

01204212 Abstract Data Types and Problem Solving

Assignment #4: Basic Complexity Analysis

ข้อกำหนด

- พัฒนาโปรแกรมตามที่เงื่อนไขกำหนด
 - งานที่มอบหมายนี้เป็นงานเดี่ยว ขอให้รับผิดชอบด้วยตนเอง
 - งานข้อ 1-3 ส่งคำตอบเป็นไฟล์นามสกุล .PDE เท่านั้น ผ่าน Google Classroom (ไม่จำเป็นต้องลอกโจทย์ เขียนกระดาษแล้วสแกน หรือถ่ายรูปได้)
 - งานข้อ 4-5 ส่งเป็นโปรแกรมผ่านเวิร์กโฟลเดอร์
 - กำหนดส่งวันจันทร์ที่ 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 $k \geq 1, \epsilon > 0$ 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 ทา $\lim_{n \rightarrow \infty} \frac{\log^k n}{n^\epsilon} = \lim_{n \rightarrow \infty} \frac{(\log n)^k}{n^\epsilon} = \lim_{n \rightarrow \infty} \left(\frac{\log n}{n^{\epsilon/k}} \right)^k = 0$ เมื่อ $\epsilon > 0$ หรือ $k \geq 1$


จะได้ $A(n) = O(B(n)) = o(B(n))$

1.2 ทา $\lim_{n \rightarrow \infty} \frac{n^k}{c^n} = 0$ เมื่อ $k \geq 1$ หรือ $c > 1$


จะได้ $A(n) = O(B(n)) = o(B(n))$

1.3 ท $\lim_{n \rightarrow \infty} \left(\frac{\sqrt{n}}{n \sin n} \right) = \lim_{n \rightarrow \infty} n^{\left(\frac{1}{2} - \sin n \right)}$

พิจารณา $\frac{1}{2} - \sin n$ ว่า $-1 \leq \sin n \leq 1$
 $-\frac{1}{2} \leq \frac{1}{2} - \sin n \leq \frac{3}{2}$


จะเห็นว่า $n^{\left(\frac{1}{2} - \sin n \right)}$ จะมีค่าที่สลับกันไปมา ไม่แน่นอนตามค่า $\sin n$ ได้ลักษณะกราฟเป็น 

∴ จึงสามารถสรุปได้ว่า $\lim_{n \rightarrow \infty} n^{\left(\frac{1}{2} - \sin n \right)}$ เป็นฟังก์ชันที่สลับไปมาในพ้อง 0 ถึง ∞ แม้ว่า n จะเข้าใกล้ ∞ ก็ตาม

ดังนั้น DATE จึงไม่สามารถหาความสัมพันธ์ระหว่าง $A(n)$ กับ $B(n)$ ได้ เพราะเป็นฟังก์ชันที่สลับไปมา 

1.4

ท $\lim_{n \rightarrow \infty} \frac{2^n}{2^{n/2}} = \lim_{n \rightarrow \infty} 2^{n - \frac{n}{2}} = \lim_{n \rightarrow \infty} 2^{\frac{n}{2}} = \infty$

∴ $A(n) = \Omega(B(n)) = \omega(B(n))$ 

1.5

ท $\lim_{n \rightarrow \infty} \frac{n^{\log m}}{m^{\log n}} = \lim_{n \rightarrow \infty} \frac{m^{\log n}}{n^{\log n}} = 1$

∴ $A(n) = O(B(n)) = \Theta(B(n)) = \Omega(B(n))$ 

↓ log ในข้อนี้ใช้หมายถึง log₂

1.6 ท $\lim_{n \rightarrow \infty} \frac{\log(n!)}{\log(n^n)}$ | พิจารณา $\ln(n!) = \ln(n \times (n-1) \times (n-2) \times \dots \times 2 \times 1) = \ln(n) + \ln(n-1) + \dots + \ln(1)$
 $= \sum_{i=1}^n \ln(i)$

$= \lim_{n \rightarrow \infty} \frac{\ln(n!)}{\ln(n^n)}$ | จากสูตรของ Stirling ซึ่ง $\sum_{i=1}^n \ln(i) \approx \int_1^n \ln(x) dx = [x \ln x - x]_1^n$
 $= n \ln n - n + 1$

$\lim_{n \rightarrow \infty} \frac{\log(n!)}{\log(n^n)} = \lim_{n \rightarrow \infty} \frac{\ln(n!)}{\ln(n^n)} = \lim_{n \rightarrow \infty} \frac{n \ln n - n + 1}{n \ln n} = \lim_{n \rightarrow \infty} \frac{n \ln n}{n \ln n} + \lim_{n \rightarrow \infty} \frac{-n}{n \ln n} + \lim_{n \rightarrow \infty} \frac{1}{n \ln n} = 1$

∴ $A(n) = O(B(n)) = \Theta(B(n)) = \Omega(B(n))$ 

1.5	$n^{\log m}$	$m^{\log n}$	ใช่	ใช่	ใช่	ใช่	ใช่
1.6	$\log(n!)$	$\log(n^n)$	ใช่	ใช่	ใช่	ใช่	ใช่

2. พิจารณาโปรแกรมต่อไปนี้ แสดงวิธีการหา big-oh notation

```

2.1  1: #include <stdio.h>
      2:
      3: long long euclidGCD(long long n, long long m) {
      4:     long long t;
      5:     while (m != 0) {
      6:         t = m;
      7:         m = n % m;
      8:         n = t;
      9:     }
     10:     return n;
     11: }
     12:
     13: int main(void) {
     14:     long long n, m;
     15:     scanf("%lld %lld", &n, &m);
     16:     if (n > m)
     17:         printf("%lld\n", euclidGCD(m, n));
     18:     else
     19:         printf("%lld\n", euclidGCD(n, m));
     20:     return 0;
     21: }
```

$n \quad m$

3 5

5 3

3 2

} $O(1)$

} $O(\text{euclidGCD}(\min(n,m), \max(n,m)))$

2.1 ถ้าพิจารณากรณีที่ค่าที่ส่งมา มาหา gcd ของเลขในลำดับ a, b (ให้ $a > b$)

จะพบค่าที่ส่งมา เมื่อ 1) a และ b มี gcd คือ 1

2) a และ b ไม่ลงตัวโดย $1 < \frac{a}{b} < 2$

ซึ่งจะพบ จำนวนที่เข้าเงื่อนไข คือ a และ b จะเป็นเลข Fibonacci 2 จำนวนที่ติดกัน (ให้ F_n แทนเลข Fibonacci ลำดับที่ n)

($1 < \frac{F_n}{F_{n-1}} < 2$, $\text{gcd}(F_n, F_{n-1}) = 1$)

พิจารณา $\text{euclidGCD}(a, b)$ เมื่อ $a > b$ แทนด้วยเลข Fibonacci ใช้ $\text{euclidGCD}(F_n, F_{n-1})$

ดูการไล่มา ดังนี้ จากขวาไป

(F_n, F_{n-1})

(F_{n-1}, F_{n-2})

(F_{n-2}, F_{n-3})

(F_2, F_1)

ซึ่งจะทำการลด $n-1$ ครั้ง

เราจะได้จาก $F_n = \frac{\phi^n}{\sqrt{5}}$ เมื่อ $\phi = \frac{1+\sqrt{5}}{2}$

ได้ $n = \log_\phi(F_n \sqrt{5})$ ดังนั้น

หรือ $n = \log_\phi(a \sqrt{5})$ ดังนั้น

* ในขั้นถัดไป ค่าจะลดลงเรื่อยๆ ถ้าค่าที่ใส่เข้าไป

ในตอนที่ลดนั้น a จะน้อยกว่า b

ถ้าไปเรื่อยๆ จะทำซ้ำสลับกัน 1 ครั้งแล้วทำ

จะทำการวนซ้ำจนกว่าจะจบ

เพื่อลดค่าลงเรื่อยๆ จนถึงค่าที่น้อยที่สุด

การวนซ้ำ 1 ครั้งนั้น

ถ้าพิจารณากรณีที่ค่าที่ส่งมาที่มากที่สุดกับที่น้อยที่สุด จะพบค่าที่ส่งมา a จะเป็นค่าที่มากที่สุดกับค่าที่น้อยที่สุด $a = \max(m, n)$

ดังนั้นจำนวนการวนซ้ำจะขึ้นกับค่าที่มากที่สุดกับค่าที่น้อยที่สุด $\log_\phi(\max(m, n) \cdot \sqrt{5})$

\therefore Complexity ของ program นี้ = $O(\log(\max(m, n)))$


```

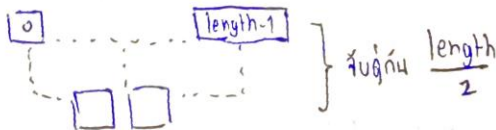
2.2 1: #include <stdio.h>
      2: #include <stdlib.h>
      3:
      4: int isPalin(char* text, int beginPos, int endPos) {
      5:     if(beginPos == endPos)
      6:         return 1;
      7:     else if(beginPos+1 == endPos)
      8:         return text[beginPos] == text[endPos];
      9:     else if(text[beginPos] != text[endPos])
      10:         return 0;
      11:     return isPalin(text, beginPos+1, endPos-1);
      12: }
      13:
      14: int main(void) {
      15:     int length;
      16:     char* text;
      17:     scanf("%d", &length);
      18:     text = (char*)malloc(sizeof(char)*(length+1));
      19:     scanf("%s", text);
      20:     printf("%d\n", isPalin(text, 0, length-1));
      21:     return 0;
      22: }
    
```

} $O(1)$

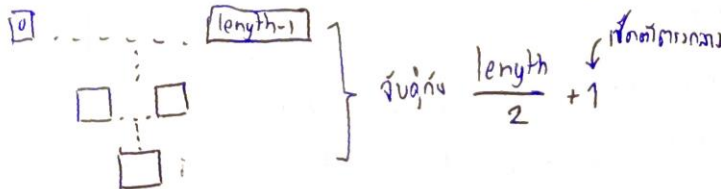
} $O(isPalin(text, 0, length-1))$

2.2 ทำการพิจารณา กรณีต่างๆ ที่เกิดขึ้นใน isPalin (...) จะพบว่า มีได้ 2 กรณี คือ

① กรณีที่ text ที่ใส่เข้ามา มีลักษณะเป็นเลขคู่ ซึ่งกรณี isPalin จะทำการดูตัวรวมที่ตัวสุดท้ายแล้วไปหาคู่จนครบ



② กรณีที่ text ที่ใส่เข้ามา มีลักษณะเป็นเลขคี่ ซึ่งกรณี isPalin จะทำการดูตัวรวมที่ตัวสุดท้ายแล้วไปหาคู่จนครบ แต่จะเหลือตัวที่ไม่คู่ที่ตัวกลาง



ดังนั้น กรณี isPalin (...) มี complexity คือ $O(length)$

ได้ Program มี complexity = $O(1) + O(length)$

∴ Program มี complexity = $O(length)$ ✓

```

1: #include <stdio.h>
2:
3: int P(int x, int n) {
4:     int y;
5:     if (n == 0)
6:         return 1;
7:     if (n%2 == 1) {
8:         y = P(x, (n-1)/2);
9:         return x*y*y;
10:    } else {
11:        y = P(x, n/2);
12:        return y*y;
13:    }
14: }
15:
16: int main(void) {
17:     int x, n;
18:     scanf("%d %d", &x, &n);
19:     printf("%d\n", P(x, n));
20:     return 0;
21: }

```

(2.3) กำหนดฟังก์ชัน $P(x, n)$ ซึ่งจะมีค่าเท่ากับ 1 เมื่อ n หาร x ลงตัว และ 2 (ไม่ลงตัว) เมื่อ n หาร x ลงตัว

ถ้า $n=0$ ฟังก์ชันจะเท่ากับ 1

จะได้ลักษณะคือ

(// = หารลงตัว)

ถ้า k หาร n ลงตัว $P(x, n)$ เท่ากับ 1
ถ้า $\frac{n}{2^k} = 1$ (กรณี 1 หารลงตัวเสมอ 2 หารลงตัวจนได้ 1 หารลงตัวเสมอ)
 $2^k = n$
 $k = \log n$
 $\therefore O(P(x, n)) = O(\log n)$

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

```

2.4 1: #include <stdio.h>
2:
3: int main(void) {
4:     int m, n, p, q, c, d, k;
5:     int first[10][10], second[10][10], multiply[10][10];
6:
7:     printf("Enter # of rows and columns of 1st matrix: "); } O(1)
8:     scanf("%d %d", &m, &n);
9:
10:    for (c=0; c<m; c++)
11:        for (d=0; d<n; d++) {
12:            printf("Enter matrix1[%d][%d]: ", c, d);
13:            scanf("%d", &first[c][d]);
14:        } } O(mn)
15:
16:    printf("Enter # of rows and columns of 2nd matrix: "); } O(1)
17:    scanf("%d %d", &p, &q);
18:
19:    if (n != p) {
20:        printf("The multiplication isn't possible.\n");
21:    } else {
22:        for (c=0; c<p; c++)
23:            for (d=0; d<q; d++) {
24:                printf("Enter matrix2[%d][%d]: ", c, d);
25:                scanf("%d", &second[c][d]);
26:            } } O(pq)
27:
28:        for (c=0; c<m; c++)
29:            for (d=0; d<q; d++) {
30:                multiply[c][d] = 0;
31:                for (k=0; k<p; k++)
32:                    multiply[c][d] += first[c][k]*second[k][d];
33:            } } + O(mqp)
34:
35:        printf("Product of the matrices:\n");
36:        for (c=0; c<m; c++) {
37:            for (d=0; d<q; d++)
38:                printf("%d\t", multiply[c][d]);
39:            printf("\n");
40:        } } + O(mq)
41:    }
42:    return 0;
43: }

```

หา complexity ที่คิดมาจน

ได้ $O(1) + O(mn) + O(1) + O(pq) + O(mqp) + O(mq)$

ดังนั้น จะได้

Complexity of this program

$= O(mqp)$

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

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 ใช้ Master theorem (โดยมองในรูปของ big-oh notation)

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

พิจารณา $n^{\log_b a + \epsilon} = n^{\log_{\frac{10}{9}} 1 + \epsilon} = n^\epsilon$

ได้ $f(n) = \Omega(n^{\log_b a + \epsilon})$

พิจารณา $af(\frac{n}{b}) \leq cf(n)$ เมื่อ $c < 1$

$$f(\frac{n}{10}) \leq cf(n)$$

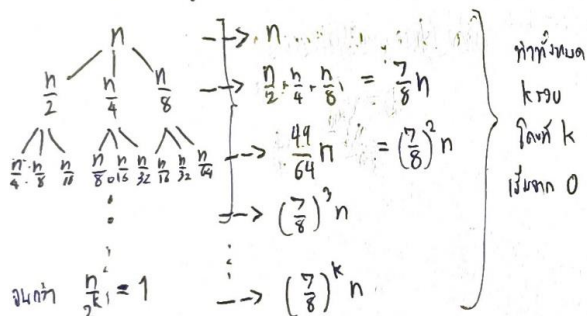
$$\frac{n}{10} \leq cn$$

$$\frac{1}{10} \leq c < 1 \text{ เป็นจริง}$$

ดังนั้น $T(n) = O(n)$

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

วาดแผนภาพ



จาก $\frac{n}{2^k} = 1$; ได้ $k = \log(n)$ รอบ

$$T(n) = \sum_{i=0}^{\log(n)} (\frac{7}{8})^i n = n \sum_{i=0}^{\log(n)} (\frac{7}{8})^i$$

เนื่องจาก $\sum_{i=0}^{\log(n)} (\frac{7}{8})^i$ มีค่าใกล้ 1 เมื่อ $i \rightarrow \infty$

จึงได้ $T(n) = n$

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

3.2 ใช้ Master theorem (โดยมองในรูปของ big-oh notation)

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

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

ได้ $f(n) = O(n^{\log_b a} \log n)$

$$f(n) = O(\sqrt{n} \log n)$$

ดังนั้น $T(n) = O(\sqrt{n} \log n)$

3.3 $T(n) = T(\sqrt{n}) + 1$ (ใช้ Master theorem ของ big-oh)

ให้ $n = 2^k$ โดย $k = \log(n)$

ได้ $T(2^k) = T(2^{\frac{k}{2}}) + 1$

ให้ $T(2^k) = S(k)$

ได้ $S(k) = S(\frac{k}{2}) + 1$

$$a = 1, b = 2, f(k) = 1$$

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

ได้ $f(k) = O(k^{\log_b a} \log k)$

$$f(k) = O(\log k) = S(k)$$

จาก $S(k) = T(2^k) = T(n) = O(\log k)$

ดังนั้น $T(n) = O(\log(\log(n)))$

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

วาดแผนภาพ

