FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110327 Algorithm Design

YEAR III, First Semester, Final Examination, September 23, 2013, Time 8:30 – 11:30

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

หมายเหตุ

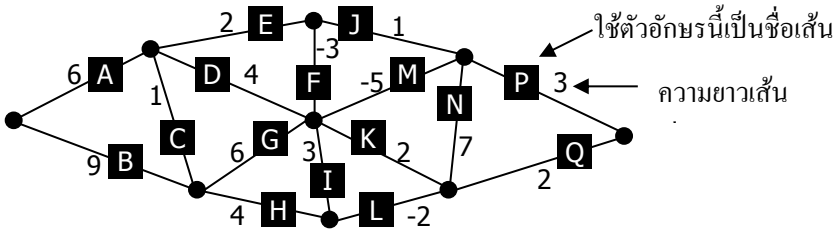
1. ข้อสอบมีทั้งหมด 11 ข้อในกระดาษคำถามคำตอบจำนวน 9 แผ่น 9 หน้า คะแนนเต็ม 95 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. เขียนตอบในกระดาษข้อสอบชุดนี้ (ถ้าเนื้อที่ไม่พอ ให้เขียนต่อเฉพาะที่หน้าหลังของกระดาษคำถามแผ่นนั้น)
4. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน
5. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
6. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
7. ผู้ที่ประสงค์จะออกจากห้องสอบก่อนหมดเวลาสอบ แต่ต้องไม่น้อยกว่า 45 นาที
8. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
9. ผู้ที่ปฏิบัติเข้าข่ายทุจริตในการสอบ ตามประกาศคณะวิศวกรรมศาสตร์
   มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

รับทราบ

ลงชื่อนิสิต (.....)

---

1 (5 คะแนน) จงเขียน<u>ลำดับของชื่อเส้นเชื่อม</u>ที่ถูกเลือกให้เป็นส่วนหนึ่งของ minimum spanning tree ด้วยการใช้ Kruskal's algorithm กับกราฟข้างล่างนี้ (ไม่ต้องแสดงวิธีทำ)

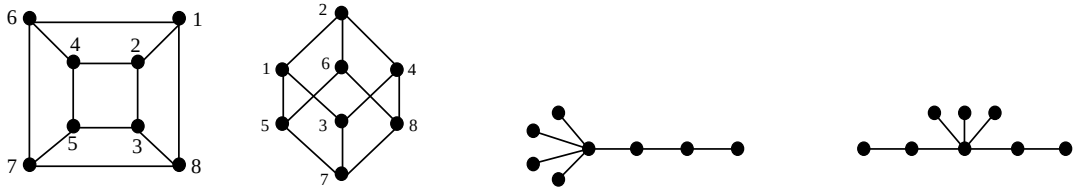


……

2 (15 คะแนน) จงระบุว่า ข้อย่อยต่อไปนี้ข้อใดถูก ข้อใดผิด (<u>ไม่ต้องอธิบายที่มาของคำตอบ เขียนตอบแค่ถูกหรือผิด</u>)

1) …… อัลกอริทึมของ Floyd-Warshall อาศัยการแก้ปัญหาแบบ dynamic programming
2) …… อัลกอริทึมของ Prim, Kruskal, และ Dijkstra ล้วนจัดเป็น greedy algorithm ทั้งสิ้น
3) …… Depth-first search หาวิถีสั้นสุดระหว่างคู่ปมคู่หนึ่งในกราฟที่เส้นเชื่อมยาวเท่ากันหมดได้ในเวลา $O(v + e)$
4) …… เราสามารถใช้อาเรย์ 2 มิติขนาด 10×10 เพื่อแทน simple undirected graph ที่มี 10 ปม เส้นเชื่อม 40 เส้นได้
5) …… ถ้า G เป็น directed graph เราสามารถหา topological sort ของกราฟ G ได้
6) …… การใช้ depth-first search เพื่อแวะผ่านทุก ๆ ปมในกราฟ จะใช้เวลา (แบบ asymptotic) น้อยกว่าการใช้ breadth-first search
7) …… ปัญหา 0/1 Knapsack ที่ใช้กับการเลือก/ไม่เลือก ของ 5 ชิ้นใส่ถุง จะมี solution states จำนวน $5^2 = 25$ states
8) …… ไม่ว่าจะใช้ breadth-first search หรือ depth-first search เพื่อหาคำตอบของปัญหา 0/1 Knapsack ก็ต้องวิ่งพิจารณา<u>ทุกปม</u>ใน state space
9) …… ปัญหาของน้องมะนาว คือ “กราฟ G มี path จากปม s ถึง t ที่มีระยะทางรวมไม่เกิน k หรือไม่ ?” คิดสักครู่ มะนาวก็สรุปว่า ปัญหานี้ง่ายมากใช้ Bellman-Ford ปัญหาของน้องมะนาวจึง<u>ไม่จัดอยู่</u>ในกลุ่มปัญหา *NP*
10) …… การพิสูจน์ได้ว่า $P = NP$ เป็นจริง ย่อมสะเทือนวงการอัลกอริทึมมากกว่า การพิสูจน์ได้ว่า $P \neq NP$ เป็นจริง
11) …… ปัญหาที่ถามว่า ‘‘กราฟ G เป็น DAG หรือไม่’’ เป็นปัญหาในกลุ่ม *NP*-complete
12) …… เราเรียก decision-problem q ว่า เป็น *NP*-complete ก็ต่อเมื่อทุก ๆ ปัญหาใน *NP* สามารถลดรูป (reduce) ไปเป็นปัญหา q ได้
13) …… ทุกปัญหาใน P มีอัลกอริทึมหาคำตอบได้ใน polynomial time ดังนั้น ปัญหาใน P จึงมีความง่ายเท่ากันหมด
14) …… ปัญหาใน *NP* ที่ไม่อยู่ใน P ก็ต้องเป็น NP-complete
15) …… หากมีวิธี reduce (ในเวลา polynomial) ปัญหา S ไปเป็นปัญหา q สรุปได้ว่า ปัญหา S ไม่ง่ายกว่าปัญหา q

3 (5 คะแนน) ปัญหา graph isomorphism ถามว่า กราฟ G และ H ที่ได้รับ มีลักษณะทางโครงสร้างเหมือนกันหรือไม่ (เหมือนกันแบบจุดและเส้นเชื่อมคล้องจองกันจุดต่อจุดและเส้นต่อเส้น โดยการเชื่อมต่อของจุดและเส้นเชื่อมที่คล้องจองกันต้องเหมือนกัน) ตัวอย่างเช่น กราฟสองรูปทางซ้ายเหมือนกัน ในขณะที่สองกราฟทางขวาไม่เหมือนกัน



จงแสดงให้เห็นจริงว่า ปัญหา graph isomorphism จัดอยู่ในปัญหาประเภท NP โดยการเขียนตัว verifier ที่ตรวจคำตอบ yes ของปัญหานี้ให้ได้ใน polynomial time

```
isValidIsomorphic( G[1..n][1..n], H[1..m][1..m], g2h[1..n] ) {
    // G และ H เป็น adjacency matrices ของกราฟ
    //   ถ้ามีเส้นเชื่อมระหว่างจุด i กับ j ในกราฟ ช่องที่ [i][j] มีค่าเป็น 1, ไม่เช่นนั้น มีค่าเป็น 0
    // g2h[k] เก็บหมายเลขจุดของกราฟ H ที่เทียบได้กับจุด k ของกราฟ G
```

4 (5 คะแนน) Eulerian graph คือ undirected connected graph ที่แต่ละ vertex มีเส้นเชื่อมต่ออยู่เป็นจำนวนคู่ จงเขียน pseudo-code ข้างล่างนี้ให้สมบูรณ์ เพื่อตรวจสอบว่า กราฟที่ได้รับ (ในรูปแบบ adjacency matrix) เป็น Eulerian graph หรือไม่ (ให้ถือว่า กราฟที่รับมาเป็น undirected connected graph แน่ๆ)

```
isEulerianGraph( G[1..n][1..n] ) {
   // ให้ถือว่า G ที่รับมาเป็น undirected connected graph
   // G[i][j] = 1 แทนการมีเส้นเชื่อมระหว่างปม i กับ j ถ้าเท่ากับ 0 แทนการไม่มีเส้นเชื่อม
```

5 (10 คะแนน) ปัญหาการทอนเหรียญ ถามว่า จะทอนเหรียญให้ได้มูลค่ารวมเท่ากับ V ด้วยเหรียญมูลค่าต่างๆ ที่มีจากเซต $C = \{ c_1, c_2, \ldots, c_n \}$ อย่างไร ให้ได้เหรียญจำนวนน้อยสุด (กำหนดให้เรามีเหรียญมูลค่าต่างๆ อยู่จำนวนไม่จำกัด)

ก. ปัญหาการทอนเงินไม่สามารถใช้ greedy กับเซต C ได้ทุกรูปแบบ (บางแบบก็ได้จำนวนเหรียญน้อยสุด บางแบบก็ไม่ได้) อย่างไรก็ตาม ในข้อนี้ จงออกแบบ greedy อัลกอริทึมเขียนบรรยายด้วย pseudo code ตามข้อกำหนดข้างล่างนี้ (ถึงแม้ว่าจะไม่สามารถได้จำนวนเหรียญน้อยสุดทุกกรณี แต่ก็น่าจะได้ "น้อยๆ" และน้อยสุดได้ในบางกรณี)

```
coinChange( V, C[1..n] , X[1..n] ) {
   // V คือมูลค่าที่ต้องการแลกเป็นเหรียญ
   // C คืออาเรย์เก็บมูลค่าของเหรียญแต่ละชนิดที่มี (มี n ชนิด) โดย C[1] < C[2] < … < C[n]
   // X คืออาเรย์เก็บผลลัพธ์ของการทอนเหรียญ โดย X[k] เก็บจำนวนเหรียญที่ทอนชนิดที่ k
   // เช่น V = 14, C = [1,2,5,10], จะได้ X = [0,2,0,1]
   //     คือทอนเหรียญ 2 สองเหรียญ และเหรียญ 10 หนึ่งเหรียญ









}
```

ข. จงยกตัวอย่างเซต C มูลค่า V และคำตอบที่ **coinChange** ที่เขียนในข้อ ก หาแล้วจะไม่ได้จำนวนเหรียญน้อยสุด

ค. อยากทราบว่า หากนำจำนวนเหรียญที่ได้จากวิธีที่ได้นำเสนอมาในข้อ ก (ซึ่งคือผลรวมของทุกช่องในอาเรย์ **X**) มาเป็น cost ในการค้นคำตอบแบบ least cost search จะได้จำนวนเหรียญที่น้อยที่สุดหรือไม่ จงอธิบายพร้อมกับใช้ตัวอย่างที่เขียนตอบในข้อ ข วาดเป็นปมต่างๆ (กับ cost) ระหว่างการค้นด้วย least cost search ประกอบ

6 (10 คะแนน) กำหนดให้ "ลำดับไม่ลด" คือ ลำดับของจำนวนเต็มบวก A = < a_1, a_2, ..., a_k > โดยที่ $a_1 \le a_2 \le ... \le a_k$ อยากทราบว่ามี "ลำดับไม่ลด" ใดบ้างที่ผลรวมของสมาชิกในลำดับมีค่าเท่ากับ N ตัวอย่างเช่น ให้ N = 3 ลำดับไม่ลดที่มีผลรวมเป็น 3 ได้แก่ <1,1,1>, <1,2> และ <3>

ก. จงเขียน State Space Tree ของการแจกแจง "ลำดับไม่ลด" ทั้งหมดที่ผลรวมเป็น N = 4

ข. จงเขียน pseudo code เพื่อ print "ลำดับไม่ลด" ทั้งหมด จากค่า N ที่ได้รับ โดยใช้ depth first search ค้นใน state space tree ในลักษณะที่นำเสนอในข้อ ก จากค่า N ที่ได้รับ

```
showAllNonDecreasingSumToN(  N  ) {
```

7 (5 คะแนน) ให้ G เป็น simple undirected graph ที่มี v ปม และ e เส้น การทำ depth first search (DFS) ใน G (ที่แทนด้วย adjacency list) ย่อมใช้เวลา $\Theta(v+e)$ ดังนั้นการใช้ DFS เพื่อตรวจว่า G มี cycle หรือไม่ จึงสามารถตรวจได้ในเวลา $O(v+e)$ จงแสดงให้เห็นจริงว่า แท้จริงแล้ว การใช้ DFS เพื่อตรวจว่า G มี cycle หรือไม่นั้น ใช้เวลาเพียง $O(v)$ (ข้อแนะนำ: แยกแสดงสองกรณี: กรณีที่ G มี cycle กับ กรณีที่ G ไม่มี cycle)

8 (10 คะแนน) ให้ G เป็นกราฟที่แทนด้วย adjacency matrix ขนาด $n \times n$ สมมติว่า เราได้หา all-pair shortest paths ของ G ได้ความยาวของ shortest paths ของทุกคู่ เก็บในเมทริกซ์ D (ซึ่งก็มีขนาด $n \times n$ เช่นกัน โดยที่ $D[i][j]$ เก็บระยะทางของเส้นทางสั้นสุดจากจุด i ไปยังจุด j ของ G) หากเราเพิ่มเส้นเชื่อมเส้นใหม่ใน G การปรับค่า all-pair shortestes paths ใน D หลังเพิ่มเส้นเชื่อมใหม่นี้จะทำได้อย่างไรในเวลา $O(n^2)$?

ให้นิสิตเขียนรหัสเทียม **`UpdateAllPairShortestPath( D,n,a,b,w )`** เพื่อปรับค่าใน D หลังเพิ่มเส้นเชื่อมใหม่จากจุด **`a`** ไป **`b`** ที่มีความยาว **`w`** (คำตอบที่ยอมรับได้จะต้องใช้เวลาในการทำงานไม่ช้ากว่า $O(n^2)$)

```
UpdateAllPairShortestPath( D, n, a, b, w ) {
```

9 (10 คะแนน) ให้คุณเป็น project manager ของบริษัทซอฟต์แวร์ยักษ์ใหญ่แห่งหนึ่ง คุณเพิ่งรับโปรเจคขนาดใหญ่ที่ประกอบด้วยส่วนของโปรแกรมย่อยๆ ทั้งหมด n ส่วน โดยส่วนที่ k $(1 \le k \le n)$ จะต้องใช้เวลาในการทำ t_k วัน และคุณก็มี deadline ว่าต้องทำโปรเจคนี้ให้เสร็จภายในเวลา x วัน คำถามในข้อนี้คือ คุณจะต้องใช้โปรแกรมเมอร์ อย่างน้อยที่สุดกี่คน เพื่อที่จะสามารถทำทุกส่วนให้เสร็จตามกำหนด

ในข้อนี้ โปรแกรมเมอร์ทุกคนสามารถทำส่วนของโปรแกรมใดก็ได้ ส่วนใดก่อนหรือหลังก็ได้ แต่มีข้อแม้ว่า แต่ละส่วนต้องทำโดยโปรแกรมเมอร์เพียงคนเดียว (ไม่สามารถช่วยเหลือกันได้) โปรแกรมเมอร์ทุกคนสามารถเริ่มงานได้ทันที และ เมื่อโปรแกรมเมอร์ทำส่วนของโปรแกรมใดเสร็จแล้ว ก็จะสามารถเริ่มทำส่วนอื่นได้ทันที (สรุปอีกครั้งว่า โปรแกรมเมอร์หนึ่งคนรับทำงานได้หลายส่วน แต่หนึ่งส่วนของโปรแกรมต้องให้โปรแกรมเมอร์หนึ่งคนทำเท่านั้น)

เนื่องจากคุณได้เรียนวิชาอัลกอริทึมมาแล้ว คุณจึงดูออกทันทีว่า สามารถใช้ Branch and Bound ในการแก้ปัญหานี้ได้
<u>คำถามคือ จะหา lower bound ของจำนวนโปรแกรมเมอร์ที่จะต้องใช้อย่างไร ?</u>

จงเติมรหัสเทียมในช่องว่างเพื่อหา lower bound ของจำนวนโปรแกรมเมอร์ที่ต้องใช้ในการทำส่วนที่ j ถึงส่วนที่ n ให้เสร็จภายในเวลา q วัน และเขียนคำอธิบายประกอบด้านล่างด้วย (lower bound ของคุณจะต้องไม่ใช่ค่าคงที่ และใช้เวลาในการคำนวนไม่เกิน $O(n)$ และคุณจะได้คะแนนตามคุณภาพของ lower bound ที่ระบุ)

```
ComputeLowerBoundOfNumProgrammers( t[1..n], j, n, q ) {
   // คืนจำนวนโปรแกรมเมอร์น้อยสุดในการทำส่วนที่ j ถึง n ให้เสร็จภายใน q วัน
   //  โดยส่วนที่ k ใช้เวลา t[k] วันในการพัฒนา (โดยโปรแกรมเมอร์คนใดก็ได้)
```

```
}
```

คำอธิบาย

10 (10 คะแนน) มีงานอยู่ n งานที่ต้องการใช้ห้องจัดงานห้องเดียวกัน แต่ละงานมีหมายเลขกำกับ (1,2,…, n) โดยงานที่ i ต้องการเริ่มใช้ห้อง ณ เวลา s_i และใช้ห้องเสร็จ ณ เวลา f_i ปัญหา Activity Selection (ที่ได้เรียนมา) ถามว่า จะเลือกงานใดบ้างจาก n งาน ให้ได้จำนวนงานมากที่สุด และไม่มีงานที่ถูกเลือกใดใช้ห้องในเวลาซ้อนเหลื่อมกันเลย

เราหาคำตอบของปัญหานี้ได้ด้วยอัลกอริทึมแบบละโมบ เขียนเป็นรหัสเทียมได้ดังนี้

```
GreedyActivitySelect( A[1..n] )
  // A เก็บงาน n งาน โดยที่ A[i].s คือเวลาที่งาน i เริ่ม และ A[i].f คือเวลาที่งาน i เลิก
  sort A by finished times //  <------------------
  X = {1}
  last_index = 1
  for (i = 2; i <= n; i++) {
    if ( A[i].s >= A[last_index].f )  {
      X = X U { i }
      last_index = i
    }
  }
  return X
}
```

เวลาส่วนใหญ่ของอัลกอริทึมนี้ใช้ไปกับการเรียงข้อมูล O(n log n) อย่างไรก็ตาม ถ้า A นั้นถูกเรียงมาแล้ว ก็น่าจะออกแบบอัลกอริทึมที่มีประสิทธิภาพดีกว่า จงเขียนรหัสเทียมสำหรับแก้ปัญหานี้ กำหนดให้ A ถูกเรียงมาเรียบร้อยแล้ว แต่เรียงตามเวลาเริ่ม ไม่ใช่เวลาสิ้นสุด โดยรหัสเทียมที่นิสิตจะเขียนต้องใช้เวลาเป็น O(n)

```
GreedyActivitySelect( A[1..n] ) {
  // A เก็บงาน n งาน โดยที่ A[i].s คือเวลาที่งาน i เริ่ม และ A[i].f คือเวลาที่งาน i เลิก
  // งานใน A ถูกเรียงแล้วตามเวลา "เริ่ม" จาก น้อยไปมาก A[1].s ≤ A[2].s ≤ ... ≤ A[n].s
```

11 (10 คะแนน) กำหนดให้ปัญหาการเดินทางในตารางเป็นดังนี้

- มีห้องสี่เหลี่ยมจัตุรัสขนาดเท่ากันหมด เรียงต่อกันเป็นตารางจำนวน $n \times m$ ห้อง
- แต่ละห้องมีพิกัด (x,y) กำกับ โดยห้องมุมซ้ายบนคือ (1,1) และห้องมุมขวาล่างคือ (n,m)
- เราสามารถเดินจากห้อง (x,y) ไปยังห้องสี่ห้องที่อยู่ติดกันได้ คือห้อง $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ และ $(x, y-1)$ (ยกเว้นกรณีที่อยู่ที่ขอบตารางอาจเดินไปบางทิศไม่ได้)
- ในแต่ละห้องมีสิ่งกีดขวางทำให้ต้องเสียเวลาเดินผ่านห้อง (x,y) เป็นเวลา $T[x][y]$ วินาที สมมติว่าอยู่ที่ห้อง (1,1) เดินไปยัง (2,3) ด้วยลำดับ (1,1) → (1,2) → (2,2) → (2,3) ใช้เวลารวมเป็น T[1][2] + T[2][2] เพราะผ่านห้อง (1,2) กับ (2,2) ไม่นับห้องต้นทางกับห้องปลายทาง

จงออกแบบอัลกอริทึมเพื่อหาระยะเวลาน้อยสุดที่ต้องใช้ในการผ่านสิ่งกีดขวางทั้งหมด ในการเดินทางจากห้อง (1,1) ไปยังห้อง (n,m) เพื่อความง่าย นิสิต**ต้องแก้ปัญหานี้โดยการเรียกใช้อัลกอริทึมของ Dijkstra** กำหนดให้มีฟังก์ชัน **`Dijkstra(k, G[1..k][1..k], s, D[1..k])`** สำหรับการหา shortest path บน directed graph ที่มี **`k`** ปม โดยเริ่มจากปม **`s`** ในกราฟ G ที่แทนด้วย adjacency matrix (**`G[a][b]`** เก็บความยาวของเส้นเชื่อมจากปม **`a`** ไปยังปม **`b`**, **`G[a][b]`** = ∞ แทนกรณีไม่มีเส้นเชื่อม) ฟังก์ชันนี้ให้ผลลัพธ์กลับมาที่ตัวแปร **`D`** โดยที่ **`D[i]`** เก็บระยะทางสั้นสุดจาก **`s`** ไปยัง **`i`** (ถือว่า ฟังก์ชัน **`Dijkstra`** ข้างต้นนี้มีให้ใช้แล้ว เรียกใช้ได้เลย ไม่ต้องเขียนรายละเอียดภายใน)