## CMPS 2200 Assignment 1

Alex Name:

In this assignment, you will learn more about asymptotic notation, parallelism, functional languages, and algorithmic cost models. As in the recitation, some of your answer will go here and some will go in main.py. You are welcome to edit this assignment-01.md file directly, or print and fill in by hand. If you do the latter, please scan to a file assignment-01.pdf and push to your github repository.

- 1. (2 pts ea) Asymptotic notation
- 1a. Is  $2^{n+1} \in Q(2^n)$ ? Why or why not? . Yes  $f(n) = \int_{-\infty}^{n} f(1) dx$ ?

$$f(n) = 2^{n+1}$$

$$f(n)$$

Jun  $f(n) = \frac{1}{2^n}$  and  $f(n) = \frac{1}{2^n} = \frac{1}{2^n}$ . 1b. Is  $2^{2^n} \in Q(2^n)$ ? Why or why not? 2 nt (0 ()")

g(n) = 1"

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\frac{1}{2^n}=\sum_{n\to\infty}^n=\infty$$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\frac{1}{2^n}=0$$

niol grows much faster than byth

since the limit is not a constant,

No

6×5×4×1× 2

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

$$f(n) = n^{1.01} \quad \text{(es)} \quad \log(n + \log(n)) = \log(n + \log(n)) = \log(n + \log(n))$$

• 
$$g(h) \ge \log n$$
  
• we want to find  $C$ ,  $h_0$   
•  $f(h) \ge C$   $g(h)$  for all  $n \ge h_0$   
•  $f(h) \ge C$   $g(h)$   $g(h)$   $g(h)$   $g(h)$   $g(h)$ 

As a approach infinity: hlol > c login

1.01 > 0+2.0" which : 1 alway true • 1e. Is  $\sqrt{n} \in O((\log n)^3)$  Yes. • We want to find  $\ell$ , N

$$\frac{1}{2} \int_{-\infty}^{\infty} \frac{1}{(\log n)^5} \int_{-\infty}^{\infty}$$

the limit is 0 -> Tes • 1f. Is  $\sqrt{n} \in \Omega((\log n)^3)$ ?

We want to find; 
$$c^2 \leq 0$$
,

 $ch \geq c \cdot (\log n)^3$ 
 $c \geq c \cdot (\log n)^6$ 
 $c^2 \leq \frac{n}{(\log n)^6}$ 
 $c \leq \frac{n}{(\log n)^6}$ 
 $c \leq \frac{n}{(\log n)^6}$ 

## 2. SPARC to Python

Consider the following SPARC code of the Fibonacci sequence, which is the series of numbers where each number is the sum of the two preceding numbers. For example, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 ...

```
\begin{array}{l} foo\ x=\\ \text{ if }\ x\leq 1 \ \text{ then }\\ x\\ \text{ else }\\ \text{ let }(ra,rb)=(foo\ (x-1))\ ,\ (foo\ (x-2))\ \text{ in }\\ ra+rb\\ \text{ end. } \end{array}
```

- 2a. (6 pts) Translate this to Python code fill in the def foo method in main.py
- 2b. (6 pts) What does this function do, in your own words?
  This function uses recurring to compute Fibonacci numbers by breaking channels problem into smaller subproblems and combine the result to get the Fibonacci number for the given index X.

.

## 3. Parallelism and recursion

Consider the following function:

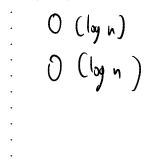
```
def longest_run(myarray, key)
    """
    Input:
        `myarray`: a list of ints
        `key`: an int
    Return:
        the longest continuous sequence of `key` in `myarray`
```

E.g.,  $longest_run([2,12,12,8,12,12,12,0,12,1], 12) == 3$ 

- 3a. (7 pts) First, implement an iterative, sequential version of longest\_run in main.py.
- 3b. (4 pts) What is the Work and Span of this implementation?

0 (n) 0 (n)

- 3c. (7 pts) Next, implement a longest\_run\_recursive, a recursive, divide and conquer implementation. This is analogous to our implementation of sum\_list\_recursive. To do so, you will need to think about how to combine partial solutions from each recursive call. Make use of the provided class Result.
- 3d. (4 pts) What is the Work and Span of this sequential algorithm?



• 3e. (4 pts) 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 and Span of this algorithm?

