Anh Pham

1.

a) Prove: $2^{n+1} \le c \cdot 2^n$ for $c > 0$ and $n \ge n_0$

$2^n \cdot 2 \le c \cdot 2^n$

$2 \le c$

Let $c = 2$: $2^{n+1} \le 2 \cdot 2^n$ ✓

since there exist a constant c such that $2^{n+1} \le c \cdot 2^n$ for all n,

$\boxed{\text{yes, } 2^{n+1} \in O(2^n)}$

b) Prove: consider ratio $\frac{2^{2^n}}{2^n} = 2^{2^n - n}$

as $n \to \infty$, $2^n - n \to \infty$

hence, $2^{2n-n} \to \infty$ as $n \to \infty$.

this implies for any constant c, there exist an n_0 such that: $2^{2^n} > c \cdot 2^n$ for all $n \ge n_0$

since 2^{2^n} grows much faster than 2^n,

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

c) $\lim_{n \to \infty} \frac{n^{1.01}}{\log^2 n} = \infty$ since polynomial functions grow significantly faster than logarithmic functions.

$\Rightarrow$ $n^{1.01}$ eventually dominates $\log^2 n$ as $n \to \infty$

since $n^{1.01}$ grows much faster than $\log^2 n$,

$\boxed{\text{no } n^{1.01} \notin O(\log^2 n)}$

d) $\boxed{\text{yes } n^{1.01} \in \Omega(\log^2 n)}$ because $\lim_{n \to \infty} \frac{n^{1.01}}{\log^2 n} = \infty$

e) consider whether $\sqrt{n} \le (\log n)^3$

$n^6 \le (\log n)^6$

$\lim_{n \to \infty} \frac{n}{\log n^6} = \infty$

hence $n \not\le (\log n)^6$

since $\sqrt{n}$ grows much faster than $(\log n)^3$,

$\boxed{\text{no } \sqrt{n} \notin O((\log n)^3)}$

f) $\boxed{\text{yes, } \sqrt{n} \in \Omega((\log n)^3)}$ because $\lim_{n \to \infty} \frac{\sqrt{n}}{(\log n)^3} = \infty$

this is because logarithmic functions is always bounded by a horizontal assymtote, hence polynomial functions will dominate as $n \to \infty$

2. b) this function returns the value of an inputed index within the Fibonacci sequence. It uses recursive calls until it reaches the base case, then starts computing the value as it exits out of the levels.

3. b) comparison mylist[n]==k : $O(1)$

update current_num+=1 : $O(1)$

update max_run : $O(1)$

for loop of n elements : $O(n)$

$\Rightarrow W(n) = O(1) + O(1) + O(1) + O(n)$

$= \boxed{O(n)}$

Each step requires information from it's previous step, Hence, there is no parallelism.

$\Rightarrow S(n) = \boxed{O(n)}$

d) $W(n) = 2W(n/2) + O(1)$

splitting in left and right

constant work of computing left/right / longest spanning size and merging

$= 2(2W(n/4) + O(1)) + O(1)$

$= 4W(n/4) + 2O(1) + O(1)$

expand until reach $W(n/n) = W(1)$

We perform $O(n)$ work each level with $O(\log n)$ levels with $O(1)$ extra work

Hence $W(n) = O(n)$

$$S(n) = S(\tfrac{n}{2}) + O(1)$$

expand: $S(n) = S(\frac{n}{2}) + O(1)$
$= S(\frac{n}{4}) + O(1) + O(1)$
$= S(\frac{n}{8}) + O(1) + O(1) + O(1)$

recursion halved hence span is depth of tree

$$S(n) = O(\log n)$$

e) Both longest_run_recursive and sumList for divide & conquer

1. Divide : both split list in halves

hence $W(n) = 2(W(\frac{n}{2})) + O(1)$

2. Conquer : both computes the sum / longest run in both halves in parallel.

3. combine : sumList adds results and longest_run_recursive merges results which are $O(1)$ extra work.

Hence both have

$$W(n) = n$$
$$S(n) = \log n$$