วิชา 204451 Algorithm Design and Analysis

รหัสนักศึกษา 630510600 ชื่อสกุล [illegible] ตอนที่ 2

**Assignment4 (8 ก.ค. 65) : กำหนดส่งงาน : จ. 25 ก.ค. 65 (เวลา 23.59 น.)**

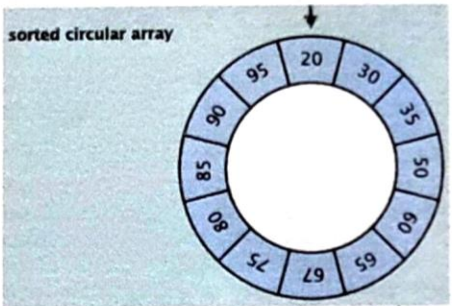
ให้นักศึกษา

1. เขียนคำตอบตามโจทย์กำหนดด้วยลายมือ แล้วถ่ายรูป (นามสกุล .jpg) หรือไฟล์ pdf ส่งที่เว็บส่งการบ้านภาควิชาฯ
2. ตั้งชื่อไฟล์ในรูปแบบ assign_x_id เมื่อ x คือหมายเลข Assignment และ id คือ รหัสนักศึกษา
   (กรณีส่งหลายไฟล์ให้ตั้งชื่อเป็น assign_01_id_a.jpg โดย a หมายถึง ลำดับไฟล์ แล้วทำการ zip รวมทุกไฟล์ส่งในงาน Assignment เดียวกันด้วยชื่อ assign_01_id.zip แทน )
3. ส่งงานภายในวันเวลาที่กำหนด หากส่งเลยกำหนดให้ชี้แจงเหตุผลกับอ. ประจำ section (พิจารณาคะแนนตามเหตุผล)

**ให้คำนวณหาค่า T(n) ของการแก้ปัญหาต่อไปนี้**

**(a) การค้นหาใน array ที่เรียงแล้วแบบวงกลม**

กำหนดอาเรย์ที่เรียงแล้วแบบวงกลมขนาด n ช่องและสมาชิก x มาให้ จงหาอัลกอริทึมที่มีประสิทธิภาพที่ตัดสินได้ว่า x อยู่ในอาเรย์หรือไม่




เขียนในหน้าถัดไป

ตัวอย่างในรูปสมมติว่าข้อมูลในอาเรย์เป็น 80,85,90,95,20,30,35,50,60,65,67,75 และ x คือ 20

**(b) ช่วงว่างที่กว้างที่สุด**

กำหนด *n* timestamps $x_1, x_2, \ldots, x_n$ ของไฟล์ที่ถูกส่งมาให้เครื่องเซิร์ฟเวอร์ จงหาอัลกอริทึมที่มีประสิทธิภาพที่หาช่วงว่างที่นานที่สุดที่ไม่มีไฟล์ถูกส่งมายังเครื่องเซิร์ฟเวอร์

เขียนในหน้าถัดไป

**(2) กำหนดฟังก์ชันต่อไปนี้**

```
Void Method1(A[1..n]):
    if n = 2 and A[1] > A[2]                 ____ O(1)
        swap (A[1],A[2])                     ____ O(1)
    else if n > 2                            ____ O(1)
        m ← ⌈ 2n/3 ⌉                         ____ O(1)
        method1(A[1 .. m])                   ____ T(2n/3)
        method1(A[n − m + 1 .. n])           ____ T(2n/3)
        method1( A[1 .. m])                  ____ T(2n/3)
// else n < 2 : do nothing — also in t_A : O(1)
```

$\left.\begin{matrix} O(1) \\ O(1) \end{matrix}\right\} t_A = O(1)$

$\left.\begin{matrix} O(1) \\ O(1) \end{matrix}\right\} t_C = O(1)$

$\left.\begin{matrix} T(\frac{2n}{3}) \\ T(\frac{2n}{3}) \\ T(\frac{2n}{3}) \end{matrix}\right\} t_B = 3.T(\frac{2n}{3})$

**ให้หาสมการ Tn() ของเวลาในการทำงานของ method1 ข้างต้น**

$$T(n) = \begin{cases} O(1) & ; \text{best-case} : n \le 2 \\ 3T(\frac{2n}{3}) + O(1) & \end{cases}$$

**และวิเคราะห์เพื่อแก้สมการ T(n) ข้างต้น**

เขียนในหน้าถัดไป

นาย [illegible]
Section 2 รหัส 630510000

คำนวณหาค่า $T(n)$ ของการแก้ปัญหา

ข้อ a.) Expected performance: $O(\log n)$.

Since the array is sorted — though it has uncertain order — we can modify the binary seach algorithm. The process is divided in to 2 parts: searching for the maximum value; the anchor, and the binary search that the position calculation is modified specifically for such the purpose.

---

i.) Algorithm to find the index of the maximum value in a sorted array.

Algorithm find_index_max(A: array, n: size of A, a: beginning index, b: ending index){

1. if $n = 0$: ____ $O(1)$
2. return Not Found and terminate ____ $O(1)$
3. if $A[b] \geq A[a]$: ____ $O(1)$
4. return b and terminate ____ $O(1)$
5. $m := \left\lfloor \frac{a+b}{2} \right\rfloor$ ____ $O(1)$
6. if $A[m] > A[a]$: ____ $O(1)$
7. $a := m$ ____ $O(1)$
8. else:
9. $b := m$ ____ $O(1)$
10. return find_index_max(A, n, a, b) and terminate ____ $T\left(\frac{n}{2}\right)$

}

Analysis: $t_A$ = line 1 − 4 $= \sum_{i=1}^{4} O(1) = O(1)$

$t_B$ = line 10 $= T\left(\frac{n}{2}\right)$

$t_C$ = line 5 − 9 $= \sum_{i=1}^{4} O(1) = O(1)$

$\therefore$ Given $T_1(n) = \begin{cases} t_A = O(1) & ; n = 0 \text{ or } A[b] \geq A[a] \\ t_B + t_C = T\left(\frac{n}{2}\right) + O(1) \end{cases}$

ii.) Algorithm to find if a particular value is in the array — the sorted but uncertain order array.

Algorithm bsc_helper (A: array, n: size of A, a: beginning index, b: ending index, key: value to find, max_anchor: index of the maximum element) {

1. $a' := (a - \text{max\_anchor} - 1) \bmod n$ ____ $O(1)$
2. $b' := (b - \text{max\_anchor} - 1) \bmod n$ ____ $O(1)$
3. $m := \left(\text{max\_anchor} + \left\lfloor \frac{a' + b'}{2} \right\rfloor + 1\right) \bmod n$ ____ $O(1)$
4. if $A[m] = \text{key}$: ____ $O(1)$
5. return m and terminate ____ $O(1)$
6. else if $a = b$: ____ $O(1)$
7. return Not Found and terminate ____ $O(1)$
8. if $\text{key} < A[m]$: ____ $O(1)$
9. return bsc_helper(A, n, a, m, key, max_anchor) and terminate ____ $T\left(\frac{n}{2}\right)$
10. else:
11. return bsc_help(A, n, (m+1) mod n, b, key, max_anchor) and terminate ____ $T\left(\frac{n}{2}\right)$

Analysis: $t_A$ = line 4 -> $= \sum_{i=1}^{7} O(1) = O(1)$

$t_B$ = either line 8-9 or line 10-11 $= T\left(\frac{n}{2}\right) + O(1)$

$t_C$ = line 1 ~ 3 $= \sum_{i=1}^{3} O(1) = O(1)$

$\therefore$ Given $T_2(n) = \begin{cases} t_A = O(1) & ; \text{key found immediately or impossible to find.} \\ t_B + t_C = T\left(\frac{n}{2}\right) + O(1) \end{cases}$

คำนวณหาค่า $T(n)$ ของการแก้ปัญหา

นาย [illegible]
section 2 รหัส 630510600

iii) Our main algorithm that put together the algorithms abae.

```
Algorithm binary_search_circular(A: array,
    n: size of A, key: value to find) {
    max_index := find_index_max(A, n, 0, n-1)          —— T1(n)
    return bsc_helper(A, n, (max_index+1) mod n,
        max_index, n, key, max_index)
        and terminate                                  —— T2(n)
```

**Analysis:** since this is not a recursive function, we may sum them up straightforwordly.

$$T(n) = T_1(n) + T_2(n) \qquad \text{———— } (*)$$

**Evaluate:** $T_1(n)$

Using the iterative substitution method

$$
\begin{aligned}
T_1(n) &= T_1\left(\tfrac{n}{2}\right) + O(1) \\
&= \left(T_1\left(\tfrac{n}{4}\right) + O(1)\right) + O(1) \\
&= T_1\left(\tfrac{n}{4}\right) + 2\cdot O(1) \\
&= \left(T_1\left(\tfrac{n}{8}\right) + O(1)\right) + 2\cdot O(1) \\
&= T_1\left(\tfrac{n}{8}\right) + 3\cdot O(1) \\
&= \cdots = T_1\left(\tfrac{n}{2^k}\right) + k\cdot O(1)
\end{aligned}
$$

consider $\frac{n}{2^k} = 1 \rightarrow k = \log_2 n$

then $T_1(n) = O(1) + (\log_2 n)\,O(1)$

$= O(\log n)$ □

**Evaluate:** $T_2(n)$

Using the recursive tree method

| level | | sum |
|---|---|---|
| 0 | $n : O(1)$ | $O(1)$ |
| 1 | $\frac{n}{2} : O(1)$ | $+\ O(1)$ |
| 2 | $\frac{n}{4} : O(1)$ | $+\ O(1)$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $k$ | $\frac{n}{2^k} : O(1)$ | $+\ O(1)$ |
| | | $T_2(n)$ |

consider $T_2(n) = T_2\left(\frac{n}{2^k}\right) + k\cdot O(1)$

consider $\frac{n}{2^k} = 1 \rightarrow k = \log_2 n$

then $T_2(n) = O(1) + (\log_2 n)\,O(1)$

$= O(\log n)$ □

Substitute $T_1(n)$ and $T_2(n)$ in $T(n)$;

$\therefore\ T(n) = O(\log n) + O(\log n)$

$= O(\log n)$ □

นาย [illegible]
section 2 รหัส 630510600

ข้อ b.) Expected performance: $O(n)$

The idea is to apply the radix sort algorithm as it only took $O(nk)$ where $k$ is the number of the maximum possible number of digits of each element. Then we simply apply the line sweep algorithm to find such the interval, which takes $O(n)$.

i.) The radix sort algorithm

We will fix $k = 12$ — the greatest possible number of digits of seconds — calculated by converting all the time unit (i.e. year, month, day, hour, minute) into seconds. In other words, worst case — 999912-31-23-59-59 — takes 311,042,764,799 seconds; 12 digits.

So technically, $k$ is a constant, so our radix sort only takes $O(n)$.

```
Algorithm radix_sort(A: array, n: size of A,
        k = 12 : no. of digits) {
  Convert all timestamps in A into seconds ______ O(n)
  B := Array of 10 empty linked lists. ______ O(1)
  for i = 0 -> n:                      ______ O(n)
      Append A[i] to B[A[i] mod 10]    ______ O(n)
  for i = 1 -> k:                      ______ O(k)
      B' := Array of 10 empty linked lists. ______ O(k)
      for every linked list b of B:    ______ O(k)
          for every node n of b:       ______ O(nk)
              d := the i^th digit of n ______ O(nk)
              Append n to B'[d]        ______ O(nk)
      B := B' and delete old B from the memory ______ O(k)
  X := Array of size n
  for every node b of B:               ______ O(1)
      Append every node of b to X.     ______ O(n)
  return X and terminate }             ______ O(1)
```

Analysis: from the algorithm;

$$T_1(n) = 3O(nk) + 7O(n) + 4O(k) + 3O(1)$$

substitute $k = 12$;

$$T_1(n) = 3O(12n) + 7O(n) + 4O(12) + 3(1)$$
$$= 7O(n) + 7O(1)$$
$$= O(n)$$

$\therefore T_1(n) = O(n)$ ◻

ii.) The line sweep algorithm.

To fit in with the context, we will call this the longest_distance algorithm instead.

```
Algorithm longest_distance(A: array,
      n: size of A) {
  A := radix_sort(A, n) ______ T_1(n) = O(n)
  ret := Not Found      ______ O(1)
  mx := 0               ______ O(1)
  for i = 0 -> n-2:     ______ O(n)
      if A[i+1]-A[i] > mx: ______ O(n)
          mx := A[i+1]-A[i] ______ O(n)
          ret := i      ______ O(n)
  return the pair (ret, ret+1)
      or Not Found if ret hasn't been
      changed.          ______ O(1)
}
```

Analysis: from above;

$$T(n) = 5O(n) + 3O(1) = O(n) + O(1)$$
$$= O(n)$$

$\therefore T(n) = O(n)$ ◻

(ต่อ) ข้อ 2. ข้อ method1 $T(n) = 3T(\frac{2n}{3}) + O(1)$

นายศักดิ์ศิลา ทพล เพชรสุภิเดช
Section 2 รหัส 630510600

## Using the iterative substitute method

Consider

$$T(n) = 3T(\tfrac{2}{3}n) + O(1)$$
$$= 3\left(3T(\tfrac{4}{9}n) + O(1)\right) + O(1)$$
$$= 9T(\tfrac{4}{9}n) + 3O(1) + O(1)$$
$$= 9\left(3T(\tfrac{8}{27}n) + O(1)\right) + (1+3)O(1)$$
$$= 27T(\tfrac{8}{27}n) + 9O(1) + (1+3)O(1)$$
$$= \cdots = 3^k T\left(\frac{2^k n}{3^k}\right) + O(1)\sum_{i=0}^{k-1} 3^i$$

Consider the base case ($n = 2$) in the common term

$$\frac{2^k}{3^k}n = 2 \rightarrow \frac{3^k}{2^k} = \frac{n}{2} \rightarrow k = \log_{\frac{3}{2}} \frac{n}{2}$$
$$= \log_{\frac{3}{2}} n - \log_{\frac{3}{2}} 2$$

And $3^k = 3^{\log_{\frac{3}{2}} n - \log_{\frac{3}{2}} 2}$

$$= \frac{3^{\log_{\frac{3}{2}} n}}{3^{\log_{\frac{3}{2}} 2}} = \frac{1}{3^{\log_{\frac{3}{2}} 2}} \cdot n^{\log_{\frac{3}{2}} 3}$$

Then $T(n) = \frac{2}{3^{\log_{\frac{3}{2}} 2}} n^{\log_{\frac{3}{2}} 3} + \frac{n^{\log_{\frac{3}{2}} 3}}{2 \cdot 3^{\log_{\frac{3}{2}} 2}} - \frac{1}{2}$

$$= \frac{5}{2 \cdot 3^{\log_{\frac{3}{2}} 2}} n^{\log_{\frac{3}{2}} 3} - \frac{1}{2}$$
$$= \Theta(n^{\log_{\frac{3}{2}} 3})$$

$\square$

## Using the Master method

According to $T(n)$, $a = 3$, $b = \frac{3}{2}$, $f(n) = O(1)$.

Let $c$ be a constant such that $f(n) = O(n^c)$.

Since $c = 0$, (because $O(1) = O(n^0)$) and

$\log_b a = \log_{\frac{3}{2}} 3 \approx 0.369 > c$, then

$f(n) = O(n^{\log_b a - \varepsilon})$; $\varepsilon > 0. \rightarrow \varepsilon = \log_{\frac{3}{2}} 3$

$\therefore T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_{\frac{3}{2}} 3})$

$\square$

วิชา 204451 Algorithm Design and Analysis

รหัสนักศึกษา 630510600 ชื่อสกุล ศิณศิลาดล เพชรกุลจินดา ตอนที่ 1

**(2)** จง**<u>แสดงวิธีทำ</u>**เพื่อหาคำตอบของสมการ Recurrence relation ที่กำหนดให้ต่อไปนี้เพื่อวิเคราะห์หาเวลาในการทำงาน (Running Time) โดยจัดให้อยู่ในรูปที่ง่ายที่สุด

A) ให้ใช้วิธี Iterative Substitute Method หรือ Recursion Tree Method

B) และวิธี **Master Theorem**

กำหนดสมการ Recurrence relationดังนี้ (1 ข้อ ให้ทำ 2 วิธี)

1) $T(n) = T(n-1) + 1$ $= \Theta(n)$ ใช้ master ไม่ได้
2) $T(n) = 4T(n/4) + c, T(1)=c$ $= \Theta(n)$
3) $T(n) = 3T(n/3) + n, T(1) = c$ $= \Theta(n \log n)$
4) $T(n) = 2T(n/3) + n^2, T(1)=c$ $= \Theta(n^2)$
5) $T(n) = 4T(n/2) + n^2, T(1)= c$ $= \Theta(n^2)$
6) $T(n) = 2T(n/2) + n\log n, T(1)= c$ $= \Theta(n \log n)$ ใช้ master ไม่ได้
7) $T(n) = T(n/2) + T(n/8) + n, T(1)= c$ $= \Theta(n)$ ใช้ master ไม่ได้

นาย ศิณ์ศิลกาล เพชรกุลสิริเดช
section 2 รหัส 630510600

1.) $T(n) = T(n-1) + 1$

**Using the iterative method**

Consider $T(n) = T(n-1) + 1$
$= (T(n-2) + 1) + 1$
$= T(n-2) + 2$
$= (T(n-3) + 1) + 2$
$= T(n-3) + 3$
$= \cdots = T(1) + (n-1) = n$
$= \Theta(n)$

$\therefore\ T(n) = \Theta(n)$ ☐

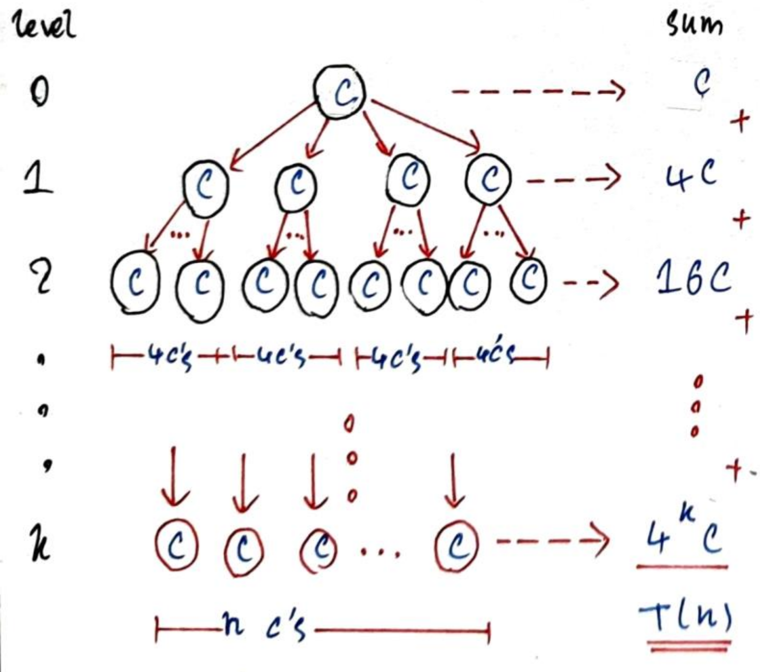
**Using the Master method**

Since $T(n)$ is not in the form of $aT(\frac{n}{b}) + f(n)$; for some constants $a \geq 1$, $b > 1$, and a function $f(n)$,

$\therefore$ The master method does not apply. ☐

2.) $T(n) = 4T(\frac{n}{4}) + c$ , $T(1) = c$

**Using the recursion tree method**




consider $\frac{n}{4^k} = 1 \rightarrow k = \log_4 n$

Consider $T(n) = c + 4c + 16c + \cdots + 4^k c$
$= (1 + 4 + 16 + \cdots + 4^k)\,c$
$= \frac{4^{k+1} - 1}{4 - 1} c = \frac{4c}{3} n - \frac{c}{3} = \Theta(n)$

$\therefore\ T(n) = \Theta(n)$ ☐

**Using the Master method**

Identifying $T(n)$, we get $a = 4$, $b = 4$, and $f(n) = c$.

Consider $f(n) = c = O(1) = O(n^0) = O(n^{\log_b a - \varepsilon})$; for some $\varepsilon > 0$, and $\log_b a = \log_4 4 = 1$,
Then we find $\varepsilon = 1$ that satisfies $f(n) = O(n^{\log_b a - \varepsilon})$

$\therefore\ T(n) = \Theta(n^{\log_b a}) = \Theta(n)$ ☐

นาย ต้นศิลกาล เพชรกุลจินดา
section 2 รหัส 630510600

3.) $T(n) = 3T\left(\frac{n}{3}\right) + n,\ T(1) = c$

**Using the iterative substitution method**

Consider $T(n) = 3T\left(\frac{n}{3}\right) + n$

$$= 3\left(3T\left(\frac{n}{9}\right) + \frac{n}{3}\right) + n$$

$$= 9T\left(\frac{n}{9}\right) + n + n$$

$$= 9\left(3T\left(\frac{n}{27}\right) + \frac{n}{9}\right) + 2n$$

$$= 27T\left(\frac{n}{27}\right) + n + 2n$$

$$= \ldots = 3^k T\left(\frac{n}{3^k}\right) + kn$$

Consider $\frac{n}{3^k} = 1 \rightarrow k = \log_3 n$

then $T(n) = \cancelto{n}{3^{\log_3 n}} \cdot T(1) + n \log_3 n$

$$= c \cdot n + n \log_3 n$$

$$= \Theta(n \log n)$$

$\therefore\ T(n) = \Theta(n \log n)$ $\square$

**Using the Master method**

Identifying $T(n)$, we get $a = 3$, $b = 3$, and $f(n) = n$.

Consider $f(n) = n = \Theta(n^{\log_b a})$, and because $\log_b a = \log_3 3 = 1$, then $f(n) = \Theta(n^{\log_b a})$.

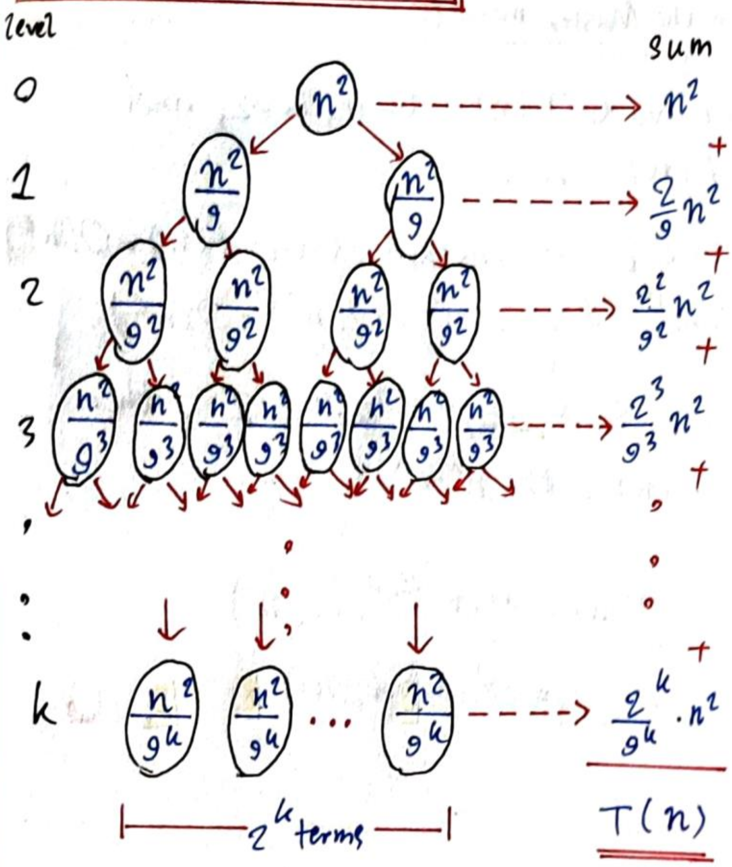
$\therefore\ T(n) = \Theta(n^{\cancelto{1}{\log_b a}} \log n)$

$$= \Theta(n \log n)$$ $\square$

# 4. $T(n) = 2T(\frac{n}{3}) + n^2$ , $T(1) = c$

นาย [illegible]
section 2 รหัส 630510600

**Using the recursion tree method**




consider $\frac{n}{3^k} = 1 \rightarrow k = \log_3 n$

then $T(n) = \left(\sum_{i=0}^{k-1} \left(\frac{2}{9}\right)^i\right) n^2 + 2^k \cdot c$

$= \dfrac{1 - \left(\frac{2}{9}\right)^k}{1 - \frac{2}{9}} n^2 + 2^k \cdot c$

$= \frac{9}{7}\left(1 - \left(\frac{2^{\log_3 n}}{\cancel{n^2}}\right)\right)\cancel{n^2} + 2^{\log_3 n} \cdot c$

$= \frac{9}{7}n^2 - \frac{9+7c}{7}\left(\frac{2^{\log_3 n}}{1}\right)$

$= \Theta(n^2)$

$\therefore T(n) = \Theta(n^2)$ ☐

**Using the Master method**

According to $T(n)$, $a = 2$, $b = 3$, and $f(n) = n^2$.

Let $c$ be a constant such that $f(n) = n^2 = \Theta(n^c)$.

since $c > \log_b a = \log_3 2$,

then $c = \log_b a + \varepsilon$

$2 = \log_3 2 + \varepsilon$

$\varepsilon = \log_3 4.5 > 0$

And for sufficiently large $n$,

$a \cdot f(\frac{n}{b}) = 2f(\frac{n}{3}) = \frac{2}{9}n^2$

$\leq c \cdot n^2 = c \cdot f(n)$

is true when $c = \frac{1}{3} < 1$

$\therefore T(n) = \Theta(f(n)) = \Theta(n^2)$ ☐

นายสิทธิ์ศิลกาล เพชรจุลจินดา
section 2 รหัส 630510600

5.) $T(n) = 4T\left(\frac{n}{2}\right) + n^2,\ T(1) = c$

**Using the iterative substitute method**

Consider $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

$$= 4\left(4T\left(\frac{n}{4}\right) + \frac{n^2}{4}\right) + n^2$$

$$= 16T\left(\frac{n}{4}\right) + 2n^2$$

$$= 16\left(4T\left(\frac{n}{8}\right) + \frac{n^2}{16}\right) + 2n^2$$

$$= 64T\left(\frac{n}{8}\right) + 3n^2$$

$$= \dots = 4^k T\left(\frac{n}{2^k}\right) + kn^2$$

Consider $\frac{n}{2^k} = 1 \longrightarrow k = \log_2 n$,

then $T(n) = 4^{\log_2 n} \cdot T(1) + n^2 \cdot \log_2 n$

$$= c \cdot n^2 + n^2 \log_2 n$$

$$= \Theta(n^2 \log n)$$

$\therefore\ T(n) = \Theta(n^2 \log n)$ $\square$

**Using the Master method**

According to $T(n)$, $a = 4$, $b = 2$, and $f(n) = n^2$.

Let $c$ be a constant that $f(n) = O(n^c)$, then $f(n) = n^2 = O(n^2) = O(n^c)$.

Since $c = \log_b a = \log_2 4 = 2$, then $f(n) = \Theta(n^{\log_b a})$.

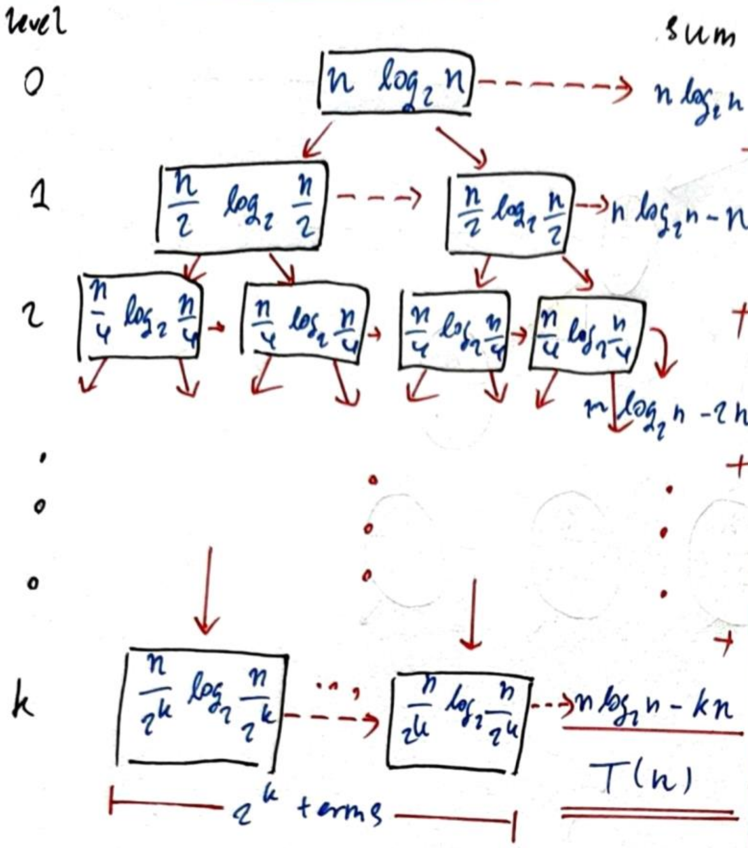
$\therefore\ T(n) = \Theta(n^{\log_b a} \log n)$

$$= \Theta(n^2 \cdot \log n)$$ $\square$

นาย ศักดิ์มงคล เพชรกุลอินทร
section 2 รหัส 630510606

6.) $2T(\frac{n}{2}) + n\log n,\ T(1) = c$

## Using the recursion tree method




Consider $\frac{n}{2^k} = 1 \longrightarrow k = \log_2 n$

Then $T(n) = k \cdot n \log_2 n - \sum_{i=1}^{k-1} i \cdot n + c \cdot 2^k$

$= n \cdot \log_2{}^2 n - \frac{(k-1)k}{2} \cdot n + cn$

$= n \cdot \log_2{}^2 n - \frac{n \log_2{}^2 n}{2} + \frac{n \log_2 n}{2} + cn$

$= \frac{1}{2} n \cdot \log_2{}^2 n + \frac{1}{2} n \log_2 n + cn$

$= \Theta(n \cdot \log_2{}^2 n)$

$\therefore T(n) = \Theta(n \cdot \log_2{}^2 n)$

## Using the master method

According to $T(n)$, $a = 2$, $b = 2$, and $f(n) = n\log_2 n$

However, $f(n)$ cannot represent in term of $\Omega(n^{\log_b a + \varepsilon})$ ; $\varepsilon > 0$, as below

Evaluate $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$; $c < 1$

$\cancel{2} \cdot \frac{n}{\cancel{2}} \log_2 \frac{n}{2} \leq c \cdot n \log_2 n$

$\cancel{n \log_2 n} - n \leq c \cdot \cancel{n \log_2 n}$

$c \geq 1 - \frac{1}{\log_2 n}$ ; $n > 1$

For significantly large $n$, $c \nless 1$, then we cannot find such constant.

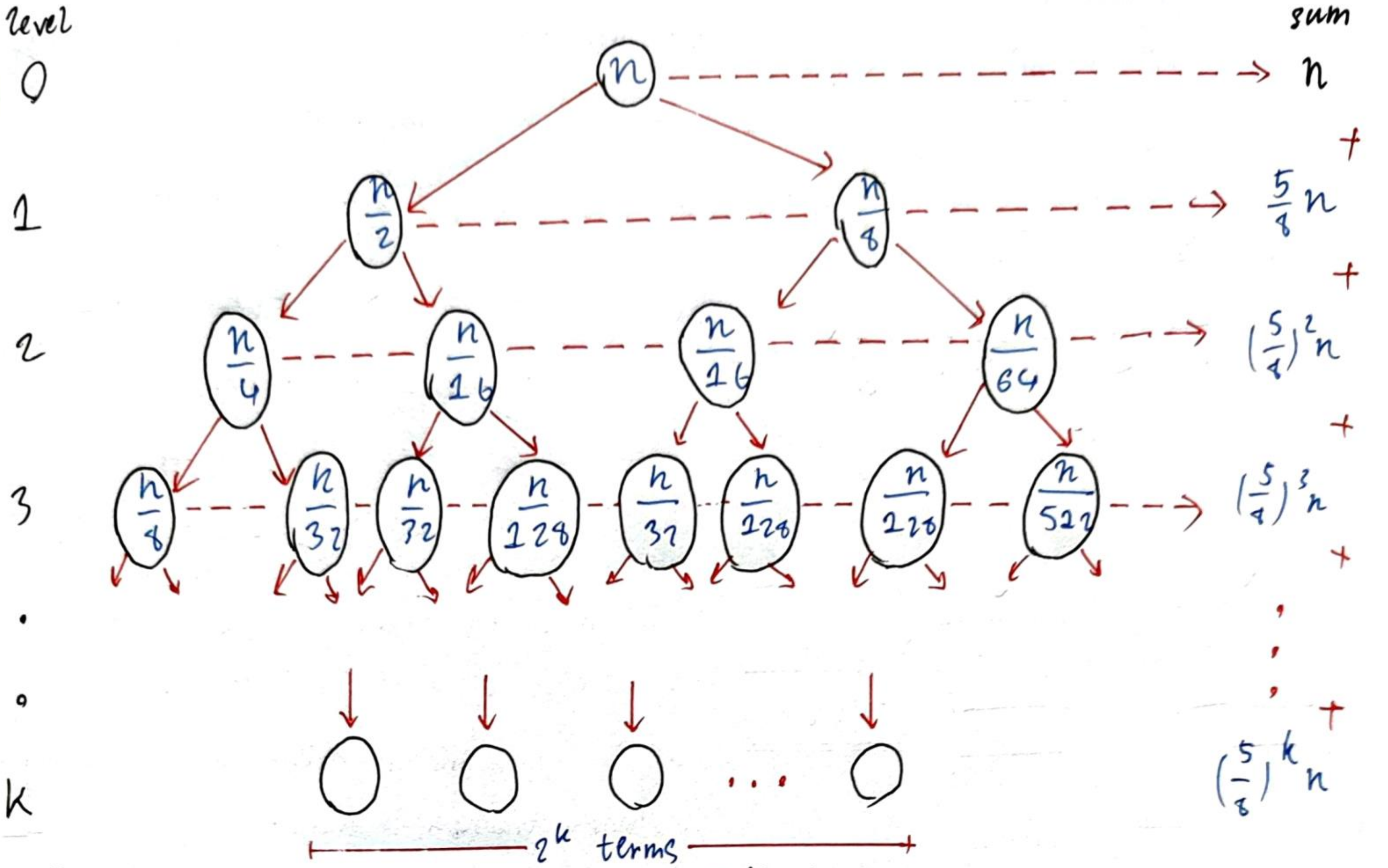
And $f(n)$ cannot represent as other cases of Master theorem as well.

$\therefore$ The Master method does not apply

นาย ศักย์ศิลกนล เพชรกุลจินดา
section 2 รหัส 630520600

7.) $T(n) = T(\frac{n}{2}) + T(\frac{n}{8}) + n$, $T(1) = C$

Using the recursion tree method




**Best-case :**

consider $\frac{n}{8^k} = 1 \rightarrow k = \log_8 n$

then $T(n) = \left(\sum_{i=1}^{k} (\frac{5}{8})^i\right) \cdot n + c \cdot 2^k = \frac{1 - \frac{5^{\log_8 n}}{8^{\log_8 n}}}{1 - \frac{5}{8}} \cdot n + c \cdot n$

$= \frac{8}{3}\left(n - n^{\log_8 5}\right) + cn = \Omega(n)$

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

**Worst-case :**

consider $\frac{n}{2^k} = 1 \rightarrow k = \log_2 n$

then $T(n) = \left(\sum_{i=1}^{k} (\frac{5}{8})^i\right) \cdot n + c \cdot 2^k = \frac{1 - \frac{5^{\log_2 n}}{8^{\log_2 n}}}{1 - \frac{5}{8}} \cdot n + cn$

$= \frac{8}{3}\left(n - \frac{n^{\log_2 5}}{n^2}\right) + cn = \frac{8}{3}n - \frac{8}{3}n^{\log_2 2.5} + cn$

$= O(n)$

Using the master method

Since $T(n)$ is not in the form corresponding to the Master theorem.

$\therefore$ The Master method does not apply.