CMPS Assignment #1

Zoe Murphy

1) Asymptotic Notation

1a) Is $2^{n+1} \in O(2^n)$? Why or why not?

- Seeing if 2^{n+1} grows at most as fast as 2^n

$2^{n+1} = 2^n \times 2^1$ (Product Rule)

See if there's a value c that satisfies $2^n \times 2^1 \leq c \times 2^n$

$\frac{2^n}{2^n} \times 2^1 \leq c \times \frac{2^n}{2^n}$

$2^1 \leq c$

$2 \leq c$

c can be 2

So yes, $2^{n+1} \in O(2^n)$ because there is a value $c = 2$ that $2^n \times 2^1 \leq c \times 2^n$

Note

Big O Notation

Let f, g be two functions $f, g : N \to R^+$. We say that $f(n) \in O(g(n))$

f is big O of g if there exists a constant $c \in R^+$ & $n_0 \in N$ such that for every integer $n \geq n_0$

$f(n) \leq c \cdot g(n)$

1b) Is $2^{2^n} \in O(2^n)$? Why or why not?

Need to find value c such that

$2^{2^n} \leq c \times 2^n$

2^{2^n} is giant & grows SUPER fast compared to 2^n, 2^{2^n} will never be smaller

So, no, $2^{2^n} \notin O(2^n)$

$f(n) \leq c \cdot g(n)$ will never happen!

1c) Is $n^{1.01} \in O(\log^2(n))$?

$n^{1.01}$ $f(n) \leq c \cdot g(n)$ $\log^2(n) = (\log(n))^2$

X	Y
10	10.23
50	52
100	104.71
500	532.1

$n^{1.01}$ grows much faster than

X	Y
10	1
50	2.89
100	4
500	7.28

$\log^2(n)$ which grows really slowly

$n^{1.01}$ will not be less than $\log^2(n)$ with value c

$n^{1.01}$ dominates $\log^2(n)$

No, $n^{1.01} \notin O(\log^2(n))$

1d) Is $n^{1.01} \in \Omega(\log^2 n)$?

Big Ω definition

$f(n) \in \Omega(g(n))$

f is big omega of g if there exist $c \in \mathbb{R}^+$ & $n_0 \in \mathbb{N}$ such that for every integer $n \geq n_0$

$$f(n) \geq c \cdot g(n)$$

Tables from previous problem

$n^{1.01}$ →

X	Y
10	10.23
50	52
100	104.71
500	532.1

X	Y	$\log^2(n)$
10	1	
50	289	
100	4	
500	7.29	

$$f(n) \geq c \cdot g(n)$$

$$n^{1.01} \geq c \cdot \log^2(n)$$

$n^{1.01}$ is much faster & dominates $\log^2(n)$, so

yes, $n^{1.01} \in \Omega(\log^2 n)$

1e) Is $\sqrt{n} \in O((\log n)^3)$?

$f(n) \leq c \cdot g(n)$

$n^{0.5}$ $\quad (\log n)^3$

x	y
10	3.16
50	7.07
100	10
500	22.3

grows faster

x	y
10	1
50	4.9
100	8
500	19.67

grows slower (log function)

Obviously as n gets larger, the diff btwn the two will get bigger.

$\lim_{n \to \infty} \frac{\sqrt{n}}{(\log n)^3}$, $\sqrt{n}$ dominates!

So no, $\sqrt{n} \notin O((\log n)^3)$

1f) Is $\sqrt{n} \in \Omega((\log n)^3)$?

$f(n) \geq c \cdot g(n)$

Same this as above

$\sqrt{n}$ dominates $(\log n)^3$

So yes, $\sqrt{n} \in \Omega((\log n)^3)$

2b) What does this function do in your own words?

This function receives a value & will recursively call itself with the two previous numbers before the input value & add them together. This process continues until the base case, when the value is 1 or less. It's computing the Fibonacci sequence.

3b) What is the work & span of this function?

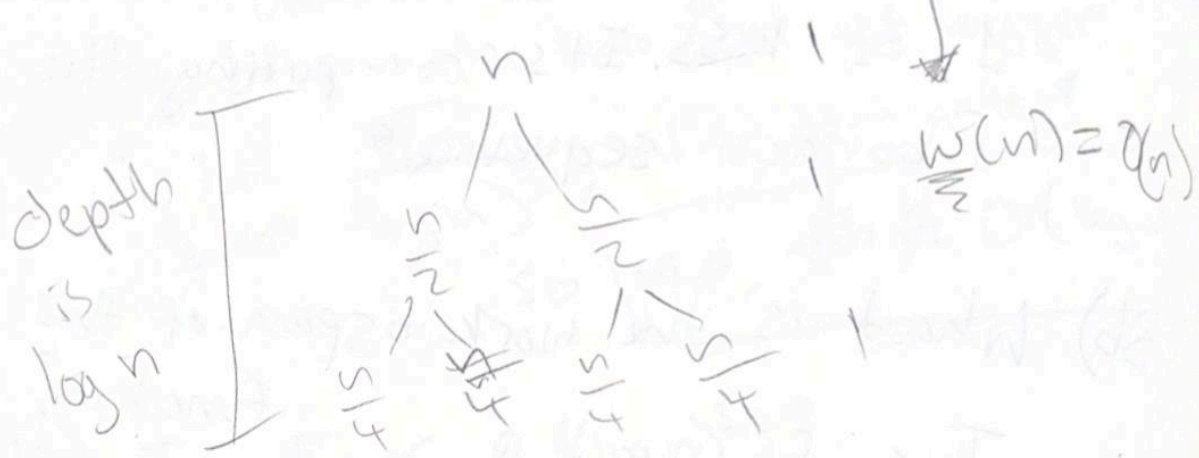
This function just checks each number in the list & just one for loop, so it's O(n) work.

Span is the longest chain of operations, & we're just checking each list item once in the for loop, so it's also O(n) for the span

O(n) = work
O(n) = span

3d) What is the work & span of this sequential algorithm?

All of this work is still constant work, just split up, so the work is still $O(n)$



The span is the depth of the recursion tree, so the span is $\log n$

3e) Assume that we parallelize in a similar way we did with sum_list_recursive. That is, each recursive call spawns a new thread. What is the work & span of this algorithm?

Still doing constant work at each leaf of tree so work is O(n)

Span depends on depth of tree (It's the longest number of operations dependent on each other) so span is also O(log n)