Network Flow

Flow graph modeling

- เรามี code ของ Edmond Karp's algorithm แล้ว ซึ่ง code นี้สามารถแก้ปัญหา Network Flow แบบธรรมดาได้ เวลาเจอโจทย์สิ่งที่เราต้องทำคือ
- มองก่อนว่าปัญหานี้ว่าเป็น Network Flow Problem หรือไม่ (จะคล่องขึ้นถ้าแก้ปัญหา Network Flow Problem บ่อยๆ)
- สร้าง flow graph ที่เหมาะสม (ปรับจาก code ก่อนหน้า กำหนด residual matrix, s และ t ให้เหมาะสม)
- รัน Edmond Karp's algorithm

Other applications

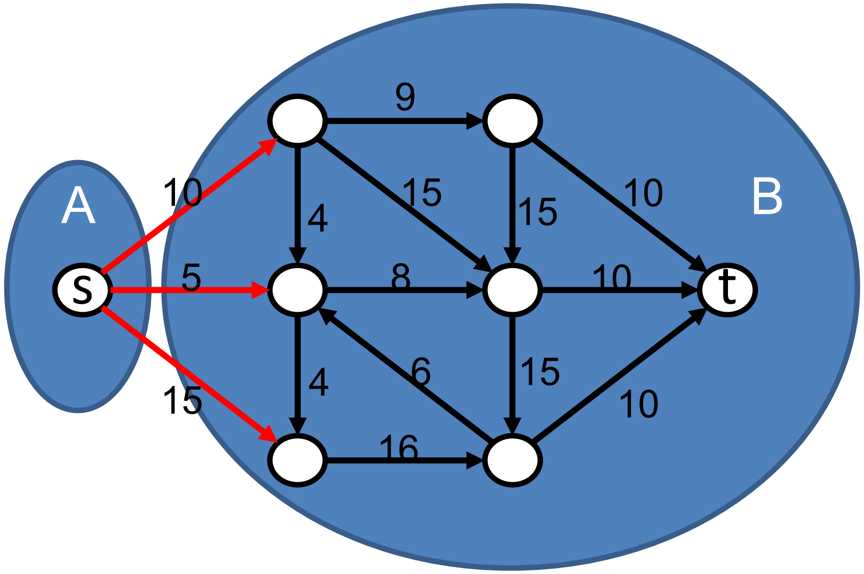
- ยังมี application อื่นๆ ที่เกี่ยวกับ flow ใน network
- Minimum Cut
- นิยาม s–t cut C คือ (S–component, T–component) ที่เป็นการแบ่งเซตของโหนด V ของ G ออกเป็นสองส่วนโดยที่ $s \in S - component$ และ $t \in T - component$
- นิยาม cut–set ของ C เป็นเซต $\{(u, v) \in E | u \in S - component, v \in T - component\}$ ที่ถ้าทุกเส้นเชื่อมใน cut–set ของ C ถูกเอาออกแล้ว Max Flow จาก s ไป t จะเป็น 0 (นั่นคือ s ขาดจาก t)

Minimum cut problem

<u>นิยาม</u> st-cut (cut) เป็นการแบ่งโหนดออกเป็นเซต (A,B) ที่มี $s \in A$ และ $t \in B$

<u>นิยาม</u> ความจุของ cut คือผลรวมความจุของเส้นเชื่อมที่ออกจาก A ไป B

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



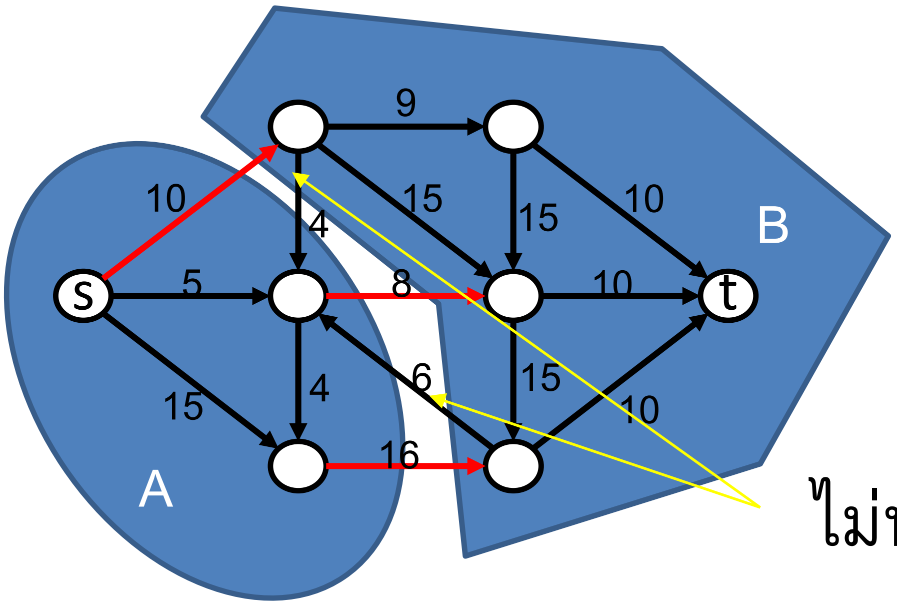
capacity=10+5+15=30

Minimum cut problem

นิยาม st–cut (cut) เป็นการแบ่งโหนดออกเป็นเซต (A,B) ที่มี $s \in A$ และ $t \in B$

นิยาม ความจุของ cut คือผลรวมความจุของเส้นเชื่อมที่ออกจาก A ไป B

$$cap(A,B) = \sum_{e \text{ out of } A} c(e)$$



capacity=10+8+16=34

ไม่นับเส้นเชื่อมจาก B ไป A

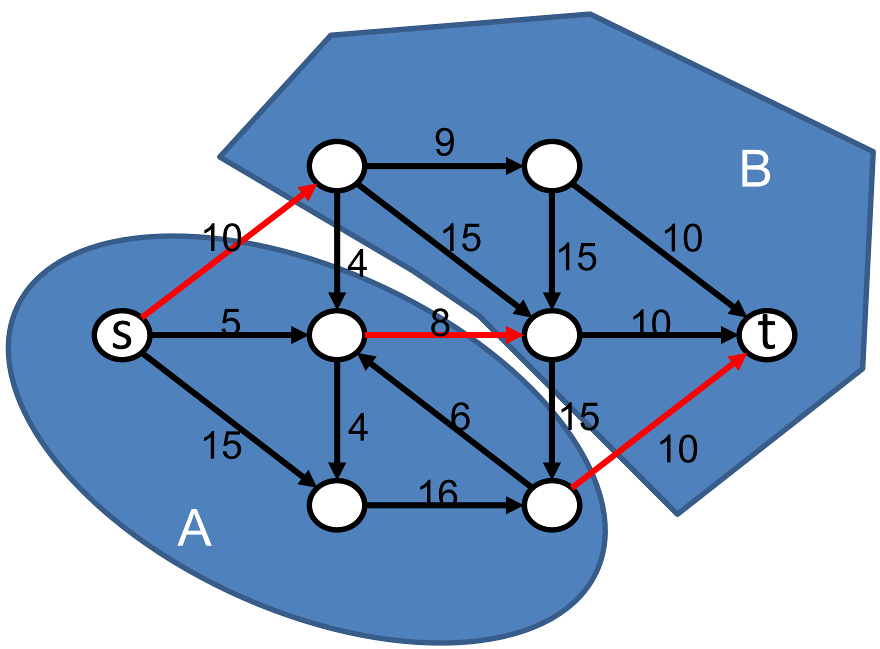
Minimum cut problem

<u>นิยาม</u> st-cut (cut) เป็นการแบ่งโหนดออกเป็นเซต (A,B) ที่มี $s \in A$ และ $t \in B$

<u>นิยาม</u> ความจุของ cut คือผลรวมความจุของเส้นเชื่อมที่ออกจาก A ไป B

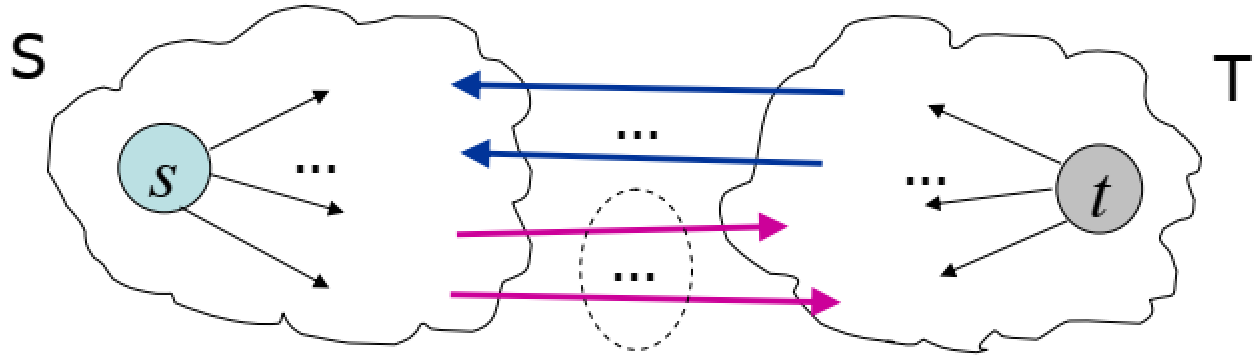
$$cap(A,B) = \sum_{e \text{ out of } A} c(e)$$

Min-cut problem หา cut ที่มีความจุ<u>ต่ำ</u>ที่สุด

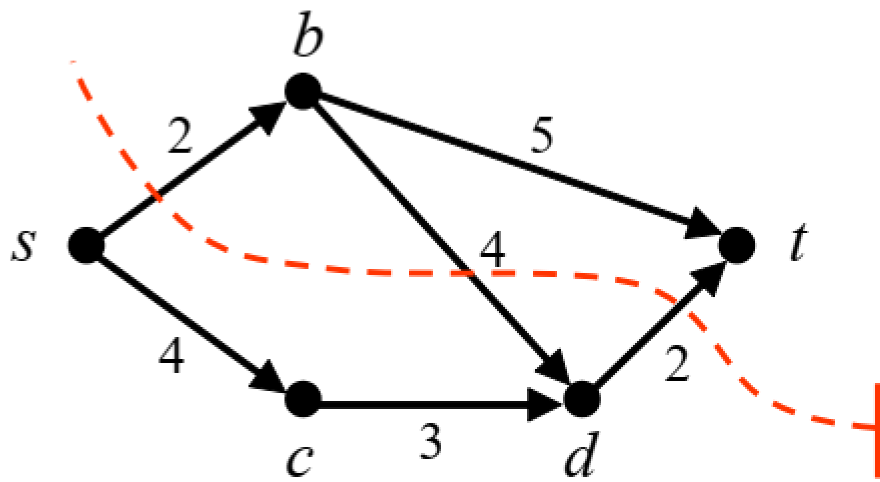


capacity=10+8+10=28

Max-Flow Min-Cut Theorem

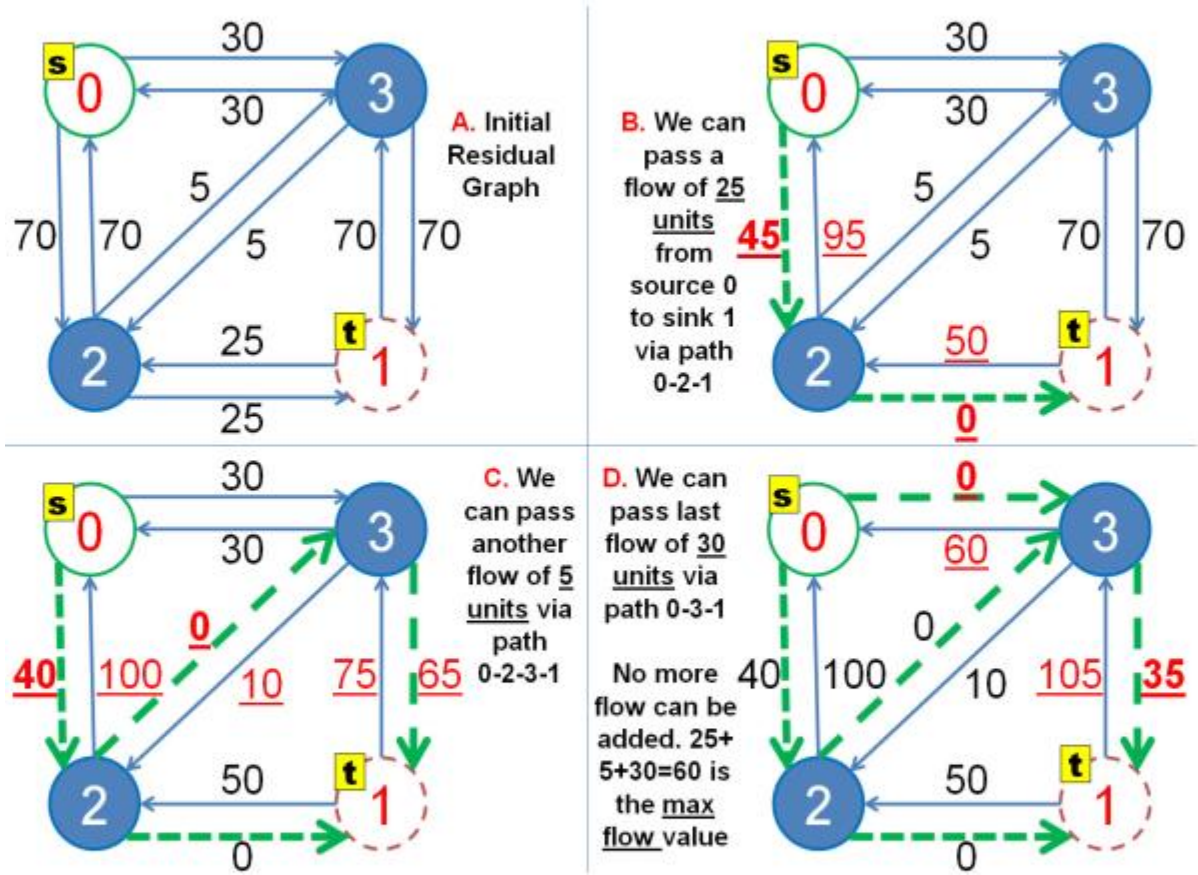


$$f_{\max}(S,T) = c_{\min}(S,T)$$



S	c(S,T)
$\{s\}$	6
$\{s, b\}$	13
$\{s, c\}$	5
$\{s, d\}$	8
$\{s, b, c\}$	12
$\{s, b, d\}$	11
$\boxed{\{s, c, d\}}$	$\boxed{4}$
$\{s, b, c, d\}$	7

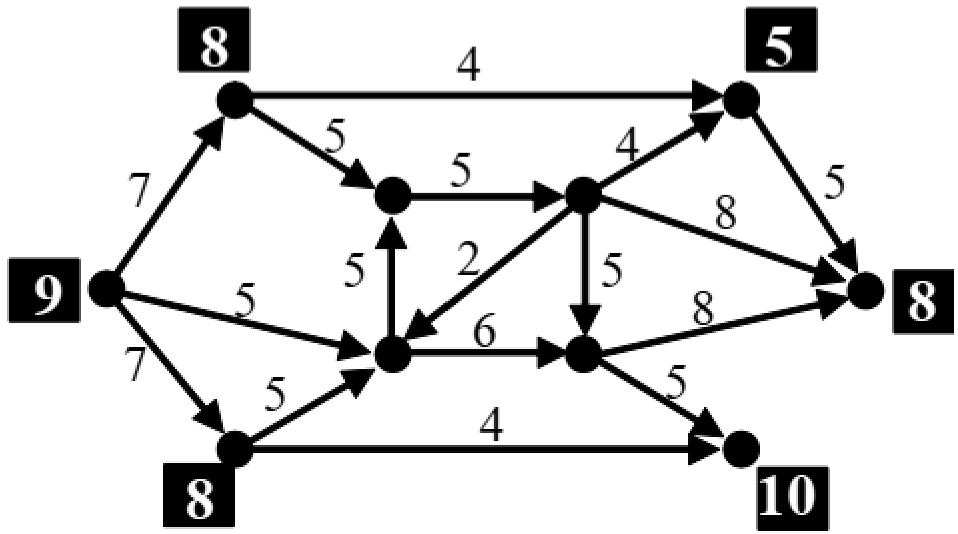
- วิธีการจัดการปัญหานี้ไม่ยาก เนื่องจาก Min Cut เป็นผลพลอยได้จาก Max Flow
- หลังจากที่ Max Flow algorithm หยุด เราก็เพียงรัน graph traversal (DFS/BFS) จาก source s อีกครั้ง
- ทุกโหนดที่ไปถึงได้จาก s โดยใช้ เส้นเชื่อมที่มีค่าน้ำหนักเป็นบวกใน residual graph จะอยู่ใน S-component
- ส่วนโหนดที่เหลือที่ไปไม่ถึงจะอยู่ใน T-component
- ทุกเส้นเชื่อมที่เชื่อมระหว่าง S-component และ T-component จะอยู่ใน cut-set C

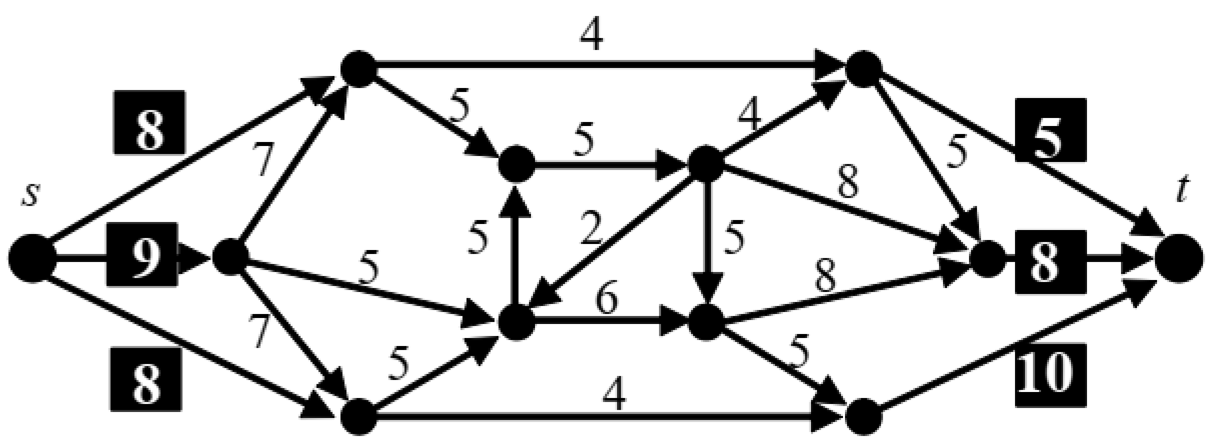


- Min Cut คือ 30+5+25=60

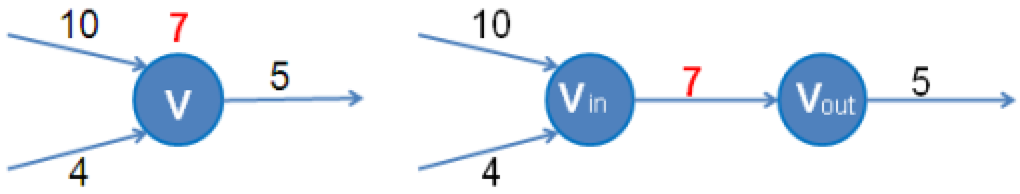
Multi-source/Multi-sink

- ในบางครั้ง เราสามารถมีได้มากกว่า 1 source และ/หรือ มากกว่า 1 sink
- อย่างไรก็ตามในปัญหาในรูปแบบนี้ไม่ได้ยากกว่าเดิมที่มี 1 source 1 sink
- เราจัดการได้โดยสร้าง super source SS และ super sink ST
- จากนั้นเชื่อม SS กับ s ด้วย infinity capacity และเชื่อมทุก t กับ ST ด้วย infinity capacity เช่นเดียวกัน
- จากนั้น run Edmond Karp's algorithm ตามปกติ


8
5
4
5
5
4
7
5
8
9
5
2
5
8
5
8
7
6
8
5
5
4
8
10


4
8
5
5
4
7
5
5
8
s
t
9
5
2
5
8
6
8
7
5
5
4
8
10

Node Capacities



- ในบางรูปแบบของปัญหาเมื่อความจุนั้นไม่ได้มีเพียงบนเส้นเชื่อมแต่มีที่โหนดด้วย
- วิธีแก้ก็ไม่ยาก เราก็ใช้วิธีการ node splitting ซึ่งเราแบ่งโหนดออกเป็นสองโหนดใน flow graph
- Weighted graph with a vertex weight สามารถแปลงให้เป็นแบบที่ไม่มีน้ำหนักได้ โดยแบ่งแต่ละโหนด v ออกเป็น v_{in} และ v_{out} แล้วกำหนดเส้นเชื่อมขาเข้าให้กับ v_{in} และขาออกให้กับ v_{out} ตามลำดับและค่าน้ำหนักของโหนด มากำหนดให้เป็นค่าน้ำหนักของเส้นเชื่อมจาก v_{in}->v_{out}

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

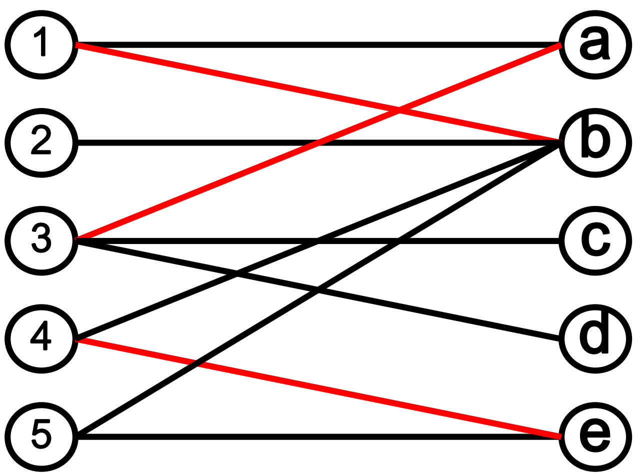
Application ที่นำไปใช้ได้แก่ เครือข่ายการติดต่อสื่อสาร

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



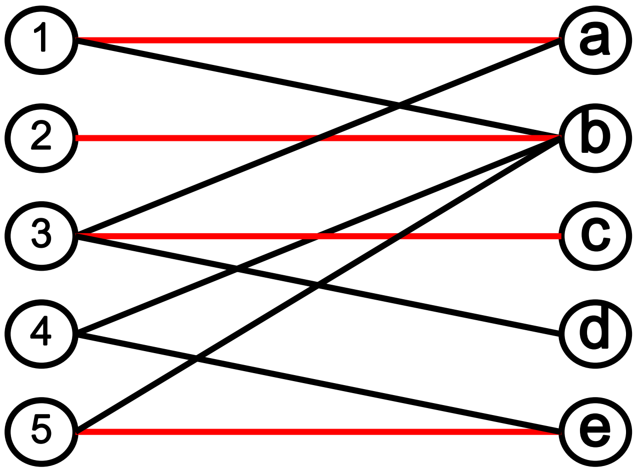
Matching
1-b,3-a,4-e

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

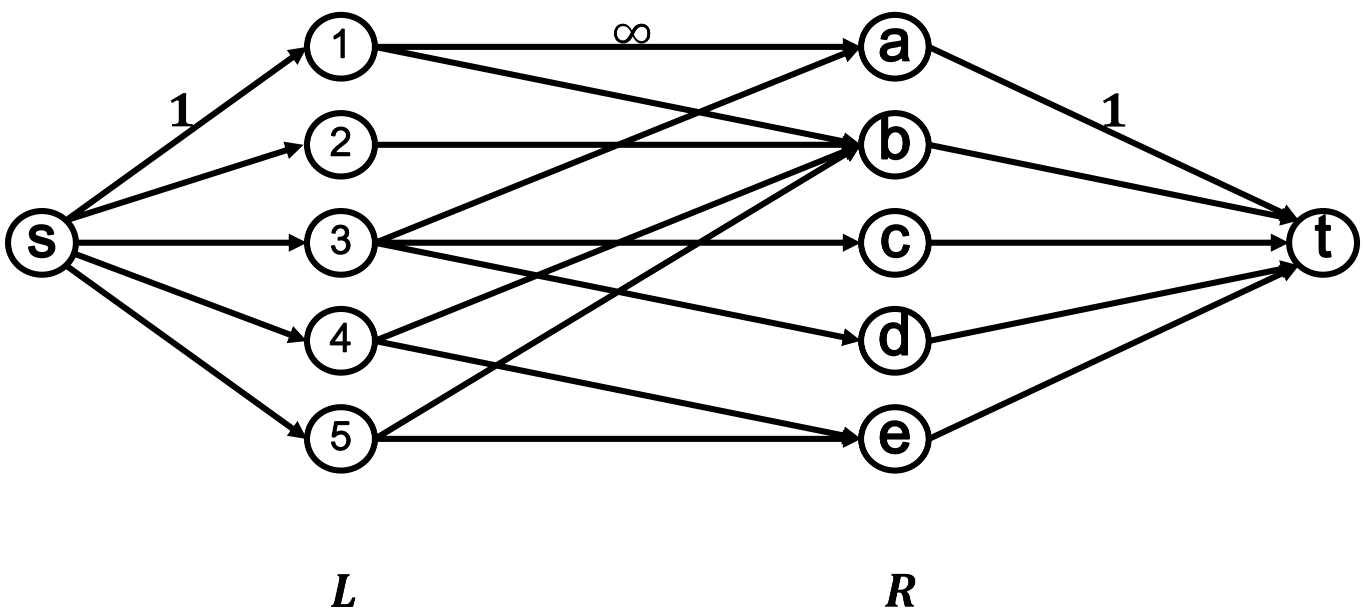
Bipartite Matching

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 หน่วยจาก 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