ABDUL SABOOR

20L-1113 | BDS-5A

HOMEWORK #1:

Question #1:

(a) $T(n) = T(n/3) + O(n^2)$

Last term equates.

$$\frac{n}{3^k} = 1$$

$$n = 3^k$$

$$\log_3 n = k \log_3 3$$

$$\boxed{\log_3 n = k}$$

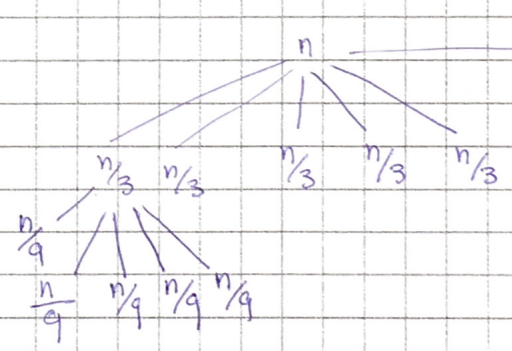
	terms	total
n	$\frac{n}{3^0}$	n^2
$n/3$	$\frac{n}{3^1}$	$(n/3)^2$
$n/9$	$\frac{n}{3^2}$	$(n/3^2)^2$
$\vdots$	$\vdots$	$\vdots$
1	$\frac{n}{3^k}$	$(n/3^k)^2$

So, height of tree is $\log_3 n$ and there are n^2 steps at each level. Hence.

$$n^2 \times \log_3 n$$

$$\Rightarrow n^2 \log_3 n \longrightarrow O(n^2) \text{ in Big-Oh}$$

(b) $T(n) = 5T(n/3) + O(n)^2$



total
n
$\frac{5n}{3}$
$\frac{25n}{9}$
$\vdots$
$\left(\frac{5}{3}\right)^k n$

$\Rightarrow$ Tree height

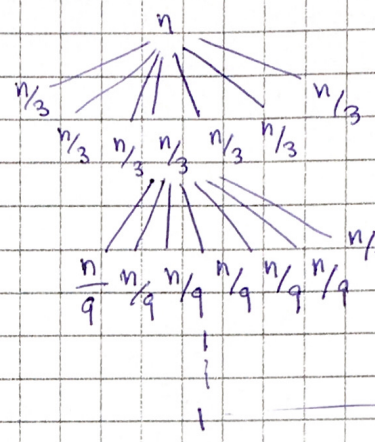
$$\frac{n}{3^k} = 1$$

$$\boxed{\log_3 n = k}$$

$\Rightarrow$ Since n^2 steps at each level, So total time is

$$n^2 \log_3 n \longrightarrow O(n^2) \text{ in Big-Oh.}$$

(c) $T(n) = 7T(n/3) + O(1)$



Steps.

$n \dots \frac{n}{3^0}$

$\frac{n}{3} \quad \frac{n}{3^1}$

$\frac{n}{9} \quad \frac{n}{3^2}$

$\frac{n}{3^k} \dots \frac{n}{3^k}$

Tree height =

$1 = \frac{n}{3^k}$

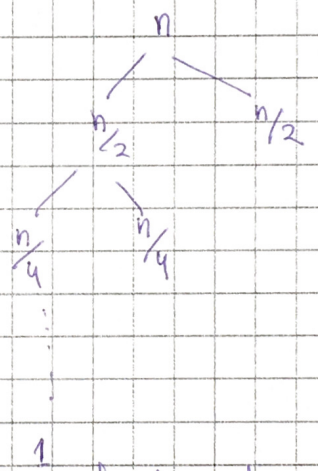
$3^k = \log_3 n$

$k \log_3 3 = \log_3 n$

$\boxed{k = \log_3 n}$

Since, there are constant time steps at each level. So total time is $\log_3 n$ —— $O(\log n)$ in Big-Oh.

(d) $T(n) = 2T(n/2) + O\left(\frac{n}{\lg(n)}\right)$



Steps

$\frac{n}{2^0}$

$\frac{n}{2^1}$

$\frac{n}{2^1}$

$\frac{n}{2^k}$

Tree - height

$\frac{n}{2^k} = 1$

$n = 2^k$

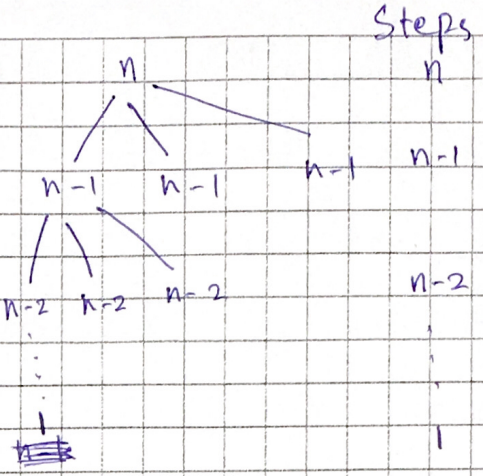
$\boxed{\log_2 n = k}$

Since, there are $\frac{n}{\lg(n)}$ steps at each level.

$\Rightarrow \frac{n}{\lg(n)} \times \log_2 n$

So, there are n steps and $O(n)$ in Big-Oh.

(e) $T(n) = 3T(n-1) + \Theta(1)$



Tree height:

$(n-1)+(n-2)---+1$

Since tree ~~is~~ is balanced we can assume calculation for single node as.

$1 + 2 + 3 - - - + (n-2) + (n-1)$, which forms an arithmetic series.

$\frac{n(n-1)}{2}$, which equates to n^2.

There are constant time steps at each node. So

$1 \cdot n^2 \longrightarrow O(n)^2$ in Big-Oh.

QUESTION # 2:

(a) At each step, we are dividing j by 2 ~~which ma~~ and this loop stops at point when j is 1 or greater.

This can be written as.

$$\frac{j}{2^k} = 1$$

$$j = 2^k$$

$$\log_2 j = k \log_2 2$$

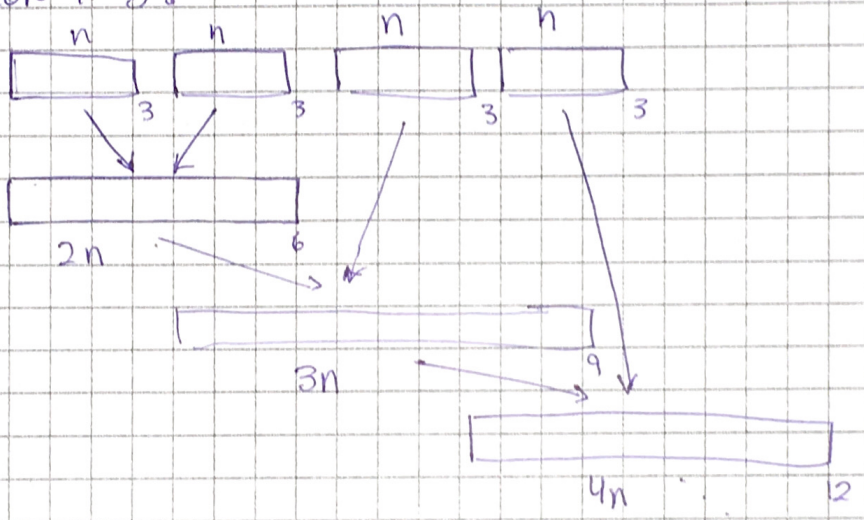
$$\log_2 j = k$$

So, $\Theta(\log_2 n)$ in Big-Oh. terms

(b) In this problem we have two loops performing a constant time and constant space operation of addition. So, total running time is equal to ~~number~~ sum of iterations of both loops. Which is equal to N+M. Hence, time complexity is $O(N+M)$ and space complexity is $O(1)$.

QUESTION # 3:

(a)



Suppose,
$k = 4$
$n = 3$.

So, Solving by doing two arrays at a time and analyzing time complexity for worst case. We have something like:

$$n + (n+n) + (n+n+n) \cdots kn$$

This forms an arithmetic series which can be written as.

$$n + (k-1)n.$$

So, computing this, the worst case complexity will be $O(kn)$

(b) A much efficient way to solve this would be initializing a pointer at the start of each array. Comparing these pointers and adding minimum value to another array. Doing this way we would manage to do merging in kn time. Which is not much faster but compared to arithmetic series, it is better.

Question #4:

$T(n) = \frac{1}{5} n^4 - 3n^2$ Prove $\Theta(n^4)$

$$k_1 n^4 \leq \frac{1}{5} n^4 - 3n^2 \leq k_2 n^4$$

$$\frac{k_1 n^4}{n^4} \leq \frac{\frac{1}{5} n^4}{n^4} - \frac{3n^2}{n^4} \leq \frac{k_2 n^4}{n^4}$$

$$k_1 \leq \frac{1}{5} - \frac{3}{n^2} \leq k_2 \qquad \boxed{n_0 = 1} \text{ (Suppose)}$$

~~$k_1 \leq -2.8 \leq k_2$~~

Let $n_0 = 1$

$$k_1 \leq \frac{1}{5} - 3 \leq k_2$$

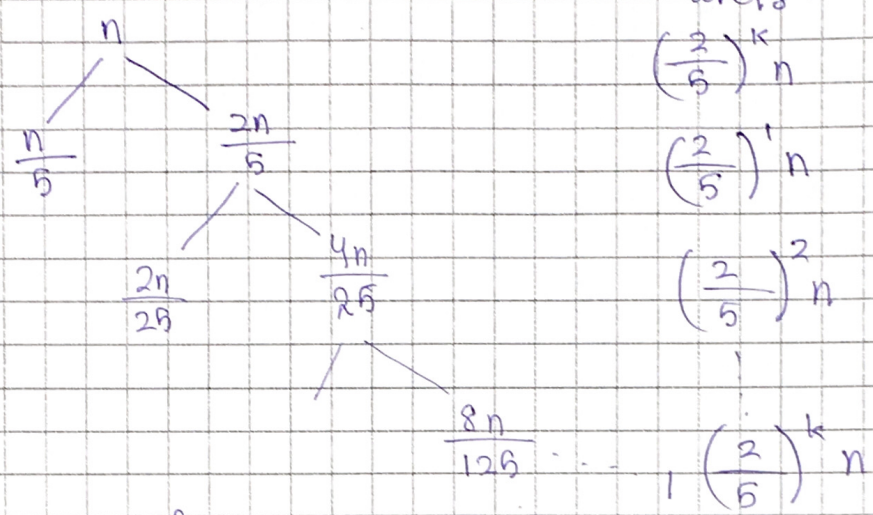
$$k_1 \leq -2.8 \leq k_2$$

So, $\boxed{k_1 = -3}$ and $\boxed{k_2 = 1}$

↳ Hence, $\Theta(n^4)$ fulfills $T(n)$

QUESTION # 5:

Trying the recursion tree method for worst-case



So, height of tree:

$$\left(\frac{2}{5}\right)^k n = 1$$

$$n = \left(\frac{5}{2}\right)^k$$

$$\log_{5/2} n = k \log_{5/2}(5/2)$$

$$\log_{5/2} n = k \cdot 1$$

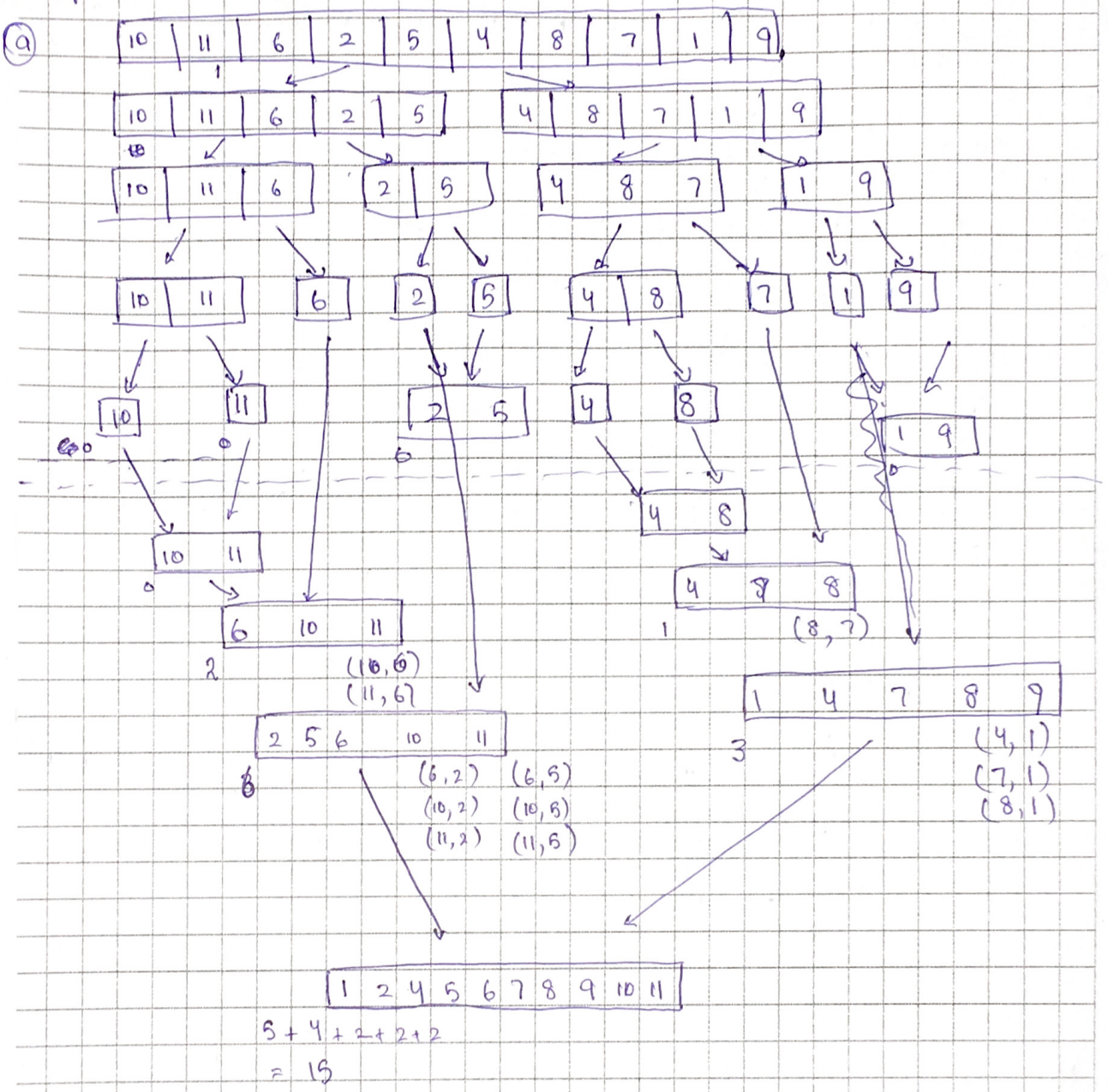
$$\boxed{k = \log_{5/2} n}$$

Now, Since there are further n steps at each node from the loop. we can multiply n by $\log_{5/2} n$ as

$n \cdot \log_{5/2} n \longrightarrow O(n \log n)$ in Big-Oh.

QUESTION # 6:

(a)



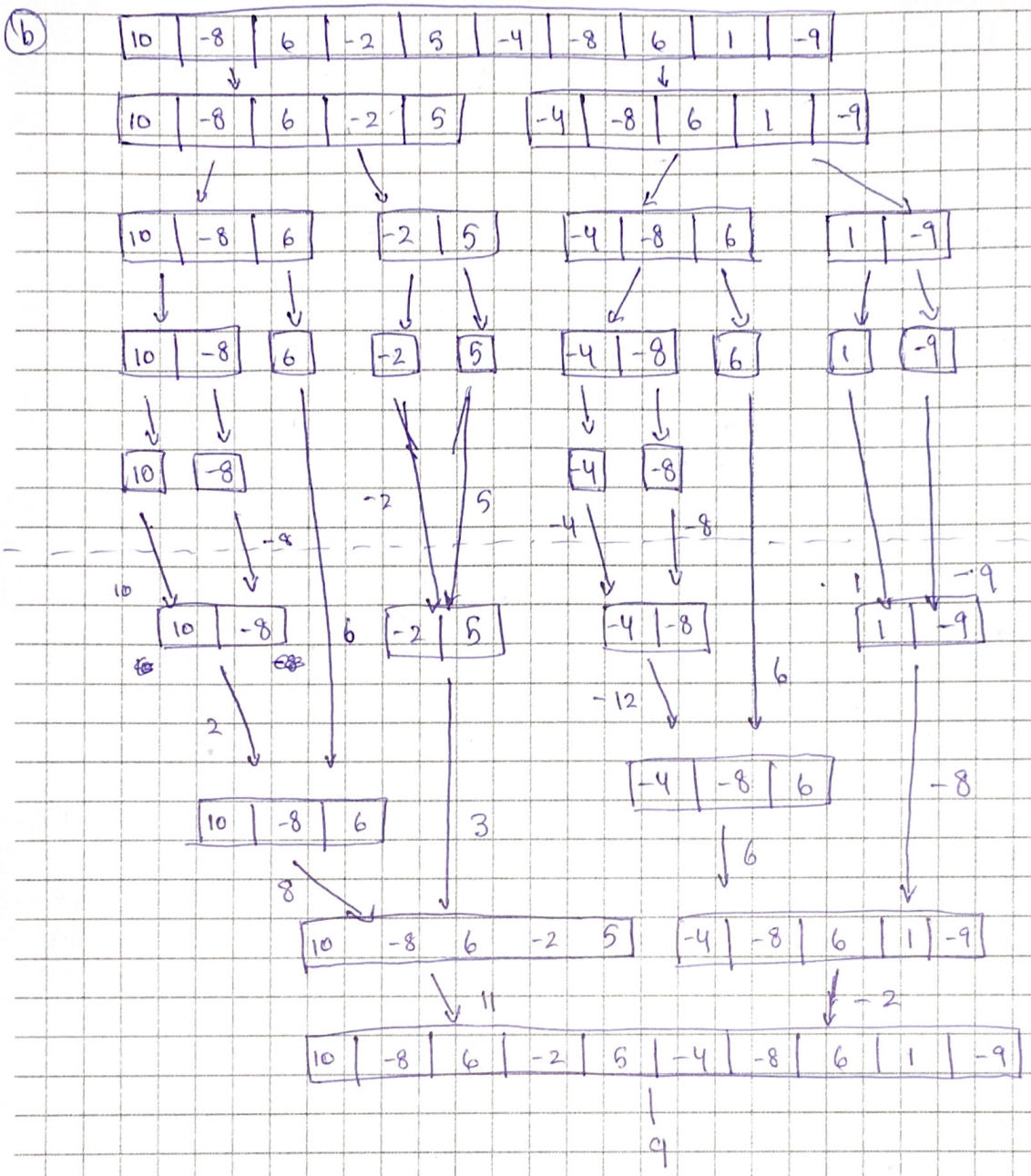
5 + 4 + 2 + 2 + 2

= 15

So, total count of split inversion pairs is:

2 + 1 + 6 + 3 + 15 = 27

(b)



Now, the maximum crossing sub-array sum is 11 and that gives the maximum sub-array sum from.

10 | -8 | 6 | -2 | 5 ⟶ max-subarray sum is 11