a) $1+\frac{1}{N}$ | As $N \to \infty$, $\frac{1}{N} \to 0$

$\therefore \quad 1+\frac{1}{N} \sim 1$

b) $\frac{\log_2(N^2+N)}{\sqrt{N}} \sim \frac{\log_2(N^2)}{\sqrt{N}}$

c) $\left(1+\frac{1}{N}\right)\left(1+\frac{2}{N^2}\right) = 1+\frac{2}{N^2}+\frac{1}{N}+\frac{2}{N^3} \sim 1$

As $N \to \infty$, expression $\to 1$

d) $\sim 2N^3$

e) $\frac{\log_2(3N)}{\log_2 N} = \frac{\log_2 3 + \log_2 N}{\log_2 N} = \frac{\log_2 3}{\log_2 N} + 1$

As $N \to \infty$, expression $\to 1$

$\therefore \quad \sim 1$

$\lg N^2 = \frac{\log_2 N^2}{\log_2 10}$

b) $\frac{\lg(N^2+1)}{\log_2 N} \sim \lg N^2$ $\sim \frac{\log_2 N^2}{\log_2 10 \cdot \log_2 N} \sim \frac{2\log_2 N}{\log_2 N} \sim 2$

g) $\frac{N^{100}}{2^n}$ if n & N are different, no approximation

for $\frac{N^{100}}{2^N}$, for a very large N

~ 0

h) $\frac{n \log_2 (n+1) + 2}{n+1} \sim \frac{n \log_2 n}{n} \sim \log_2 n$

i) $\frac{\log (N!)}{N \log N} = \frac{\log (N(N-1)(N-2) \cdots (N-x))}{N \log N}$

$\sim \frac{\log N^N}{N \log N} \sim 1$

2) a) first loop: $N, \frac{N}{2}, \frac{N}{4}, \frac{N}{8}, \cdots, \frac{N}{2^k}$

$\downarrow$

$N \cdot \frac{1}{2^k} = 1$ (executes until it is 1)

$N = 2^k$

$\Rightarrow \log_2 N = k$

second loop : N times

sum ++ : $N + \frac{N}{2} + \frac{N}{4} + \dots + 1$

$a = N, \quad r = \frac{1}{2}, \quad n = \log_2 N + 1$

$\log_2 N + \log_2 2$

$= \log_2(2N)$

$$S = N \cdot \frac{1 - (1/2)^{\log_2 N + 1}}{1 - \frac{1}{2}} = 2N \left(1 - \frac{1}{2}^{\log_2 N + 1}\right)$$

$\to 1/2$

$$= 2N \left(1 - \frac{1}{2^{\log_2 2N}}\right)$$

$$= 2N \left(1 - \frac{1}{2N}\right)$$

$\to \approx 0$ for large N

$$\therefore S \sim 2N$$

∴ Order of growth is $O(N)$

b) Outer loop: $1, 2, 4, \dots, 2^k$ (last term $= N$) $\Rightarrow O(\log_2 N)$

$$2^k \leq N$$

$$\log_2 N = k$$

Inner loop: i times for each i

Sum ++ : Executions: $1 + 2 + 4 + \dots + \frac{N}{2}$

Summation, $S = \frac{1(2^{\log_2 N} - 1)}{2 - 1}$

$= N - 1$

∴ Order of growth is $O(N)$

c) Outer loop: $O(\log_2 N)$ times

Inner loop: $O(N)$ times

Sum ++: $O(N(\log_2 N))$ times

∴ Order of growth is $O(N \log_2 N)$

d) Outer loop: $ck = n \Rightarrow k = \frac{n}{c}$

executes k times?

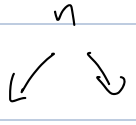
Inner loop: 10 times

sum : $10k$ times

$\Rightarrow 10\frac{n}{c}$ times

∴ Order of growth is $O(n)$

3)



$\frac{n}{2}$ $\frac{n}{2}$

$\frac{n-1}{2}$ comparisons $\frac{n-1}{2}$ comparisons

3 comparisons per group

$\frac{3}{2} \times 6$

min = 1

max = 2

6 + 1 = 7

n = 4

[1, 2, 3, 4, 5, 6]

[3, 4] [5, 6]

max > 4?

min < 3? min < 5?

max > 6?

For an even 'n':

Compare the first two elements of the input array. (1 comparison)

Initialize min to the smaller value, max to the larger value.

For the remaining pairs, compare array[i] and array[i+1], i.e. in pairs.

3 comp. { The smaller value of the pair (assigned candidate minimum) is compared to the 'min' assigned earlier. If it is smaller, 'min' is updated.

3 comp. { Similarly the larger value (assigned candidate maximum) is compared to the 'max' assigned earlier, and it is updated if larger.

Total no. of comparisons $= \frac{3(n-2)}{2} + 1$

For an odd 'n', instead assign both max and min to the first value in the input array (leaving n-1 elements)

The other steps remain the same.

Total no. of comparisons $= \frac{3(n-1)}{2}$

∴ Both cases require less than $\frac{3n}{2}$ comparisons.

4) $[0, 4n]$ with repititions

- Create an integer array B of size $4n+1$ (all elements initialized to 0) to

O(n) { Keep record of the frequency of each element chosen in A. For instance, if $A[0] = 0$ is chosen 5 times, then $B[0] = 5$.

O(n) • Iterate from $i = 0$ to $4n$ and perform $B[A[i]]++$. This updates the count of each element from 0 to $4n$.

- Initialize maxCount and maxVal (=0) so that we can assign the greatest count and corresponding maximum value.
- Iterate through array B and compare each value to maxCount. If $B[i] > \text{maxCount}$, then $\text{maxCount} = B[i]$ and $\text{maxVal} = A[i]$.
- The value k with the greatest frequency is obtained.

Updating an array is constant run time.

∴ Running time is $O(n)$

5) a) push 1, 2, 3, 4

pop 4, 3, 2, 1

push 5, 6, 7, 8, 9

pop 9, 8, 7, 6, 5

∴ valid

b) push 0, 1, 2, 3, 4

pop 4

push 5, 6

pop 6

push 7, 8

pop 8, 7, 5, 3, 2

push 9

pop 9

cannot pop 0

∴ invalid ←

c) push 0, 1, 2

pop 2

push 3, 4, 5

pop 5

push 6

pop 6

push 7

pop 7

pop 4

push 8

pop 8

push 9

pop 9, 3, 1, 0

∴ valid

d) push 0, 1, 2, 3, 4

pop 4, 3, 2, 1, 0

push 5

pop 5

⋮

∴ valid

e) valid

f) push 0

pop 0

push 1, 2, 3, 4

pop 4

push 5, 6

pop 6, 5

pop 3

push 8

pop 8

cannot pop 1 since 2 is still in the stack.

∴ invalid ←

g) invalid at 0 2 ←

h) valid