Intro to Computer Science, HW 2

Guy Taggar - 206260762

Q1.

We have 3 fixed operations - defining both "digits" and "result" and returning "result". As for the while loop, it will keep looping exactly len(i) times until $i = 0$. Every loop does 2 assignments (result, i), 3 operations ($+, \%, //$) and 1 item accessing (digits[item]). That being said, letting $n = \text{len}(i)$ yields the required complexity of:

$$6n + 3$$

Q2

Let $n = \text{len}(\text{lst})$. We are calling the function recursively n times (until $n = 0$). Every function calling does a "for" loop on a list of length $n-1$, namely "p", with 2 operations per loop (adding and accessing a list item). Each function calling also performs 2 fixed operations (Assigning "p" and adding the loop result).

Also, the last calling will return the base case (1 operation). Let us now analyze and prove the complexity:

$$t_n = t_{n-1} + 2 + 2(n-1) = t_{n-1} + 2n$$

Hence:

$$t_n = t_{n-1} + 2n = t_{n-2} + 2(n-1) + 2n =$$
$$t_{n-3} + 2(n-2) + 2(n-1) + 2n =$$
$$t_{n-k} + 2(n-k+1) + 2(n-k+2) + \ldots + 2n =$$
$$t_{n-k} + 2[(n-k+1) + (n-k+2) + (n-k+3) + \ldots + n]$$

This will run until t_0 iff $k = n$. thus:

$$t_n = 2\left[1+2+3+4+5+\ldots n\right]+1 = 1+2\sum_{i=1}^{n} i$$

We conclude:

$$t_n = 1+2\sum_{i=1}^{n} i = 2\cdot\frac{n\left(n+1\right)}{2}+1 = n^2+n+1 = O\left(n^2\right)$$

■

Q4.

1. $f\left(n\right)=5n^2+1,\ \ g\left(n\right)=10n^2+30.$

   It's easy to see that $\forall n\in\mathbb{N}$ we have $f\left(n\right)\leq g\left(n\right)$, hence $f=O\left(g\right)$, while we also have:

   $$\underbrace{g\left(n\right)}=10n^2+30\underset{n\in\mathbb{N}}{\leq}40n^2\underbrace{\leq}40n^2+8=8\left(5n^2+1\right)=\underbrace{8f\left(n\right)}$$

   for every $n\in\mathbb{N}$, so $g=O\left(f\right)$ as well.

2. $f\left(n\right)=\log_3 n+1,\ \ g\left(n\right)=\log^3 n.$

   $$\lim_{n\to\infty}\frac{\log_3 n+1}{\log^3 n}=\lim_{x\to\infty}\frac{\log_3 x+1}{\log^3 x}\underset{\frac{\infty}{\infty}}{\overset{L}{=}}\lim_{x\to\infty}\frac{\frac{1}{x\ln 3}}{\frac{3\log^2 x}{x\ln 10}}=\lim_{x\to\infty}\frac{\ln 10}{\ln 3\cdot 3\log^2 x}=0$$

   Where we used both Lhopital's rule and Heine's lemma. By the definition of the limit, there exists some $n_0\in N$ such that for every $n\geq n_0$ we have:

   $$\left|\frac{\log_3 n+1}{\log^3 n}-0\right|<1\Rightarrow$$

   $$\frac{\log_3 n+1}{\log^3 n}<1\Rightarrow\log_3 n+1<1\cdot\log^3 n\Rightarrow$$

   $$\log_3 n+1=O\left(\log^3 n\right)$$

3. $f\left(n\right)=8^{\log n},\ \ g\left(n\right)=3n^3+2.$

   Assuming $\log n=\log_{10} n$, we have $f\left(n\right)=8^{\log n}\leq 10^{\log n}=n$ for every $n\in\mathbb{N}$. So clearly, $f=O\left(g\right)$ and not vice versa.

4. $f(n) = n^n, \quad g(n) = 2^n$

   As:

$$\lim_{n\to\infty} \frac{n^n}{2^n} = \lim_{n\to\infty} \left(\frac{n}{2}\right)^n = \infty$$

   By the definition of the limit, there exists some $n_0 \in N$ such that for every $n \geq n_0$ we have:

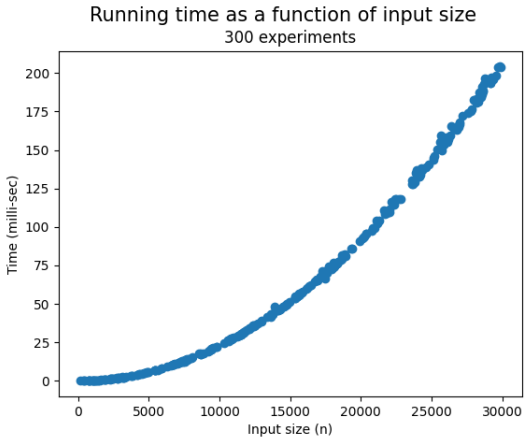
$$\frac{n^n}{2^n} > 1 \Rightarrow n^n > 1 \cdot 2^n \Rightarrow 2^n = O(n^n)$$

   By the way, both here and in item (2) we actually proved that $f = o(g)$ (small o notation).

Q5.

Having a nested "while" loop inside a for loop with constant operations in both loops, we can say the complexity of my "insertion sort" algorithm is $O(n^2)$. We avoid proving it mathematically as done at Q2, as we rather use an actual running time experiment instead to test our assumption (say, "physicists proof").

While the experiment code can be found at "Q5.py" file, consider the following, 300 sized, experiment of random sized lists, filled by randomized integers ranging from 1 to 1000, as a function of time taken to sort them using my "insertion sort" algorithm (BTW, that took a total of about 30 minutes of running time):



We can see the "squared" behavior just from looking at the graph. It's also worth noticing the squared (x, y) relations, i.e when x triple itself from $10k$ to $30k$, the y value gets about 3^2 times larger.

Moreover, consider the following function:

$$f(x) = 0.00000022x^2$$

or:

$$f(x) = 2.2 \times 10^{-7}x^2$$

Now let us create the following (x, y) data-table of f (Credit - Desmos):

$$f(x) = 0.00000022x^2$$

x	$f(x)$
5000	5.5
10000	22
15000	49.5
20000	88
25000	137.5
30000	198

Clearly, that function well describes our experiment graph. We conclude the average running time of "insertion sort" is indeed $O\left(n^2\right)$.