

FACULTY OF ENGINEERING  
CHULALONGKORN UNIVERSITY  
2110327 Algorithm Design

YEAR III, First Semester, Final-term Examination, November 30, 2015, Time 8:30 – 11:30

ชื่อ-นามสกุล \_\_\_\_\_ เลขประจำตัว \_\_\_\_\_ CR58 \_\_\_\_\_

หมายเหตุ

1. ข้อสอบมีทั้งหมด 9 ข้อในกระดาษคำถามคำตอบ รวม จำนวน 8 หน้า คะแนนเต็ม 92 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

ห้ามนิสิตพกโทรศัพท์หรืออุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ ให้ได้รับ F และอาจจะพิจารณาสั่งพักการศึกษา \*\*

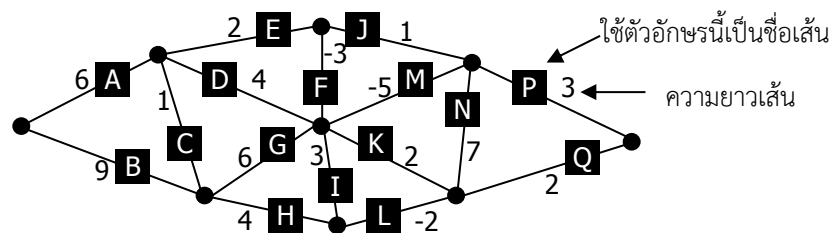
ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือในการทำข้อสอบนี้

ลงชื่อนิสิต .....

วันที่.....

1. (18 คะแนน) จงระบุว่า ข้อย่อต่อไปนี้ข้อใดจริง ข้อใดเท็จ (ไม่ต้องอธิบายที่มาของคำตอบ เขียนตอบแค่ T หรือ F คำตอบที่ถูกต้องจะได้ 1 คะแนน คำตอบที่ผิดจะติดลบ 0.5 คะแนน คะแนนติดลบในข้อนี้จะไม่ส่งผลกระทบต่อข้ออื่น)
  - 1) ..... อัลกอริทึมของ Floyd-Warshall อาศัย recurrence  $d_{ij}(k) = \min(d_{ij}(k-1), d_{ik}(k-1)+d_{kj}(k-1))$
  - 2) ..... เราหา maximum spanning tree ของกราฟ G ได้ง่าย ๆ ด้วยการสร้างกราฟ H ที่เหมือน G แต่เส้นเชื่อมของ H มีน้ำหนักเป็นค่าติดลบของน้ำหนักเส้นเชื่อมใน G จะได้ว่า minimum spanning tree ของ H (ที่หาด้วย อัลกอริทึม Prim) ก็เป็น maximum spanning tree ของ G เช่นกัน
  - 3) ..... ให้ G เป็น directed acyclic graph เราหา longest path ใน G ได้ง่าย ๆ ด้วยการสร้างกราฟ H ที่เหมือน G แต่เส้นเชื่อมของ H มีน้ำหนักเป็นค่าติดลบของน้ำหนักเส้นเชื่อมใน G จะได้ว่า shortest path ของ H (ที่หาด้วยอัลกอริทึมของ Dijkstra) ก็เป็น longest path ของ G เช่นกัน
  - 4) ..... ให้ G เป็น dense graph ซึ่งคือกราฟที่มีจำนวนเส้นเชื่อม  $e = \Theta(v^2)$  การหา minimum spanning tree ด้วยอัลกอริทึมของ Prim กับ Kruskal จะมี time complexity เท่ากัน ไม่ว่าจะเลือกใช้โครงสร้างข้อมูลแบบใดก็ตาม
  - 5) ..... เป็นได้ไหม ที่ directed graph G ที่มี  $v$  ปม เป็น weakly connected graph แต่มี  $v-1$  strongly connected components
  - 6) ..... โดยปกติเรานิยามให้ความยาวของ path หนึ่งคือผลรวมของ edge ทุกเส้นใน path นั้น ถ้าขอเปลี่ยนนิยามให้ความยาวของ path หนึ่งเป็น ผลคูณของ edge ทุกเส้นใน path นั้น อยากทราบว่า ถ้าไม่มีเส้นเชื่อมใดที่ความยาวเป็นลบ จะใช้อัลกอริทึม Dijkstra หา shortest path (ด้วยนิยามความยาวแบบผลคูณ) ได้หรือไม่
  - 7) ..... โดยปกติเรานิยามให้ความยาวของ path หนึ่งคือผลรวมของ edge ทุกเส้นใน path นั้น ถ้าขอเปลี่ยนนิยามให้ความยาวของ path หนึ่งเป็น ผลคูณของ edge ทุกเส้นใน path นั้น อยากทราบว่า จะใช้อัลกอริทึม Bellman-Ford หา shortest path (ด้วยนิยามความยาวแบบผลคูณ) ได้หรือไม่

- 8) ..... กราฟที่มี topological sort แคแบบเดียว เป็นกราฟที่หา shortest path ระหว่างปมใด ๆ ได้ใน  $O(v)$
  - 9) ..... การเพิ่มความยาวให้กับเส้นเชื่อมทุกเส้นในกราฟด้วยค่าคงตัว  $K$  ไม่ได้ทำให้ shortest path ในกราฟเปลี่ยน
  - 10) ..... ให้  $G$  เป็น directed graph ขนาด 10,000 ปม ที่มี 1 strongly connected component และทุก vertex มี in-degree = 1 และ out-degree = 1 จะได้ว่าหน่วยความจำที่ใช้เพื่อทำ depth-first search มีปริมาณน้อยกว่าเมื่อทำ breadth-first search กับกราฟ  $G$
  - 11) ..... การทำ depth-first search กับกราฟ  $G$  ที่มี  $v$  ปม  $e$  เส้น ที่แทนด้วย adjacency matrix ใช้เวลา  $O(v^2)$
  - 12) ..... หลังจากอัลกอริทึมของ Ford-Fulkerson หยุดทำงานแล้ว เราสามารถหาปริมาณ max-flow ได้ง่าย ๆ ด้วยการรวม flow ของเส้นเชื่อมทุกเส้นที่พุ่งออกจากปม source ของ network
  - 13) ..... หลังจากอัลกอริทึมของ Ford-Fulkerson หยุดทำงานแล้ว เราสามารถหาปริมาณ min-cut ได้ง่าย ๆ ด้วยการรวม flow ของเส้นเชื่อมทุกเส้นที่พุ่งเข้าหาปม sink ของ network
  - 14) ..... การทดสอบว่ากราฟ  $G$  เป็น directed acyclic graph หรือไม่ ทำได้ในเวลา  $O(v+e)$  ด้วย depth-first search ดังนั้น ปัญหาการทดสอบว่ากราฟ  $G$  เป็น directed acyclic graph หรือไม่ จึงไม่จัดอยู่ในกลุ่ม  $NP$
  - 15) ..... เราเรียก decision-problem  $q$  ว่าเป็น  $NP$ -complete ก็ต่อเมื่อทุก ๆ ปัญหาใน  $NP$  สามารถลดรูป (reduce) ไปเป็นปัญหา  $q$  ได้
  - 16) ..... ทุกปัญหาใน  $P$  มีอัลกอริทึมหาคำตอบได้ใน polynomial time ดังนั้น ปัญหาใน  $P$  จึงมีความง่ายเท่ากันหมด
  - 17) ..... ปัญหาใน  $NP$  ที่ไม่อยู่ใน  $P$  ก็ต้องเป็น  $NP$ -complete
  - 18) ..... หากมีวิธี reduce (ในเวลา polynomial) ปัญหา  $S$  ไปเป็นปัญหา  $q$  สรุปได้ว่า ปัญหา  $S$  ไม่ง่ายกว่าปัญหา  $q$
2. (4 คะแนน) จงเขียนลำดับของชื่อเส้นเชื่อมที่ถูกเลือกให้เป็นส่วนหนึ่งของ minimum spanning tree ด้วยการใช้ Kruskal's algorithm กับกราฟข้างล่างนี้ (ไม่ต้องแสดงวิธีทำ)



3. ตอบคำถามต่อไปนี้ โดยบรรยายการทำงานของอัลกอริทึมอย่างกระชับได้ใจความ สามารถใช้อัลกอริทึมที่นำเสนอในวิชานี้ได้เลย
- 1) (5 คะแนน)  $G$  เป็น connected undirected graph ในรูปของ adjacency matrix จงบรรยายอัลกอริทึมที่ใช้เวลาที่มีประสิทธิภาพที่สุด ๆ เพื่อหาว่า มี edge ใน  $G$  ใหม่ ที่เมื่อลบออกแล้ว  $G$  ยังคง connected

เชื่อกว่า  $G$  เป็น tree แน่หรือไม่

$$\sum \sum a[i][j] = v-1$$

$$O(n^2)$$

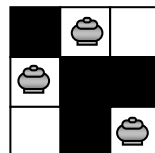
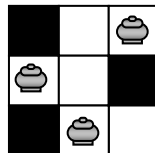
- 2) (5 คะแนน) จงบรรยายอัลกอริทึมที่หา spanning tree สักหนึ่งต้น (ไม่จำเป็นต้อง minimal) ของ undirected graph  $G$  ในเวลาเชิง asymptotic ที่เร็วที่สุด ๆ

DFS / BFS  $O(|V| + |E|)$

- 3) (5 คะแนน) จงบรรยายขั้นตอนการหา shortest path ระหว่างคู่ปมที่กำหนดให้ของกราฟ  $G$  ที่เส้นเชื่อมทุกเส้นยาวเท่ากันหมดในเวลาเชิง asymptotic ที่เร็วที่สุด ๆ

BFS  $O(|V| + |E|)$

4. (5 คะแนน) จงแสดงวิธีการเปลี่ยนปัญหาข้างล่างนี้ให้เป็นปัญหา max flows ยกตัวอย่างการแปลงประกอบด้วย (โดยใช้ตัวอย่างของรูปที่แสดง) จงหาวิธีการวาง "เรือ" ที่ไม่ "กินกัน" ให้ได้จำนวนมากสุด ในช่องต่าง ๆ ของตารางขนาด  $r \times c$  (แถวแนวนอน  $r$  แถว, แถวแนวตั้ง  $c$  แถว) โดยตารางนี้มีบางช่องที่กำหนดให้ห้ามวางเรือ (หมายเหตุ เรือในหมากรุก มีการเดินและกินยาวได้ในแนวตั้ง-แนวนอน ดังนั้นจึงห้ามวางเรือมากกว่าหนึ่งตัวในแถวแนวนอน หรือแถวแนวตั้งเดียวกัน) ดังสองตัวอย่างข้างล่างนี้ ช่องสีดำคือช่องที่ห้ามวางเรือ



5. (10 คะแนน) จงแสดงให้เห็นจริงว่า ปัญหาต่อไปนี้ อยู่ในกลุ่ม *NP*

1) "กราฟ  $G$  มี spanning tree ที่ผลรวมของน้ำหนักของเส้นเชื่อมไม่เกิน  $K$  หรือไม่?"

นั่น minimum spanning tree

$$\text{ผล} = \text{sum}(\text{tree}) \leq K$$

2) "กราฟ  $G$  มี clique ขนาดไม่น้อยกว่า  $K$  หรือไม่?"

(เราจะเรียกกราฟย่อยใดของกราฟ  $G$  ว่าเป็น clique ถ้ากราฟย่อยนั้นเป็น complete subgraph คือมีเส้นเชื่อมระหว่างทุกคู่ปมในกราฟย่อยนั้น ขนาดของ clique ก็คือจำนวนปมของ clique นั้น)

จับจุด เรา sum  $\Phi_n$  adj matrix ข้อข ว่า  $\# \text{ edge} = \binom{V}{2}$

6. (5 คะแนน) เราจะเรียกกราฟย่อยใดของกราฟ  $G$  ว่าเป็น clique ถ้ากราฟย่อยนั้นเป็น complete subgraph (คือมีเส้นเชื่อมระหว่างทุกคู่ปมในกราฟย่อยนั้น) จงเขียน pseudo-code ของอัลกอริทึมที่ใช้ state space search แบบ depth-first เพื่อหาว่ากราฟ  $G$  มี clique ที่ใหญ่สุดขนาดเท่าใด (ขนาดของ clique คือจำนวนปมใน clique) ให้รับกราฟ  $G$  ในรูปของ adjacency matrix (depth-first search ธรรมดา ไม่ต้องมี backtrack ไม่ต้อง branch & bound)

1) Generate  $(0, 1, 0, \dots)$  ตามปม

2) ที่ใบ เช็คว่าได้ Complete graph หรือไม่

7. ข้อนี้เราจะพิจารณาปัญหาชื่อว่า Minimum Set Cover ซึ่งนิยามได้ดังต่อไปนี้ กำหนดให้มี set  $U = \{1, 2, \dots, N\}$  และมี subset จำนวน  $m$  subsets คือ  $s_1, s_2, \dots, s_m$  โดยที่  $s_i \subseteq U$  เราต้องการเลือก subset เหล่านี้มาสักชุดหนึ่ง (กำหนดให้ตัวที่เลือกคือ  $s_{c1}, s_{c2}, \dots, s_{ck}$ ) โดยมีข้อบังคับคือ  $s_{c1} \cup s_{c2} \cup \dots \cup s_{ck}$  จะต้องเท่ากับ  $U$  แน่นอนว่าเราสามารถเลือก subset ดังกล่าวได้ง่าย ๆ โดยการเลือก  $s_i$  ทุกตัว แต่โจทย์ข้อนี้มีการกำหนดเพิ่มเติมคือ การเลือก  $s_i$  นั้นจะต้องเสีย cost  $p_i$  ( $p_i > 0$  เสมอ) ดังนั้น เราต้องการเลือก  $s_{c1}, s_{c2}, \dots, s_{ck}$  ที่ทำให้  $p_{c1} + p_{c2} + \dots + p_{ck}$  นั้นมีค่าน้อยที่สุด จงตอบคำถามต่อไปนี้
- 1) (5 คะแนน) จงพิสูจน์ว่า Minimum Set Cover เป็น NP-Complete โดยให้ Reduce ปัญหา minimum vertex cover ให้เป็นปัญหา Minimum Set Cover กำหนดให้ปัญหา minimum vertex cover เป็นดังนี้ มี Graph  $G = (V, E)$  โดยที่ ปม  $v_i$  นั้นกำกับด้วยน้ำหนัก  $w_i$  เราต้องการหา  $V' \subseteq V$  ที่ทำให้เส้นเชื่อมทั้งหมดของ  $G$  นั้นตกกระทบกับอย่างน้อย ปมใดปมหนึ่งใน  $V'$  และทำให้น้ำหนักรวมของปมใน  $V'$  มีค่าน้อยที่สุด
  - 2) (5 คะแนน) จงออกแบบ State Space Search algorithm แบบ DFS ที่ไม่ใช่ backtracking หรือ branch & bound ใด ๆ สำหรับปัญหา Minimum Set Cover โดยให้ระบุ state ที่ใช้ และวิเคราะห์ประสิทธิภาพในการทำงาน พร้อมกับ ยกตัวอย่าง 1 ตัวอย่างของ minimum set cover ที่มีค่า  $m$  ไม่น้อยกว่า 3 และเขียน state space tree ประกอบ

- 3) (5 คะแนน) ถ้าต้องการใช้เทคนิค Backtracking และ/หรือ Branch & Bound จงบรรยายละเอียดของวิธีการดังกล่าว ว่าใช้อะไรเป็นข้อจำกัดในการ backtracking และ/หรือ ยกตัวอย่าง bounding function ที่ใช้ พร้อมทั้งยกตัวอย่างประกอบให้เห็นภาพ (อย่าลืมว่า  $p_i > 0$  เสมอ) พร้อมทั้งระบุประสิทธิภาพเชิงเวลาของ bounding function มาด้วย

8. (10 คะแนน) ให้คุณเป็นผู้คุมเหมืองเพชรแห่งหนึ่งโดยเหมืองนี้มีอุโมงค์อยู่  $N$  อุโมงค์ที่จะสามารถเข้าไปขุดเพชรออกมาได้ คุณมี **คนงานอยู่  $M$  ( $1 \leq M \leq 10N$ ) คน** โดยที่แต่ละคนจะสามารถเข้าไปในอุโมงค์เพียงหนึ่งอันเพื่อทำการขุดเพชรขึ้นมา และเพื่อความปลอดภัยแต่ละอุโมงค์นั้นจะมีคนงานเข้าไปขุดได้ไม่เกิน 10 คน เพื่อการวางแผนการแบ่งคนงานที่ดี คุณจึงได้ใช้เครื่อง DMPS (Diamond Mine Positioning System) ในการวัดมูลค่าของเพชรที่มีอยู่ในแต่ละอุโมงค์ โดยเครื่อง DMPS นี้ได้คำนวณ Array 2 มิติ ของตัวเลข  $V[i][j]$  ( $1 \leq i \leq N$ ,  $1 \leq j \leq 10$ ) ให้คุณโดย  $V[i][j]$  ระบุมูลค่าเพชรที่จะขุดได้ในอุโมงค์ที่  $i$  หากใช้คนงาน  $j$  คนขุดในอุโมงค์ที่  $i$  โดยที่  $V[i][j]$  นั้นมีคุณสมบัติดังนี้  $V[i][k] \geq V[i][k-1]$  ( $2 \leq k \leq 10$ ) เนื่องจากยิ่งคนงานมากก็ยิ่งได้เพชรมาก และ  $V[i][l] - V[i][l-1] \geq V[i][l+1] - V[i][l]$  ( $2 \leq l \leq 9$ ) เนื่องจากกฎผลิตภาพหน่วยท้ายสุดลดลง (Law of diminishing return) งานของคุณคือออกแบบอัลกอริทึมที่มีประสิทธิภาพดีที่สุดเพื่อจะหาว่าจะให้คนงานเข้าไปในอุโมงค์แต่ละอุโมงค์กี่คน เพื่อให้ผลรวมของมูลค่าเพชรที่ขุดได้สูงสุดเท่าที่จะทำได้

ตัวอย่าง มีอุโมงค์ 3 อุโมงค์ คนงาน 6 คน

อุโมงค์	1 คน	2 คน	3 คน	4 คน	5 คน	6 คน	7 คน	8 คน	9 คน	10 คน
1	100	180	250	300	330	350	350	350	350	350
2	120	230	330	400	450	500	540	570	600	620
3	200	300	320	330	330	330	330	330	330	330

วิธีที่ดีที่สุดคือให้คนงาน 1 คนขุดเพชรในอุโมงค์ที่ 1, คนงาน 3 คนขุดเพชรในอุโมงค์ที่ 2, คนงาน 2 คนขุดเพชรในอุโมงค์ที่ 3 โดยมูลค่ารวมคือ  $100 + 330 + 300 = 730$

## Greedy

ให้  $B[i][j] = A[i][j] - A[i][j-1]$  และ  $B[i][1] = A[i][1]$

ให้  $\text{vector<int> } v[N] = \{1, 1, \dots, 1\}$  เก็บจำนวนคนที่เข้าไปดูโมเมนต์ +1

ให้  $\text{pq<int,int>}$  เก็บ  $\langle \text{มูลค่าเพชรที่เพิ่มขึ้น, เลขดูโมเมนต์} \rangle$

```
for (int i = 1; i ≤ n; i++)
```

```
    pq.push(< B[i][v[i]], i >)
```

```
sum = 0
```

```
while (m --) {
```

```
    cost, id = pq.top()
```

```
    pq.pop()
```

```
    sum += cost
```

```
    v[id] ++
```

```
    if (v[id] ≤ 10) pq.push(< B[id][v[id]], id >)
```

```
}
```

9. (10 คะแนน) ในข้อนี้ให้ถือว่าคุณมีฟังก์ชัน `float shortestPath(int n, int s, int t, vector<Edge>& e)` ซึ่งสามารถเรียกใช้ได้เลย ฟังก์ชันนี้คืนค่าความยาวของ shortest path จาก ปม s ไป ปม t ของกราฟมีน้ำหนักระบุทิศทางที่มี n ปมที่มีเส้นเชื่อม  $e[0]..e[e.size()-1]$  โดยนิยามของ Edge คือ

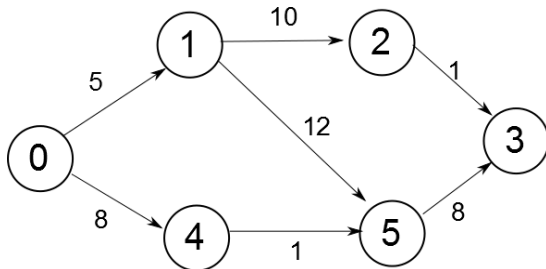
```
class Edge{
public:
    int i; //ปมต้นทาง
    int j; //ปมปลายทาง
    float w; //ระยะทาง โดยที่ w ≥ 0 เสมอ
};
```

ปัญหาที่คุณต้องแก้ในข้อนี้คือคุณจะต้องเขียนฟังก์ชัน

```
float shortestPathDiv2Twice(int m, int a, int b, vector<Edge>& f)
```

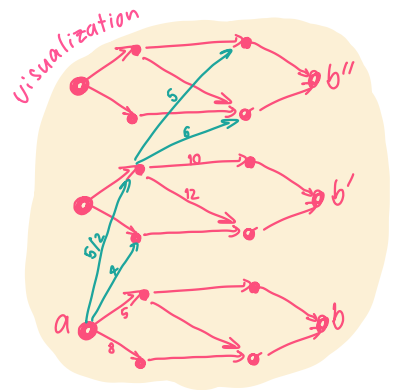
เพื่อหาระยะทางที่สั้นที่สุดจากปม a ไปยังปม b โดยเดินทางผ่านเส้นเชื่อมต่างๆที่ระบุใน f โดยที่คุณสามารถที่จะสามารถลดระยะทาง เส้นเชื่อมใดๆก็ได้ 2 เส้นลงครึ่งหนึ่ง

ตัวอย่าง



เดิน (0 → 1 → 2 → 3)  
 for 2.5 + 5 + 1 = 8.5

อ่าน  
 อันนี้



~~shortestPathDiv2Twice(6,0,3,f) เมื่อ f เก็บเส้นในกราฟระบุทิศทางข้างบนเอาไว้ จะต้องคืนค่า 9 ออกมา (โดยเดินทาง 0 → 4 → 5 → 3 และใช้การลดระยะครึ่งหนึ่งที่เส้น 0 → 4 และ 5 → 3 ทำให้ระยะรวมคือ 4 + 1 + 4 = 9)~~

จงเขียน code เพื่อเรียกใช้ฟังก์ชัน shortestPath (ด้วยกราฟใหม่ที่สร้างจาก input) เพื่อให้คำนวณค่า shortestPathDiv2Twice ให้ถูกต้องในช่องว่างข้างล่างนี้

```
float shortestPathDiv2Twice(int m, int a, int b, vector<Edge>& f) {
    int n,s,t;
    vector<Edge> e;
    // เขียน code ตรงนี้
    s = a    t = b
    n = 3 * m
    e = f
    for (Edge edge : f)
        edge2 = Edge(edge.i',edge.j', edge.w)
        edge3 = Edge(edge.i'',edge.j'',edge.w)
        edge4 = Edge(edge.i',edge.j'', edge.w/2)
        edge5 = Edge(edge.i, edge.j', edge.w/2)
        e.add(edge2,edge3,edge4,edge5)

    return shortestPath(n,s,t,e);
}
```

Suggest ให้อ่านตัวชื่อ by P'Top

เดิม node ชื่อ 0, 1, ..., n-1

ชั้น 2 ชื่อ n, n+1, ..., 2n-1

ชั้น 3 ชื่อ 2n, 2n+1, ..., 3n-1

copy  
 อีก 2 ชั้น

สร้าง node  
 2 ชั้น