

ชื่อ-นามสกุล.....เลขประจำตัว.....ตอนเรียนที่.....เลขที่นั่ง CR58.....

หมายเหตุ

1. ข้อสอบมีทั้งหมด 9 ข้อ ในกระดาษคำถามคำตอบ 7 หน้า
2. ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ
3. ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ
4. ห้ามการหยิบยื่นสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยื่นให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับ สัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่า
นิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และ
อาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

*** รวมนรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยไม่ได้รับการช่วยเหลือ
หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....

วันที่.....

1. (10 คะแนน) จงวิเคราะห์ประสิทธิภาพเชิงเวลาของขั้นตอนวิธีแต่ละข้อดังต่อไปนี้

สำหรับบางข้อที่สามารถใช้ Master Theorem ได้ กำหนดให้ Master Theorem มีนิยามคือ

สำหรับ Recurrence relation $T(n) = aT(n/b) + f(n)$ เมื่อ $a \geq 1$ และ $b > 1$ เป็นค่าคงที่ และ $f(n)$ เป็นฟังก์ชันที่มีค่าเป็นบวกเสมอ เราสามารถคำนวณ $T(n)$ ได้ตามกรณีต่าง ๆ ดังต่อไปนี้

$T(n) = \Theta(n^{\log_b(a)})$	ถ้า $f(n) = O(n^{\log_b(a)-\epsilon})$
$T(n) = \Theta(n^{\log_b(a)} \log^{k+1} n)$	ถ้า $f(n) = \Theta(n^{\log_b(a)} \log^k(n))$ และ $k \geq 0$
$T(n) = \Theta(f(n))$	ถ้า $f(n) = \Omega(n^{\log_b(a)+\epsilon})$ และ $\epsilon > 0$ และ $af(n/b) \leq kf(n)$ เมื่อมีค่าคงที่ $k < 1$ และ n ขนาดใหญ่มาก

และให้ถือว่า $T(1) = 1$ เสมอ

ข้อ		Big O หรือ Θ (...)
1.	<pre>sum = 0 for(int i=n; i>0; i/=2) { for(int j=1; j<n; j*=2) { for(int k=0; k<n; k+=2) { sum++; } } }</pre> <p>$\log_2 n$ $\log_2 n$ $\left(\frac{n}{2}\right)$ $n \log_2^2 n$ $\frac{\log^2 n}{\log n}$</p>	$\Theta(n \log^2 n)$
2.	<pre>void c2110327() { for (int i=1; i<=n; i++) for (int j=1; j<=log(i); j++) printf("2110327"); }</pre> <p>n $\log(i)$ $\log(1) + \log(2) + \dots + \log(n)$ $n \rightarrow \frac{n}{2} \quad T(n) = T(\frac{n}{2}) + O(n)$</p>	$\Theta(\log(n!))$
3.	<pre>A(n) { if(n<=1) return 1; else return A(sqrt(n)); // sqrt(n) is square root of n }</pre> <p>$T(n) = 1 + T(\sqrt{n})$ $T(\frac{n}{2}) = T(\frac{n}{4}) + O(1)$ $n \quad n^{\frac{1}{2}} \quad n^{\frac{1}{4}} \quad n^{\frac{1}{8}} \quad T(1) \rightarrow O(1)$</p>	$O(1)$
4.	<pre>for(i = 0; i < n; i++) { for(j = 0; j < n; j++) a[j] = randomValue(i); // randomValue is O(1) goodSort(a); // goodSort is O(n log n) }</pre> <p>$O(1)(n)$ $O(n^2 \log n)$</p>	$O(n^2 \log n)$
5.	การเรียงลำดับข้อมูลของ Quicksort เมื่อใช้ pivot เป็นค่า mean ที่ใช้เวลาหา $O(n)$.
6.	$T(n) = T(n-1) + O(1)$ $a \quad b \quad d$	$O(n)$
7.	$T(n) = 3T(n/2) + n$ $c = \log_b a = \log_2 3$	$\Theta(n^{\log_2 3})$
8.	$T(n) = 64T(n/8) + n^2 \log n$ $2 \log 64 = 8$	$\Theta(n^2 \log^2 n)$
9.	$T(n) = T(n/2) + n^{(2 - \cos n)}$ $\log_2 n$ $\left(\frac{n}{2}\right)^2$ $(\log_2 n) \left(\frac{n}{2}\right)^{2 - \cos n}$ $\frac{2 - \cos n}{2 - \cos \frac{n}{2}}$ $n^3 \left(\frac{n}{2}\right)^3$	$O(n^3)$
10.	$T(n) = 16T(n/4) + n!$ $T(n/4) = \frac{n!}{4!} + \frac{n!}{16}$	

2. (5 คะแนน) สำหรับปัญหา Matrix Chain Multiplication กำหนดให้เราคำนวณจำนวนครั้งในการคูณน้อยสุดการหาผลคูณของ matrix จำนวน 6 matrix คือ A_1, A_2, \dots, A_6 เข้าด้วยกัน โดยมีตารางค่าคำตอบ M ดังต่อไปนี้ (กำหนดให้ $M[i][j]$ คือ จำนวนครั้งการคูณน้อยสุดของการคูณ matrix A_i ถึง matrix A_j จงระบุวิธีการใส่วงเล็บที่ทำให้ได้จำนวนครั้งการคูณน้อยสุด

	j=1	j=2	j=3	j=4	j=5	j=6
i = 1	0	750	200	230	370	720
i = 2		0	50	70	170	470
i = 3			0	10	70	320
i = 4				20	220	
i = 5					0	400
i = 6						0

กำหนดให้ขนาดของ แต่ละ matrix เป็นดังนี้

Matrix	A1	A2	A3	A4	A5	A6
จำนวนแถว	15	10	5	1	2	10
จำนวนคอลัมน์	10	5	1	2	10	20

$$470 + 15 \times 10 \times 20 = 420 + 15 \times 1 \times 20 = 300$$

การคูณ matrix เหล่านี้ทำได้โดยใช้จำนวนครั้งในการคูณเลขน้อยสุด ด้วยการใส่วงเล็บดังนี้ $(A_1(A_2 A_3))(A_4 A_5) A_6$

(ให้ตอบโดยการใส่วงเล็บ ตัวอย่างเช่น $((A_1 A_2)(A_3(A_4 A_5)))$)

3. (5 คะแนน) ปัญหาการทอนเหรียญเป็นดังนี้ เราต้องการจ่ายเงิน n บาท ด้วยเหรียญต่าง ๆ ให้ใช้จำนวนเหรียญน้อยที่สุด เรามีเหรียญอยู่ k แบบ แต่ละแบบมีค่าแตกต่างกันคือ $v[1..k]$ โดยเรามีเหรียญแต่ละเหรียญไม่จำกัด และรับประกันว่า $v[1] = 1$ เสมอ เช่น ถ้า $n = 11$ และ $v = [1, 3, 4, 6]$ เราจะใช้สามเหรียญในการจ่ายเงิน 11 บาท (คือเหรียญ 4 + 4 + 3) กำหนดให้ $C(a, b)$ คือจำนวนเหรียญน้อยสุดที่ใช้ในการทอนเงิน b บาท เมื่อพิจารณาเฉพาะเหรียญ $v[1..a]$ เราจะได้ว่า $C(a, b)$ มี recurrence relation ดังนี้

$$C(a, b) = b \quad ; \text{ if } (a == 1 \parallel b == 0)$$

$$C(a, b) = C(a-1, b) \quad ; \text{ if } (a > 1 \ \&\& \ b < v[a])$$

$$C(a, b) = \min(C(a, b - v[a]) + 1, C(a-1, b)) \quad ; \text{ if } (a > 1 \ \&\& \ b \geq v[a])$$

จงเติมตารางสำหรับค่า $C(a, b)$ ด้านล่างนี้ เมื่อกำหนดให้ v มีค่าดังนี้ $v = [1, 3, 4, 7, 12]$

a\b	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	1	2	1	2	3	2	3	4	3	4	5	4	5	6	5	6	7	
3																		
4																		
5																		

4. (5 คะแนน) จาก Recurrence Relation ต่อไปนี้ จงระบุว่า หากเราเขียนโปรแกรมเพื่อคำนวณ Relation ดังกล่าวด้วยวิธีการแบบ Dynamic Programming แล้ว เราจะต้องใช้ตารางขนาดกี่มิติในการเก็บข้อมูล และการคำนวณนั้นทำได้เร็วสุดมีประสิทธิภาพเชิงเวลาเป็นเท่าไร

Recurrence Relation	ขนาดของตาราง (มิติ)	ประสิทธิภาพเชิงเวลา (ตอบเป็น O หรือ Θ)
(ตัวอย่าง) $F(n) = F(n-1) + F(n-2)$	1	$O(n)$
$B(n, k) = k * B(n-1, k) + B(n-1, k-1)$		
$C(n) = \max(C(1), C(2), \dots, C(n-1))$		
$D(n, b, c) = D(n-1, b-1, c-1) + D(n-1, b-1, c+1) + D(n-1, b+1, c+1) + D(n-1, b+1, c-1)$		
$E(n) = E(n/2) + a[n]$ // ให้ $a[n]$ เป็นอาเรย์ 1 มิติ		
$G(n, k) = \max(G(n-1, 1) + a[1], G(n-1, 2) + a[2], \dots, G(n-1, k-1) + a[k-1])$		

5. (10 คะแนน) ปัญหาเชิญคนมางานปาร์ตี้ เป็นดังนี้ มีแขกทั้งหมด n คน กำกับด้วยหมายเลข 1 ถึง n เราต้องการเชิญคนเหล่านี้งานปาร์ตี้ อย่างไรก็ตาม แขกบางคนไม่ถูกกัน ให้ $\text{dislike}(a, b)$ เป็นฟังก์ชันที่คืนค่า true เมื่อแขก a และ b นั้นเกลียดกัน แขกที่เราเชิญมา จะต้องไม่มีคู่ไหนเลยที่เกลียดกัน

- 5.1. จงเขียน Recurrence relation สำหรับการคำนวณหาจำนวนคนมากที่สุดที่เราสามารถเชิญมางานปาร์ตี้ได้ กำหนดให้ $\text{MaxGuest}(i, S)$ คือ จำนวนคนที่สามารถเชิญมาร่วมงานได้มากที่สุด เมื่อเราพิจารณาเฉพาะคนหมายเลขที่ 1 ถึง i โดยที่ S คือคนหมายเลข $i+1$ ถึง n ที่เราเลือกเชิญมาก่อนแล้ว และ S ไม่มีคนที่เกลียดกันเลย ซึ่งคำตอบที่เราต้องการคือ $\text{MaxGuest}(n, \{ \})$

// กรณีพื้นฐาน (trivial case)

$\text{MaxGuest}(0, S) =$

// ทัวไป (trivial case)

$\text{MaxGuest}(i, S) =$

- 5.2. กำหนดให้ $\text{prefer}(a)$ คือค่าที่บอกว่าแขก a นั้นเป็นที่ชื่นชอบขนาดไหน เราต้องการให้ผลรวมของค่า prefer ของแขกที่เชิญทั้งหมดนั้นมากที่สุด กำหนดให้ $\text{Best}(i, S)$ คือค่ามากที่สุดของผลรวมของค่า prefer เมื่อเราพิจารณาเฉพาะคนหมายเลขที่ 1 ถึง i โดยที่ S คือคนหมายเลข $i+1$ ถึง n ที่เราเลือกเชิญมาก่อนแล้ว และ S ไม่มีคนที่เกลียดกันเลย ซึ่งคำตอบที่เราต้องการคือ $\text{Best}(n, \{ \})$ จงเขียน Recurrence relation นี้

// กรณีพื้นฐาน (trivial case)

$\text{Best}(0, S) =$

// ทัวไป (trivial case)

$\text{Best}(i, S) =$

สำหรับข้อ 6 – 9 นั้น จะเป็นการออกแบบอัลกอริทึม ในแต่ละข้อนั้นสามารถอธิบายอัลกอริทึมที่ออกแบบด้วย รหัสเทียม (pseudo code) หรือว่า programming language ภาษาใดที่เคยเรียนมาก็ได้ แต่ทุกข้อให้ระบุประสิทธิภาพเชิงเวลาด้วย

6. (10 คะแนน) มีอาร์เรย์ $A[1..5][1..n]$ ซึ่งเป็นอาร์เรย์ 2 มิติขนาด 5 แถว n คอลัมน์ เราต้องเลือกข้อมูลบางตัวมาจากอาร์เรย์นี้ โดยต้องเลือกข้อมูล 1 ตัวมาจากแต่ละคอลัมน์ โดยมีกฎดังนี้ 1) ต้องเลือกทีละตัว จาก คอลัมน์ 1 ไปยังคอลัมน์ n 2) ในคอลัมน์ที่ 1 จะเลือกข้อมูลตัวใดก็ได้ 3) ถ้าคอลัมน์ที่ i เลือกข้อมูลจากแถวที่ j แล้วในคอลัมน์ที่ $i+1$ สามารถเลือกได้เฉพาะแถวที่ $j-1, j, j+1$ (เมื่อมีแถวดังกล่าวอยู่) เท่านั้น เราต้องการเลือกให้ได้ผลรวมมากที่สุด จากกฎดังกล่าว หากเรากำหนดให้ $B(r, c)$ คือผลรวมมากที่สุดของข้อมูลที่เลือก เมื่อเราได้ทำการเลือกจนถึงการเลือกข้อมูลแถวที่ r จากคอลัมน์ที่ c เราจะได้ Recurrence relation ของ $B(r, c)$ ดังนี้

$B(r, c) = A[r][c] \quad ; \text{ if } (c == 1)$

$B(r, c) = \max(B(r, c-1), B(r+1, c-1)) + A[r][c] \quad ; \text{ if } (c > 1 \ \&\& \ r == 1)$

$B(r, c) = \max(B(r-1, c-1), B(r, c-1), B(r+1, c-1)) + A[r][c] \quad ; \text{ if } (c > 1 \ \&\& \ r > 1 \ \&\& \ r < 5)$

$B(r, c) = \max(B(r-1, c-1), B(r, c-1), \quad) + A[r][c] \quad ; \text{ if } (c > 1 \ \&\& \ r == 5)$

จงเขียนรหัสเทียมในการคำนวณค่า B ใด ๆ โดยใช้วิธี Dynamic Programming แบบ bottom up และให้เก็บผลลัพธ์ลงในตัวแปร B

Solve(n, A[1..5][1..n])

ยกตัวอย่างประกอบ

return B; // ต้องคืนค่า B ซึ่งเป็นตาราง 2 มิติกลับมา

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ _____

7. (10 คะแนน) กำหนดให้มีอาร์เรย์ $A[1..n]$ โดยที่ค่าใน a มีเฉพาะตัวเลข 0, 1, 2 เท่านั้น และ a ถูกเรียงจากน้อยไปมากแล้ว เราต้องการทราบว่าในอาร์เรย์ a มีตัวเลข 1 อยู่กี่ตัว (รับประกันว่า a มีขนาดมากกว่า 3 ช่อง และมีตัวเลข 0, 1, 2 อย่างน้อยเลขละ 1 ตัวแน่นอน) ตัวอย่างเช่น $A = [0, 1, 1, 2, 2, 2]$ จะต้องตอบว่า 2 เพราะมี 1 อยู่ 2 ตัว จงออกแบบอัลกอริทึมสำหรับแก้ปัญหาดังกล่าว

Count(A, n)

ยกตัวอย่างประกอบ

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ _____

8. (10 คะแนน) เราต้องการนับจำนวน string ความยาว n ที่ประกอบด้วยตัวเลข 0 หรือ 1 เท่านั้นที่มีเงื่อนไขคือ string นั้นจะต้องไม่มีเลข 1 ติดกัน เช่น เมื่อ n เป็น 3 จะมี string ที่ตรงกับเงื่อนไขคือ “000”, “001”, “010”, “100” และ “101” เท่านั้น รวมทั้งหมด 5 แบบ จงออกแบบอัลกอริทึมที่นับจำนวน string ดังกล่าวเมื่อระบุค่า n เป็น input ด้วยวิธี Divide & Conquer หรือ Dynamic Programming
- 8.1. จงกำหนดนิยามของ Recurrence relation ของปัญหานี้ โดยให้นิยาม function ที่ใช้ พร้อมทั้ง parameter ที่เกี่ยวข้อง

- 8.2. จงเขียนสมการของ Recurrence relation ในข้อ 8.1

- 8.3. จงออกแบบอัลกอริทึมสำหรับปัญหานี้ (ไม่จำเป็นต้องทำตามข้อ 8.1 หรือ 8.2 ก็ได้ ขอให้ทำงานได้ถูกต้อง)

ยกตัวอย่างประกอบ

9. (10 คะแนน) เกมตำรวจจับโจรเป็นดังนี้ มีโจรและตำรวจเข้าแถวเป็นเส้นตรงอยู่ เราสามารถแทนตำแหน่งของโจรและตำรวจด้วย อาร์เรย์ $A[1..n]$ โดยที่ $A[i]$ มีค่าเป็น 0 หมายความว่าตำแหน่งที่ i มีโจรอยู่ แต่ถ้า $A[i]$ มีค่าเป็น 1 แสดงว่าตำแหน่ง i มีตำรวจยืนอยู่ ตำรวจแต่ละคนสามารถจับโจรได้คนเดียวเท่านั้น และตำรวจ ที่อยู่ ณ ตำแหน่ง i สามารถจับโจรที่อยู่ตำแหน่งตั้งแต่ $i-k$ ถึง $i+k$ ได้เท่านั้น เราอยากทราบว่าจาก A ที่กำหนดให้ มีโจรโดนจับมากที่สุดกี่คน ตัวอย่างเช่น $A = [0, 1, 1, 1, 0, 0, 0]$ และ $k = 1$ จะจับโจรได้สองคน คือ โจรที่ตำแหน่ง 1 และ 5 (ตำรวจ ณ ตำแหน่ง 3 ไม่สามารถจับโจรได้เลย) หรือให้ $A = [1, 0, 0, 0, 1, 1, 1, 0, 0]$ และ $k = 2$ จะสามารถจับโจรได้มากที่สุด 4 คน จงออกแบบอัลกอริทึมสำหรับแก้ไขปัญหานี้

Catch($A[1..n]$, k)

```

int count = 0;
for i : 1 ; i ≤ n ; i++
    if (A[i] == 1) {
        start = max(0, i-k);
        stop = min(n-1, i+k);
        for (j : start ; j ≤ stop ; j++) {
            if (A[j] == 0) {
                A[j] = 1;
                count++;
                break;
            }
        }
    }

return count;

```

ยกตัวอย่างประกอบ