Honors Data Structures
Theoretical Assignment 1

Mark Kirichev, UNI: mmk2243

Problem 1:

For the function $f(n) = 2n^2 - 3n + 2$, we know that it's polynomial and that the highest power of n is 2. Therefore, we'd expect there to be some polynomial functions with highest power 2 that could be used as an upper and a lower bound around $f(n)$. Since the coefficient ~~of~~ in front of n^2 is 2 we can try to set the lower boundary function ~~to~~ to n^2 and the upper boundary function to ~~3n~~ $3n^2$. This is plausible because we know that kn^2 grows with the same rate for any $k = const$; $k \in \mathbb{R}^+$. Thus, we'd expect ~~the sam~~ for higher values of k the same growth rate to be scaled higher and, analogously, for ~~k~~ smaller k's -> lower. So, here we have $g(n) = n^2$, according to the definition of Big-Θ. Now, we need to find c_1, c_2, and n_0 that satisfy: $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for $\forall n \ge n_0$

=> For $g(n) = n^2$: $0 \le c_1 n^2 \le f(n) \le c_2 n^2$ for $\forall n \ge n_0$

If we take ~~into~~ into account the above logic and set:

$c_1 = 1$; $C_2 = 3$, then if we manage to find a const. n_0, we'll have our desired result.

$$(1)\, n^2 \le f(n) \le (3)\, n^2$$

$$\Rightarrow n^2 \le 2n^2 - 3n + 2 \le 3n^2 \iff \begin{cases} n^2 \le 2n^2 - 3n + 2 \\ 2n^2 - 3n + 2 \le 3n^2 \end{cases}$$

$$\Leftrightarrow \begin{cases} 0 \leq n^2 - 3n + 2 \\ -3n + 2 \leq n^2 \end{cases} \Leftrightarrow \begin{cases} 0 \leq n^2 - 3n + 2 \\ 0 \leq n^2 + 3n - 2 \end{cases}$$

From the last set of quadratic inequalities, it isn't hard to see that if we choose some random n_0 which is relatively large (like 100, for example), we'll satisfy the condition. However, if we try to get the minimal possible n_0, in order to find where exactly does $f(n)$ get in between $c_1 g(n)$ and $c_2 g(n)$:

1) $n^2 - 3n + 2 \geq 0 \Rightarrow n^2 - 2n - n + 2 \geq 0 \Rightarrow n(n-2) - (n-2) \geq 0$

$\Rightarrow (n-1)(n-2) \geq 0 \Rightarrow n \in (-\infty; -1] \cup [2; +\infty)$

2) $n^2 + 3n - 2 \geq 0 \Rightarrow \left(n^2 + 3n + \frac{9}{4}\right) - \frac{9}{4} - 2 \geq 0$

$\Rightarrow \left(n + \frac{3}{2}\right)^2 - \frac{17}{4} \geq 0 \Rightarrow \left(n + \frac{3}{2}\right)^2 - \left(\frac{\sqrt{17}}{2}\right)^2 \geq 0$

$\Rightarrow \left(n + \frac{3}{2} - \frac{\sqrt{17}}{2}\right)\left(n + \frac{3}{2} + \frac{\sqrt{17}}{2}\right) \geq 0 \Rightarrow n \in \left(-\infty; \frac{-\sqrt{17}}{2} - \frac{3}{2}\right] \cup \left[\frac{\sqrt{17}}{2} - \frac{3}{2}; +\infty\right)$

$$\frac{\sqrt{17} - 3}{2} < \frac{\sqrt{25} - 3}{2} = \frac{5 - 3}{2} = \frac{2}{2} = 1 < 2$$

$\Rightarrow$ Choosing $n_0 = 2$ would be the least possible n_0 that would satisfy the condition. However, since we found a set of working (c_1, c_2, n) constants, then we proved that, indeed, our $g(n) = n^2 \Rightarrow \boxed{\Theta(g(n)) = \Theta(n^2)}$

Problem 2.

$$p_d(n) = \sum_{i=0}^{d} a_i n^i$$

$$\sum_{i=0}^{d} a_i n^i = a_0 n^0 + a_1 n^1 + a_2 n^2 + \dots + a_d n^d =$$

$$= a_0 + a_1 n + a_2 n^2 + \dots + a_d n^d$$

We need to prove that the above expression is

$$p_d(n) = O(n^k)$$

for $k \in \mathbb{N}$ and $k \geq d$

The above would be true if we find a value for c_1 and n_0 where $c_1 n^k = c_1 g(n) \geq p_d(n)$

for $\forall n \geq n_0$,

where c_1 and n_0 are const.

Now, since we know that $a_d > 0$ and $a_i \geq 0$ for $i \in \mathbb{N}_0$ and $i \in [0; d-1]$, we can transform the above inequality: $\frac{c_1 n^k}{p_d(n)} \geq 1$

If we "unfold" $p_d(n)$: $\frac{c_1 n^k}{a_0 + a_1 n + a_2 n^2 + \dots + a_d n^d} \geq 1$

Since we know that $k \geq d$, we can get a factor of n^d from both the nominator and denominator: $\frac{c_1 n^d \times n^{k-d}}{n^d \left(\frac{a_0}{n^d} + \frac{a_1}{n^{d-1}} + \dots + \frac{a_{d-1}}{n} + a_d \right)}$

Now, to prove that the numerator function dominates the denominator one, we can take the limit as n approaches infinity:

$$\lim_{n \to \infty} \frac{c_1 \cancel{n^d} \times n^{k-d}}{\cancel{n^d}\left(\frac{a_0}{n^d} + \frac{a_1}{n^{d-1}} + \dots + \frac{a_{d-1}}{n} + a_d\right)} =$$

$$= c_1 \lim_{n \to \infty} \frac{n^{k-d}}{\frac{a_0}{n^d} + \frac{a_1}{n^{d-1}} + \dots + a_d} =$$

$$= c_1 \frac{\lim_{n \to \infty}\left(n^{k-d}\right)}{\lim_{n \to \infty}\left(\frac{a_0}{n^d}\right) + \lim_{n \to \infty}\left(\frac{a_1}{n^{d-1}}\right) + \lim_{n \to \infty}\left(\frac{a_2}{n^{d-2}}\right) + \dots + \lim_{n \to \infty}\left(a_d\right)} =$$

$$= c_1 \frac{\lim_{n \to \infty}\left(n^{k-d}\right)}{0 + 0 + 0 + \dots + 0 + a_d} = \frac{c_1}{a_d} \lim_{n \to \infty}\left(n^{k-d}\right)$$

I case: $k > d$: $\lim_{n \to \infty} \frac{c_1 n^k}{p_d(n)} = +\infty$

=> The above function always dominates the denominator one, i.e. every c_1 works! => Since every c_1 works, then there'll definitely be some arbitrarily large n_0 value for which the condition is satisfied since the difference between the numerator & denominator will eventually $\to +\infty$

II case: $k = d$: $\frac{c_1}{a_d} \lim_{n \to \infty}\left(n^{k-d}\right) = \frac{c_1}{a_d} \lim_{n \to \infty}\left(n^{d-d}\right) =$

$$= \frac{c_1}{a_d} \lim_{n \to \infty} n^0 = \frac{c_1}{a_d} \lim_{n \to \infty} 1 = \boxed{\frac{c_1}{a_d}}$$

↳ Note: Here we don't reach ∞^0 because we firstly need to evaluate the function and then take the limit and not simultaneously.

Now, we want to find a value for c_1, where the limit satisfies the condition $\frac{c_1 n^k}{p_d(n)} \geq 1$

Thus, if we take n_0 arbitrarily large, i.e. assume that $n_0 \to \infty$, then any $c_1 \geq 1$ will work and the condition will be satisfied. If we, insted, fix a value for n_0:

$$\frac{c_1 n_0^d}{a_0 + a_1 n_0 + a_1 n_0^2 + \dots + a_d n_0^d} = \frac{\cancel{n_0^d} \times c_1}{\cancel{n_0^d}\left(\frac{a_0}{n_0^d} + \frac{a_1}{n_0^{d-1}} + \dots + a_d\right)} =$$

$$= \frac{c_1}{\frac{a_0}{n_0^d} + \frac{a_1}{n_0^{d-1}} + \dots + a_d}$$; We want the last expression to be ≥ 1, which is

equivalent to: $\boxed{c_1 \geq \frac{a_0}{n_0^d} + \frac{a_1}{n_0^{d-1}} + \frac{a_2}{n_0^{d-2}} + \dots + a_d}$,

which is obviously correct for some arbitrarily large value, because on the right side of the inequality we just have constants and the sum of several constant terms is always going to be a constant. Therefore, there will ~~exist~~ always exist a bigger constant because the set of real / natural numbers is infinite!

$\Rightarrow$ In case 2, also $\boxed{p_d(n) = O(n^k)}$

Problem 3

Firstly, we can recognize that 256 is a constant, and, thus, does not grow for any $n \in \mathbb{R}$. For the function $(2/N)$ we can see that as N progresses to increase, the output gets smaller and, more precisely, $\lim_{N \to \infty} (2/N) = 0$. For any other function, besides those two, we have $\lim_{N \to \infty} f(N) = +\infty$

In the next step, we'll prove that $\log N$ has a slower growth rate than $\sqrt{N}$

If we take the lim: $\lim_{N \to \infty} \frac{\log N}{N^{1/2}} \overset{L'Hopital}{=} \lim_{N \to \infty} \frac{1/N}{\frac{1}{2} N^{-1/2}} =$

$$= \lim_{N \to \infty} \frac{1/N}{1/2\sqrt{N}} = \lim_{N \to \infty} \frac{2\sqrt{N}}{N} = \lim_{N \to \infty} \frac{2}{\sqrt{N}} = 0$$

$\Rightarrow$ $\boxed{\log N \text{ gets dominated by } N^{1/2}}$

Next up, it's trivial to see that $\boxed{5N}$ grows faster than $\sqrt{N}$: $\lim_{N \to \infty} \frac{5N}{\sqrt{N}} = \lim_{N \to \infty} (5\sqrt{N}) = +\infty$

After that, we can argue that $N \log N$ grows faster than $5N$: $\lim_{N \to \infty} \frac{\cancel{N} \log N}{5\cancel{N}} = \lim_{N \to \infty} \frac{\log N}{5} = +\infty$

Earlier, we saw that $\log N$ grows slower than $N^{1/2}$ and it's trivial that $N^{1/2}$ grows slower than N.

$\Rightarrow$ $\log N$ grows slower than N

$\Rightarrow$ $\boxed{N \log N \text{ grows slower than } N^2}$

The last polynomial function in the set of given functions is $18N^3$. Since it's highest power is 3, and N^2's is 2, it's obvious that $18N^3$ grows slower than N^2

Next, we have the exponential functions and the factorial. First, we'll argue that the factorial one is the slowest. Let's have an arbitrary constant $k \in \mathbb{N}$.

Now, if we examine the function k^n and try to compare it to $n!$, we can observe that however big k is, at each "step" k^n gets multiplied by k and $n!$ gets multiplied by a bigger number than it was multiplied before ~~that~~. Thus, there will come a moment when $n!$ will surpass k^n and it will continue growing in a faster rate ~~to~~ than k^n since at this point n will have become bigger than k.

To give a more formal proof to the above:

If we assume that $\frac{n!}{k^n} \to \infty$ for $n \to \infty$,

then we know that $\log\left(\frac{n!}{k^n}\right) \to \infty$ because $\log(\infty) = \infty$

Now $\log\left(\frac{n!}{k^n}\right) = \log(n!) - \log(k^n) =$

$= \log\left(\prod_{i=1}^{n} i\right) - n \log k = \left[\log 1 + \log 2 + \log 3 + \dots + \log n\right] - n \log k =$

$= \left(\sum_{i=1}^{n} \log i\right) - n \log k$. Now, since we know that n gets large, we can assume that at some point it will surpass k

Now, as we continue to increase n, we only add another log k to the part that we subtract but add log(n+1) to the sum. Thus, the sum will, at some point, become arbitrarily large as n becomes large.

And, even more rigorously: We want to prove that

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} \log i}{n \log k} = \infty$$

; We clearly know that:

$$\sum_{i=1}^{n} \log i \geq \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n} \log i$$

Now, for all $i \in [\lfloor \frac{n}{2} \rfloor ; n]$, we know that:

$$\log(i) \geq \log\left(\frac{n}{2}\right) = \log(n) - \log(2) = \log(n) - 1$$

$$\Rightarrow \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n} \log i \geq \frac{n}{2} \log(n) - \frac{n}{2} \Rightarrow \sum_{i=1}^{n} \log i > \frac{n}{2} \log(n) - \frac{n}{2}$$

Now: $$\lim_{n\to\infty} \left(\frac{\frac{n}{2} \log(n) - \frac{n}{2}}{n \log k} \right) = \lim_{n\to\infty} \left(\frac{\frac{1}{2} \log(n) - \frac{1}{2}}{\log k} \right) =$$

$$= \lim_{n\to\infty} \left(\frac{\log(n)}{2 \log k} - \frac{1}{2 \log k} \right) = +\infty$$

$$\Rightarrow \boxed{\lim_{n\to\infty} \left(\frac{\sum_{i=1}^{n} \log i}{n \log k} \right) = +\infty}$$

$\Rightarrow$ We proved that $n!$ is the slowest growing function.

Now, we just need to order the exponential functions. We can see that $2^{n+1} = 2 \times 2^n$ and since 2 is just a constant, we can conclude that the growth rate of 2^n and 2^{n+1} is the same

Now, comparing 2^n and 4^n : $4^n = (2^2)^n = 2^{2n} =$

$= 2^n \times 2^n$

If we assume that 2^n and 4^n have the same growth ~~rate~~ rate, then there should exist a constant $c \in \mathbb{R}^+$ such that $4^n \leq c2^n$ for all $n \geq n_0$ for some const. n_0

However: $4^n \leq c\, 2^n$

$2^n \times 2^n \leq c\, 2^n \;/ : 2^n$

$2^n \leq c$ -> this is obviously a contradiction, because c cannot be a constant to ~~satisfy~~ satisfy the inequality for all n.

=> $\boxed{4^n \text{ grows slower than } 2^n}$

Lastly, $\left(\frac{2}{N}\right) = 2N^{-1}$ has a slower "growth" rate than $256 = 256N^0$ => Finally:

$\left(\frac{2}{N}\right) < 256 < \log N < \sqrt{N} < 5N < N\log N <$

$< N^2 < 18N^3 < 2^N = 2^{N+1} < 4^n < n!$

Problem 4

a) We know that $T(1) = 2$; $T(2) = 4$; $T(3) = 16$...

$\Rightarrow$ We can encounter the following recursive sequence:

$T(n) = [T(n-1)]^2$; This is because we know that "each subsequent year, the tribute was squared"

$\Rightarrow$ If we "unfold" the last expression:

$$T(n) = [T(n-1)]^2 = \left[[T(n-2)]^2\right]^2 = [T(n-2)]^4$$

$$T(n-2) = [T(n-3)]^2 \Rightarrow T(n) = \left[[T(n-3)]^2\right]^4 =$$

$$= [T(n-3)]^8$$

$\Rightarrow$ We can prove a stronger relation through induction:

$$T(n) = [T(n-k)]^{2^k}$$

Base case: $k = 0$: $T(n) = T(n-0)^{2^0} = T(n)^1$ ✓

$k = 1$: $T(n-1) = [T(n-1)]^{2^1} = [T(n-1)]^2$ ✓

Induction step: Let's assume that the induction claim is true for a fixed number k, where $k \in \mathbb{N}_0$ and $k \in [0; n-2]$

$\Rightarrow$ We need to prove that the induction claim is true for $(k+1)$.

We have: $T(n) = [T(n-k)]^{2^k}$

From the problem we know that $T(n-k) =$

$= [T(n-(k+1))]^2$

$$\Rightarrow T(n) = [T(n-k)]^{2^k} = \left[[T(n-(k+1))]^2\right]^{2^k} =$$

$$= [T(n-(k+1))]^{2 \times 2^k} =$$

$$= \left[T(n-(k+1))\right]^{2^{k+1}}$$

$$\Rightarrow \boxed{T(n) = \left[T(n-(k+1))\right]^{2^{k+1}}}$$

$\Rightarrow$ We proved the induction!

$\Rightarrow$ We know that the following is true:

$$T(n) = \left[T(n-k)\right]^{2^k}$$

for $k \in [0; n-1]$, $k \in \mathbb{N}_0$

$\Rightarrow$ If we let $k = n-1$:

$$T(n) = \left[T(n-(n-1))\right]^{2^{n-1}} =$$

$$= \left[T(n-n+1)\right]^{2^{n-1}} =$$

$$= \left[T(1)\right]^{2^{n-1}}$$

However, we know that $T(1) = 2$

$\Rightarrow \boxed{T(n) = 2^{2^{n-1}}}$ $\Rightarrow$ In year N the tribute is: $\boxed{2^{2^{N-1}}}$

b) If we let $D = 2^{2^{N-1}}$ for some $N \in \mathbb{N}_0$

$$\Rightarrow \log_2 D = 2^{N-1}$$

$$\Rightarrow \log_2(\log_2 D) = N-1$$

$$\Rightarrow \boxed{N = \log_2(\log_2 D) + 1}$$

Note: $\log_2$ operation is ledgit since we know that D and $2^{2^{N-1}}$ are both ≥ 1 and $\log_2 1 = 0$

i.e. $\log_2 2^{N-1} > \log_2 1 > 0$ and we can take logs on expressions > 0

Problem 5.

a) We see that the outer loop will run for n iterations. The inner loop will run for:

- 23 times the first time
- 22 times the second time
- 21 times the third time

⋮

- 2 times the 22nd time
- 1 time the 23rd time
- 0 times the 24th time & onwards.

$\Rightarrow$ If $n \geq 23$, the whole code piece will run for $n \times \left(\sum_{i=1}^{23} i\right) = n \times \left(\frac{23 \times 24}{2}\right) =$

$= n \times (23 \times 12) = 276n$ times

If $n < 23$, the code piece will run for $k \times n$, where $k = const$, $k \in \mathbb{N}$ and $k < 276$

However, we know that $O(276n) = O(n)$

$\Rightarrow$ $\boxed{a) \text{ is } O(n)}$ (also $O(kn) = O(n)$)

b) The outer loop will run for 2 iterations.
The inner loop will run for:

- $k = 0$: 0 iterations
- $k = 1$: (n^2-1) iterations and then the outer loop breaks because $k = n^2 - 1 > n$ for $n > 1$

$\Rightarrow$ The ~~[illegible]~~ code snippet will do $2(n^2-1)$ iterations and $O(2n^2-2) = O(n^2)$, because the 2's are constants!

c) We have 2 options every time the function is ran:

I: $n \leq c$: terminate

II: $n > c$: divide n by c and call the function again.

Let's choose a fixed constant $k \in \mathbb{N}$ for which $c^k \geq n > c^{k-1}$. We know that such constant exists, because this is the same as saying: n is between some number and a number bigger than it.

In order to reach the base case, n should become less than or equal to c. Since we divide n by c each step and n starts at a value for which $n \leq c^k$, then after:

- 1 step: $n \leq c^{k-1}$
- 2 steps: $n \leq c^{k-2}$

  $\vdots$

- i steps: $n \leq c^{k-i}$

We want $n \leq c = c^1$; Then if we set $i = k-1$:

$n \leq c^{k-(k-1)} = c^{\cancel{k}-\cancel{k}+1} = c^1 = c$ will happen after $(k-1)$ steps. ~~[illegible]~~

$$k-1 = \log_c(c^{k-1}) < \log_c n \leq \log_c c^k = k$$

$\Rightarrow$ The ~~[illegible]~~ algorithm's complexity is $\log_c n$. However, since the base of the algorithm is irrelevant, the algorithm has complexity $O(\log_c n) = \boxed{O(\log n)}$