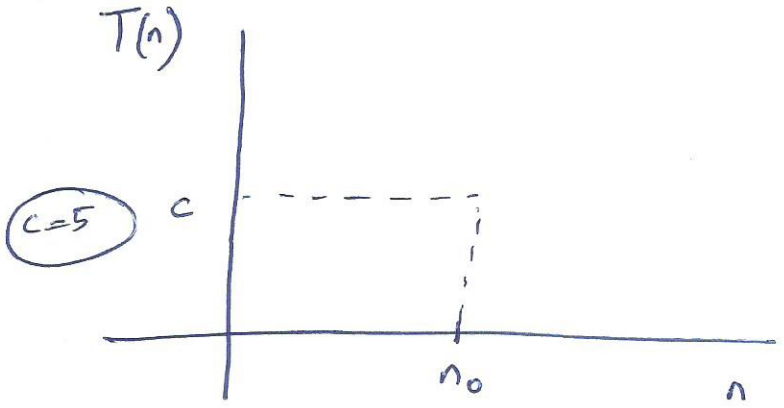
1. Explain what you mean by $T(n) = O(f(n))$

[A] $T(n)$ is $O(f(n))$



if $n_0 \geq 0$,
$n \geq n_0$,
$c > 0$,
then we have
$T(n) \leq c f(n)$
$T(n) \leq 5f(n)$

Let $c = 5$

$f(n) = n$ $\quad T(n) \leq 5n$
$f(n) = n^2$ $\quad T(n) \leq 5n^2$
$f(n) = n^3$ $\quad T(n) \leq 5n^3$

} T is asymptotically upper bounded by f.

we say, $T(n) \leq cf(n) \Rightarrow T$ is asymptotically upper bounded by f.

2. Express the upper bounds on running times for an algorithm whose running time is $T(n) = pn^2 + qn + r$.
Here, p, q, r are positive scalars.

[A] We say that $T(n)$ has $O(n^2)$ since for $n \geq 1$,

$qn \leq qn^2$,
$r \leq rn \leq rn^2$

So, $T(n) = pn^2 + qn + r$
$\leq pn^2 + qn^2 + rn^2$
$= (p+q+r)n^2$

(or), $T(n) \leq (p+q+r)n^2$ for all $n \geq 1$.

ie; $T(n) \leq cn^2$, where $c = p+q+r$

ie; $T(n) \leq c f(n)$, where $f(n) = n^2$

We can also say $T(n) \leq (p+q+r)n^3$ since $n^3 \geq n^2$ for $n \geq 1$.

So, $T(n)$ has $O(n^2)$, $O(n^3)$

While $T(n)$ has $O(n^2)$ as its tightest upper bound,
$T(n)$ has $O(n^3)$ as weak upper bound.

3. Express the lower bounds on $T(n) = pn^2 + qn + r$

[A] For $n \geq 1$, $T(n) = pn^2 + qn + r \geq pn^2$ where
p, q, r are positive scalars/constants.

ie; $T(n) = \Omega(n^2)$ where $\Omega(n^2) = pn^2$

Thus, $T(n) = \Omega(f(n))$ where $f(n) = n^2$

Just like strong and weak upper bounds, we also have strong and weak lower bounds!

Strong lower bound for $T(n)$ is pn^2 and weak lower bound for $T(n)$ is pn. We say
$T(n) \geq pn^2 \geq pn$.

4. What is poly-time ?

[A] Input size = N

Scalars/constants = c, d

Running time = cN^d

5. What if input size is doubled/tripled ?

[A] (a) Input size = $2N$

Running time for input size $2N$ $= c(2N)^d = c \cdot 2^d \cdot N^d$

$= 2^d (cN^d)$

$= 2^d$ (Running time for input size N)

(or) $\frac{\text{Running time (size } 2N)}{\text{Running time (size } N)} = 2^d > 1$

(or) Algorithm slows down in exponential order.

(b) Input size = $3N$

Running time for input size $3N$ $= c(3N)^d = c \cdot 3^d N^d$

$= 3^d (c \cdot N^d)$

$= 3^d$ (Running time for size N)

(or) $$\frac{\text{Running time (size } 3N)}{\text{Running time (size } N)} = 3^d >>> 1$$

Generalizing,

$$\frac{\text{Running time (size } MN)}{\text{Running time (size } N)} = M^d >>> 1$$

6. What is worst-case run-time? Can random inputs be good?

[A] Worst-case run time $\Rightarrow$ Slowest possible run time

Random inputs are not good $\Rightarrow$ All distributions cannot perform well on all inputs.

7 Distinguish between Upper & Lower Bounds.

[A] Let computational complexity be $T(n)$

$$T(n) \leq c\,f(n), \quad c > 0, \quad n_0 \geq 0$$

Upper bounds:

$T(n) = O(f(n))$ where $f(n)$ is any kind of polynomial function

$$T(n) \geq c\,f(n)$$

Lower bounds:

$T(n) = \Omega(f(n))$

6: Consider $T(n) = 32n^2 + 17n + 32$

where $p = r = 32$, $q = 17$

[A] From Q3, we find that $T(n) \geq 32n^2 = \Omega(n^2)$

We write $T(n) = \Omega(f(n))$ where $f(n) = n^2$ ← strong lower bound

Similarly, $T(n) \geq 17n = \Omega(n)$

We write $T(n) = \Omega(f(n))$ where $f(n) = n$ ← weak lower bound

For upper bounds, we can write

$$T(n) = 32n^2 + 17n + 32$$
$$\leq 32n^2 + 17n^2 + 32n^2$$
$$= (32 + 17 + 32)\, n^2$$
$$= 81n^2$$

Thus, $T(n) \leq 81n^2 = O(n^2)$

Thus, $T(n) = O(f(n))$ where $f(n) = n^2$ ← strong upper bound

Similarly, we can write $T(n) = O(f(n))$ where $f(n) = n^3$ ← weak upper bound

We also say $T(n)$ is $\Theta(n^2)$ since $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$ where $f(n) = n^2$

(7) What is transitivity?

[A] Let there be two functions $f(n)$ and $g(n)$,

$f(n) \neq g(n)$

If both $f(n)$ and $g(n)$ have the same upper bound, say $O(n^3)$, then just because they have the same upper bound, it does not mean $f(n) = g(n)$

Eg. $f(n) = 5n^3 \leq cn^3$ where $c \geq 5$
$= O(n^3)$

$g(n) = 3n^2 \leq kn^3$ where k is any positive integer such that the inequality holds good!

(8) Computational complexity of all sorting algorithms → UB

LB ↑

Algorithm	Best	Average	Worst
Selection Sort	$\Omega(n^2)$	$\theta(n^2)$	$O(n^2)$
Bubble sort	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$
Insertion Sort	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$
Heap Sort	$\Omega(n \log n)$	$\theta(n \log n)$	$O(n \log n)$
Quick sort	$\Omega(n \log n)$	$\theta(n \log n)$	~~$\theta(n \log n)$~~ $O(n^2)$
Merge sort	$\Omega(n \log n)$	$\theta(n \log n)$	$O(n \log n)$
Bucket sort	$\Omega(n+k)$	$\theta(n+k)$	$O(n^2)$
Radix sort	$\Omega(nk)$	$\theta(nk)$	$O(nk)$

10. Let $f_1(n) = 10^n \le n^n$ for $n \ge 10$ — Let $z = \log_2 n$

$f_2(n) = n^{1/3} \Rightarrow \log_2 f_2(n) = \log_2 n^{1/3} = \frac{1}{3}\log_2 n = \frac{1}{3}z$

$f_3(n) = n^n$

$f_4(n) = \log_2 n \Rightarrow \log_2 f_4(n) = \log_2(\log_2 n) = \log_2 z$

$f_5(n) = 2^{\sqrt{\log_2 n}}$

From (2-8), for every $b > 1$, $x > 0$, $\log_b n = O(n^x)$. Apply this to $f_4(n)$. So, $\log_2 n = O(n^x) = O(n^{1/3})$ where $x = 1/3 > 0$

From (2-9), for every $r > 1$, $d > 0$, $n^d = O(r^n)$

$\log_2 f_5(n) = \log_2 2^{\sqrt{\log_2 n}} = \sqrt{\log_2 n}$

From this, we have $f_4(n) = \log_2 n = O(n^{1/3}) = O(f_2(n))$

$= (\log_2 n)^{\frac{1}{2}} = \sqrt{f_4(n)}$

$\sqrt{f_4(n)} = 2^{\sqrt{\log_2 z}} = 2^{(\log_2 z)^{\frac{1}{2}}}$

(or) $f_5(n) = 2$

$= \left(2^{\log_2 z}\right)^{\frac{1}{2}} = z^{1/2}$

Functions	Bounds
$\log_2 f_1(n)$	$\log_2 n^n = n\log_2 n$
$\log_2 f_2(n)$	$\frac{1}{3}\log_2 n$
$\log_2 f_3(n)$	$n\log_2 n$
$\log_2 f_4(n)$	$\log_2 \log_2 n$
$\log_2 f_5(n)$	$(\log_2 n)^{1/2}$

Graph these functions for their bounds for $1 \le n \le 100$ (say)

(Pge 65)

11. Let $f = O(g)$. Then show that $g = \Omega(f)$ (8)

upper bound — lower bound.

[A] By definition, if $f = O(g)$, then for any constants c and n_0, we must have $f(n) \leq c\,g(n)$

$\Rightarrow g(n) \geq \frac{1}{c} f(n)$ if c is > 0 for all $n \geq n_0$.

(or) $g = \Omega(f)$