Algorithm Design and Analysis

บทที่ 10

Network flow Part II

อ. ดร. จักริน ชวชาติ
อ. เบญจมาศ ปัญญางาม

Ford-Fulkerson Augmenting Path Algorithm

```
while(there exists an augmenting path){
        Find augmenting path P
        Compute bottleneck capacity of P
        Augment flow along P
}
```

คำถาม

Q:การทำเช่นนี้ทำให้ได้ max flow ใช่หรือไม่

A:ใช่

Ford-Fulkerson Algorithm: Analysis

Assumption: ให้ความจุมีค่าเป็นจำนวนเต็ม

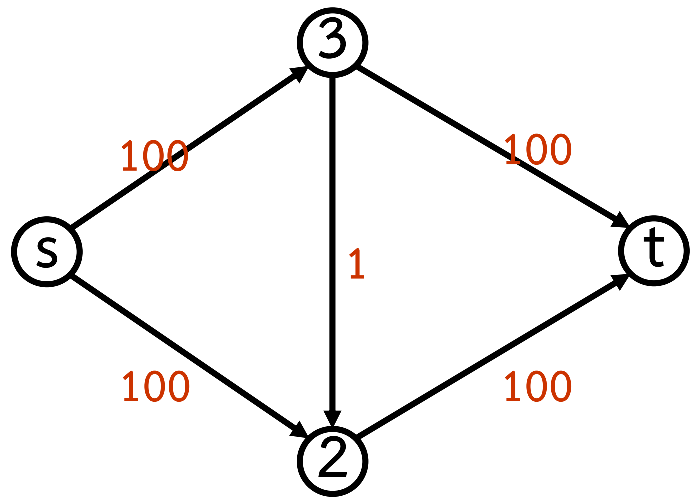
ในแต่ละรอบการทำงานจะมีการหา st-path ซึ่งใช้เวลาในการทำงาน O(N+M)

เราจะส่ง flow อย่างน้อย 1 หน่วยผ่าน path นี้

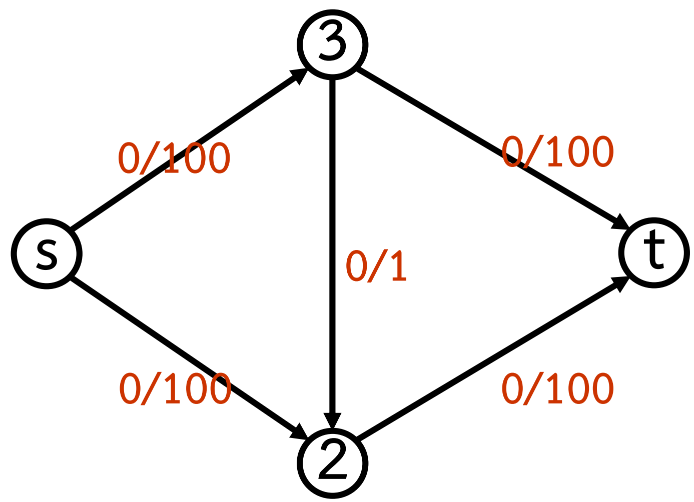
ถ้า max-flow มีค่าเป็น f* แล้วเวลาในการทำงานของอัลกอริทึมมีค่าเป็น O((N+M)*|f*|)

Choosing Good Augmenting Path

residual graph

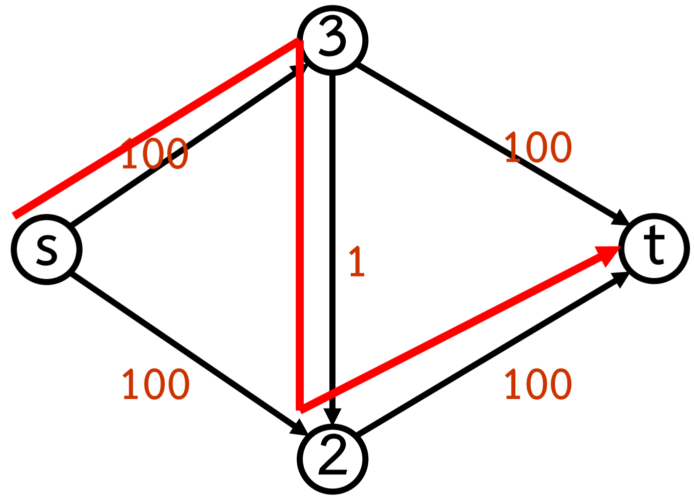


original graph

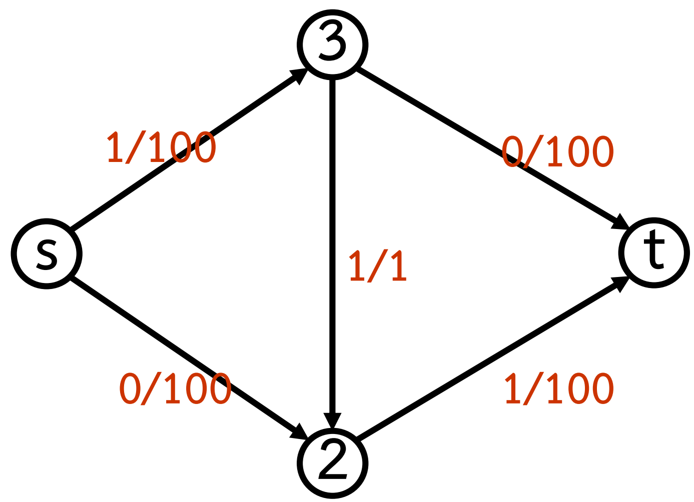


Choosing Good Augmenting Path

residual graph

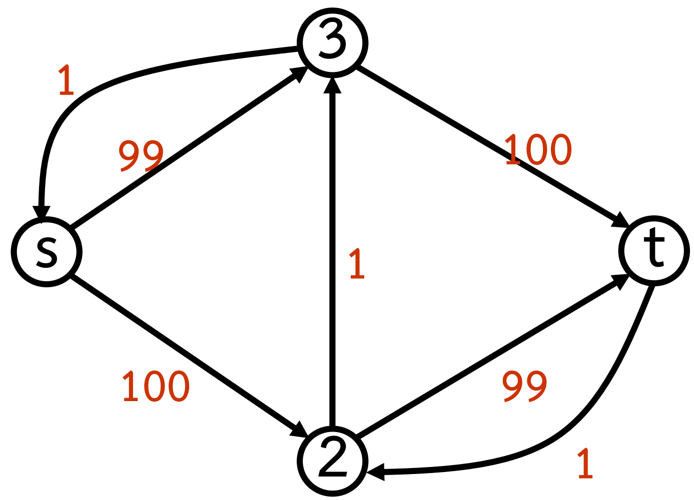


original graph

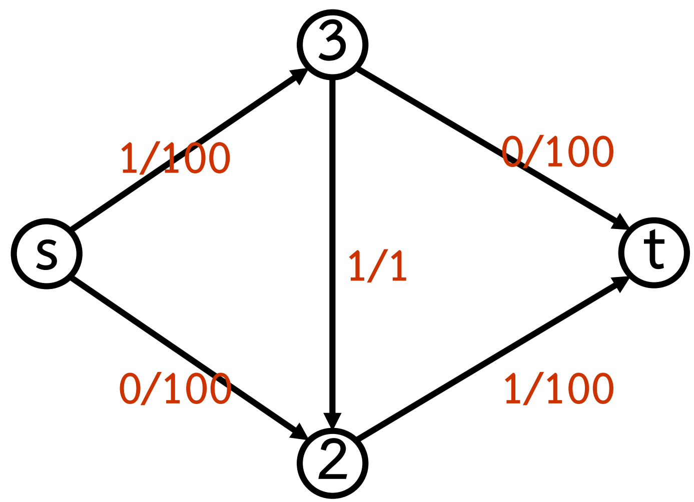


Choosing Good Augmenting Path

residual graph

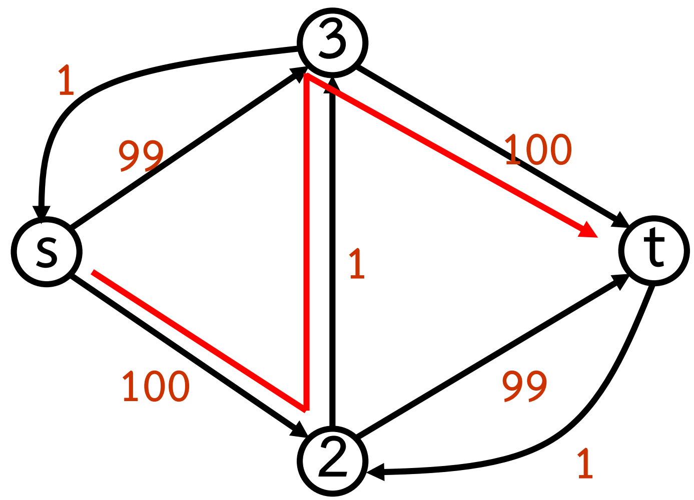


original graph

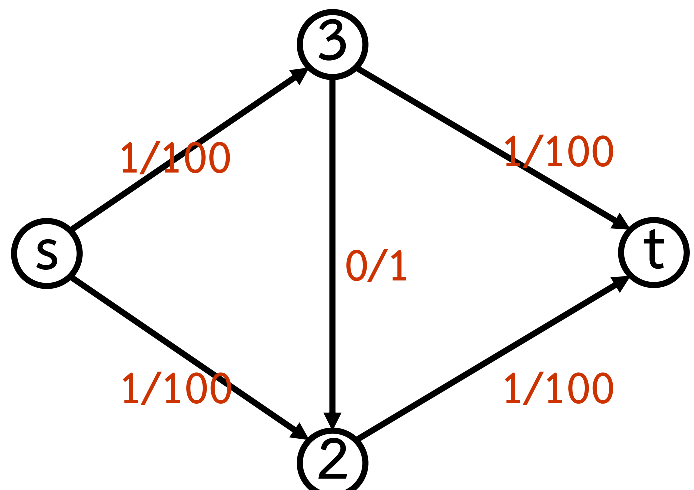


Choosing Good Augmenting Path

residual graph



original graph



หากเลือกไม่ได้ต้องทำถึง 200 รอบ

Choosing Good Augmenting Path

ควรระวังในการเลือก augmenting path

- การเลือกบางวิธีทำให้ได้ exponential algorithm
- การเลือกบางวิธีทำให้ได้ polynomial algorithm

ออกแบบรูปแบบของ augmenting path ที่อยากได้

- หา augmenting path ได้อย่างมีประสิทธิภาพ
- ทำจำนวนรอบที่น้อย

เลือก augmenting path ด้วยวิธี

- ใช้จำนวนเส้นเชื่อมที่น้อยที่สุด (shortest path)
- เลือกเส้นที่ bottleneck capacity ใหญ่สุด (fattest path)

Shortest augmenting path

หาได้ง่าย สามารถใช้ BFS ได้

หา augmenting path ที่มีจำนวนเส้นเชื่อมน้อยที่สุด

ในแต่ละรอบความยาวของ shortest augmenting path จะเพิ่มขึ้น

- ความยาวเพิ่มไม่เกิน E
- ไม่เกิน EV augmenting path ทั้งหมด
- ดังนั้นเวลาในการทำงานเป็น O(E^2V)

Fattest augmenting path

หา augmenting path ที่มี bottleneck capacity ที่มีค่ามากที่สุด

ส่ง flow ไปยัง sink

solve โดยใช้ dijkstra-style (Priority-first search) algorithm

การหา fattest path ใช้ O(E log V) ต่อการ augment โดยใช้ binary heap

การหา min-cut

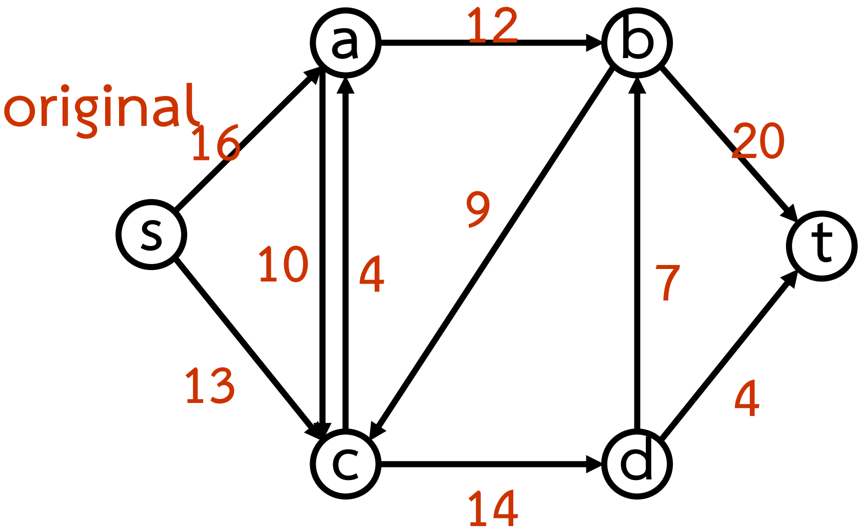
- เรารู้ว่า max flow – min cut
- ตอนนี้เรารู้วิธีการหา max flow

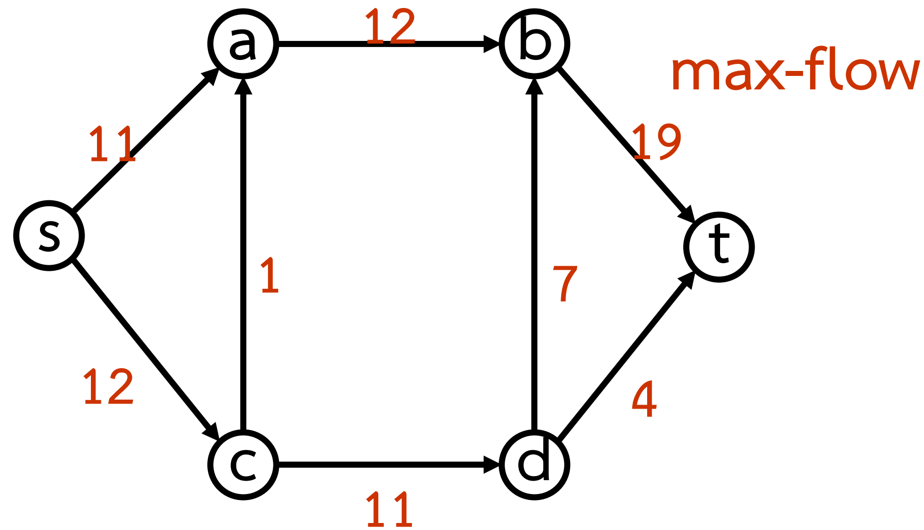
Q: เราจะหา min-cut ได้อย่างไร

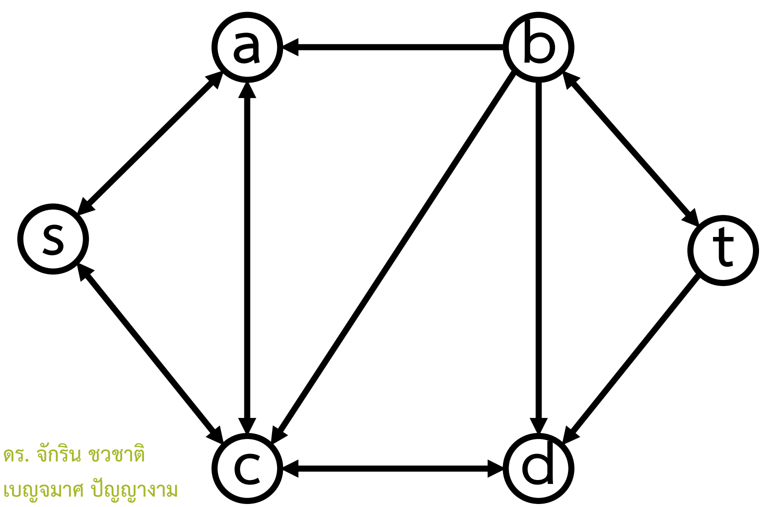
A: เราจะใช้ residual graph

การหา min-cut

- เราจะลบ max flow ออกจาก original graph





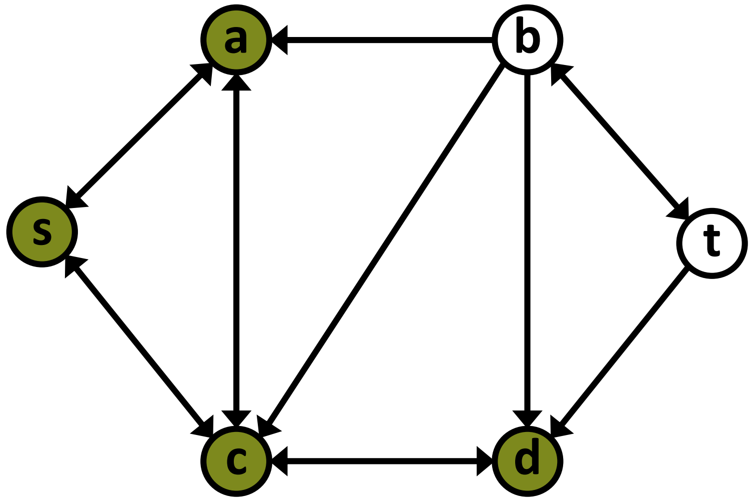


แสดงแค่ topology ของ residual graph

อย่าลืมเพิ่มเส้นย้อนกลับ

การหา min-cut จาก topology ของ residual grap

- mark ทุกโหนดที่ไปถึงจาก s
  - เรียก set ของโหนดที่ไปถึงจาก s ว่า A

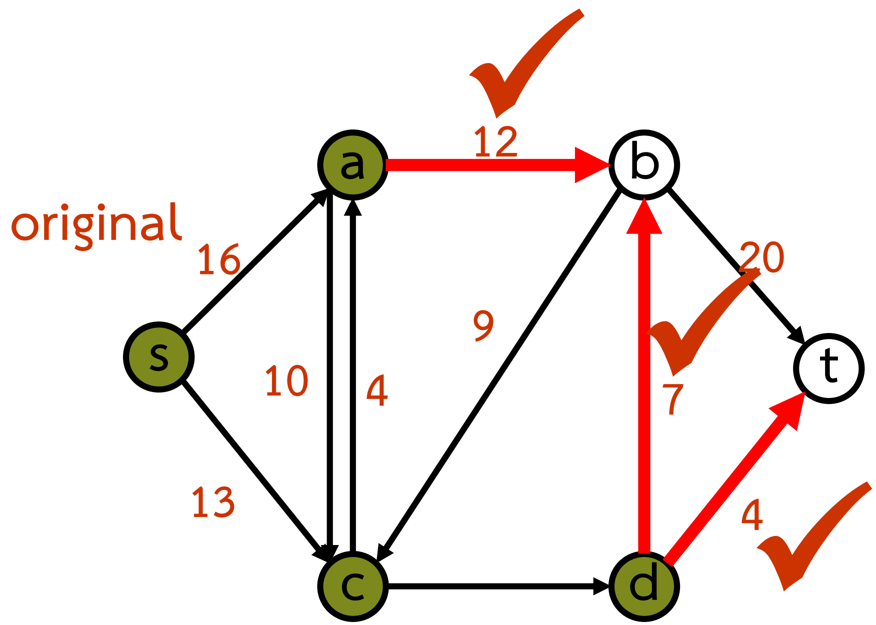


แยกโหนดเหล่านี้ออกจากกลุ่ม

เส้นเชื่อมที่วิ่งจาก A ไปยัง V-A คือ cut นั่นเอง

การหา min-cut

- พิจารณาใน original graph เพื่อให้ cut



Contents

- Disjoint paths
- Network connectivity
- Bipartite matching
- Vertex cover

Disjoint Paths

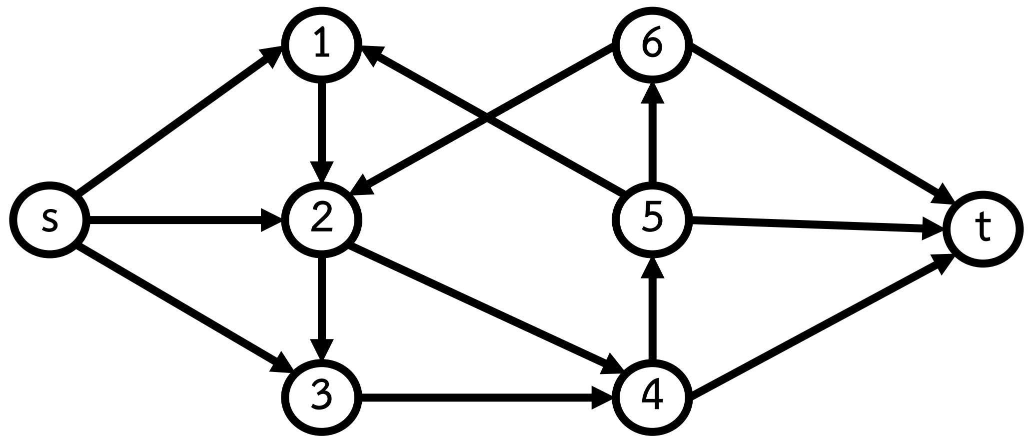
Disjoint path network: G=(V,E,s,t)

- กราฟแบบมีทิศทาง (V,E) source s และ sink t
- Path 2 path จะเป็น edge-disjoint path ถ้าทั้งสอง path นั้นไม่มีการใช้เส้นเชื่อม (edge) ที่เหมือนกันเลย
- Disjoint path problem: ต้องการหา edge-disjoint s-t path จำนวนมากที่สุด

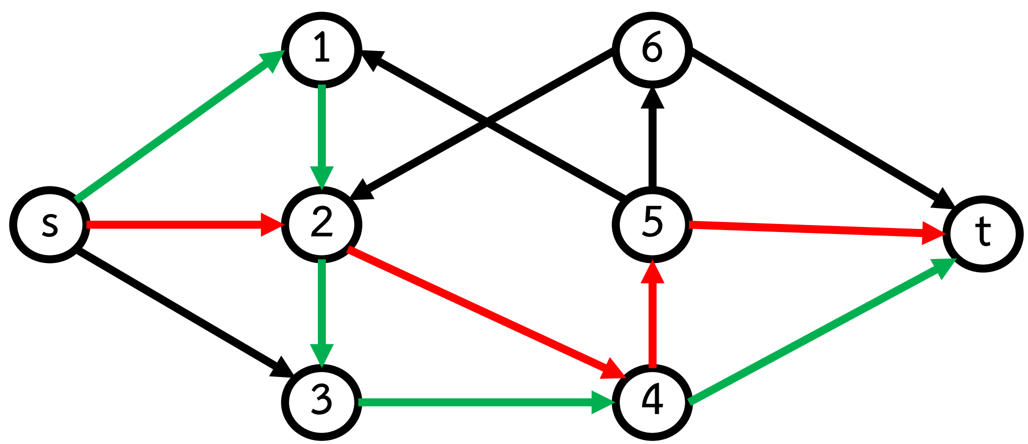
Application ที่นำไปใช้ได้แก่ เครือข่ายการติดต่อสื่อสาร

Disjoint Paths

Input



Solution



2 paths

Disjoint Paths

เราจะแก้ปัญหานี้ได้อย่างไร

มีเงื่อนไขว่าเส้นเชื่อมหนึ่งเส้นถูกใช้ได้เพียงครั้งเดียว

เส้นเชื่อมที่เลือกต้องต่อกันเป็น path มีวิธีการอะไรคล้ายไหม

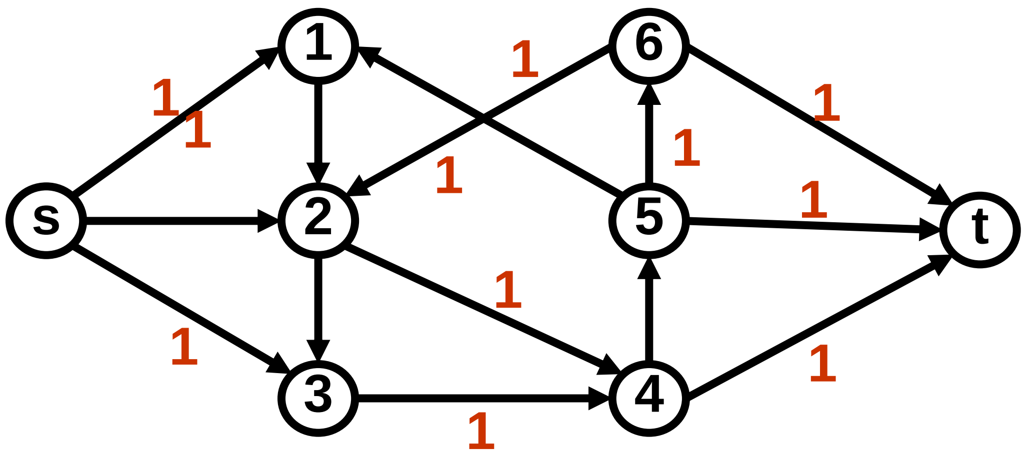
เราพบว่าปัญหา Max-flow คล้าย เพราะว่ามีการส่ง flow จาก s ไป t

ซึ่งการส่ง flow ต้องต่อเนื่องกัน

เราจะเปลี่ยนไปเป็นปัญหา Max-flow นั้นต้องมีการปรับอะไรบ้าง เพิ่มอะไรบ้าง หรือลดอะไรบ้าง

Max-flow formulation

กำหนดให้แต่ละเส้นเชื่อมมีความจุ 1 หน่วยทุกเส้นเชื่อม



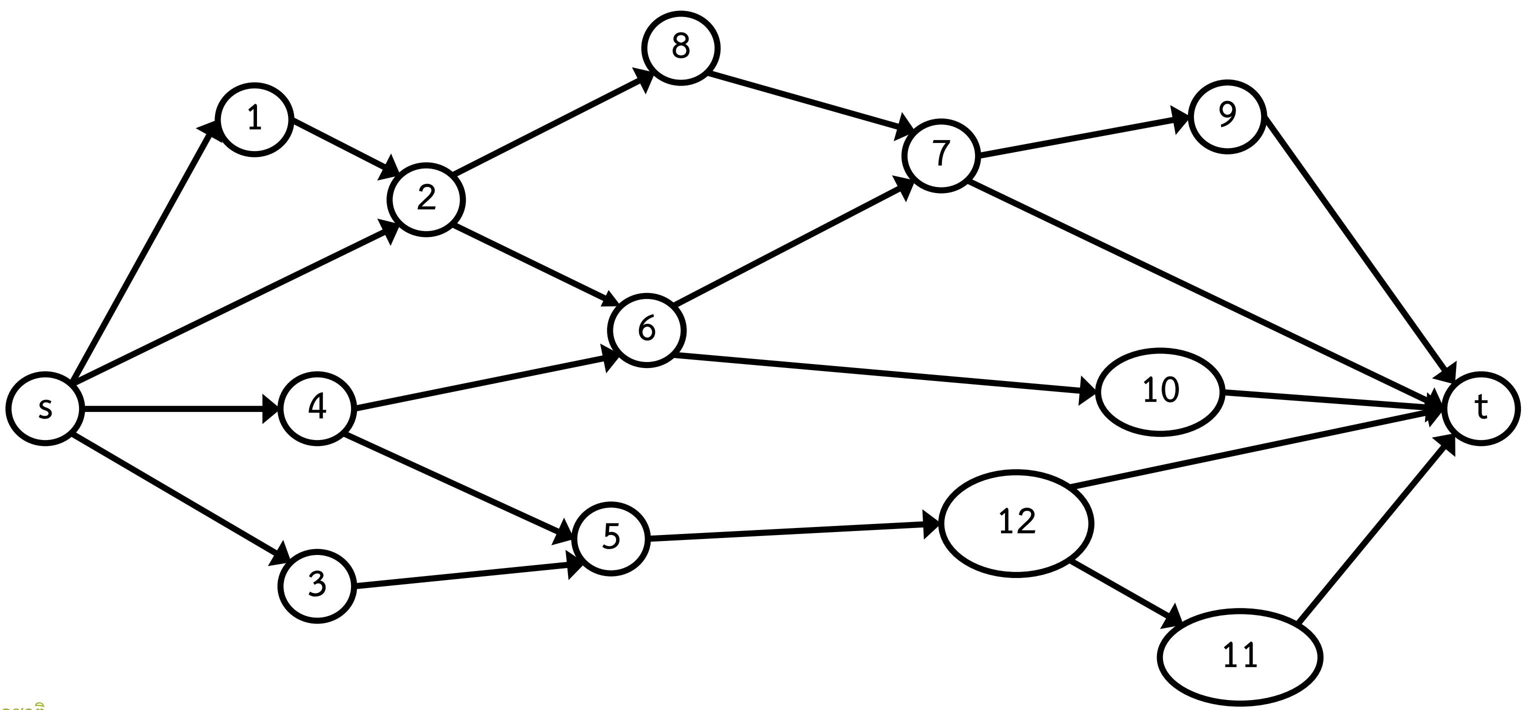
จากนั้นใช้ Max-flow algorithm ในการแก้ปัญหา

คำถาม ถ้า Max-flow algorithm หาคำตอบได้ k หน่วยแสดงว่ามี edge-disjoint paths กี่ path

Disjoint Paths Problem & Max Flow Problem

Theorem

มี k edge-disjoint paths จาก s ไป t ก็ต่อเมื่อ max flow มีค่าเป็น k



Network Connectivity

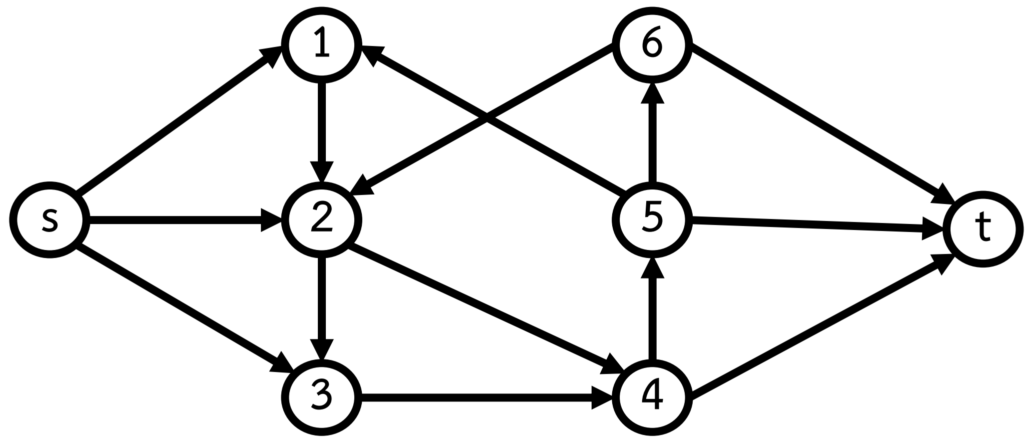
Network connectivity network: G=(V, E, s, t)

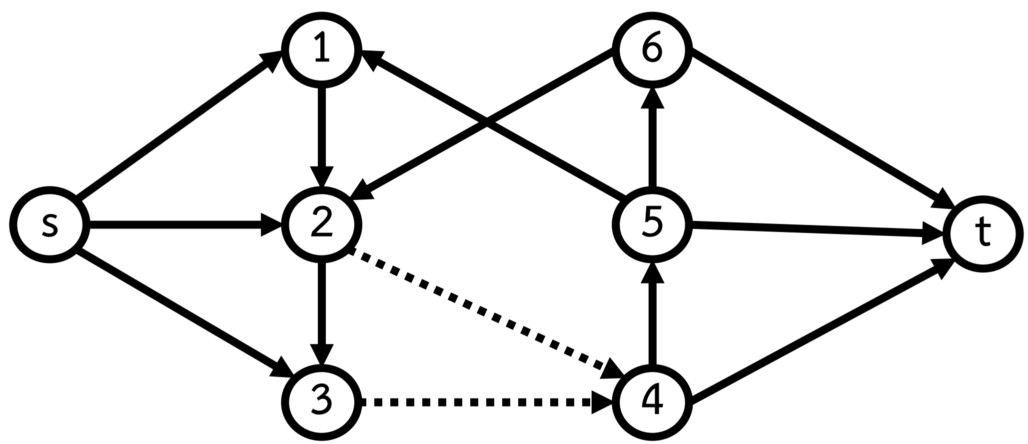
- กราฟแบบมีทิศทาง (V,E) source s และ sink t
- เซตของเส้นเชื่อม $F \subseteq E$ ที่ตัดการเชื่อมต่อ (disconnect) ระหว่าง t กับ s ถ้าทุกๆ s-t paths ใช้อย่างน้อย 1 เส้นเชื่อมใน F

Network connectivity: หาจำนวนเส้นเชื่อมที่น้อยที่สุดที่เมื่อเอาเส้นเชื่อมออกแล้วจะตัดการเชื่อมต่อระหว่าง t กับ s

Network Connectivity

Input





Network Connectivity

ข้อสังเกต จำนวนของเส้นเชื่อมที่ต้องเอาออกมีค่าเท่ากับอะไร

จำนวนของ edge-disjoint s-t path

Theorem (Menger's Theorem)

จำนวนของ edge-disjoint s-t paths ที่มากที่สุด

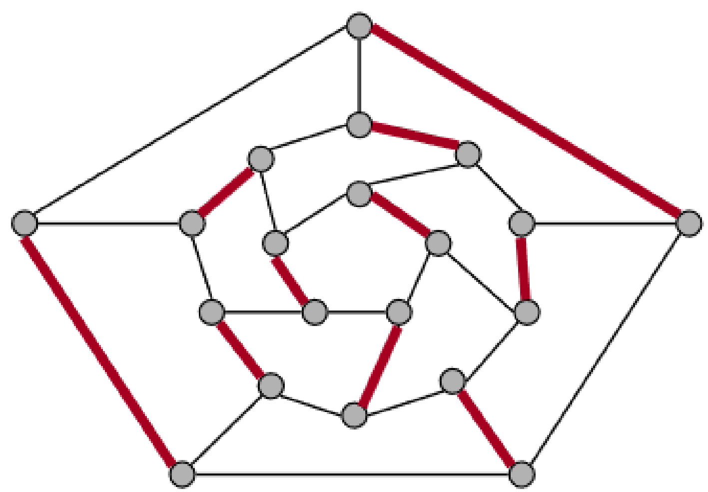
จะเท่ากับ จำนวนของเส้นเชื่อมที่น้อยที่สุดที่ตัดการเชื่อมต่อระหว่าง s กับ t

Matching

Matching

- Input: กราฟแบบไม่มีทิศทาง G=(V,E)
- $M \subseteq E$เป็น matching ถ้าแต่ละโหนดปรากฏอยู่ในเส้นเชื่อม M ไม่เกิน 1 ครั้ง

Max matching: หาจำนวนของ matching ที่มากที่สุด

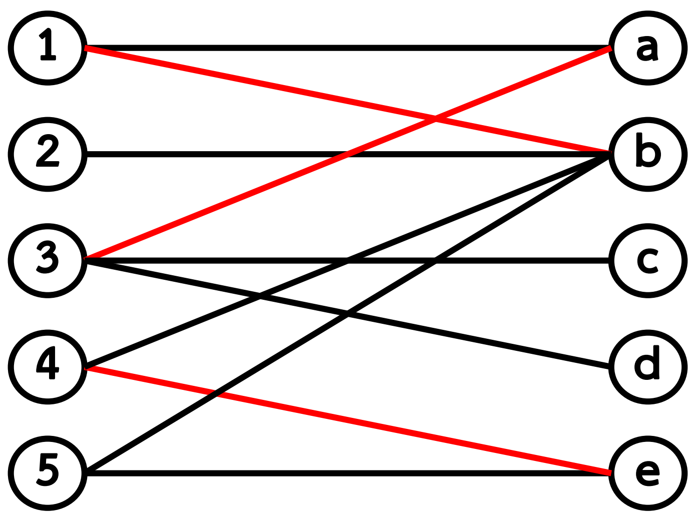


Bipartite Matching

Bipartite matching

- Input: กราฟแบบไม่มีทิศทาง G=(L U R,E)
- $M \subseteq E$เป็น matching ถ้าแต่ละโหนดปรากฏอยู่ในเส้นเชื่อม M ไม่เกิน 1 ครั้ง

Max matching: หาจำนวนของ matching ที่มากที่สุด



Matching

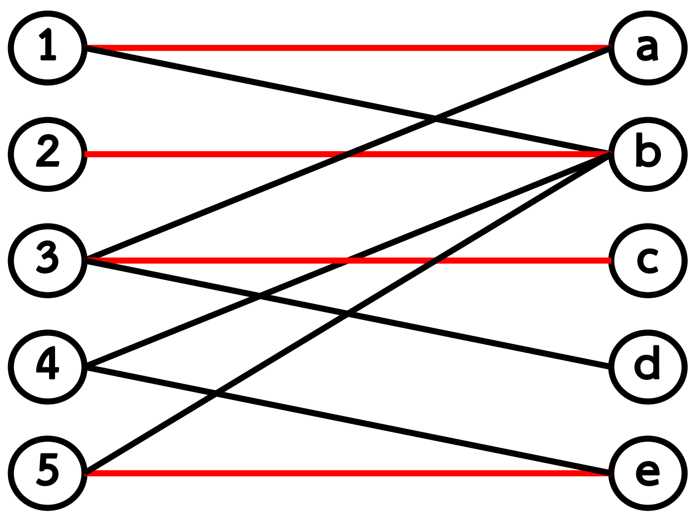
1-b,3-a,4-e

Bipartite Matching

Bipartite matching

- Input: กราฟแบบไม่มีทิศทาง G=(L U R,E)
- $M \subseteq E$เป็น matching ถ้าแต่ละโหนดปรากฏอยู่ในเส้นเชื่อม M ไม่เกิน 1 ครั้ง

Max matching: หาจำนวนของ matching ที่มากที่สุด



Matching

1-a,2-b,3-c,5-e

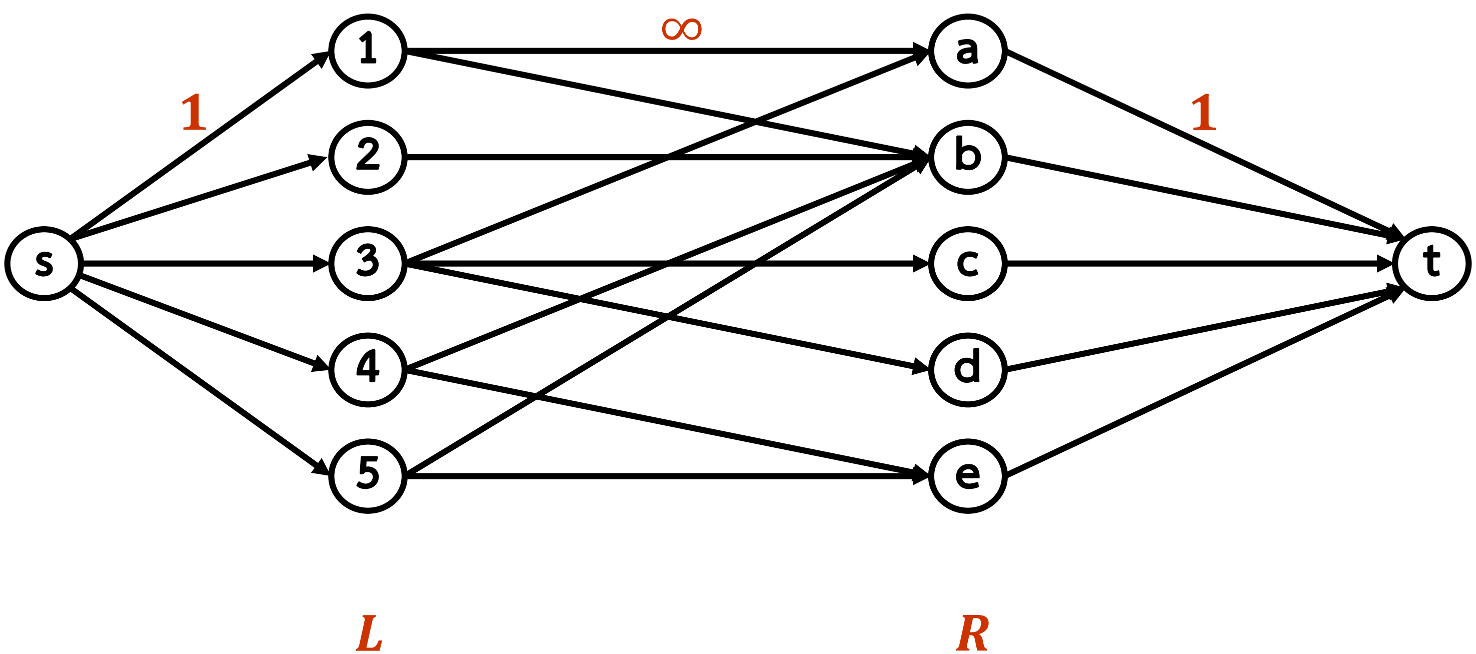
Bipartite Matching

Max flow formulation

- สร้างกราฟแบบมีทิศทาง G’=(L U R U {s,t},E’)
- กำหนดทิศทางจาก L ไป R โดยให้ capacity เป็น infinity
- เพิ่ม source s และเพิ่มเส้นเชื่อมแบบมีทิศทางความจุ 1 หน่วย จาก s ไปยังแต่ละโหนดใน L
- เพิ่ม sink t และเพิ่มเส้นเชื่อมแบบมีทิศทางความจุ 1 หน่วยจากแต่ละโหนดใน R ไปยัง t

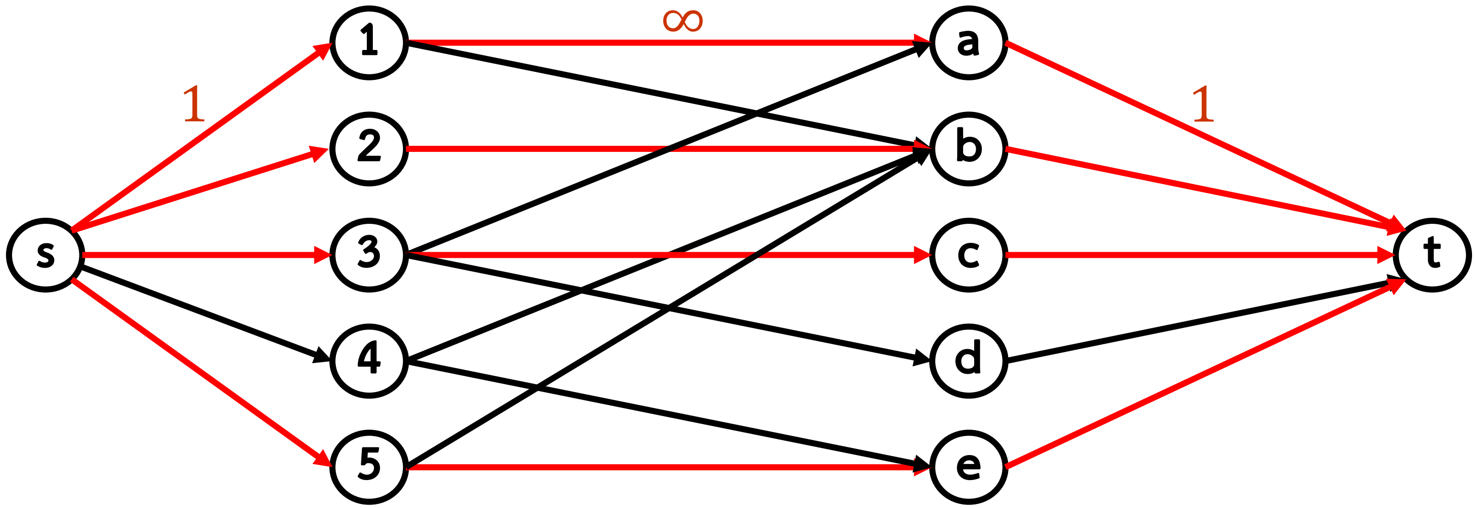
Bipartite Matching

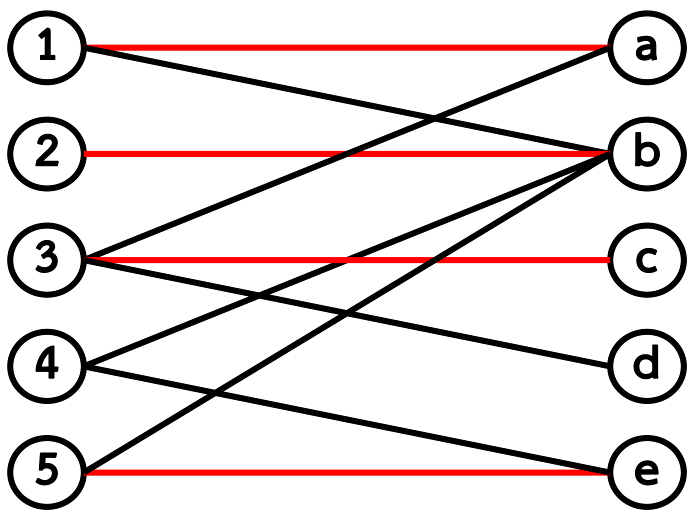
Max flow formulation



Bipartite Matching

flow f ที่มีค่า k ใน G’จะทำให้ได้ matching ขนาด k ใน G



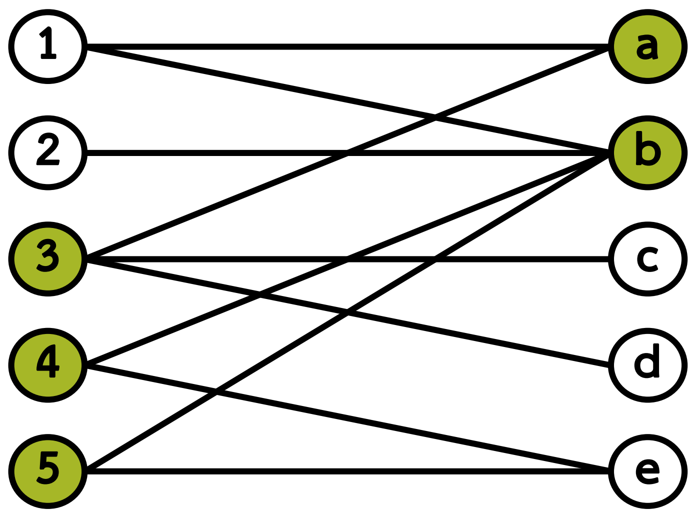


อ. ดร. จักริน ชวชาติ
อ. เบญจมาศ ปัญญางาม

Vertex cover

กำหนดกราฟแบบมีทิศทาง G=(V,E)

vertex cover คือ subset ของ vertices $C \subseteq V$ ที่ ทุกเส้นเชื่อม $(v, w) \in E$ มี $v \in C$ หรือ $w \in C$ หรือทั้งคู่



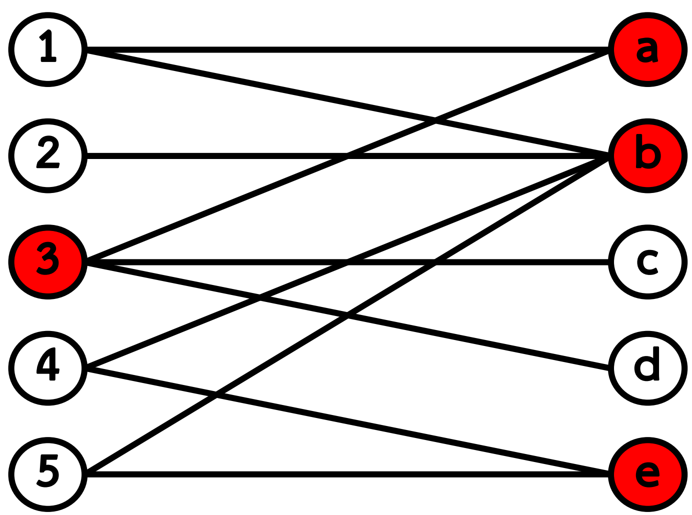
C={3,4,5,a,b}

|C|=5

Vertex cover

กำหนดกราฟแบบมีทิศทาง G=(V,E)

vertex cover คือ subset ของ vertices $C \subseteq V$ ที่ ทุกเส้นเชื่อม $(v, w) \in E$ มี $v \in C$ หรือ $w \in C$ หรือทั้งคู่



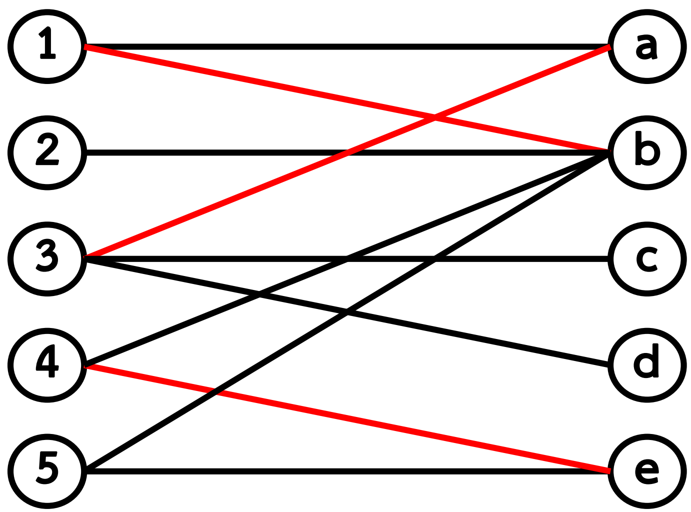
C={3,a,b,e}

|C|=4

Vertex cover

ข้อสังเกต ให้ M เป็น matching และให้ C เป็น vertex cover เราสังเกตได้ว่า $|M| \leq |C|$

แต่ละ vertex สามารถ cover ได้ไม่เกิน 1 เส้นเชื่อมใน matching



Matching

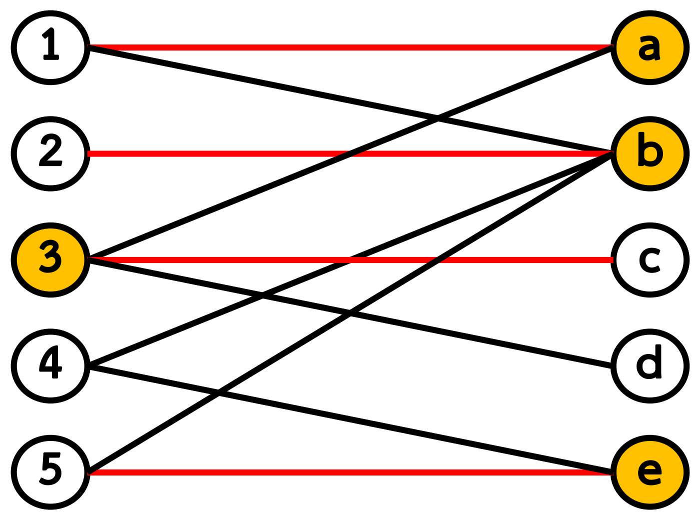
1-b,3-a,4-e

Vertex cover

Konig-Egervary Theorem: ใน bipartite undirected graph, จำนวนของ matching ที่มากที่สุดจะเท่ากับจำนวนของ vertex cover ที่น้อยที่สุด

M*={1-a,2-b,3-c,5-e}

|M*|=4



C*={a,b,3,e}

|C*|=4