FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110327 Algorithm Design
YEAR III, First Semester, Mid-term Examination, July 29, 2010, Time 8:30 – 11:30

ชื่อ-นามสกุล_____เลขประจำตัว | | | | | | | | | 2 | 1 | CR58_____

หมายเหตุ

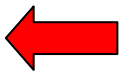
1. ข้อสอบมีทั้งหมด 6 ข้อในกระดาษคำถามคำตอบจำนวน 2 แผ่น 3 หน้า คะแนนเต็ม 60 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
6. ผู้ที่ประสงค์จะออกจากห้องสอบก่อนหมดเวลาสอบ แต่ต้องไม่น้อยกว่า 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. ผู้ที่ปฏิบัติเข้าข่ายทุจริตในการสอบ ตามประกาศคณะวิศวกรรมศาสตร์
   มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

รับทราบ

ลงชื่อนิสิต (……..……)

หมายเหตุ (เพิ่มเติม)

1. ข้อใดที่ให้ออกแบบอัลกอริทึมนั้น นิสิตสามารถตอบโดยเขียนบรรยายแนวคิดที่ implement ได้ในทางปฏิบัติ หรือจะเขียนเป็นรหัสเทียมประกอบแนวคิดที่นำเสนอด้วยก็ได้ และ**ต้องวิเคราะห์ประสิทธิภาพเชิงเวลาของอัลกอริทึมที่นำเสนอด้วย**
2. ต้องแสดงวิธีทำทุกข้อ การเขียนคำตอบเพียงอย่างเดียวจะไม่มีคะแนนให้ (ยกเว้นว่าจะเขียนในคำสั่ง)
3. ให้เขียนตอบในสมุดคำตอบ โดยเขียน**ตอบข้อที่ *k* ไว้ที่หน้าที่ *2k-1* และ *2k* ในสมุดคำตอบ** (k = 1,..,6)



1.1 (5 คะแนน) ข้อย่อยต่อไปนี้ เป็นจริงหรือเท็จ อธิบายเหตุผลประกอบสั้น ๆ ด้วย

- $n \log n = O(n\sqrt{n})$
- $1/n = O(\log n)$
- $n^2(1+\sqrt{n}) = O(n^2 \log n)$
- $n! = \Omega(2^n)$
- The solution to the recurrence $T(n) = 100T(n/99) + \log(n!)$ is $T(n) = \Theta(n \log n)$
- In the worst case, merge sort runs in $O(n^2)$ time.
- The sum of the smallest $\sqrt{n}$ elements in an unsorted array of distinct numbers can be found in $O(n)$

1.2 (5 คะแนน) กำหนดให้ปัญหาการหาค่าต่ำสุดและสูงสุดของอาเรย์ A ขนาด n ช่องสามารถทำได้โดยฟังก์ชันภาษา C ดังต่อไปนี้

```
void FMM(int* A,int p,int q,int &min,int &max) {
    if (p == q) { min = max = A[p]; }
    else if (p + 1 == q) {
        if (A[p] < A[q]) { min = A[p]; max = A[q]; }
        else  { max = A[p]; min = A[q]; }
    } else {
        int m = (p + q) / 2;
        int max_l,max_r,min_l,min_r;
        FMM(A,p,m,min_l,max_l);
        FMM(A,m+1,q,min_r,max_r);
        if (min_l < min_r) min = min_l; else min = min_r;
        if (max_l > max_r) max = max_l; else max = max_r;
    }
}
```

จงเขียน Recursion Tree เมื่อมีการเรียกฟังก์ชัน `FMM(A,0,9,min,max)` สำหรับอาเรย์ขนาด 10 ช่องโดยให้แต่ละปมใน Recursion Tree นั้นระบุถึงค่า `p` และ `q` ในการเรียกฟังก์ชันแต่ละครั้ง กำหนดให้ปมแรกสุดคือ `FMM(0,9)`

2. (10 คะแนน) ให้ $A = \{a_1, a_2, \ldots, a_n\}$ จงออกแบบอัลกอริทึมแบบ divide and conquer เพื่อหาว่า มีข้อมูลใดใน A ที่มีปรากฏซ้ำกันใน A เป็นจำนวนเกินครึ่งหรือไม่ (เรียกกันว่าเป็นตัวหมู่มากหรือ majority) เช่น $\{1,1,2,1,3,1,4\}$ มี 1 เป็นตัวหมู่มาก แต่ $\{1,2,2,1,0,1,1,0\}$ ไม่มีตัวหมู่มาก ทั้งนี้อนุญาตให้นำ a_k ใด ๆ มาเปรียบเทียบความเท่ากันได้เท่านั้น ห้ามเปรียบเทียบแบบมากกว่า น้อยกว่า (ซึ่งรวมการห้ามนำไป sort) ให้ออกแบบอัลกอริทึมที่ใช้เวลาทำงานเป็น $O(n \log n)$ หากต้องการคะแนนอีก 20% ของคะแนนเต็มของข้อนี้ ให้ออกแบบเป็นอัลกอริทึมที่ใช้เวลาเป็น $O(n)$

3. (10 คะแนน) Hanamard matrices $H_0, H_1, H_2, \ldots$ มีนิยามดังนี้

H_0 คือ 1 x 1 matrix [1]

สำหรับ $k > 0$, H_k คือ $2^k \times 2^k$ matrix

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$ ตัวอย่างเช่น $H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ และ $H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$

จงออกแบบอัลกอริทึมในการหาค่าของ $H_k v$ เมื่อ v เป็น matrix ขนาด $2^k \times 1$ โดยอัลกอริทึมดังกล่าวควรจะใช้เวลาการทำงานเป็น $O(n \log n)$ เมื่อกำหนดให้ $n = 2^k$ (ให้คิดว่าการคูณและการบวกเลขใด ๆ นั้นใช้เวลาคงที่)

4. (10 คะแนน) ให้ A คืออาเรย์ของจำนวนเต็มขนาด n ตัว ที่มีตัวซ้ำกันมากพอควรโดยมีข้อมูลที่ไม่เหมือนกันเลยใน A เป็นจำนวน $O(\log n)$ ตัว จงออกแบบอัลกอริทึมที่สามารถเรียงลำดับข้อมูลใน A ได้ในเวลา $O(n \log \log n)$

5. (10 คะแนน) กำหนดให้ปัญหาการหาจัตุรัสใหญ่สุดใน binary image เป็นดังนี้ มีตารางสองมิติขนาด $n \times n$ ช่องของตัวเลข 1 หรือตัวเลข 0 ให้หาบริเวณรูปสี่เหลี่ยมจัตุรัสที่ใหญ่ที่สุดใน A ที่ทุก ๆ ช่องในบริเวณนั้นเป็นตัวเลข 1 ทั้งหมด โดยให้ตอบเป็นความกว้างของจัตุรัสที่ใหญ่ที่สุด กำหนดให้ A เป็น Array ของเลขจำนวนเต็มที่เก็บตารางดังกล่าว โดยที่ $A[0][0]$ เก็บมุมบนซ้ายของตาราง และกำหนดให้ SQ(a, b) เป็นฟังก์ชันที่หาความกว้างของจตุรัสใหญ่ที่สุดที่มีมุมซ้ายบนอยู่ที่ (a,b) เมื่อพิจารณาเฉพาะตารางตั้งแต่ช่องที่มีค่า x ในช่วง $a \le x < n$ และมีค่า y ในช่วง $b \le y < n$ เราสามารถเขียน Recurrent Relation ของฟังก์ชัน SQ ได้ดังนี้

$$\mathrm{SQ}(x,y) = \begin{cases} 1 + \min(\mathrm{SQ}(x+1,y), \mathrm{SQ}(x+1,y+1), \mathrm{SQ}(x,y+1)) & ; A[x][y] = 1 \\ 0 & ; A[x][y] = 0 \\ 0 & ; x \ge n \text{ หรือ } y \ge n \end{cases}$$

จงเขียนฟังก์ชันภาษา C ที่ใช้วิธี Dynamic Programming เพื่อคำนวณค่าSQ(0,0) โดยให้เขียนสองฟังก์ชัน ฟังก์ชันแรกเป็นแบบ top-down memoization และฟังก์ชันที่สองเป็น dynamic แบบไม่มี recursive (แบบ bottom-up) กำหนดให้ใช้โครงฟังก์ชันดังนี้

```
int SQ(int A[][], int n, int x, int y) {
  // A คืออาเรย์ขนาด n x n ช่องที่ประกอบด้วยเลข 0,1
  // ฟังก์ชันนี้ต้องคืนค่าความกว้างจตุรัสที่ใหญ่ที่สุดใน A
  // เมื่อพิจารณาตั้งแต่ช่อง A[x][y] ถึง A[n-1][n-1]
}
```

6. (10 คะแนน) กำหนดให้ปัญหาคะแนนเกม pachinko มากที่สุดเป็นดังนี้ กำหนดให้มีหมุดปักเรียงเป็นรูปสามเหลี่ยมดังรูปด้านล่าง โดยหมุดจะเรียงตัวเป็นชั้นจำนวน n ชั้นและจำนวนหมุดในแต่ละชั้นจะเริ่มจาก 1, 2, 3, … จนครบ n ชั้น กำหนดให้หมุด pin(p, q) คือหมุดลำดับที่ q จากซ้ายมือในชั้นที่ p จากด้านบน ผู้เล่นจะทิ้งลูกเหล็กจากยอดบนสุดของ

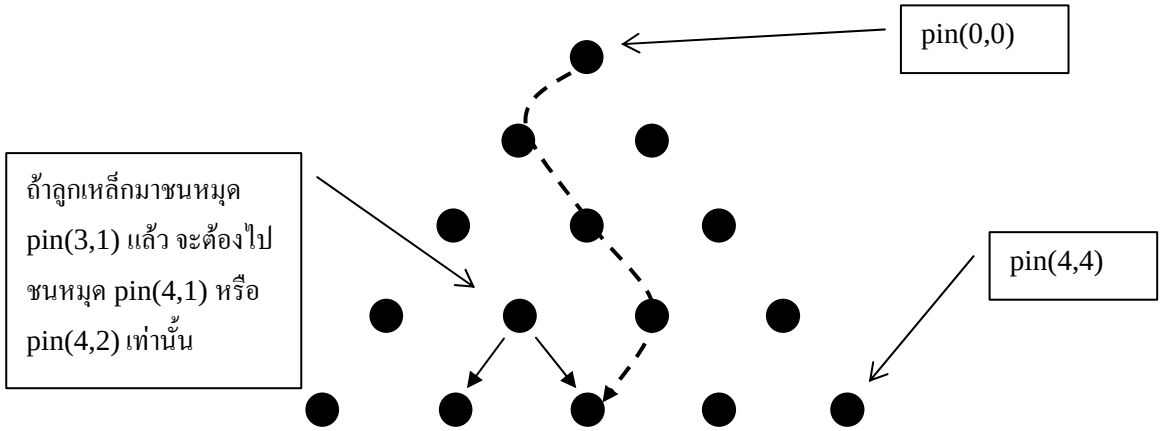
สามเหลี่ยม (หมุด pin(0, 0)) เมื่อลูกเหล็กกระทบหมุดใด ก็จะได้คะแนนตามที่กำหนดไว้ในหมุดดังกล่าว แล้วลูกเหล็กจะไหลลงมายังชั้นถัดไป ซึ่งลูกเหล็กจะต้องชนกับหมุดใดหมุดหนึ่งในสองหมุดที่อยู่ติดกับหมุดในชั้นที่แล้วเท่านั้น เราจะอธิบายได้ว่า เมื่อลูกเหล็กชนกับหมุด pin(p, q) แล้ว ลูกเหล็กจะต้องไหลไปชนกับหมุด pin(p+1, q) หรือ pin(p+1, q+1) เท่านั้น เกมจะจบเมื่อลูกเหล็กกลิ้งผ่านชั้นที่ n โดยที่คะแนนที่ผู้เล่นได้จะได้เท่ากับผลรวมของหมุดในแต่ละชั้นที่ลูกเหล็กกลิ้งไปชน

กำหนดให้ A เป็นอาเรย์สองมิติที่เก็บเลขจำนวนเต็มที่บอกคะแนนของหมุดแต่ละหมุด โดยที่ $A[p][q]$ จะเป็นคะแนนของหมุด pin(p, q) จงออกแบบอัลกอริทึมแบบ Dyanmic Programming เพื่อหาว่า คะแนนมากที่สุดที่เป็นไปได้ในการทิ้งลูกเหล็กหนึ่งลูกนั้นเป็นเท่าใด โดย

ก) จงเขียน Recurrent Relation พร้อมทั้งระบุนิยามของ Recurrent Relation ดังกล่าวรวมถึงกรณีเริ่มต้นด้วย

ข) จงเขียนโปรแกรมภาษา C สำหรับแก้ปัญหาดังกล่าวโดยใช้โครงฟังก์ชันดังนี้

```
int maxpin(int A[][], int n) {
  // A คืออาเรย์ขนาด n x n
  // ฟังก์ชันนี้ต้องคืนค่าคะแนนมากที่สุดที่เป็นไปได้เมื่อเริ่มทิ้งลูกเหล็กที่หมุด pin(0,0)
}
```



เส้นประคือทิศทางการกลิ้งของลูกเหล็กที่เป็นไปได้แบบหนึ่ง โดยจะได้คะแนนเท่ากับ A[0][0] + A[1][0] + A[2][1] + A[3][2] + A[4][2]