Algorithm Design and Analysis

วิชาบังคับก่อน: 204251 และ 206281
ผู้สอน: ตอน 1 ผศ. เบญจมาศ ปัญญางาม เรียน ห้อง 207
ตอน 2 ผศ. ดร. จักริน ชวชาติ เรียน ห้อง 209

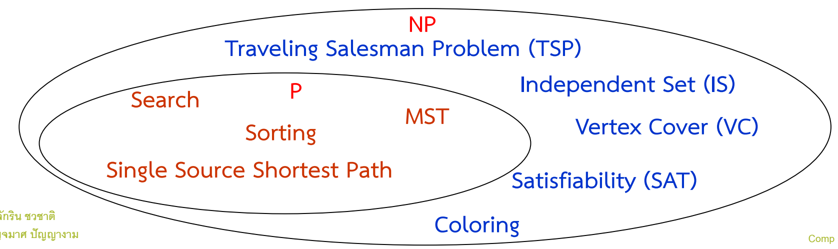
วันสอบปลายภาค : วันศุกร์ที่ 13 พ.ย. 2563
เวลา 8:00 -11:00 น. (ตามประกาศมหาวิทยาลัย)

บทที่ 11

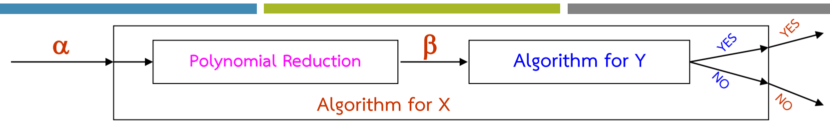
NP-Completeness Part II

P และ NP

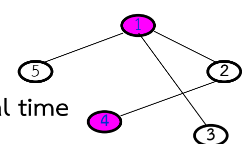
- เราจัดกลุ่ม Decision problem ต่างๆ เป็น กลุ่ม P และ NP และทราบว่า P $\subseteq$ NP
  - ปัญหาในกลุ่ม P เป็นจะปัญหาง่าย เพราะมีอัลกอริทึมที่ใช้หาคำตอบของปัญหาได้ภายใน Polynomial time
  - ปัญหาในกลุ่ม NP เป็นปัญหาที่สามารถทวนสอบ (Verify) ได้ใน Polynomial-time โดยมีปัญหาต่างๆ มากมายใน NP ยังไม่มีใครหาอัลกอริทึมที่ใช้เวลาในการหาคำตอบภายใน Polynomial time ได้ เช่น TSP , IS, VC, SAT เป็นต้น

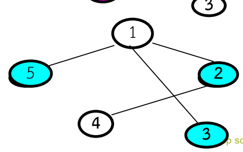


Reduction : VC ≤p IS

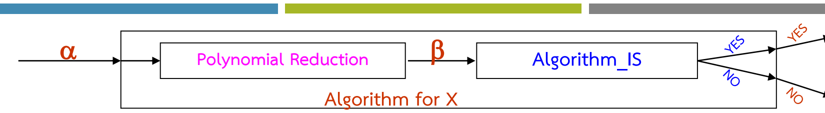


- Decision problem สำหรับ Vertex Cover : กำหนด G (V,E) จะมีเซตย่อย $V_{vc} \subseteq V$ ที่ cover ทุกเส้นเชื่อมโดยที่ขนาดของ V_{vc} ขนาดเท่ากับ k หรือไม่
- Decision problem สำหรับ Independent Set: กำหนด G (V,E) จะมีเซตย่อย $V_I \subseteq V$ ที่จะไม่มีเส้นเชื่อมระหว่างโหนดใดๆ ใน V_I โดยที่ขนาดของ V_I ขนาดเท่ากับ k หรือไม่
- หาก VC ≤p IS แสดงว่ามีอัลกอริทึมสำหรับแปลง instance ของปัญหา VC ไปเป็น instance ของปัญหา IS ภายใน polynomial time
- โดยที่ใช้อัลกอริทึมที่แก้ปัญหา IS เพื่อหาคำตอบของ IS
- คำตอบที่ได้จะเป็นคำตอบของ VC ด้วย (Yes/No)





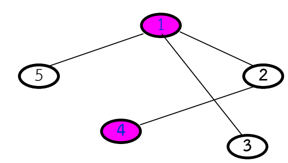
Reduction : VC ≤p IS



- ตัวอย่าง Instance สำหรับปัญหา VC (α)

V={1,2,3,4,5} E={{1,2}, {1,3},{1,5}, {2,4}}

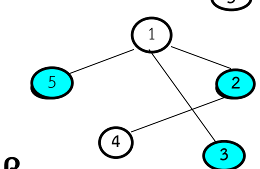
k = 2 V_{vc}={1,4}



- เปลี่ยนเป็น Instance สำหรับปัญหา IS (β)

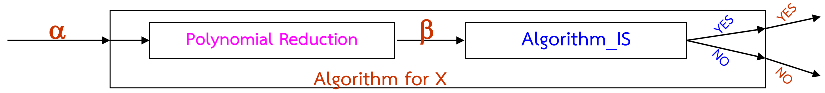
V={1,2,3,4,5} E={{1,2}, {1,3},{1,5}, {2,4}},

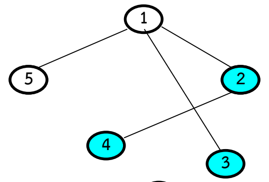
k = |V| - k=3 V_I= V-V_{vc}={2,3,5}



จะพบว่า Algorithm_IS ให้คำตอบเป็น Yes สำหรับ instance β ซึ่งคำตอบสำหรับ instance α จะให้คำตอบเป็น Yes เช่นกัน

Reduction : VC ≤p IS

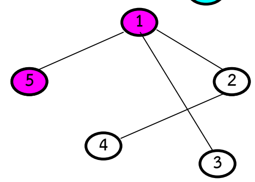




- ตัวอย่าง Instance สำหรับปัญหา VC (α)

  V={1,2,3,4,5} E={{1,2}, {1,3},{1,5}, {2,4}}

  k = 3 V_{vc}={2,3,4}



- เปลี่ยนเป็น Instance สำหรับปัญหา IS (β)

  V={1,2,3,4,5} E={{1,2}, {1,3},{1,5}, {2,4}},

  k = |V| - k=2 V_I= V-V_{vc}={1,5}

จะพบว่า Algorithm_IS ให้คำตอบเป็น No สำหรับ instance β

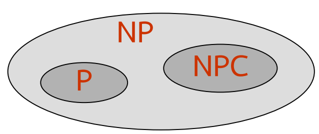
ซึ่งคำตอบสำหรับ instance α จะให้คำตอบเป็น No เช่นกัน

NP-complete (NPC)

- ในปี ค.ศ.1971 คุก (Cook) ได้แสดงให้เห็นว่า ปัญหา SAT เป็นปัญหาที่ยากที่สุด (NP-hard) ใน NP โดยพิสูจน์ว่า
  - ทุกปัญหาใน NP สามารถลดรูป (reduce) ภายใน Polynomial time ไปเป็นปัญหา SAT ได้ทั้งหมด (for every p' ∈ NP , $p' \leq_p p$)
- ในปี ค.ศ.1971 คาร์ป (Karp) ได้แสดงให้เห็นว่ามีปัญหามากมายที่มีความยากง่ายเทียบเท่ากับปัญหา SAT
- ซึ่งเป็นที่มาของกลุ่มปัญหา NP-complete คือกลุ่มปัญหาที่ยากที่สุดใน NP และเป็นปัญหาที่ยากง่ายเท่าเทียมกันหมด

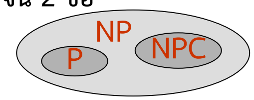
NP-complete (NPC)

- ดังนั้นหากใครพบอัลกอริทึมที่ใช้เวลาภายใน Polynomial time ในการหาคำตอบของปัญหาสักปัญหาหนึ่งในกลุ่ม NP-complete ได้แสดงว่าทุกๆ ปัญหาในกลุ่ม NPC เป็นปัญหาง่ายทั้งสิ้น
  - นั่นคือพิสูจน์ได้ว่า P = NP
- ในทางกลับกันหากมีใครพิสูจน์ได้ว่า มีสักปัญหาหนึ่งใน NPC เป็นปัญหายากก็สรุปได้ว่าทุกๆ ปัญหาใน NPC เป็นปัญหายากทั้งสิ้น นั่นคือ P ≠ NP หรือ P ⊆ NP
- ดังนั้นกลุ่มปัญหา P NP และ NPC



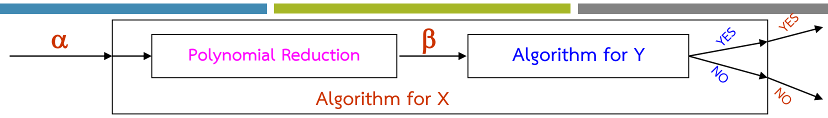
NP-complete (NPC)

- การจะสรุปว่าปัญหา x เป็น NP-complete จะต้องพิสูจน์ 2 ข้อ
  1. ปัญหา A ∈NP
  2. ปัญหา A เป็น NP-hard



  คือพิสูจน์ว่าทุกๆ ปัญหาใน NP สามารถลดรูปไปเป็นปัญหา A ได้ภายใน polynomial time (for all B ∈ NP , $B \leq_p A$)
- เนื่องจาก SAT เป็น NP และ SAT ก็เป็น NP-hard ด้วย
  - ดังนั้น SAT เป็น NPC
- การพิสูจน์ในข้อ 1 สามารถทำได้ไม่ยากนัก โดยการหาอัลกอริทึมที่ทวนสอบคำตอบของปัญหา A ได้ภายใน polynomial time
- แต่การพิสูจน์ในข้อ 2 เป็นเรื่องยากมากที่จะแสดงให้เห็นว่าทุกปัญหาใน NP สามารถลดรูปไปเป็นปัญหา A ได้ภายใน polynomial time

Reduction & NP-complete



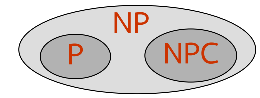
- ถ้า X ≤p Y สรุปได้ว่าปัญหา X ไม่ยากกว่าปัญหา Y หรือปัญหา Y ไม่ง่ายกว่าปัญหา X (นั่นคือ X และ Y เป็นปัญหายากง่ายเท่ากัน)
- หาก X ≤p Y และ Y ≤p Z แสดงว่า X ≤p Z ได้เช่นกัน

- หากสมมุติว่าปัญหา B เป็น NPC นั่นคือทุกปัญหาใน NP สามารถลดรูปไปเป็นปัญหา B ได้ภายใน polynomial time (คุณสมบัติของ NP-complete)
- ถ้า B ≤p A ก็แสดงว่าทุกปัญหาใน NP สามารถลดรูปไปเป็นปัญหา A ได้ภายใน polynomial time ได้เช่นกัน

Reduction & NP-complete

- ซึ่งเราทราบว่ามีปัญหาอื่นๆ ใน NP (อย่างน้อยก็ SAT) เป็น NPC อยู่แล้ว การจะสรุปได้ว่า ปัญหา A จะอยู่ในกลุ่ม NPC สามารถใช้การพิสูจน์ 2 ข้อนี้แทน

1. ถ้าปัญหา A ∈NP และ
2. ถ้า B ≤p A โดยที่ B เป็นปัญหาในกลุ่ม NP-Complete

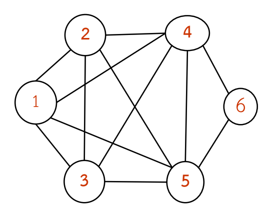


The SAT Problem

- SAT เป็นปัญหาของนิพจน์บูลีนสำหรับ n ตัวแปร ที่เรามีวิธีในการกำหนดค่าตัวแปรในนิพจน์แล้วสามารถทำให้นิพจน์บูลีนให้ค่าเป็นจริงได้หรือไม่ เช่น
  - $((x_1 \rightarrow x_2) \vee ((x'_1 \leftrightarrow x_3) \vee x_4))' \wedge x'_2$
- CNF (Conjunctive Normal Form)-SAT เป็นปัญหาของนิพจน์บูลีนในรูปแบบ $E_1 \wedge E_2 \wedge E_3 .... \wedge E_n$ โดยแต่ละนิพจน์ย่อย E_i จะอยู่ในรูปแบบของการ OR เช่น
  - $(x_1 \vee x'_2) \wedge (x'_1 \vee x_3 \vee x_4) \wedge (x'_5)$
  - $(x_1 \vee x'_2 \vee x'_3) \wedge (x'_1 \vee x_3 \vee x_4) \wedge (x'_5 \vee x_3 \vee x_4)$

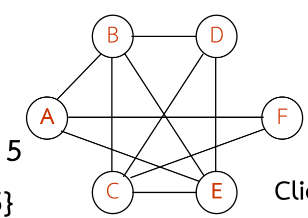
Clique Problem

- กำหนด G=(V,E) เป็นกราฟแบบไม่ระบุทิศทาง และ k คือจำนวนเต็ม
- Clique : G มีคลิก (clique) ขนาดอย่างน้อย k หรือไม่
- คลิกคือกราฟย่อยที่เป็น complete graph
- ขนาดของคลิก (V_c) คือจำนวนโนดในกราฟย่อย



Clique ขนาด 5

V_c={1,2,3,4,5}



Clique ขนาด 4

V_c={B,C,D,E}

คำถาม:

ปัญหานี้เป็น NP หรือไม่ ?

CLIQUE is NPC

- จะแสดงว่า CLIQUE เป็น NPC ต้องแสดงให้เห็นว่า

1. CLIQUE เป็น NP
2. CNF-SAT $\leq_p$ CLIQUE (รู้มาก่อนแล้วว่า CNF-SAT เป็น NPC) นั่นคือจะแสดงว่าสามารถลดรูปปัญหา CNF-SAT ไปยังปัญหา CLIQUE ภายใน polynomial time

CLIQUE ∈ NP?

- กำหนดกราฟ G=(V,E) และเซต V‘ ⊆ V ซึ่งเป็นเซตคำตอบที่ต้องการทวนสอบ เราต้องการหาอัลกอริทึมสำหรับทวนสอบเซตคำตอบ V’ ว่าเป็น k- CLIQUE ?

VerifyingClique{

1) ตรวจสอบว่า | V' | = k (มีจำนวน k โหนด) และทุกโหนดใน V' มีชื่อต่างกัน ซึ่งใช้เวลา linear time
2) ตรวจสอบทั้ง k โหนด ใน V' ว่าแต่ละโหนด (v) มีเส้นเชื่อมถึงโหนดอื่นๆ (u) ครบหรือไม่ โดยที่ <v,u> ∈ E ด้วยซึ่งใช้เวลา quadratic time.

}

- เนื่องจากอัลกอริทึมข้างต้นทำงานภายใน polynomial time ดังนั้น สรุปได้ว่า CLIQUE ∈ NP

CNF-SAT $\leq_p$ CLIQUE

- จะแสดงว่าสามารถลดรูปปัญหา k-CNF-SAT ไปเป็น k-CLIQUE ภายใน polynomial time
- กำหนดนิพจน์ $F=C_1\wedge C_2\wedge\ldots\wedge C_k$ เป็นอยู่ในรูป CNF-SAT ที่ประกอบด้วยนิพจน์ย่อยจำนวน k นิพจน์ เช่น

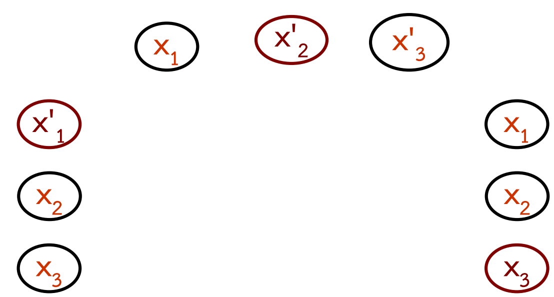
$$F=C_1\wedge C_2\wedge C_3=(x_1\vee x'_2\vee x'_3)\wedge(x'_1\vee x_2\vee x_3)\wedge(x_1\vee x_2\vee x_3)$$

CNF-SAT $\leq_p$ CLIQUE

$$F=C_1\wedge C_2\wedge C_3=(x_1\vee x'_2\vee x'_3)\wedge(x'_1\vee x_2\vee x_3)\wedge(x_1\vee x_2\vee x_3)$$

- ขั้นตอนการลดรูป k-CNF-SAT ไปเป็น k-CLIQUE (สร้างกราฟ G=(V,E)) ดังนี้

1) แต่ละนิพจน์ย่อย $C_A=(l_1^A\vee l_2^A\vee\ldots\vee l_n^A)$ ให้แทนตัวแปรแต่ละตัวแปรใน C_A ไปสร้างเป็นโหนดต่างๆ ($v_1^A, v_2^A, \ldots, v_n^A \in V$) ของกราฟ

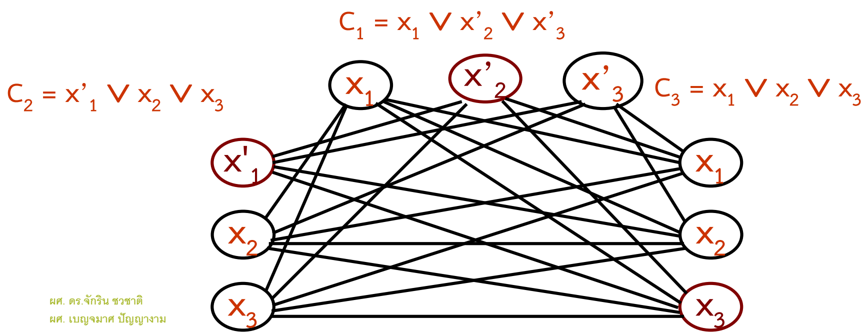


CNF-SAT ≤p CLIQUE

$F=C_1 \wedge C_2 \wedge C_3=(x_1 \vee x'_2 \vee x'_3)\wedge(x'_1 \vee x_2 \vee x_3)\wedge(x_1 \vee x_2 \vee x_3)$

2) ให้เพิ่มเส้นเชื่อมระหว่างโหนด v_i^A และ v_j^B โดย
- v_i^A และ v_j^B ต้องอยู่คนละนิพจน์ย่อยกัน ($A \neq B$)
- v_i^A และ v_j^B ต้องไม่เป็น negation กัน เช่น ถ้า $v_i^A = x'_1$ ค่า v_j^B ต้องไม่เป็น x_1

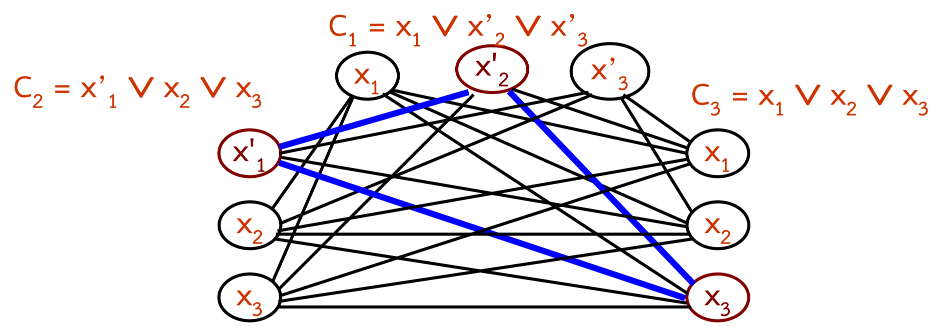
ตัวอย่าง 3-CNF-SAT ไปเป็น 3-CLIQUE



CNF-SAT ≤p CLIQUE

$F=C_1 \wedge C_2 \wedge C_3=(x_1 \vee x'_2 \vee x'_3)\wedge(x'_1 \vee x_2 \vee x_3)\wedge(x_1 \vee x_2 \vee x_3)$

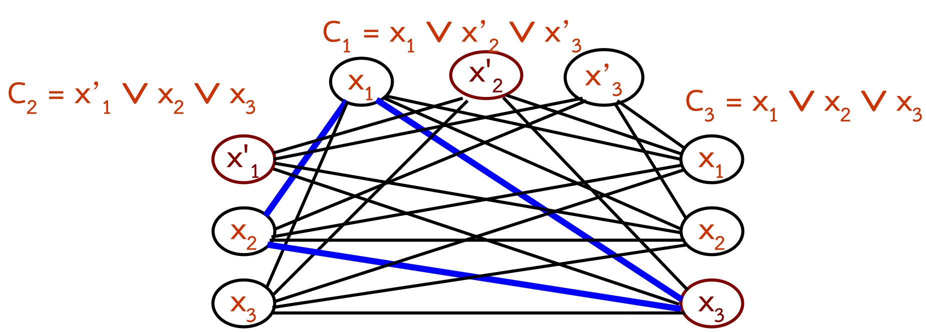
หากมีเซตคำตอบ $\{x_2=0,\ x_1=0$ และ $x_3=1\}$ สำหรับ 3-CNF-SAT
เราจะพบ V' ที่เป็นคำตอบของปัญหา 3-clique



CNF-SAT ≤p CLIQUE

$F=C_1 \wedge C_2 \wedge C_3=(x_1 \vee x'_2 \vee x'_3)\wedge(x'_1 \vee x_2 \vee x_3)\wedge(x_1 \vee x_2 \vee x_3)$

หากกำหนด $V' = \{x_1, x_2, x_3\}$ ที่เป็นคำตอบของปัญหา 3-clique
จะได้เซตคำตอบ $\{x_1=1,\ x_2=1$ และ $x_3=1\}$ สำหรับ 3-CNF-SAT



Assignment#07 ส่วนที่ 2

- กำหนดนิพจน์ในรูป CNF-SAT สำหรับ 4 clauses ดังนี้

F = (x' + y + z) (x + y' + z) (y + z) (x' + y' + z')

จงแสดงขั้นตอนการลดรูปไปเป็นปัญหา 4-CNF-SAT ไปเป็น 4-CLIQUE

NP-complete problems & reductions