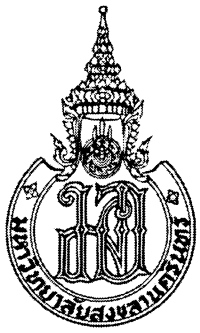
ชื่อ _____ รหัส _____ section ___



สอบกลางภาค: ภาคการศึกษาที่ 2 **ปีการศึกษา**: 2556

วันสอบ: 7 มกราคม 2557 **เวลาสอบ**: 13.30 – 16.30 น.

ห้องสอบ: R200, R201

ผู้สอน: อ.เสกสรรค์ สุวรรณมณี อ.อัมรินทร์ ดีมะการ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

รหัสและชื่อวิชา: 242-310 Introduction to Algorithm and Complexity

แนะนำขั้นตอนวิธีและความซับซ้อน

ทุจริตในการสอบมีโทษขั้นต่ำคือ ปรับตกในรายวิชาที่ทุจริตและพักการเรียน 1 ภาคการศึกษา

คำสั่ง: อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต: เครื่องเขียนต่างๆ เช่น ปากกา หรือดินสอ เข้าห้องสอบ และ กระดาษขนาด A4 หนึ่งแผ่น จดบันทึกด้วยลายมือเท่านั้น (ห้าม print หรือ ถ่ายเอกสาร)

ไม่อนุญาต: หนังสือ หรือเครื่องคิดเลขเข้าห้องสอบ และ**เอกสารใดๆ เข้าและออกห้องสอบ**

เวลา: 3 ชั่วโมง (180 นาที)

คำแนะนำ

- ☐ ข้อสอบมี 14 **หน้า** (*รวมหน้าปก*) แบ่งออกเป็น 4 **ตอน** คะแนนรวม 105 คะแนน (คิดเป็นคะแนนเก็บ 35%)
- ☐ **<u>เขียนคำตอบในข้อสอบ คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด</u>**
- ☐ อ่านคำสั่งในแต่ละข้อให้เข้าใจก่อนลงมือทำ
- ☐ เวลาที่ใช้เวลาทำตอนให้เหมาะสม ตามคำแนะนำ
- ☐ หากข้อใดเขียนคำตอบไม่พอ ให้เขียนเพิ่มที่ด้านหลังของหน้านั้นเท่านั้น

ตอน	**1** (21) 7%	**2** (30) 10%	**3** (30) 10%	**4** (24) 8%	**รวม** (105) 35%
คะแนน					

นักศึกษารับทราบ ลงชื่อ .....

ตอนที่ 1 (21 คะแนน, 7%, 30 นาที)

<u>บทนำ พื้นฐานอัลกอริทึม</u>

1. จงอธิบายความหมายและคุณสมบัติของขั้นตอนวิธี หรือ อัลกอริทึม (algorithm) มาพอสังเขป (5 คะแนน)

2. จงบรรยายปัญหาเชิงคำนวณต่อไปนี้ ด้วยลักษณะของข้อมูลขาเข้า (input) ผลลัพธ์ที่ต้องการ (output) ตัวอย่างปัญหา (problem instance) และ output ของตัวอย่างปัญหา และตัวกำหนดขนาดของปัญหา (problem size) เลือกทำข้อใดข้อหนึ่งเพียงข้อเดียว (8 คะแนน)

2.1 ผลรวมแนวทแยงมุมหลักของเมตริกซ์จัตุรัส (Sum of main diagonal elements of a square matrix)

2.2 หาค่าตัวหมู่มาก (majority) ของกลุ่มข้อมูลจำนวนเต็มบวก ที่มีขอบเขตข้อมูลจาก 1 ถึง N โดยข้อมูลมีการจัดเก็บเป็นอาเรย์ A[1..N] ค่าข้อมูล A[x] หมายถึงความถี่ของข้อมูล x , $1 \le x \le N$, หากกลุ่มข้อมูลนี้ไม่มีตัวหมู่มาก จะได้คำตอบคือ 0

2.3 หาค่าเฉลี่ย (average) จากกลุ่มของข้อมูล ดังข้อ 2.2

3. จงเขียนรหัสเทียม (pseudo code) ของอัลกอริทึมสำหรับแก้ปัญหาในข้อ 2 (เลือกเพียงปัญหาเดียว) และวิเคราะห์หาฟังก์ชันเวลาในการทำงาน (running time) *T(N)* ของอัลกอริทึมนั้น (8 คะแนน)

/********** **จบตอนที่** 1 **********/

ตอนที่ 2 (30 คะแนน, 10%, 50 นาที)

<u>การวิเคราะห์อัลกอริทึม</u>

1. จงเขียนนิยามและความหมายของสัญกรณ์เชิงเส้นกำกับ (Asymptotic notations) ต่อไปนี้ Big-O, Big Omega (Ω), Big Theta (Θ) พอสังเขป พร้อมทั้งแสดงกราฟเส้นประกอบ (10 คะแนน)

2. ถูกหรือผิด ให้ใส่เครื่องหมายถูก √ หรือ ผิด X หน้าข้อต่อไปนี้ (5 คะแนน)

_____ a) if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ then $f(n) \prec g(n)$, (g(n) grows faster than f(n))

_____ b) log $(3n^2 + 1) \prec 2n$

_____ c) if f(n) = O(g(n)) and f(n) = Ω(g(n)) then f(n) = Θ(g(n))

_____ d) if f(n) = O(g(n)) and h(n) = O(g(n)) then f(n) = O(h(n))

_____ e) if f(n) = O(g(n)) then f(n)+100 = O(g(n))

3. จงพิสูจน์ว่า $2^n = \Theta(2^{n+1})$ (5 คะแนน)

4. จงหา Big-O ของ running time *t(n)* หรืออัลกอริทึมต่อไปนี้ (10 คะแนน)

4.1 $t(n) = n^2 + 3*\log n$

4.2 $t(n) = 5n^2 - 3n + 10n^3$

4.3 $t(n) = 3^n + n^5$

4.4

```
bubbleSort(A[1..n])
{ for (i=1 to n)
  { j=i
    while(j<=n-1)
    { if (A[j]>A[j+1] )
          swap(A[j],A[j+1])
      j++
     }
}
```

4.5

```
findMin(A[1..n])
{ if (n==1) return A[1]
  m = findMin(A[1..n-1])
  if( m<A[n])
     return m
  else
     return A[n]
}
```

/********** **จบตอนที่** 2 **********/

ตอนที่ 3 (30 คะแนน, 10%, 60 นาที)

<u>โครงสร้างข้อมูล</u>

1. จงออกแบบการเก็บข้อมูลแบบตารางแฮช (Hash Table) สำหรับเก็บข้อมูลนักศึกษา ให้สร้างฟังก์ชันแฮช (hash function) และยกตัวอย่างการเก็บข้อมูล 5-6 ข้อมูล และวิธีการจัดการเมื่อเกิดการชนกัน (collision) ของข้อมูลแฮช (8 คะแนน)

2. Priority Queue with Min Binary Heap (18 คะแนน)

ใช้โครงสร้างข้อมูลแบบ Min Binary Heap สำหรับสร้าง Priority Queue จัดการข้อมูลซึ่งเป็นคีย์ที่เป็นจำนวนเต็มบวกที่มีค่าไม่ซ้ำกัน โดยมี operation สามอย่างคือ Insert (สำหรับการสร้าง Min Heap และ เพิ่มโหนด), Find_Min คือหาค่าคีย์ที่น้อยที่สุดและ Delete_Min คือลบค่าคีย์ที่น้อยที่สุดออก

อัลกอริทึมสำหรับการทำงานทั้งสามอย่างเป็นดังนี้

```
// insert key x to binary tree T
Insert(T, x)
{  if(T==NULL) // empty tree
     return Create(x)  //make a new node with key value of x
   cKey = T->key  // current key at the root node
   if(x < cKey)  // x less than root key
   {  T->key = x // replace x at the root
      if (T->left == NULL) // left subtree is empty
         T->left = Create(cKey) // make a left child node with old root key
      else if (T->right == NULL) // right subtree is empty
         T->right = Create(cKey) // make a left child node with old root key
      else // there exist left and right children nodes
      { if(T->left->key > cKey) // the left key greater than cKey
         T->left = Insert(T->left,cKey) //push cKey down to the left subtree
        else
         T->right = Insert(T->right,cKey) //push cKey down to right subtree
      }
      return T  // return the updated tree
   }
 // if x greater than root key then push x to left or right subtree
  if (T->left == NULL) // left subtree is empty
         T->left = Create(x) // make a left child node with x
  else if (T->right == NULL) // right subtree is empty
         T->right = Create(x) // make a left child node with x
  else // there exist left and right children nodes
  { if(T->left->key > x) // the left key greater than x
       T->left = Insert(T->left, x)  // move x down to insert in left subtree
      else
        T->right = Insert(T->right, x)
         // move x down to inset in right subtree
   }
     return T  // return the updated tree
 }


// Find the min key in the tree
Find_min (T)
{  if(T!=NULL) // not empty tree
     return T->key // min is the root key
   else
     return -1 // error (empty tree)
}
```

```
// Delete min from binary tree T
Delete_min(T)
{  if(T==NULL) // empty tree
     return NULL
   if (T->left == NULL) return T->right
   if (T->right == NULL) return T->left
   // there exist both left and right children nodes
   if(T->left->key < T->right->key) // if left key less than right key
   {  T->key = T->left->key // make the new key with left key
      T->left = Delete-min(T->left)
        //recursively delete_min in the left subtree
   }
   else
   { T->key = T->right->key // make the new key with right key
     T->right = Delete-min(T->right)
        //recursively delete_min in the right subtree
   }
   return T  // return the updated tree
 }
```

คำถาม

2.1 จากอัลกอริทึมข้างต้น จงเขียนรูป Min Heap ผลลัพธ์ เมื่อทำการสร้างด้วยการเพิ่มโหนดด้วยคีย์ต่อไปนี้ ตามลำดับ 15, 16, 17, 10, 6, 30, 28 (5 คะแนน)

Min Heap

2.2 จงหา เวลาที่ใช้ (Big-O) ของการทำงาน Insert, Delete_Min และ Find_Min สำหรับ ข้อมูลคีย์จำนวน n ข้อมูล (5 คะแนน)

Hint: ให้เขียนอยู่ในรูป Big-O ของฟังก์ชันของ n หรือ h เมื่อให้ค่า $n=2^h$ และมีสมมติฐานโดยกรณีเฉลี่ย สำหรับ n ข้อมูล สามารถสร้าง binary tree ได้ระดับความสูง $h = log_2 n$

2.3 ให้ min_k คือค่าคีย์ที่น้อยที่สุดในลำดับที่ k (เช่น จากตัวอย่างในข้อ 2.1 ค่า $min_3 = 15$)

จงเขียนอัลกอริทึมสำหรับการหาค่า min_k ใน binary heap *T* ด้วยค่า *k* ใดๆ ($1 \leq k \leq n$)

Hint. สามารถเรียกใช้ Insert, Delete_min หรือ Find_min ได้ (4 คะแนน)

```
Find_min_k (T, k)
```

2.4 ถ้าเราใช้โครงสร้างข้อมูลอาร์เรย์ไม่เรียงลำดับ (unsorted array) เพื่อเก็บ priority queue การค้นหาค่า min และค่า min_k จะใช้วิธีการอย่างไร และใช้เวลาเท่าใด จงอธิบายพอสังเขป และให้เปรียบเทียบกับการหา min_k ด้วย binary heap นี้ แบบใดมีประสิทธิภาพมากกว่ากัน (4 คะแนน)

3. การแทนกราฟแบบไม่มีทิศทาง (undirected graph) สามารถใช้โครงสร้างข้อมูลแบบใดได้บ้าง (ยกตัวอย่างสองแบบ) และหากมีกราฟแบบแน่น (dense graph) และ กราฟแบบโล่ง (sparse graph) ให้เลือกใช้โครงสร้างข้อมูลแบบใดจึงจะเหมาะสม (6 คะแนน)

/********** **จบตอนที่** 3 **********/

ตอนที่ 4 (24 คะแนน, 8 %, 40 นาที)

<u>การค้นหาและการเรียงลำดับ</u>

1. จากขั้นตอนวิธีในการเรียงลำดับข้อมูลที่กำหนดให้ต่อไปนี้ จงเลือกขั้นตอนวิธีที่เหมาะสมที่สุด

I. Bubble Sort II. Insertion Sort III. Merge Sort IV. Quick Sort

V. Selection Sort VI. Shell Sort VII. Heap Sort (7.5 คะแนน)

1.1 อาร์เรย์ที่เรียงลำดับเกือบทั้งหมดแล้ว มีข้อมูลส่วนน้อยที่อยู่ในตำแหน่งที่ไม่ถูกต้อง

1.2 ข้อมูลที่ต้องการเรียงลำดับมีขนาดใหญ่กว่าหน่วยความจำหลักที่จะเก็บได้หมด ข้อมูลส่วนใหญ่อยู่ในหน่วยความจำสำรอง เช่น hard disk

1.3 มีข้อมูลหลายชุดที่ต้องการเรียงลำดับข้อมูล โดยข้อมูลแต่ละชุดมีสมาชิกไม่เกิน 10 ค่า

1.4 ขั้นตอนวิธีที่มีประสิทธิภาพในกรณี Worst case เป็น $O(N^{1.5})$

1.5 ขั้นตอนวิธีที่มีประสิทธิภาพในกรณี Worst case เป็น $O(N*\log N)$ โดยไม่สามารถใช้พื้นที่ในพิเศษหน่วยความจำเพิ่มเติม (เว้นแต่ตัวแปรชนิด Local จำนวนไม่มาก)

2. จงแสดงขั้นตอนในการเรียงลำดับด้วยขั้นตอนวิธีการเรียงลำดับต่อไปนี้ จากข้อมูลที่กำหนดให้ (8 คะแนน)

2.1 ใช้ *Selection Sort* ในการการเรียงข้อมูลต่อไปนี้ จากน้อยไปมาก

16, 21, 45, 8, 11, 53, 3, 26, 49

2.2 เลือกใช้ขั้นตอนวิธีการเรียงลำดับข้อมูล(ที่ต่างจาก Selection Sort) ในการเรียงลำดับข้อมูลนี้ ให้ระบุชื่อขั้นตอนวิธีและเวลาที่ใช้ (time complexity, Big-O) ของขั้นตอนวิธีที่เลือก

62 , 83, 18, 53, 7, 17, 96, 85, 47, 69, 25, 29

3. จงตอบคำถามต่อไปนี้ (8.5 คะแนน)

3.1 จงอธิบายหลักการของ Sequential Search (1 คะแนน)

3.2 ในการพัฒนาโปรแกรมของ Sequential Search ในบทเรียนมี 2 วิธีคือ ค้นหาบน Linked List และ ใช้อาร์เรย์ จงบอกข้อแตกต่างด้านประสิทธิภาพว่ามีความแตกต่างกันหรือไม่ และอย่างไร (1.5 คะแนน)

3.3 จงเขียน Pseudo Code ของขั้นตอนวิธีแบบ Binary Search (4 คะแนน)

3.4 ประสิทธิภาพเชิงเวลาใน การวิเคราะห์ Binary Search สำหรับข้อมูลที่เป็น worst case และ average case มีค่าเป็นเท่าใด (2 คะแนน)

/********** **จบตอนที่** 4 **********/