CMPS 6610 Problem Set 01 Answers

Name: Lei Yang

1. **Asymptotic notation**

- 1a True, since $2^{n+1} = 2 * 2^n$,pick $c \geq 2$ and $n = 0$.We can get $2 * 2^n = 2^{n+1}$ for all $n > 0$. Hence,$2^{n+1} \in O(2^n)$

- 1b Fasle, $\lim_{n\to\infty} \frac{2^{2^n}}{2^n}$ approaches infinity ,it implies that we cannot find a constant c such that for every given c , there exists a sufficiently large n that satisfies the requirement.

- 1c False, as $\lim_{n\to\infty} \frac{n^{1.01}}{\log n^2}$ approaches infinity, it implies that we cannot find a constant c such that for every given c , there exists a sufficiently large n that satisfies the requirement.

- 1d True, as mentioned earlier, $n^{1.01}$ grows faster than $\log^2 n$. ,pick $c \geq 1$ and $n = 1$.We can get $n^{1.01} \geq log^2 n$ for all $n \geq 1$. Hence,$n^{1.01} \in \Omega(log^2 n)$

- 1e False, as $\lim_{n\to\infty} \frac{\sqrt{n}}{\log^3 n}$ approaches infinity, it implies that we cannot find a constant c such that for every given c , there exists a sufficiently large n that satisfies the requirement.

- 1f True, as mentioned earlier, $\sqrt{n}$ grows faster than $\log^3 n$. ,pick $c \geq 1$ and $n = 1$.We can get $\sqrt{n} \geq log^3 n$ for all $n \geq 1$. Hence,$\sqrt{n} \in \Omega(log^3 n)$

- 1g Let's assume that there exists some function f(n) that belongs to both o(g(n)) and omega(g(n)).

1. For $o(g(n))$: For every positive constant c_1, there exists a constant n_{01} such that $f(n) \leq c_1 * g(n)$ for all $n \geq n_{01}$.

2. For $\omega(g(n))$: For every positive constant c_2, there exists a constant n_{02} such that $f(n) \geq c_2 * g(n)$ for all $n \geq n_{02}$.

Let c_1 and c_2 be any two positive constants.

Now, let $n_{max} = max(n_{01}, n_{02})$. For n ¿= $n_max, bothconditionsmusthold$:

1. The first condition $f(n) \leq c1 * g(n)$ implies that $\frac{f(n)}{g(n)} \leq c1$.

2. The second condition $f(n) \geq c2 * g(n)$ implies that $\frac{f(n)}{g(n)} \geq c2$.

Since c_1 and c_2 are any two positive constants, this leads to a contradiction: $\frac{f(n)}{g(n)}$ cannot be both less than any c_1 and greater than any c_2.

Therefore, our initial assumption is false, and we can conclude that $o(g(n)) \sqcup \omega(g(n))$ must be the empty set.

2. **SPARC to Python**

- 2b The function 'foo' calculates the n-th Fibonacci number using recursion. For inputs 0 and 1, it returns the value itself. Otherwise, it calculates the sum of the (x-1)-th and (x-2)-th Fibonacci numbers by recursively calling itself.

3. **Parallelism and recursion**

- 3b Work: Since we explore each element of the array at least once, the work done by the algorithm is O(n), where n is the length of the array. Span: It implements in an iterative, sequential way, so its span is O(1)

- 3d Work: O(n) Span: At each level of recursion, the algorithm splits the problem into two independent subproblems, so the depth of the recursion tree would be log n . Thus, the span of the algorithm is O(log n).

- 3e Work:In our recursive algorithm for finding the longest run, we divide the array into two halves in each recursive call until we reach arrays of size 1. Since we explore each element of the array at least once, the work done by the algorithm is O(n), where n is the length of the array. Span: At each level of recursion, the algorithm splits the problem into two independent subproblems, so the depth of the recursion tree would be log n .And current result relies in its sub-result. Thus, the span of the algorithm is O(log n).