CMPS 2200 Assignment 1

Name: Chenyu Zhao

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**

$g(n) \leq c \cdot f(n)$ for $n \geq n_0$

- 1a. Is $2^{n+1} \in O(2^n)$? Why or why not?
  - true based on the definition of asymptotic dominance where $g(n) = 2^{n+1}$ and $f(n) = 2^n$, the inequality was true when $n_0 = 1$ and $c = 3$.

$n_0 = 1$, $c = 3$

$2^{n+1} \leq c \cdot 2^{n_0}$

$2^{1+1} \leq 3 \cdot 2^1$

$4 \leq 6$ ✓

- 1b. Is $2^{2^n} \in O(2^n)$? Why or why not?
  - False based on the proof on the right because $x - \log(x)$ is a increasing function where it will pass "c" as x approaches ∞.

proof: let $x = 2^n$

contradiction proof

$2^x \leq c \cdot x$

$\log(2^x) \leq \log(c \cdot x)$

$x \leq \log(c) + \log(x)$

$x - \log(x) \leq \log(c)$

$\log n$

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

$f(n) = n^{1.01}$

$g(n) = \log^2 n = (\log n)^2$

$\ln(10) \approx 2.30$

$1/(\ln(10)) \approx 0.43$

$\frac{d}{dx} \log_b n = \frac{1}{\ln b} \cdot \frac{1}{x}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{1.01}}{(\log n)^2} = \lim_{n\to\infty} \frac{1.01\, n^{0.01}}{2(\log n)(\ln(10)^{-1} \cdot n^{-1})} = \lim_{n\to\infty} \frac{1.01\, n^{1.01}}{2(\log n)(\ln(10))^{-1}}$$

$$= \lim_{n\to\infty} \frac{1.0201\, n^{0.01}}{2(\ln(10)^{-1} \cdot n^{-1})(\ln(10))^{-1}} = \lim_{n\to\infty} \frac{1.0201\, n^{1.01}}{2(\ln(10))^{-2}} = \boxed{\infty}$$

Based on the limit method, $n^{1.01} \in O(\log^2 n)$ is false since the $\lim_{n\to\infty} \frac{n^{1.01}}{(\log n)^2} = \infty$ instead of 0 to make it true.

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

same work as above.

Based on limit method, $n^{1.01} \in \Omega(\log^2 n)$ is true since the $\lim_{n\to\infty} \frac{n^{1.01}}{(\log n)^2} = \infty$.

- 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?

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

$f(n) = \sqrt{n} = n^{1/2}$

$g(n) = (\log n)^3$

$$\lim_{n\to\infty} \frac{n^{1/2}}{(\log n)^3} = \lim_{n\to\infty} \frac{\frac{1}{2}n^{-1/2}}{3(\log n)^2(\ln(10)^{-1}\cdot n^{-1})} = \lim_{n\to\infty} \frac{\frac{1}{2}n^{1/2}}{3(\log n)^2(\ln(10))^{-1}} =$$

$$\lim_{n\to\infty} \frac{\frac{1}{4}n^{-1/2}}{6\log n(\ln(10)^{-1}\cdot n^{-1})(\ln(10)^{-1})} = \lim_{n\to\infty} \frac{\frac{1}{4}n^{1/2}}{6(\ln(10)^{-2})\log n} = \lim_{n\to\infty} \frac{\frac{1}{4}\cdot\frac{1}{2}n^{-1/2}}{6(\ln(10)^{-3})n^{-1}}$$

$$= \lim_{n\to\infty} \frac{\frac{1}{8}n^{1/2}}{6(\ln(10)^{-3})} = \lim_{n\to\infty} \frac{n^{1/2}}{48(\ln(10)^{-3})} = \boxed{\infty}$$

Based on limit method $\sqrt{2} \in O((\log n)^3)$ is false because $\lim_{n\to\infty} \frac{n^{1/2}}{(\log n)^3} = \infty$ instead of $=0$ to make it true.

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

True based on limit method since $\lim_{n\to\infty} \frac{n^{1/2}}{(\log n)^3} = \infty$.

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 ...

```
foo x =
  if x ≤ 1 then
    x
  else
    let (ra, rb) = (foo (x − 1)) , (foo (x − 2)) in
      ra + rb
    end.
```

- 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 is for the Fibonacci sequence. For instance, foo(2) passes through the "if" statement but because 2 > 1, it does not return X. Foo(2) then goes into the "else" statement where ra, rb = (foo(2-1), foo(2-2)) = (foo(1), foo(0)). Foo(1) and foo(0) both then pass through "if" statement since they are less than or equal to 1 which means they return 0. Finally ra + rb is returned where 0 + 1 = 1 for foo(2).

3. Parallelism and recursion

Consider the following function:

```
def longest_run(myarray, key)
```