FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110327 ALGORITHM DESIGN
Year II, Second Semester, Final Examination, Mar 12, 2021 13:00-16:00

ชื่อ-นามสกุล.....เลขประจำตัว.....ตอนเรียนที่.....เลขที่ใน CR58.....

<u>หมายเหตุ</u>

1. ข้อสอบมีทั้งหมด 10 ข้อ ในกระดาษคำถามคำตอบ 8 หน้า
2. **ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ**
3. **ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ**
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....
วันที่.....

- ใช้ดินสอเขียนคำตอบได้
- ให้เขียนเลขที่ในใบเซ็นชื่อเข้าสอบทุกหน้า
- หากพื้นที่สำหรับเขียนคำตอบไม่เพียงพอ ให้เขียนไว้ด้านหลังของหน้านั้น ห้ามเขียนข้ามไปหน้าอื่น และให้ระบุไว้ในพื้นที่สำหรับเขียนคำตอบว่า “มีต่อด้านหลัง”

1. (5 คะแนน) จงวิเคราะห์ประสิทธิภาพในการทำงานโดยให้ระบุสัญกรเชิงเส้นกำกับที่ถูกต้องเหมาะสม จาก Recurrence Relation ที่ระบุเวลาในการทำงานในแต่ละข้อต่อไปนี้ โดยทุกข้อกำหนดให้ T(n) = 1 เมื่อ n<= 1

ข้อย่อย	Recurrence Relation	สัญกรเชิงเส้นกำกับ
1	$T(n) = 8T(n/8) + n^3$	
2	$T(n) = 2T(n-2) + n$	
3	$T(n) = nT(n-1) + 1$	
4	$T(n) = 4T(n/2) + n^3$	
5	$T(n) = 4T(n/2) + n^2$	

2. (5 คะแนน) ส่วนของโปรแกรมต่อไปนี้พยายามที่จะเรียงข้อมูลในอาเรย์ A[1..n] แต่มีจุดที่ทำงานผิดอยู่ จงอธิบายว่า 1) จุดที่ผิดคือจุดใดบ้าง 2) ผิดอย่างไร และ 3) ต้องแก้ไขจุดดังกล่าวอย่างไรให้ทำงานได้ถูกต้อง

ส่วนของโปรแกรม	คำตอบ
1: def my_sort(A[1..n]) 2: if (n > 1) 3: x = A[n] 4: i = n - 1 5: while (A[i] > x) 6: A[i+1] = A[i] 7: i = i - 1 8: end 9: A[i] = x 10: else 11: my_sort(A[1..n-1]) 12: end 13: end	else return A[1..n]

3. (6 คะแนน) จาก Recurrence Relation ต่อไปนี้จงวาด Recursion Tree โดยให้ระบุปริมาณภาระจริงที่เกิดขึ้นของแต่ละปมไว้ในปมดังกล่าว โดยต้นไม้ที่เขียนขึ้นจะต้องมีความลึกไม่น้อยกว่า 3 ชั้น พร้อมทั้งให้ระบุผลรวมของภาระจริงที่เกิดขึ้นในแต่ละชั้นด้วย

ข้อย่อย	Recurrence Relation	Recursion Tree
1	$T(n) = 2T(n/2) + n^2$	
2	$T(n) = 3T(n/2) + \log n$	

4. (8 คะแนน) จาก Recurrence Relation ต่อไปนี้ จงเขียนโปรแกรม (ด้วยภาษาใดที่เคยเรียนมาก็ได้) เพื่อคำนวณค่าของ Recurrence Relation ที่กำหนดให้ โดยใช้วิธีการแบบ Dynamic Programming ในรูปแบบ Bottom Up พร้อมทั้งระบุประสิทธิภาพในการทำงาน

ข้อย่อย	Recurrence Relation	Code

1	$DP(i,j) = \begin{cases} i & ;i = j \\ DP(i+1,j) + DP(i,j-1) & ;i < j \end{cases}$
	ต้องการคำนวณ DP(1,n)
	เมื่อกำหนดให้มี n เป็นข้อมูลนำเข้า
		DP(i,j){
		int table[n+1][n+1];
		for(int i=1;i≤n;i++){
		table[i][i]=i;
		}
		ประสิทธิภาพ $O(n^2)$
		int plus = 1;
		while(plus<n){
		for(int i=1; i≤n-plus; i++){
		table[i][i+plus]=table[i+1][i+plus]+table[i][i+plus-1]
		}
		}
		return table[1][n];

2	$L(i,j,k) = \begin{cases} 0 & ;k = 0 \\ \infty & ;i < 0 \\ \infty & ;j < 0 \\ \min\limits_{\substack{i-w\le a\le i,\\ j-w\le b\le j}} L(a,b,k-1) + A[i][j] & ;k > 0 \end{cases}$
	ต้องการคำนวณ L(p, q, n)
	เมื่อกำหนดให้มี w, A[][], p, q, n เป็นข้อมูลนำเข้า
	รับประกันว่า A[][] มีค่าเป็นบวกในทุกช่อง และมีค่าไม่เกิน 1,000
		L(p,q,n)
		int table[p+1][q+1];
		for(int i=1; i≤p; i++){
		for(int j=1;j≤q;j++)
		table[i][j]=1;

5. (5 คะแนน) จากโจทย์ปัญหา 0-1 Knapsack Problem จงยกตัวอย่าง Problem Instance ที่มีของ 5 ชิ้นพอดี ที่แต่ละชิ้นมีน้ำหนักแตกต่างกันทั้งหมด และมีมูลค่าแตกต่างกันทั้งหมด พร้อมด้วยขีดจำกัดของถุงที่ไม่เกิน 16 ที่ทำให้มีคำตอบที่ดีที่สุด ไม่น้อยกว่า 3 คำตอบที่แตกต่างกัน โดยให้เขียนมูลค่าและน้ำหนักของของแต่ละชิ้น รวมถึงขีดจำกัดของถุง พร้อมทั้งเขียนตาราง K[a][b] โดยให้ K[a][b] คือ ผลรวมที่ดีที่สุดของมูลค่าของของที่เลือกเมื่อพิจารณาเฉพาะของชิ้นที่ 1 ถึง a และขีดจำกัดของถุงคือ b

5.1 (2 คะแนน) เติมคำตอบในช่องว่างต่อไปนี้

ของชิ้นที่	1	2	3	4	5
มูลค่า	2	5	7	8	15
น้ำหนัก	2	5	7	8	15

ขีดจำกัดของถุง (W) = 15
มูลค่าของคำตอบที่ดีที่สุด = 15
คำตอบที่ 1 เลือกของหมายเลข = 5
คำตอบที่ 2 เลือกของหมายเลข = 3 4
คำตอบที่ 3 เลือกของหมายเลข = 1 2 4

5.2 (3 คะแนน) เขียนตาราง K[a][b] ในช่องว่างด้านล่างนี้ (เติมเฉพาะค่า b ที่ไม่เกิน W ที่ตอบในข้อที่แล้ว

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
2	0	0	2	2	2	5	6	7	7	7	7	7	7	7	7	7	
3	0	0	2	2	2	5	5	7	7	9	9	9	12	12	14	14	
4	0	0	2	2	2	5	5	7	8	9	10	10	12	13	13	15	
5	0	0	2	2	2	5	5	7	8	9	10	10	12	13	13	15	

- **สำหรับข้อที่ 6 เป็นต้นไป เป็นการออกแบบอัลกอริทึม ในแต่ละข้อสามารถตอบโดยการอธิบายอัลกอริทึม โดยใช้รหัสเทียม (Pseudocode) หรือ programming language ภาษาใดที่เคยเรียนมาก็ได้ และต้องวิเคราะห์ประสิทธิภาพในการทำงานของอัลกอริทึมด้วย**
- **คะแนนที่ได้จะแปรตามประสิทธิภาพในการทำงาน**

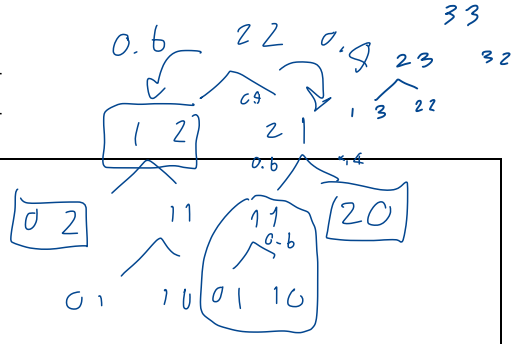
6. (10 คะแนน) หลังจากที่งานฟุตบอลประเพณีจุฬา-ธรรมศาสตร์ 2564 ถูกงดไปเนื่องจากสถานการณ์ COVID-19 เพื่อหาผู้ชนะในปีนี้ จึงได้ตกลงกันว่าจะจัดการแข่งขันแบบไม่มีผู้ชม โดยให้มีการแข่งกันระหว่างจุฬา-ธรรมศาสตร์ขึ้นเป็นจำนวน 2n-1 ครั้ง โดยทีมที่ชนะ n ครั้งได้ก่อนจะถือว่าเป็นผู้ชนะในปี 2564

เราต้องการทราบความน่าจะเป็นที่จุฬาจะเป็นผู้ชนะในปีนี้ โดยกำหนดให้ในการแข่งกันแค่ละครั้งนั้น จุฬามีโอกาสที่จะชนะเป็น p และมีโอกาสที่จะแพ้เป็น 1-p (ในการแข่งแต่ละครั้งจะไม่มีการเสมอกันเนื่องจากจะเตะลูกโทษกันจนกว่าจะรู้ผู้ชนะ) กำหนดให้ c(i,j) เป็นความน่าจะเป็นที่จุฬาจะชนะการแข่งขันถ้าจุฬายังต้องชนะอีก i รอบ และธรรมศาสตร์ยังต้องชนะอีก j รอบธรรมศาสตร์ถึงจะชนะการแข่งขัน โดยสิ่งที่จะต้องคำนวนคือ c(n, n)

6.1. (1 คะแนน) จงระบุ c(0,j) เมื่อ 0 < j <= n c(0,j) = 1

6.2. (1 คะแนน) จงระบุ c(i,0) เมื่อ 0 < i <= n c(i,0) = 0

6.3. (4 คะแนน) จงระบุความสัมพันธ์เวียนบังเกิดของ c(i,j) เมื่อ 0 < i,j <= n



c(i,j) = $p(i-1,j) + (1-p)(i,j-1)$

6.4. (4 คะแนน) จงออกแบบอัลกอริทึม calculate_win(n, p) ซึ่งจะต้องคืนค่า c(n,n) พร้อมทั้งวิเคราะห์ประสิทธิภาพในการทำงาน

```
int dp[i+1][j+1];

calculate_win(n,p){
    return c(n,n,p);
}
c(i,j,p){
    if(i==0) return 1;
    if(j==0) return 0;
    if(dp[i][j] >0) return dp[i][j]
    dp[i][j] = p*(i-1,j) + (1-p)*(i,j-1);
    return dp[i][j]
```

7. (10 คะแนน) มีหินอยู่ n ก้อน (กำกับด้วยหมายเลข 1 ถึง n) โดยทุกก้อนมีน้ำหนักเท่ากันหมดยกเว้นหินหนึ่งก้อนที่มีน้ำหนักต่างจากหินก้อนอื่น (อาจจะเบากว่าหรือหนักกว่าก็ได้) เราสามารถเรียกใช้ฟังก์ชัน w(i,j) ซึ่งจะคืนผลลัพธ์เป็นน้ำหนักรวมของหินหมายเลข i ถึงหมายเลข j มาให้ โดยการเรียก w(i,j) นั้นมีเงื่อนไขคือ 1 <= i <= j <= n

จงออกแบบอัลกอริทึม get_special_index(n) ซึ่งจะต้องคืนค่าหมายเลขหินที่มีน้ำหนักต่างไป โดยใช้จำนวนครั้งในการเรียก w(i,j) ให้น้อยที่สุด ในข้อนี้ รับประกันว่า $n = 3^k$ โดยที่ k เป็นจำนวนเต็มบวก พร้อมทั้งวิเคราะห์ประสิทธิภาพในการทำงาน

```
get_special_index(n) {
    int plus = n/3;
    int first = w(1, plus);
    int second = w(plus+1, 2plus);
    int third = w(2plus+1, 3plus);
    if (first = second) {
      return find_idx(2plus+1, 3plus, first/plus);
    }
    else if (second = third) {
      return find_idx(1, plus, second/plus);
    }
    else { return find_idx(plus+1, 2plus, first/plus); }
}

find_idx(int start, int stop, int want) {
    if (start == stop) return start;
    int m = (start+stop)/2;
    if ( w(start, m)/(m-start+1) ≠ want )
        return find_idx(start, m, want);
    else return find_idx(m+1, stop);
}
```

ประสิทธิภาพ คือ O(log n)

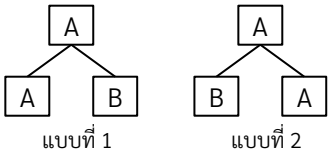
8. (10 คะแนน) กำหนดให้เรามีตัวแปร x อยู่ ในตอนเริ่มต้นนั้น x มีค่าเป็น 1 เราต้องการทำให้ x มีค่าเป็นจำนวนเต็ม n อย่างไรก็ตาม สิ่งที่เราสามารถกระทำกับ x ได้มีเพียง 2 อย่าง คือ 1) x = x +1 (การเพิ่ม x ด้วย 1) หรือ 2) x = x * 2 (การทำให้ x มีค่าเพิ่มขึ้นเป็น 2 เท่า)

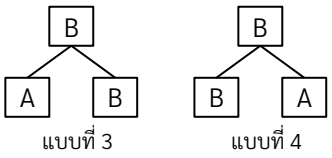
8.1. (5 คะแนน) จงออกแบบอัลกอริทึม make(n) เพื่อทำให้ x กลายเป็น n โดยใช้จำนวนการกระทำ 1) และ 2) รวมกันน้อยที่สุด พร้อมทั้งวิเคราะห์ประสิทธิภาพในการทำงาน ในอัลกอริทึมนี้ เราต้องเรียกใช้ฟังก์ชัน increase_x() เพื่อเพิ่มค่า x ด้วย 1 หรือ เรียกใช้ double_x() เพื่อทำให้ x มีค่าเป็นสองเท่า

```
make(n) {
    while(x*2 ≤ n){
        double_x();
    }
    while(x+1 ≤ n){
        increase_x();
    }
}
```

8.2. (5 คะแนน) จงให้เหตุผลว่า ทำไมอัลกอริทึมที่เสนอมาจึงใช้จำนวนการกระทำน้อยที่สุด

9. (10 คะแนน) การแข่งขันกีฬาแบบทัวร์นาเมนต์แพ้คัดออกเป็นดังนี้ มีผู้เข้าแข่งขัน n = 2^k คน (กำกับด้วยหมายเลข 1 ถึง n) และจะมีการแข่งขันเป็นรอบจำนวน k รอบ (กำกับด้วยหมายเลข 1 ถึง k) ในรอบที่ 1 จะมีการแข่งขัน n/2 ครั้ง (กำกับด้วยหมายเลข 1 ถึง n/2) โดยการแข่งครั้งที่ i จะเป็นการแข่งขันของผู้เข้าแข่งขันหมายเลข 2i-1 กับหมายเลข 2i ในการแข่งขันแต่ละครั้งจะมีผู้แพ้และผู้ชนะ โดยผู้ชนะจะได้เข้ารอบถัดไป ในรอบที่ 2 ผู้เข้าแข่งขันจะเหลืออยู่ n/2 คน ซึ่งคนเหล่านี้จะทำการแข่งในรอบที่ 2 ต่อไป

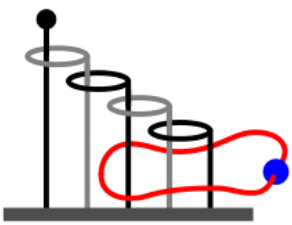




การแข่งขันในรอบที่ j (สำหรับ 2 <= j <= k) นั้น จะมีการแข่งขัน n/(2^j) ครั้ง โดยการแข่งครั้งที่ i ในรอบที่ j นั้นจะเป็นการแข่งขันของผู้ชนะจากการแข่งขันครั้งที่ 2i-1 กับ ครั้งที่ 2i ในรอบที่ j-1 ผลการแข่งขันแบบทัวร์นาเมนต์แพ้คัดออกสามารถแสดงได้ด้วยแผนภูมิต้นไม้ โดยใบของต้นไม้จะระบุผู้เข้าแข่งขัน ส่วนปมภายในแทนการแข่งขันแต่ละครั้งและระบุผู้ชนะไว้ที่ปม ผู้ชนะเลิศของทัวร์นาเมนต์จะอยู่ที่ปมรากของต้นไม้นั่นเอง รูปด้านขวานี้แสดงตัวอย่างของแผนภูมิทั้งหมดที่เป็นไปได้เมื่อ n = 2

จงออกแบบอัลกอริทึม tournament(k) ซึ่งจะต้องคืนค่าจำนวนรูปแบบของแผนภูมิต้นไม้แสดงผลลัพธ์การแข่งขันที่เป็นไปได้ทั้งหมด ของผู้เข้าแข่งขัน n = 2^k คน โดยที่แต่ละคนมีชื่อแตกต่างกันทั้งหมด โดยให้ถือว่าแผนภูมิสองแบบจะแตกต่างกันถ้าหากมีอย่างน้อย 1 ปมที่มีชื่อที่ระบุไว้ในปมนั้นแตกต่างกัน พร้อมทั้งวิเคราะห์ประสิทธิภาพในการทำงาน

10. (10 คะแนน) เกม Baguenaudier เป็นของเล่นแบบหนึ่งซึ่งประกอบด้วยเชือกต่อเป็นวงจำนวน 1 วง และห่วงโลหะจำนวน n ห่วง โดยเชือกจะร้อยอยู่กับห่วงเหล่านั้น ดังรูปด้านขวา เป้าหมายของเกมคือการเอาวงของเชือกออกจากห่วงเหล่านั้นให้ได้ โดยในตอนเริ่มต้นเชือกจะร้อยอยู่กับห่วงทุกห่วง



เพื่อความสะดวก เราสามารถพิจารณาเกมนี้เป็นตัวเลขฐาน 2 จำนวน n บิตได้ โดยให้บิตที่ i มีค่าเป็น 1 ก็ต่อเมื่อเชือกนั้นยังร้อยอยู่กับห่วงที่ i ดังนั้น (ให้บิตขวาสุดคือบิตที่ 1) ตอนเริ่มต้น สถานะของเกมสามารถระบุด้วยเลขฐานสองที่มีค่าเป็น 1 ทุกหลัก (111...1) และเป้าหมายของเกมคือทำให้เลขฐาน 2 นี้กลายเป็น 0 ในทุกหลัก (000...0) ในการเล่นเกมนี้ เรามี “ท่า” ที่สามารถกระทำกับเชือกได้อยู่ 2 “ท่า” คือ

a. การสลับค่าในบิตขวาสุด (จาก 1 เป็น 0 หรือจาก 0 เป็น 1 ก็ได้) ซึ่งเราสามารถทำท่านี้เมื่อไรก็ได้ ตัวอย่างเช่น เราสามารถทำ 11111 ให้กลายเป็น 1111**0** หรือ จาก 11010 ให้กลายเป็น 1101**1** ได้

b. การสลับค่าเฉพาะบิตที่ k + 2 (จาก 1 เป็น 0 หรือจาก 0 เป็น 1 ก็ได้) โดย k คือ จำนวนเลข 0 ที่อยู่ติดกันด้านท้าย(ขวาสุด) ของเลขฐานสอง ตัวอย่างเช่น เราสามารถทำ 11100 ให้กลายเป็น 1**0**100 ได้ หรือ จาก 001000 ให้กลายเป็น 0**1**1000 ได้ ขอให้สังเกตว่า ด้วยท่านี้ เราสามารถทำ 11111 ให้กลายเป็น 111**0**1 ได้เช่นเดียวกัน เนื่องจาก 11111 นั้นมี 0 อยู่ด้านท้ายเป็นจำนวน 0 ตัวพอดี นอกจากนี้ ให้สังเกตว่า เราไม่สามารถใช้ท่านี้เพื่อแปลง 11100 ให้เป็น 111**1**0 ได้ หรือ 11**0**00 เพราะ 11100 นั้นมี 0 อยู่ด้านท้าย 2 ตัวพอดี ดังนั้น การใช้ท่านี้จะสลับบิตที่ 4 ได้เท่านั้น

เราสามารถทำตามเป้าหมายของเกมนี้ได้โดยใช้ท่าเพียงสองท่านี้ ตัวอย่างเช่น สมมติให้ n = 3 เราสามารถแปลง 111 ให้กลายเป็น 000 ได้ตามลำดับดังต่อไปนี้ 111 a.–> 110 b.–> 010 a.–> 011 b.–> 001 a.–> 000 หรืออีกตัวอย่างหนึ่งเมื่อ n = 4 คือ 1111 b.–> 1101 a.–> 1100 b.–> 0100 a.–> 0101 b.–> 0111 a.–> 0110 b.–> 0010 a.–> 0011 b.–> 0001 a.–> 0000

จงออกแบบอัลกอริทึม bag(n) สำหรับเล่นเกมนี้ โดยเริ่มจากเลขฐานสองขนาด n bit ที่มีค่าเป็น 1 ในทุกบิต ในอัลกอริทึมนี้ เราต้องเรียกใช้ฟังก์ชัน perform_a() และ perform_b() เพื่อทำท่า a. หรือ b. พร้อมทั้งวิเคราะห์ประสิทธิภาพในการทำงาน

```
bag(n){ int count = n;
        int table[n];
        while (count != 0) {

}
```