   We also know that $5n^3 \leq 5n^3$ and $n^2 \leq n^3$ and $4 \leq n^3$, so

   $5n^3 + n^2 + 4 \leq 5n^3 + n^3 + n^3 = 7n^3$

   … so long as we choose a constant $c \geq 7$ and $n_0 \geq 1$, we can say that

   $5n^3 + n^2 + 4$ is $O(n^3)$ because $5n^3 + n^2 + 4 \leq 7n^3$.

2. $2n^4 - 3n^2 + n$ is $O(n^4)$

   We know $n > 0$ as it's the size of the problem and n is an integer.

   So $2n^4 < n^4$ and $-3n^2 < 0$ and $n < n^4$

   Therefore, $2n^4 - 3n^2 + n \geq n^4 - 0 + n^4$

   Simplified, $2n^4 - 3n^2 + n \geq 2n^4$

   So, as long as $c \geq 2$ and $n_0 \geq 1$, we can say is $2n^4 - 3n^2 + n$ is $O(n^4)$.

Part III: Mystery Functions

Big-O Runtimes

A. The runtime of function $fnA(n)$ is $O(n)$ because the for loop will index through the values at the indices from 1 to $\frac{n}{2}$, setting a variable, a, equal to the index at that given instance. $\frac{n}{2}$ is the maximum index so a will get $A[i]$, $\frac{n}{2}$ times. Disregarding the constants, $fnA(n)$ will take the linear time complexity of n.

B. The runtime of function $fnB(n)$ is $O(n^2)$ because there are two for loops from 1 to n that each take n number of steps so complexity is $n * n$.

C. The runtime of function $fnC(n)$ is $O(log_2 n)$ because for every loop, j gets j multiplied by 2. It will continue to run for $log_2 n$ steps until $j > n$.

D. The runtime of function $fnD(n)$ is $O(n^4)$ because there are two for loops, each indexing from 1 to $n * n$, that takes a step at each instance. So, the total number of steps will be $n * n * n * n = n^4$.

E. The runtime of function $fnE(n)$ is $O(n)$ because the first for loop goes from 1 to n, and the second goes from 1 to 4, making it linear because the second loop is only multiplying the number of steps by 4 to get $4n$ steps.

F. The runtime of function $fnF(n)$ is $O(n^3)$ because the indices of the while loop will increment by +1 until i is greater than n^3, taking n^3 steps.

Sorted:

1. fnC(n)
2. fnA(n)
3. fnE(n)
4. fnB(n)
5. fnF(n)
6. fnD(n)

function_timer

1. $f1 = fnF(n)$

   Function F has $O(n^3)$, so it is the second slowest algorithm, since the only one that is slower is function D which is $O(n^4)$. The two plots below show that f6 has the biggest runtime, and $f1$ has the second biggest runtime. So, $f1$ is $fnF(n)$.

2. $f2 = fnA(n)$

   Function A has $O(n)$ and takes $\frac{n}{2}$ steps, making it the second fastest algorithm. The plot of $f2$ shows that it is faster than $f4$ but slower than $f3$, so $f2$ must be $fnA(n)$.

3. $f3 = fnC(n)$

   Function C has $O(log_2 n)$, so it is the fastest algorithm. Given that logarithmic algorithms are faster than linear algorithms, we can assume Function C is faster than Functions A and E. The plot of $f3$ shows that it is faster than $f2$ and $f4$, so $f3$ must be $fnC(n)$.

4. $f4 \ = \ fnE(n)$

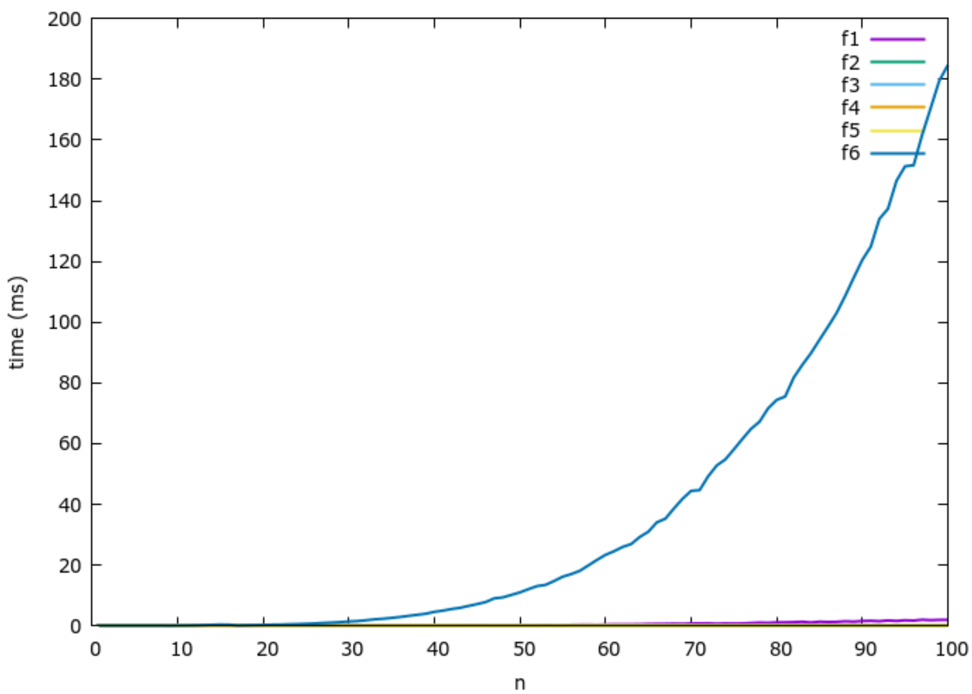
Function E has $O(n)$ but takes $4n$ steps to run, making it the slowest algorithm out of the linear ones and the third fastest algorithm overall. As plot 3 shows, $f4$ has longer runtimes than $f3$ and $f2$, so it must be $fnE(n)$.

5. $f5 \ = \ fB(n)$

Function B is $O(n^2)$, making it the third slowest algorithm behind F and D. Plot 2 shows that $f5$ is slower than $f2$, $f3$, or $f4$, since it is the only visible function. Other plots also confirmed that $f5$ is slower than those other functions. So, $f5$ is $fnB(n)$.

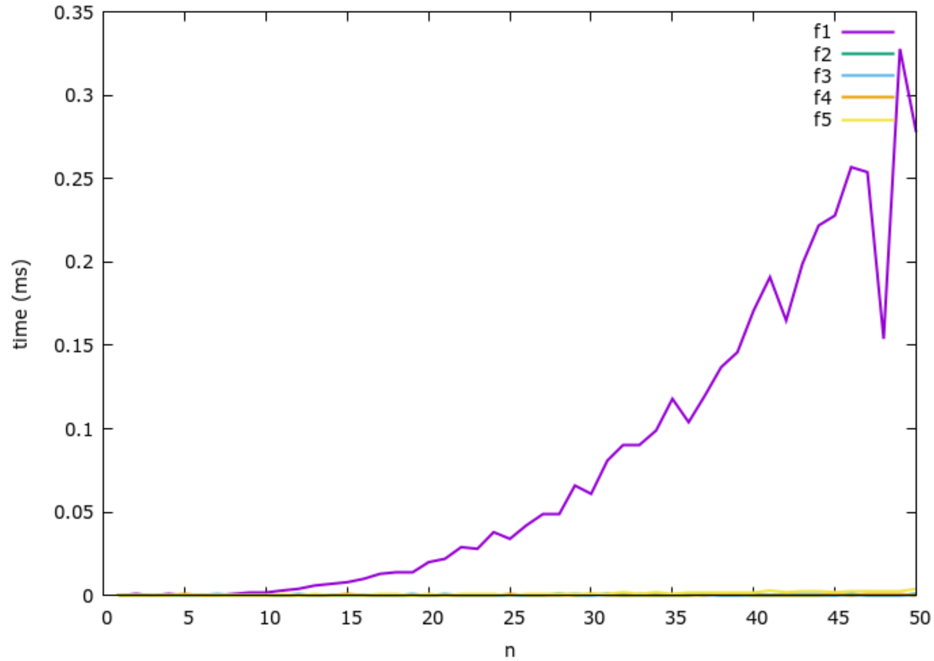
6. $f6 \ = \ fD(n)$

Function D is $O(n^4)$, so it is the slowest algorithm. The first plot below shows that $f6$ has the longest runtime by far. So, $f6$ is function $fnD(n)$.



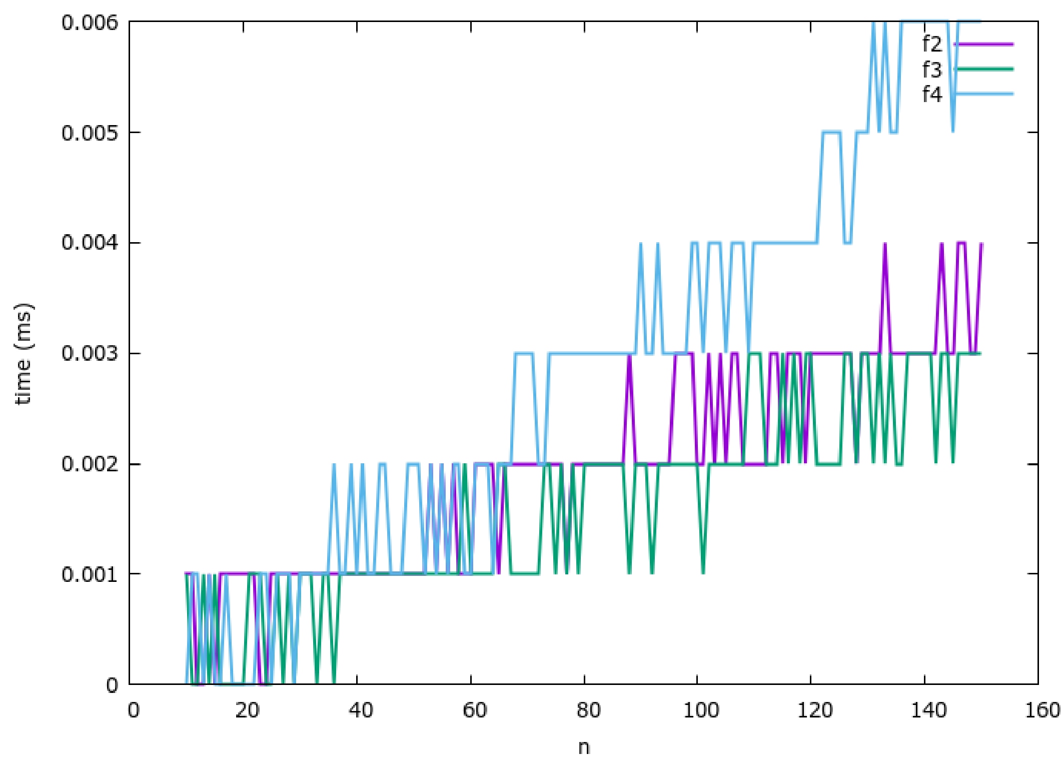
Plot 1. $f1, f2, f3, f4, f5\,,\ f6$ from $n = 1$ to $n = 100$

This plot shows that $f6$ has the longest runtimes, so it must be $fnD(n)$ which is $O(n^4)$



Plot 2. $f1$, $f2$, $f3$, $f4$, $f5$ from $n = 1$ to $n = 50$

This plot shows $f1$ has the second longest runtimes, so it must be $fnF(n)$ which is $O(n^3)$



Plot 3. $f2$, $f3$, $f4$ from $n = 1$ to $n = 150$

From the previous plots, it is clear that $f2$, $f3$, and $f4$ are the fastest functions. So, they probably correspond to the linear and logarithmic algorithms. $fnE(n)$ is $O(4n)$, $fnC(n)$ is $O(log_2 n)$, and $fnA(n)$ is $O(\frac{n}{2})$. The plot above shows that $f4$ is the slowest, $f2$ in the middle, and $f3$ is the fastest. So, $f4$ is probably $fnE(n)$, since it has a runtime of 4n. $fnA(n)$ has a runtime of approximately $O(\frac{n}{2})$, and is therefore slower than $fnC(n)$ but faster than $fnE(n)$, so $f2$ must be $fnA(n)$. Given that $f3$ has the fastest runtime and $fnC(n)$ is $O(log_2 n)$, which is faster than linear time complexities, $f3$ must be $fnC(n)$.