01204212 Abstract Data Types and Problem Solving

Assignment #4: **Basic Complexity Analysis**

ข้อกำหนด

1. พัฒนาโปรแกรมตามที่เงื่อนไขโจทย์กำหนด
2. งานที่มอบหมายนี้เป็นงานเดี่ยว ขอให้นิสิตทำด้วยตนเอง
3. งานข้อ 1-3 ส่งคำตอบเป็น**ไฟล์นามสกุล .PDF** เท่านั้น ผ่าน Google Classroom (ไม่จำเป็นต้องลอกโจทย์ เขียนกระดาษแล้วสแกนหรือถ่ายรูปได้)
4. งานข้อ 4-5 ส่งเป็นโปรแกรมผ่านเซิร์ฟเวอร์
5. กำหนดส่งวัน**จันทร์ที่ 26 กรกฎาคม 2564** ก่อนเที่ยงคืน

1. พิจารณาฟังก์ชัน $A(n)$ และ $B(n)$ ดังตาราง จงระบุว่าฟังก์ชัน A เป็น $O, o, \Omega, \omega, \Theta$ ของฟังก์ชัน B หรือไม่ (Is $A(n) = O(B(n))$?) เมื่อกำหนดให้ k, ε, c เป็นค่าคงที่ โดย $k \geq 1, \varepsilon > 0$ และ $c > 1$ ให้เขียนคำตอบว่า "ใช่" หรือ "ไม่ใช่" ลงในแต่ละช่องตาราง พร้อมแสดงวิธีการพิสูจน์

	$A(n)$	$B(n)$	O	o	Ω	ω	Θ
1.1	$\log^k n$	n^ε	ใช่	ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่
1.2	n^k	c^n	ใช่	ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่
1.3	$\sqrt{n}$	$n^{\sin n}$	ไม่ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่
1.4	2^n	$2^{n/2}$	ไม่ใช่	ไม่ใช่	ใช่	ใช่	ไม่ใช่
1.5	$n^{\log m}$	$m^{\log n}$	ใช่	ไม่ใช่	ใช่	ไม่ใช่	ใช่
1.6	$\log(n!)$	$\log(n^n)$	ใช่	ไม่ใช่	ใช่	ไม่ใช่	ใช่

1.1) $\lim_{n\to\infty} \frac{\log^k n}{n^\varepsilon} = \lim_{n\to\infty} \left(\frac{\log n}{n^{\frac{\varepsilon}{k}}}\right)^k = \left(\lim_{n\to\infty} \frac{\log n}{n^{\frac{\varepsilon}{k}}}\right)^k$

เนื่องจาก $\lim_{n\to\infty} \frac{\log n}{n^{\frac{\varepsilon}{k}}} = 0$ $\therefore \left(\lim_{n\to\infty} \frac{\log n}{n^{\frac{\varepsilon}{k}}}\right)^k = 0^k = 0$

1.2) $\lim_{n\to\infty} \frac{n^k}{c^n} = 0$ (c^n โตเร็วกว่า n^k)

1.3) $\lim_{n\to\infty} \frac{\sqrt{n}}{n^{\sin n}} = \lim_{n\to\infty} n^{\frac{1}{2} - \sin n}$

เนื่องจาก
$$-1 \leq \sin n \leq 1$$
$$1 \geq -\sin n \geq -1$$
$$\frac{3}{2} \geq \frac{1}{2} - \sin n \geq -\frac{1}{2}$$

พิจารณา $\lim_{n\to\infty} n^{\frac{1}{2} - \sin n}$ บอกลิมิตไม่ได้ เนื่องจากค่ามันไปมาระหว่าง 0 ถึง ∞

1.4) $\lim_{n\to\infty} \frac{2^n}{2^{n/2}} = \lim_{n\to\infty} 2^{\frac{n}{2}} = \infty$

1.5) $\lim_{n\to\infty} \frac{n^{\log m}}{m^{\log n}} = \lim_{n\to\infty} \frac{m^{\log n}}{m^{\log n}} = 1$

1.6) $\lim_{n\to\infty} \frac{\log(n!)}{\log(n^n)}$

พิจารณา $\log(n!) = \log 1 + \log 2 + \log 3 + \dots + \log n = \sum_{i=1}^{n} \log i$

$$\sum_{i=1}^{n} \log i \approx \int_{i=1}^{n} \log i \, di = \left[i \log i - i \ln 10 \right]_1^n$$

$$= i \log i - i \ln 10 + \ln 10$$

$$\lim_{n\to\infty} \frac{\log(n!)}{n \log n} \approx \lim_{n\to\infty} \frac{n \log n - n \ln 10 + \ln 10}{n \log n} = \lim_{n\to\infty} 1 - \cancelto{0}{\frac{\ln 10}{\log n}} + \cancelto{0}{\frac{\ln 10}{\ln n}} = 1$$

2. พิจารณาโปรแกรมต่อไปนี้ แสดงวิธีการหา big-oh notation

2.1

```
#include <stdio.h>

long long euclidGCD(long long n, long long m) {
  long long t;
  while (m != 0) {
    t = m;
    m = n % m;
    n = t;
  }
  return n;
}

int main(void) {
  long long n, m;
  scanf("%lld %lld", &n, &m);
  if(n > m)
    printf("%lld\n", euclidGCD(m, n));
  else
    printf("%lld\n", euclidGCD(n, m));
  return 0;
}
```

มีค่ามาก มีค่าน้อย

10 2

10 2

$O(1)$

2 มีค่าน้อย 10 มีค่ามาก

$O(\log(\max(m,n)))$

ให้ (a_k, b_k) แทนค่า (n,m) ของ euclidGCD ทำงานรอบที่ k เป็นจำนวนนับใดๆ จากหลักการทำงานของ euclid

$a_k = cb_k + r_1$ จะเห็นว่า $a_{k+1} = b_k$ และ $a_k = cb_k + b_{k+1}$ โดยที่ $c \geq 1$

$a_{k+1} = cb_{k+1} + r_2$

เมื่อพิจารณาที่ worst case ของ euclidGCD จะพบว่า จำนวนรอบการทำงานมากสุดก็ต่อเมื่อ

1.) a_1 และ b_1 มี GCD = 1

2.) c ต้องเท่ากับ 1 เพื่อทำให้การลดลงของจำนวนรอบลดลงน้อยที่สุด นั่นคือ $a_k = b_k + b_{k+1}$ หรือ $1 < \frac{a_k}{b_k} < 2$

เมื่อสังเกต input จะพบว่า สอดคล้องกับ คุณสมบัติ ของ fibonucci 2 ตัวที่อยู่ติดกันใดๆ

(คุณสมบัติ fibonucci คือ $1 < \frac{F_n}{F_{n-1}} < 2$ และ $GCD(F_n, F_{n-1}) = 1$ ทุก n เป็นจำนวนนับ)

เราจะได้ว่า (n,m) หรือ $(a_1,b_1) = (F_n, F_{n-1})$ ทุก n เป็นจำนวนนับ และ $a_1 > b_1$

พิจารณา

$$\left.\begin{array}{c}(F_n, F_{n-1})\\ \downarrow \\ (F_{n-1}, F_{n-2}) \\ \downarrow \\ (F_{n-2}, F_{n-3}) \\ \downarrow \\ \vdots \\ (F_2, F_1)\end{array}\right\}$$

ทำงานทั้งหมด $n-1-1+1 = n-1$ รอบ

หาค่า n จาก; $fib_n = \dfrac{\left(\frac{1+\sqrt{5}}{2}\right)^n}{\sqrt{5}}$

$$\sqrt{5}\, fib_n = \left(\frac{1+\sqrt{5}}{2}\right)^n$$

$$n = \frac{\log(\sqrt{5}\, fib_n)}{\log\left(\frac{1+\sqrt{5}}{2}\right)}$$

แต่เนื่องจากตอนการทำงานฟังก์ชันครั้งแรกมีการให้ argument สลับตำแหน่งกับฟังก์ชันหลัก ทำให้การทำงานเพิ่มอีก 1 รอบ

∴ ในฟังก์ชัน $F_n = m$ จะได้ complexity $= O(\log m)$

พิจารณาทั้งโปรแกรม จะได้ complexity $= O(1) + O(\log m)$

ถ้ามอง m คือ $\max(m,n)$ (ที่เป็น input) จะได้ว่า $O(\log(\max(m,n))) \overset{\text{อยู่ใน}}{\leq} O(\log(m+n))$

∴ complexity $= O(\log(m+n))$

2.2

```c
#include <stdio.h>
#include <stdlib.h>

int isPalin(char* text, int beginPos, int endPos) {
  if(beginPos == endPos)
    return 1;
  else if(beginPos+1 == endPos)
    return text[beginPos] == text[endPos];
  else if(text[beginPos] != text[endPos])
    return 0;
  return isPalin(text, beginPos+1, endPos-1);
}

int main(void) {
  int length;
  char* text;
  scanf("%d", &length);
  text = (char*)malloc(sizeof(char)*(length+1));
  scanf("%s", text);
  printf("%d\n", isPalin(text, 0, length-1));
  return 0;
}
```

$O(1)$

ให้ length = l

ถ้า l เป็นเลขคู่

$\_\ \_\ \_ \cdots \_\ \_\ \_$ (1, 2, 3, … $l-2$, $l-1$, l) — $1 \Rightarrow O(1)$

$\_\ \_ \cdots \_\ \_$ (2, 3, … $l-2$, $l-1$) — $2 \Rightarrow O(1)$

⋮

$\_\ \_$ → beginPos+1 = endPos

ทั้งหมด $\frac{l}{2}$ รอบ

ถ้า l เป็นเลขคี่

$\_\ \_\ \_ \cdots \_\ \_\ \_$ (1, 2, 3, … $l-2$, $l-1$, l) — $1 \Rightarrow O(1)$

$\_\ \_ \cdots \_\ \_$ (2, 3, … $l-2$, $l-1$) — $2 \Rightarrow O(1)$

⋮

$\_\ \_\ \_$

$\_$ ← beginPos = endPos

ทั้งหมด $\boxed{\frac{l+1}{2} \text{ รอบ}}$ — $O(\text{length})$

ที่ worst case (เป็นเลขคี่) จะได้ complexity $O(1)$ $O(\text{length})$

$\therefore$ complexity ของโปรแกรมนี้ = $O(\text{length}) + O(1)$

complexity ของโปรแกรมนี้ = $O(\text{length})$

2.3

```
#include <stdio.h>

int P(int x, int n) {
  int y;
  if (n == 0)
    return 1;
  if (n%2 == 1) {
    y = P(x, (n-1)/2);
    return x*y*y;
  } else {
    y = P(x, n/2);
    return y*y;
  }
}

int main(void) {
  int x, n;
  scanf("%d %d", &x, &n);
  printf("%d\n", P(x, n));
  return 0;
}
```

จากฟังก์ชัน P(x, n) มีเรียกใช้ตัวเองทุกรอบ หารสองไม่เอาเศษไปเรื่อยๆ จนกว่า n จะเป็น 0 โดยที่ เมื่อเขียนเป็นแผนภาพได้ดังรูป

ให้ // แทนการหารไม่เอาเศษ   แต่เมื่อพิจารณาเป็นสมการจะได้ว่า

$$n \to n//2 \to n//4 \to n//6 \to \cdots \to 0$$

$$\left.\begin{array}{c} n \\ \downarrow \\ \frac{n}{2} \\ \downarrow \\ \frac{n}{4} \\ \downarrow \\ \frac{n}{6} \\ \downarrow \\ \vdots \\ \downarrow \\ C \end{array}\right\}$$

เมื่อ $C<1$

ให้ p เป็นจำนวนรอบจะได้

$$\frac{n}{2^p} = C \ ; C<1$$

$$p = \log_2\left(\frac{n}{c}\right)$$

$$= \log_2 n - \log_2 c$$

จะได้ว่า $O(p) = O(\log_2 n - \log_2 c) = O(\log n)$

$\therefore$ complexity $= O(\log n)$

2.4

```c
#include <stdio.h>

int main(void) {
  int m, n, p, q, c, d, k;
  int first[10][10], second[10][10], multiply[10][10];

  printf("Enter # of rows and columns of 1st matrix: ");
  scanf("%d %d", &m, &n);

  for (c=0; c<m; c++)
    for (d=0; d<n; d++) {
      printf("Enter matrix1[%d][%d]: ", c, d);
      scanf("%d", &first[c][d]);
    }

  printf("Enter # of rows and columns of 2nd matrix: ");
  scanf("%d %d", &p, &q);

  if (n != p) {
    printf("The multiplication isn't possible.\n");
  } else {
    for (c=0; c<p; c++)
      for (d=0; d<q; d++) {
        printf("Enter matrix2[%d][%d]: ", c, d);
        scanf("%d", &second[c][d]);
      }

    for (c=0; c<m; c++)
      for (d=0; d<q; d++) {
        multiply[c][d] = 0;
        for (k=0; k<p; k++)
          multiply[c][d] += first[c][k]*second[k][d];
      }

    printf("Product of the matrices:\n");
    for (c=0; c<m; c++) {
      for (d=0; d<q; d++)
        printf("%d\t", multiply[c][d]);
      printf("\n");
    }
  }
  return 0;
}
```

$O(1)$

$O(mn)$

$O(1)$

$O(pq)$

$O(mpq)$

$O(1) + O(mq)$

complexity $= O(1) + O(mn) + O(1) + O(pq) + O(mpq) + O(1) + O(mq)$

$= O(mpq)$

3. แสดงวิธีการหา big-oh notation ของ recurrence relation ต่อไปนี้ เมื่อกำหนดให้ $T(n)$ เป็นค่าคงที่สำหรับ $n \le 1$

3.1 $T(n) = T\left(\frac{9n}{10}\right) + n$

3.2 $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

3.3 $T(n) = T(\sqrt{n}) + 1$

3.4 $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

3.5 $T(n) = T(n-1) + \lg n$

3.1) $T(n) = T\left(\frac{9n}{10}\right) + n$

master theorem; $a = 1$, $\frac{n}{b} = \frac{9n}{10} \rightarrow b = \frac{10}{9}$, $f(n) = n$

พิจารณา; $n^{\log_{\frac{10}{9}} 1} = n^0 = 1$

$f(n) = n = \Omega(n^{\log_{\frac{10}{9}} 1 + \varepsilon})$, เมื่อ $0 < \varepsilon \le 1$

พิจารณา $af\left(\frac{n}{b}\right) \le cf(n)$ for some $c < 1$

$1 f\left(\frac{9n}{10}\right) \le cf(n)$

$1\left(\frac{9n}{10}\right) \le cn$, $c = \frac{9}{10}$ ซึ่ง $c < 1$

∴ ใช้ case 3 จะได้ว่า $T(n) = \Theta(f(n)) = \Theta(n)$

∴ $T(n) = O(n)$

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

master theorem; $a = 2$, $b = 4$, $f(n) = \sqrt{n}$

พิจารณา $n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$

$f(n) = \sqrt{n} = \Theta(n^{\log_b a}) = \Theta(\sqrt{n})$

∴ ใช้ case 2 จะได้ว่า $T(n) = \Theta(n^{\log_4 2} \log n) = \Theta(\sqrt{n} \log n)$

∴ $T(n) = O(\sqrt{n} \log n)$

3.3) $T(n) = T(\sqrt{n}) + 1$

ให้ $n = 2^m$ โดย $m = \lg n$ จะได้ $T(2^m) = T(2^{\frac{m}{2}}) + 1$

ให้ $T'(m) = T(2^m)$ จะได้ $T'(m) = T'(\frac{m}{2}) + 1$

master theorem, $a = 1$, $b = 2$, $f(m) = 1$

พิจารณา $m^{\log_b a} = m^{\log_2 1} = 1$

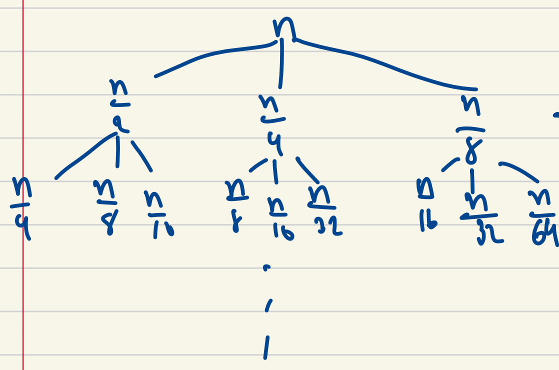
จะได้ว่า $f(m) = 1 = \Theta(m^{\log_b a}) = \Theta(1)$

จะเป็น case 2 จะได้ว่า $T'(m) = \Theta(1 \log m)$

จาก $T'(m) = T(2^m) = T(2^{\lg n}) = T(n)$ $\therefore T(n) = \Theta(\log(\lg(n)))$

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

3.4) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$



$\rightarrow n$

$\rightarrow$ workload $= \frac{n}{2} + \frac{n}{4} + \frac{n}{8} = \frac{7n}{8}$

$\rightarrow$ workload $= \frac{49}{64} n = \left(\frac{7}{8}\right)^2 n$

ทำแบบนี้ไปเรื่อยๆ

$2^l = n$

$l = \log_2 n$

พิจารณา $\sum_{i=0}^{l} \left(\frac{7}{8}\right)^i n = n\left(\frac{1\left(1 - \left(\frac{7}{8}\right)^{\log_2 n + 1}\right)}{1 - \frac{7}{8}}\right) = 8n\left(1 - \left(\frac{7}{8}\right)^{\log_2 n}\left(\frac{7}{8}\right)\right) = 8n - 7n\left(\frac{7}{8}\right)^{\log_2 n}$

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

3.5) $T(n) = T(n-1) + \lg n$

$T(n) = \lg(n) + \lg(n-1) + \lg(n-2)$

$= \lg(n!)$

$T(n) = O(\log(n!))$