Marc Clinedinst
Kelby Faessler
James Fitzwater
Tom Gariepy
Sean Reilly
Joseph Struth
CS 261 - Group 1
28 June 2015

Worksheet 9

Suppose by careful measurement you have discovered that a program has the running time as shown at right. Describe the running time of each function using bigOh notation.

3n^3 + 2n + 7	The dominating part of the function to the left is 3n^3, because this part of the function grows more rapidly than 2n or the constant 7. Thus, this function has O(n^3) complexity; another way of stating this is that this algorithm has cubic complexity.
(5 * n) * (3 + log n)	Multiplying this out, we get 15n + 5n log n. The dominating part of this function is 5n log n. Thus, this function has O(n log n) complexity.
1 + 2 + 3 + … + n	1 + 2 + 3 + . . . + n = $\frac{n(n+1)}{2}$ = $\frac{n^2+n}{2}$ The nth sum in this formula is the triangular number, given by the equations above. This function has O(n^2), or quadratic complexity.
n + log n^2	Using the properties of logarithms, we get n + 2 log n. n dominates this function, so this function has O(n), or linear, complexity.
((n+1) log n) / 2	Multiplying this out, we get (n log n + log n) / 2. Since n log n dominates log n, this function has O(n log n) complexity.
n^3 + n! + 3	The factorial function (n!) dominates both the cubic (n^3) and constant (3) terms in this function. As a result, this function has O(n!) or factorial complexity.
2^n + n^2	2^n is the dominating term in this equation.

	Therefore, this function has O(2^n) or exponential complexity.
n (sqrt(n) + log n)	Multiplying this out, we get n sqrt(n) + n log n. The first term, n sqrt(n), dominates the second term, n log n. As a result, this function has O(n sqrt(n)) complexity.

Using the idea of dominating functions, give the big-Oh execution time for each of the following sequences of code. When elipses (…) are given you can assume that they describe only constant time operations.

for (int i = n; i > 0; i = i / 2) { … } for (int j = 0; j * j < n; j++) …	The first loop is logarithmic complexity because the size is cut in half each iteration, similar to a binary search. The second conditional is j^2 < n, which can be re-written as j < sqrt(n). Thus, this loop has sqrt(n) complexity. Combining these loops together, we get the function log n + sqrt(n). Since sqrt(n) dominates log n, this function has O(sqrt(n)) complexity.
for (int i = 0; i < n; i++) { for (int j = n; j > 0; j = j / 2) { … } for (int k = 0; k < n; k++) { … } }	The outer loop executes n times. The first inner loop executes log n times. The second inner loop executes n times. Combining this information together, we get the function n(log n + n), which multiplies out to n log n + n^2. Given that n^2 dominates n log n in this function, the function has O(n^2) or quadratic complexity.
for (int i = 0; i < n; i++) … for (int j = 0; j * j < n; j++) …	The first loop executes exactly n times, therefore it has linear complexity. The second loop executes root n times, therefore it has root n complexity. Adding them together, we have n + sqrt(n). Linear complexity dominates root-n complexity, therefore this code segment has O(n) or linear complexity
for (int i = 0; i < n; i++)	The first loop executes n times. This is also

… **for (int j = n; j > 0; j--)** **…**	true of the second loop--it, too, executes n times. Adding this together, we get n + n, or 2n. This function has O(n), or linear complexity.
for (int i = 1; i * i < n; i += 2) **…** **for (int i = 1; i < n; i += 5)** **…**	The first loop executes sqrt(n) times. The second loop executes n times. Adding these together, we get sqrt(n) + n. In this case, n clearly dominates sqrt(n). Thus, the function has O(n), or linear, complexity.

Worksheet 10

Warehouse video keeps a list of titles for their 7000 item inventory in a simple unsorted list. To find out how many copies of "Kill Bill" they have using the countOccurrences algorithm takes about 45 seconds. They recently acquired a competing video store, and now their inventory has 43000 items. How long will it take to search?

Use the formula f(n1) / f(n2) = t1 / t2 where f(n1) and f(n2) are the complexity equations with n values substituted, and t1, t2 are the corresponding execution times.

In this case, they must search every item in the inventory because it is unsorted. This requires one by one search through the database at linear complexity. Substituting the values we're given, our equation becomes

7000 / 43000 = 45 sec / X

Solving for X yields 276.43 seconds to search the new inventory.

Suppose you can multiply two 17 by 17 matrices in 33 seconds. How long will it take to multiply two 51 by 51 element matrices. (By the way 51 is 17 times 3).

Using the fact that the ratio of the big-O's is equivalent to the ratio of execution times for the function we can find the execution time for multiplying two 51 x 51 matrices. Given that multiplying two matrices is $O(n^3)$ we have all the information we need to solve for the execution time.

Using f(n1) / f(n2) = t1 / t2
with n1 = 17 , n2 = 51, t1 = 33 seconds we get

$$\frac{17^3}{51^3} = \frac{33\,s}{t_2} \qquad \frac{4913}{132651} = \frac{33\,s}{t_2} \qquad \frac{1}{27} = \frac{33\,s}{t_2} \qquad \frac{33\,s}{\frac{1}{27}} = t_2 \quad 891\,s = t_2$$

It will take 891 seconds to multiply two 51 x 51 matrices using the matMult() function.

If you can print all the primes between 2 and 10000 in 92 seconds, how long will it take to print all the primes between 2 and 160000?

Using the printPrimes() function which is O(n*Sqrt(n)) and the same equation as the previous problem.
We are given:
Primes between 2 and 10000, so 10000 - 2 = 9998, n1 = 9998
Primes between 2 and 160000, so 160000 - 2 = 159998, n2 = 159998
and t1 = 92 s
Solving for t2 or the execution time it takes printPrimes() run for 159998 elements

$$\frac{9998 \cdot \sqrt{9998}}{159998 \cdot \sqrt{159998}} = \frac{93\,s}{t_2} \quad \frac{9998 \cdot 100}{159998 \cdot 400} = \frac{93\,s}{t_2} \quad \frac{999800}{63999200} = \frac{93\,s}{t_2} \quad \frac{93}{\frac{999800}{63999200}} = t_2 \quad 5889\,s = t_2$$

So to print all the primes between 2 and 160000 using the function printPrimes() it will take 5889 seconds.

Group Meeting Minutes

Google Docs does not save chat history, so I am copying and pasting the conversation Kelby and I had earlier this evening. I will summarizing this in the final document that is submitted.

Thomas Gerard Gariepy left group chat.

---

me

10:47 PM

Howdy.
Brushing up on this logarithm stuff. Haw.

Kelby Faessler

10:48 PM

Hey there!

---

me

10:48 PM

Hiya

Kelby Faessler

10:48 PM

Yes trying to keep track of things
Funny we're analyzing complexities although we haven't had algorithms yet. I guess this is a primer

---

me

10:49 PM

I'm not sure if these are right . . . I'm just trying to take a stab at them.
Yeah, I thought it was a little weird, too, but w/e.
I did most of the programs for Assignment 1 and ended up writing a couple of sorting algorithms for them.
So I think we'll be touching on algorithms a bit this term.

Kelby Faessler

10:50 PM

I'm just double checking the answers so far
1 and 2 look good

---

me

10:50 PM

Cool cool cool.

Kelby Faessler

10:50 PM

I'm wondering about 3

---

me

10:50 PM

If you want to change anything, please feel free. I am not going to be offended even if I am watching you.

Yeah . . . I honestly don't know about that one.

Kelby Faessler

10:51 PM

I can see why you wrote O(n)

---

me

10:51 PM

It's wrong.

Kelby Faessler

10:51 PM

that would've been my first guess

---

me

10:51 PM

https://en.wikipedia.org/wiki/1_%2B_2_%2B_3_%2B_4_%2B_%E2%8B%AF

Kelby Faessler

10:51 PM

I could also see why it would be constant complexity

---

me

10:52 PM

so

Kelby Faessler

10:52 PM

so quadratic?

---

me

10:52 PM

1 + 2 + 3 + . . . + n actually comes out to [n(n+1)] / 2
so yeah
i think it's quadratic

Kelby Faessler

10:52 PM

ok I'm good with that

---

me

10:53 PM

good call
It's been a while since I took calculus.
baaaaaaaaaah

Kelby Faessler

10:58 PM

this stuff is actually not too bad with basic algebra

---

me

10:58 PM

Yeah, I just like to complain about math.

Kelby Faessler

10:58 PM

I assume we'll learn in the algorithms course how to actually determine which parts of functions get which complexities
haha no doubt

---

me

11:01 PM

So that first loop
It's dividing in half each time, right?
Would that be log?

Kelby Faessler

11:01 PM

ohhh
I think you're right

---

me

11:01 PM

I feel like the second one would be the same, maybe, too but i'm not sure

---

Kelby Faessler

11:02 PM

She wants the execution time
How do we get that?

---

me

11:03 PM

Same way as above . . . we have to figure out the function first, then use that to get execution time.
So . . . just for sake of argument
if the first loop is linear and the second one is cubic
we'd write down n + n^3
and then just get the complexity same way as above
at least that's how i'm reading it

Kelby Faessler

11:05 PM

right, we get the complexity
but somehow we have to use the complexity to get the execution time
unless the complexity IS the execution time

---

me

11:06 PM

Yeah, I think that's what BigOh notation is. It's just abstracting away the concept of wall time, since that's not the best way to measure it.

Kelby Faessler

11:07 PM

yep, agree
I was thinking we were somehow supposed to calculate wall time using the complexity
I think we're just supposed to find the complexity again
the first loop is definitely logarthmic complexity
think about a binary search where it's using divide and conquer

---

me

11:07 PM

Yep

Kelby Faessler

11:09 PM

a binary search eliminates half every time and it is log complexity
this is the same concept
eleminating half each time
is there a mathematical way we can prove that?
the second loop might be root n complexity

---

me

11:11 PM

I think what you have written is fine. I'm looking at the reading, and it's pretty similar to how they explain it.
i think the second loop is sqrt

Kelby Faessler

11:17 PM

I agree on that second problem
nice job

Kelby Faessler

11:20 PM

Ok I've got to take off. Have to pack for my trip tomorrow

---

me

11:20 PM

All right. Have a good trip!

Kelby Faessler

11:21 PM

I'll try and check in throughout the weekend when I get some downtime
Thanks, have a nice weekend
laters

---

me

11:21 PM

Cool cool cool.
Kelby Faessler left group chat.