FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2110327 ALGORITHM DESIGN

Year II, Second Semester, Final Examination, Aug 5, 2020 13:00-16:00

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

<u>หมายเหตุ</u>

1. ข้อสอบมีทั้งหมด 10 ข้อ ในกระดาษคำถามคำตอบ 7 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 ได้รับ สัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

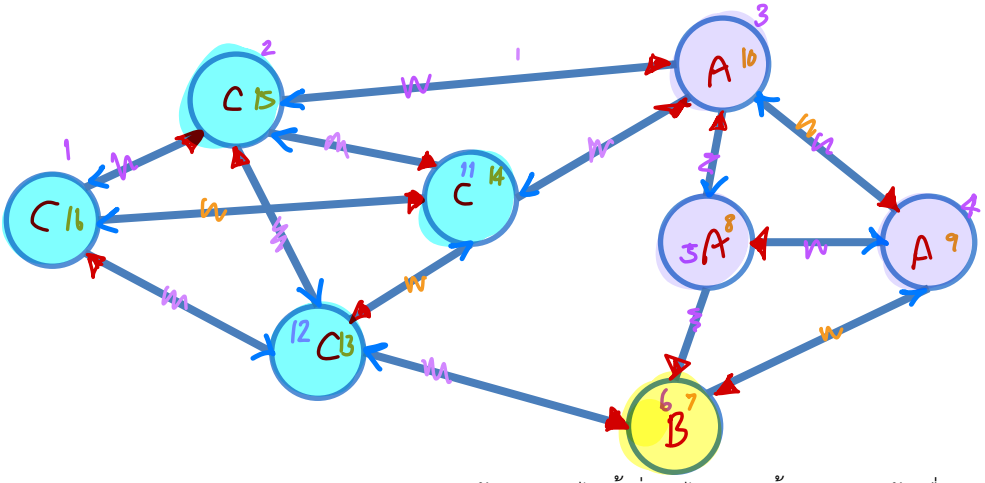
*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือหรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

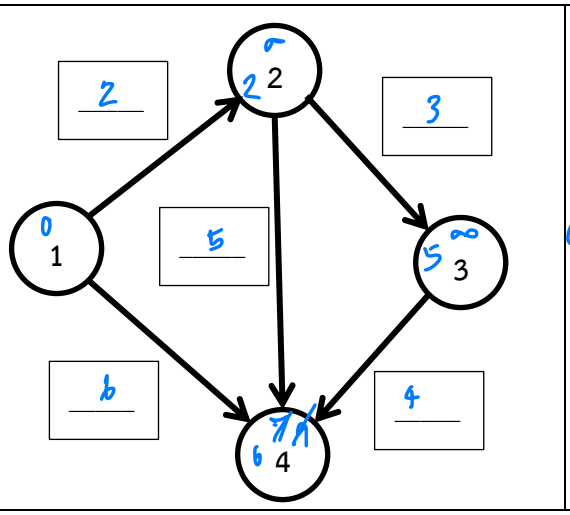
ลงชื่อนิสิต.....

วันที่.....

1. (5 คะแนน) จาก undirected graph ในรูปด้านล่างนี้ ให้ใส่หัวลูกศรลงในเส้นเชื่อมทุกเส้นเพื่อแปลงกราฟนี้ให้เป็น simple directed graph โดยที่ กราฟที่ได้จะต้องประกอบด้วย strongly connected component จำนวน 3 component โดยจะต้องมี 1 component ที่มี 4 ปมพอดี และมีอีก 1 component ที่มี 3 ปมพอดี และอีก 1 component ที่มี 1 ปมพอดี ให้ระบุตัวอักษรลงบนปมเพื่อบอก component ด้วย โดยให้ component เดียวกันมีตัวอักษรเดียวกัน

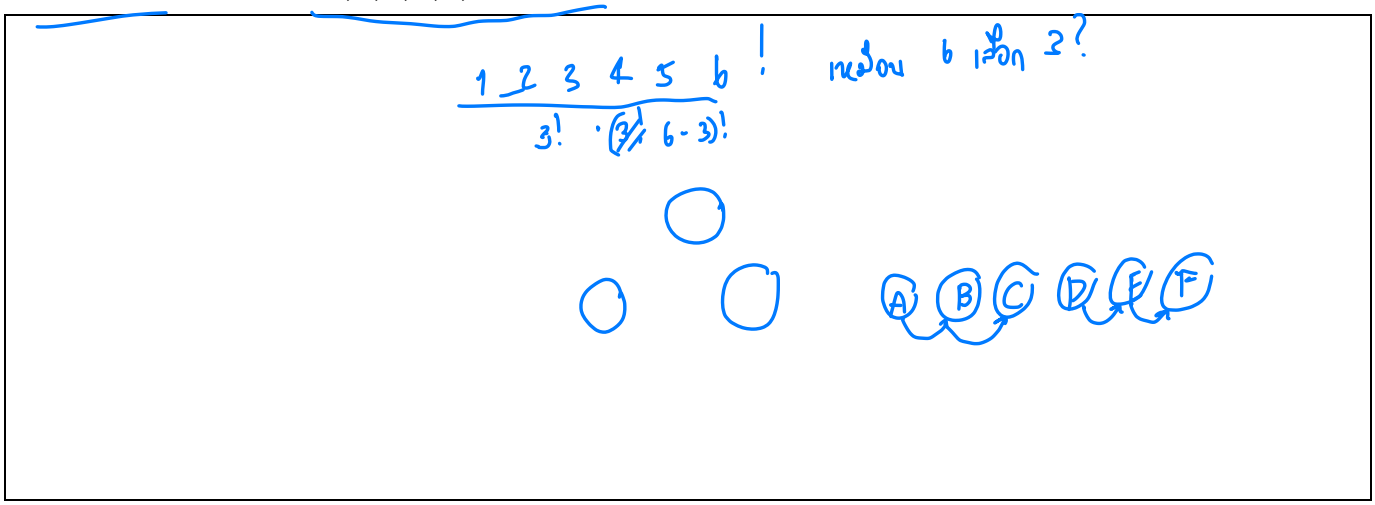


2. (5 คะแนน) มีกราฟแบบ directed weighted graph ดังรูปด้านล่างต่อไปนี้ ซึ่งเราไม่ทราบน้ำหนักของเส้นเชื่อมแต่ละเส้นเชื่อม อย่างไรก็ตาม เราทราบว่า เมื่อเราใช้ Bellman-Ford Algorithm ตาม pseudocode ด้านล่างนี้กับกราฟดังกล่าว โดยให้ปมเริ่มต้นคือปมหมายเลข 1 จะพบว่า บรรทัดหมายเลข 6: มีการทำงานทั้งหมด 5 ครั้งพอดี จงระบุน้ำหนักของเส้นเชื่อมของกราฟดังกล่าว โดยการเขียนตัวเลขลงบนเส้นเชื่อมในรูป เพื่อให้บรรทัดที่ 6: ทำงาน 5 ครั้งพอดี (ถ้าเป็นไปได้หลายคำตอบ ให้ตอบคำตอบใดก็ได้ที่ถูกต้อง)

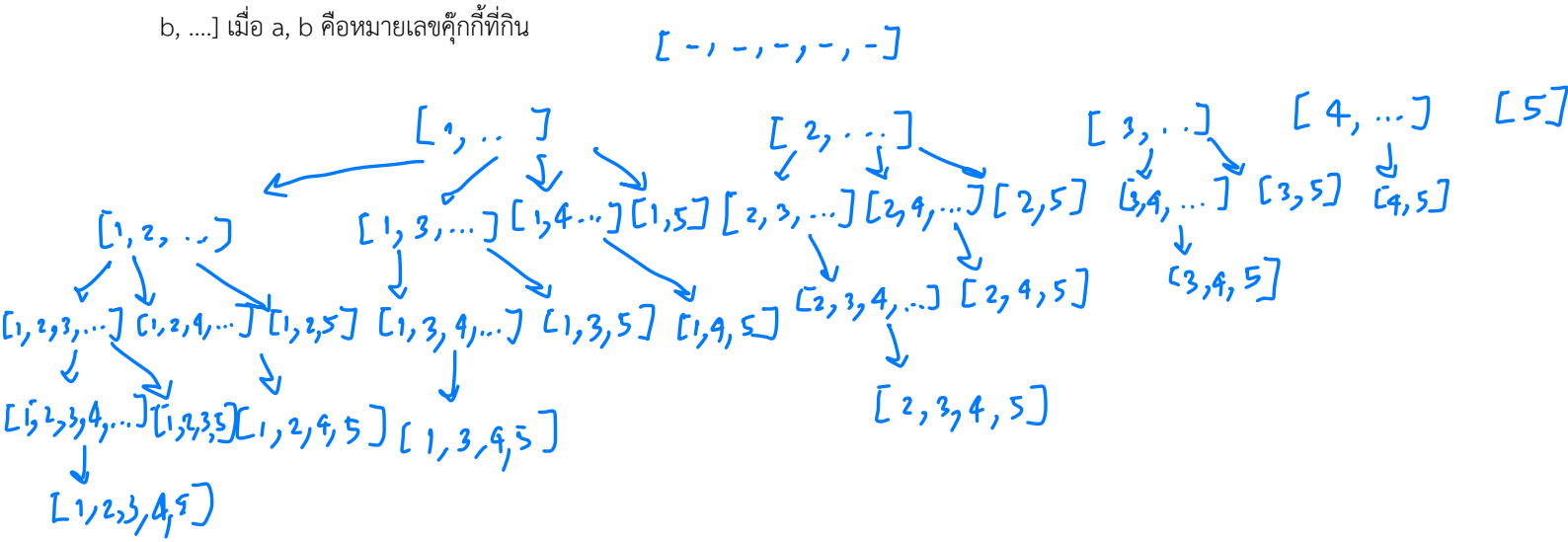



```
1: init D[1..4] to [0, ∞, ∞, ∞]
2: for round in 1..3
3:   B = D
4:   for each edge e = (u,v) in the graph
5:     if B[v] > D[u] + weight[e]
6:       B[v] = D[u] + weight[e]
7:   D = B
```

3. (5 คะแนน) มี directed graph ขนาด 6 ปม 4 เส้นเชื่อมอยู่กราฟหนึ่ง กำหนดให้ปมของกราฟนี้คือปม A, B, C, D, E, F เราทราบว่า topological sorting ของกราฟนี้มีทั้งหมด 6! / (3! * 3!) = 20 รูปแบบที่แตกต่างกันพอดี จงวาดกราฟดังกล่าว โดยกราฟจะต้องมี 6 ปม และ 4 เส้นเชื่อม และกราฟมีปม A, B, C, D, E, F เท่านั้น



4. (5 คะแนน) ปัญหาเรามีคุ๊กกี้อยู่ N ชิ้น (กำกับด้วยหมายเลข 1 – N) เรากำลังจะกินคุ๊กกี้เหล่านี้ทีละชิ้น อย่างไรก็ตาม เราตั้งกฎการกินคุ๊กกี้ไว้ว่าหากเรากินคุ๊กกี้หมายเลข i ไปแล้ว ครั้งต่อไปเราจะต้องกินคุ๊กกี้หมายเลข**มากกว่า** i+1 เสมอ (คุกกี้ชิ้นแรกจะกินชิ้นหมายเลขใดก็ได้) จงวาด State Space Tree ที่สร้างรูปแบบการกินคุ๊กกี้ทั้งหมดที่เป็นไปได้ เมื่อ N = 5 โดยกำหนดให้ state ที่วาดในแต่ละปมอยู่ในรูปแบบ [a, b, ...] เมื่อ a, b คือหมายเลขคุ๊กกี้ที่กิน

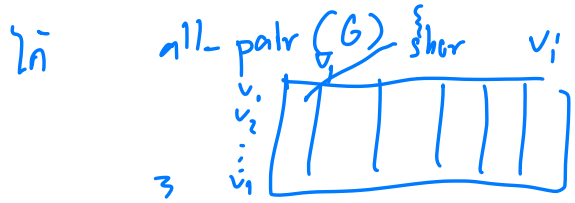


5. (10 คะแนน) กำหนดให้ shortest(G,s,t) เป็นปัญหาการหาระยะทางสั้นสุดในกราฟ G จากปม s ไปยังปม t และกำหนดให้ all_pair(G) เป็นปัญหาการหาระยะทางสั้นสุดของทุก ๆ คู่ปมในกราฟ G จงตอบคำถามต่อไปนี้

5.1. ปัญหา shortest(G,s,t) สามารถ polynomially reduce ไปเป็นปัญหา all_pair(G) ได้หรือไม่ เพราะเหตุใด จงตอบโดยสังเขป

ไปได้

5.2. ปัญหา all_pair(G) สามารถ polynomially reduce ไปเป็นปัญหา shortest(G,s,t) ได้หรือไม่ เพราะเหตุใด จงตอบโดยสังเขป



5.3. สมมติว่า shortest(G,s,t) และ all_pair(G) สามารถ polynomially reduce ไปหากันและกันได้ และเรามีอัลกอริทึมสำหรับแก้ปัญหา shortest(G,s,t) ที่ใช้เวลา O(f(n)) เราสามารถตอบได้หรือไม่ว่า เราสามารถสร้างอัลกอริทึมสำหรับแก้ปัญหา all_pair(G) ได้ในเวลา O(f(n)) อย่างแน่นอน เพราะเหตุใด

✓ เพราะ shortest ∈ all
all ∈ shortest

5.4. สมมติว่า shortest(G,s,t) และ all_pair(G) สามารถ polynomially reduce ไปหากันและกันได้ เราสามารถตอบได้หรือไม่ว่า P = NP เป็นจริง เพราะเหตุใด

ไม่สามารถตอบได้

สำหรับข้อ 6 – 10 นั้น จะเป็นการออกแบบอัลกอริทึม ในแต่ละข้อนั้นสามารถอธิบายอัลกอริทึมที่ออกแบบด้วย รหัสเทียม (pseudo code) หรือว่า programming language ภาษาใดที่เคยเรียนมาก็ได้ แต่ทุกข้อให้ระบุประสิทธิภาพเชิงเวลาด้วย
นอกจากนี้ นิสิตสามารถเรียกใช้ data structure หรือว่า อัลกอริทึมใด ๆ ที่อยู่ในบทเรียนได้เลย โดยไม่จำเป็นต้องเขียน code ของอัลกอริทึมนั้น แต่ต้องระบุว่า input ของอัลกอริทึมที่เรียกใช้ให้ชัดเจน ไม่กำกวม และ output ที่ได้เป็นอะไรอยู่ในรูปแบบใด และนำไปใช้อย่างไรให้ชัดเจน ไม่กำกวม

6. (10 คะแนน) มี directed simple graph G = (V,E) อยู่ เราต้องการทราบว่า ในกราฟนี้ มี simple cycle (cycle ที่ไม่วิ่งผ่านปมใดซ้ำเลย) ที่มีความยาว k เส้นเชื่อม หรือไม่ จงออกแบบอัลกอริทึมแบบ state space search สำหรับปัญหานี้ และหากทำได้ ให้เสนอด้วยว่าควรจะใช้ backtracking อย่างไร *กำหนดให้ input ของปัญหานี้คือ G = (V,E) และ K* โดยให้ตอบคำถามต่อไปนี้

6.1. จงระบุ state สำหรับปัญหานี้ โดยให้ระบุว่า state นั้นเก็บข้อมูลอะไรบ้าง และเก็บในตัวแปรประเภทใด พร้อมยกตัวอย่าง และระบุถึงขนาดของ search space

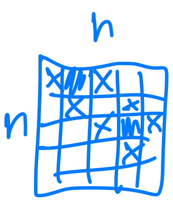
6.2. จงเขียน pseudocode หรือ code สำหรับแก้ปัญหาดังกล่าว โดยกำหนดให้ input ของโจทย์คือจำนวนเต็ม k และ n ซึ่งระบุจำนวนปมในกราฟ กำหนดให้ปมในกราฟแทนด้วยตัวเลข 1 ถึง n และ adjacency list e ซึ่งระบุเส้นเชื่อมในกราฟ กล่าวคือ e[a] จะเก็บอาเรย์ที่ระบุปมทั้งหมดที่มีเส้นเชื่อมมาจาก ปม a

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ_____

6.3. ระบุถึงเทคนิค backtracking ที่ใช้ในข้อ 6.2 (ถ้ามี) พร้อมทั้งยกตัวอย่างเหตุการณ์ที่เกิดการ backtrack

7. (10 คะแนน) ปัญหาการเจาะน้ำมันเป็นดังนี้ มีแหล่งน้ำมันแหล่งหนึ่ง แผนที่ของแหล่งเจาะน้ำมันนี้สามารถเขียนแทนได้ด้วยอาเรย์สองมิติ A[1..n][1..n] โดยที่ A[i][j] นั้นจะระบุว่า หากเราสร้างแท่นเจาะน้ำมันที่ช่อง (i, j) แล้ว เราจะได้กำไรเป็นเงินกี่บาท อย่างไรก็ตาม หากเราสร้างแท่นเจาะน้ำมันที่ช่องใดแล้ว เราจะไม่สามารถสร้างแท่นเจาะน้ำมันที่ช่องที่มีด้านติดกันได้อีก (กล่าวคือ หากสร้างที่ช่อง (i, j) แล้ว จะไม่สามารถสร้างที่ช่อง (i-1, j) , (i, j-1), (i+1, j) หรือ (i, j+1) ได้) จงออกแบบอัลกอริทึมแบบ Least Cost Search สำหรับการหาว่ากำไรรวมมากสุดจากการสร้างแท่นเจาะน้ำมันจำนวน K แท่นเป็นเท่าไร *กำหนดให้ input ของปัญหานี้คือ A, n และ K* โดยให้ตอบคำถามต่อไปนี้

7.1. จงระบุ state สำหรับปัญหานี้ โดยให้ระบุว่า state นั้นเก็บข้อมูลอะไรบ้าง และเก็บในตัวแปรประเภทใด พร้อมยกตัวอย่าง และระบุถึงขนาดของ search space



vector <int,int> dig → vector เลือก เก็บช่องน้ำมันที่จะขุด
int total → บอกว่าขุดไปทั้งหมดกี่ช่อง

7.2. จงเขียน pseudocode หรือ code สำหรับแก้ปัญหาดังกล่าว โดยกำหนดให้ input ของโจทย์คือจำนวนเต็ม k และอาเรย์ 2 มิติ A[1..n][1..n]

```
dig, A[1..n][1..n] (copy), total
B = A
sum = 0
for ( (x,y) : dig) {
    sum += B[x][y]
    B[x][y] = B[x+1][y] = B[x-1][y] = B[x][y-1] = ... = -∞
}
for (i) (j) if B[i][j] ≠ -∞
    pri_order.push( B[i][j] );
}
while (k - total --) → sum += pri.top ; pri.pop()

searchForOil (A[1..n][1..n], total, vector<int>)
```

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ____________________________

7.3. อธิบายถึงเทคนิค Branch and Bound ที่ใช้ในข้อ 7.2 (ถ้ามี) พร้อมด้วย Heuristic Function ที่ใช้

8. (10 คะแนน) เรามี undirected graph G = (V,E) อยู่กราฟหนึ่ง และต้องการสร้างกราฟ directed acyclic graph (กราฟที่ไม่มี cycle) H = (V,E') โดยที่กราฟ H มีปมเหมือน G ทุกประการ และเส้นเชื่อม E' นั้นเกิดจากการกำหนดทิศทางให้กับเส้นเชื่อมใน E กล่าวอีกนัยหนึ่งคือ เส้นเชื่อม e ซึ่งเชื่อมปม a และ b ใด ๆ จะเป็นสมาชิกของ E ได้ ก็ต่อเมื่อ มี e1 = (a,b) หรือมี e2 = (b,a) อยู่ใน E' เท่านั้น และขนาดของ E และ E' จะต้องเท่ากัน จงออกแบบอัลกอริทึมสำหรับการสร้าง H ดังกล่าวจาก G *กำหนดให้ input ของปัญหานี้คือ G = (V,E)*

```
vector<bool> visited (V, false)
  for (i = 0; i < V; i++) {
        if (!visited[i]) {
              dfsAddEdge(i);
        }
  }

dfsAddEdge(i) {
      visited[i] = true                 direct edge e->i
      for ((i,e) ∈ E)                       ↑
          if (visited[e]) edge.push( (e,i) )
          else   edge.push((i,e)); dfsAE(e)
}
```

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ______________________________

9. (10 คะแนน) นิสิตรุ่น CP-XX จำนวน N คน เมื่อจบการศึกษาออกไปแล้ว ได้ตั้งบริษัทขึ้นมาคนละ 1 บริษัท (กำหนดให้แต่ละบริษัทระบุได้ด้วยหมายเลข 1 ถึง N) เมื่อเวลาผ่านไป บริษัทต่าง ๆ เหล่านี้ได้จับมือเป็นพันธมิตรกันเพื่อความแข่งแกร่งทางธุรกิจ ให้ merge[1..k] เป็นอาเรย์ที่เก็บข้อมูลว่าบริษัทใดมีการจับมือกันบ้าง เรียงตามลำดับเวลาในการจับมือ กล่าวคือ การจับมือครั้งแรกเก็บอยู่ใน merge[1] และครั้งถัดไปอยู่ใน merge[2] ไปเรื่อย ๆ โดยให้ merge[i].first และ merge[i].second นั้นเป็นหมายเลขของบริษัทที่จับมือกัน บริษัทที่มีความสัมพันธ์กันผ่านการจับมือกันจะถือว่าอยู่ในกลุ่มบริษัทเดียวกัน (เช่น บริษัท 1 จับมือกับ 2 และ 2 จับมือกับ 3 เราจะถือว่า บริษัท 1,2,3 นั้นอยู่ในกลุ่มบริษัทเดียวกัน) เราอยากทราบว่า หลังจากเกิดการจับมือจนถึงครั้งที่ k นั้น มีกลุ่มบริษัททั้งหมดกี่กลุ่ม จงออกแบบอัลกอริทึมสำหรับแก้ปัญหานี้ *กำหนดให้ input ของปัญหานี้คือ N, merge และ k*

```
vector<int> parent [1,2,...,N]
vector<int> height (0, N)
mergeParent (p1, p2) {
        if (h[p1] > h[p2])
              parent[p2] = p1
        else {    p1 = p2
                  if ==    height[p2]++

}
find (p) {     while (p ≠ Parent[p]) { p = parent p } return p }
  for a, b : merge {
          mergePa ( find(a), find(b) );
  }
  for int i = 1  i ≤ N ; i++     if    parent[i] = i    count ++
      return count
```

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ KlogN

10. (10 คะแนน) กำหนดให้มี undirected weighted graph G = (V,E) โดยให้ w(e) เป็นฟังก์ชันซึ่งระบุระยะทางของเส้นเชื่อม e และรับประกันว่า w(e) มีค่าเป็นบวกเสมอสำหรับ e ใด ๆ เราต้องการเดินทางจากปม s ไปยังปม t โดยใช้ระยะทางรวมน้อยสุด อย่างไรก็ตาม เรามีสิทธิ์ที่จะเพิ่มเส้นเชื่อม x ซึ่ง w(x) = K เข้าไปในกราฟ G โดยให้เส้นเชื่อมนี้เชื่อมระหว่างปมใด ๆ ก็ได้**ที่ไม่ใช่ s หรือ t** จงออกแบบอัลกอริทึมเพื่อหาว่า เราควรจะเพิ่มเส้นเชื่อม x ระหว่างคู่ปมใด เพื่อให้ระยะทางสั้นสุดจาก s ไปยัง t นั้นมีค่าน้อยที่สุดเท่าที่เป็นไปได้ โดยให้อัลกอริทึมนี้ตอบระยะทางสั้นสุดหลังจากเพิ่มเส้นเชื่อม x *กำหนดให้ input ของปัญหานี้คือ G, w, s, t และ K*

```
bellman ford (s, t) {
    parent (-1, V)
    weight (∞, V)
    wei[s] = 0;
    for (i = 0; i < V-1; i++) {
        for ( e in eadge {
            a, b → e
            if  weigh[b] > weigh[a] + W(e)
                p[b] = a
        }
    }
    x = parent[t]   , max = 0 , a, b ?
    while (x != s) {
        max = max(max, W[x] - W[parent[x]])
        if > max   a, b = x, parent[x]
    }
    return a, b
```

ประสิทธิภาพในการทำงานของอัลกอริทึมนี้คือ O(V)