Solutions to 1st Mid-Semester Exam

Note that the explanations given here are more detailed than required in the answers given by the examinees!

Question 1 (6 Marks)

Write the following functions writing in *increasing* order of asymptotic growth.

- $\lfloor\sqrt{n}\rfloor$ (Integer part of square root of n.)
- $2^n/10^{10}$
- $(n-100)/2$
- $\log(n+6)$
- $\sin(n)$
- n^2-n-1

Solution. The order is

$$\sin(n), \log(n+6), \lfloor\sqrt{(n)}\rfloor, (n-100)/2, n^2-n-1, 2^n/10^{10}$$

Some justifications:

- $\sin(n)$ is $\Theta(1)$.
- $\log(n+6) \sim \log(n)$.
- $\lfloor\sqrt{(n)}\rfloor \sim \sqrt{n}$.
- $(n-100)/2$ is $\Theta(n)$.
- n^2-n-1 is $\Theta(n^2)$.
- $2^n/10^{10}$ is $\Theta(2^n)$.

Question 2 (7 Marks)

Given the function $f(n)$ defined by the following recursion:

- $f(n) = n$ for $n \leq 2$.
- $f(n) = 5 \cdot f(n-3) + 2 \cdot f(n-2) + f(n-1)$ for $n \geq 3$

Compare $f(n)$ using o or O (as appropriate), to the following functions

a. $g_0(n) = n^3$.
b. $g_1(n) = 2^n$.
c. $g_2(n) = 3^n$.

Solution. We note by induction on n:

1. $f(n) \geq 0$ for all n and $f(n) > 0$ for all $n > 1$.
2. This implies that $f(n) > f(n-1)$ for all n. So $f(n)$ is increasing.

3. This implies that

$$\begin{aligned} f(n) &= 5 \cdot f(n-3) + 2 \cdot f(n-2) + f(n-1) \\ &> (5+2+1) f(n-3) = 8 f(n-3) = 2^3 f(n-3) \end{aligned}$$

By induction on n, this means that for $n > 0$, we have:

$$\begin{aligned} f(3n+1) &> 2^{3n+1} f(1)/2 \\ f(3n+2) &> 2^{3n+2} f(2)/4 \\ f(3n+3) &> 2^{3n+3} f(3)/8 \end{aligned}$$

Since $f(1), f(2), f(3)$ are positive, it follows that 2^n is $O(f(n))$.

Since n^3 is $o(2^n)$, we see that n^3 is $o(f(n))$.

We note that $f(0) < 3^0$, $f(1) < 3^1$ and $f(2) < 3^2$. By induction on n:

$$f(n) < 5 \cdot 3^{n-3} + 2 \cdot 3^{n-2} + 3^{n-1} = (5 + 2 \cdot 3 + 3^2) 3^{n-3} = 20 \cdot 3^{n-3} < 3^n$$

It follows that $f(n)$ is $O(3^n)$.

With a little sharper analysis we can show that 2^n is $o(f(n))$ and $f(n)$ is $o(3^n)$.

Question 3 (7 Marks)

Given an algorithm `add` that takes two multi-digit numbers $\underline{a}$ and $\underline{b}$ and returns the sum $\underline{c} = \underline{a} + \underline{b}$. Also given that it uses $p + q$ basic operations where $\underline{a}$ has length p and $\underline{b}$ has length q.

Count the number of operations of the following algorithm `addlist` which adds together a list $L = (\underline{a_1}, \ldots, \underline{a_k})$ of multi-digit numbers *assuming* that:

- All the multi-digit numbers $\underline{a_i}$ have the same length p.
- The only operations to be counted occur during the computation of `add`.
- We want the number of operations in terms of k and p.
- We want to look at the worst case (the maximum number of operations).

define `addlist`(L):
  if *length of* L *is* 1:
    return $\underline{a_1}$, *the first (and only!) element of* L
  else:
    Put L' *as the list obtained from* L *by removing* $\underline{a_1}$.
    return `add`$(\underline{a_1}$, `addlist`$(L'))$

Solution. Let $T(k, p)$ denote the number of invocations of `add` when the input consists of a list of length k of multi-digit numbers of size p.

We note that $T(1, p) = 0$ since the algorithm returns the single element of the list without the need to invoke `add`.

Next, we note that for $k \geq 2$, we have $T(k,p) = T(k-1,p) + (p + U(k-1,p))$ where $U(k-1,p)$ is the length of the output of the algorithm on the input of $k-1$ multi-digit numbers of size p.

Now, each addition can result in one extra digit. So $U(k,p) \leq 1 + U(k-1,p)$ for $k \geq 2$ and $U(1,p) = p$.

By induction on k, we deduce that $U(k,p) \leq (k-1) + p$.

Since we are looking at the *worst-case*, we use $T(k,p) = T(k-1,p)+p+((k-2)+p)$ for $k \geq 2$.

By induction on k, this gives us $T(k,p) = 2(k-1)p + (k-1)(k-2)/2$.

A finer analysis will note that the extra-digit is not obtained for *all* k.

We note that when k is $o(p)$ (small lists of large numbers), then the growth in complexity is $O(p)$ (linear), whereas when p is $O(k)$ (large lists of numbers), the growth is $O(k^2)$ (quadratic).