Discrete Computational Structures – Dijkstra's algorithm

อ. ภูริวัจน์ วรวิชัยพัฒน์

อ้างอิงจากหนังสือ: Discrete Computational Structure, คทา ประดิษฐ์วงศ์

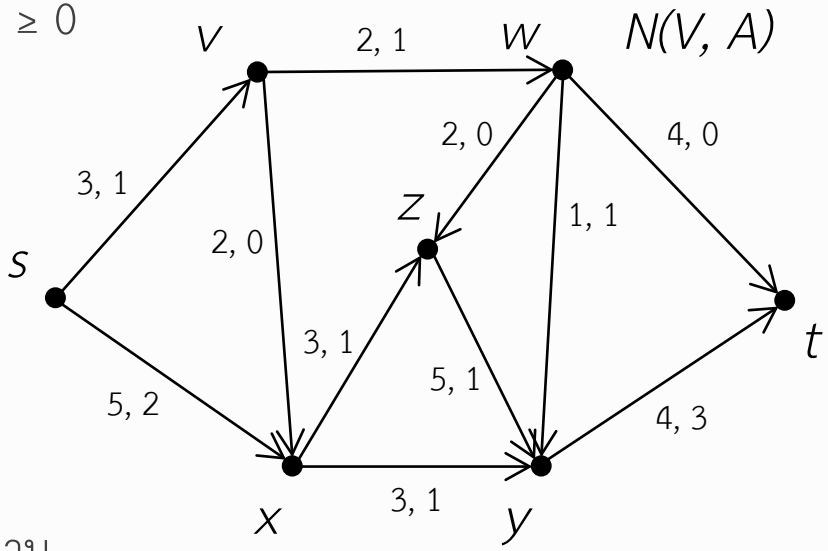
ทบทวนคาบที่แล้ว (Recap)

- **ข่ายงาน – Network, N(V,A)**
- นิยาม: ข่ายงานต้องเป็นกราฟทิศทาง, มีการเชื่อมโยงแบบอ่อน, เส้นเชื่อมมีเลขความจุ ≥ 0

  จุดยอด 3 แบบ: แหล่งต้นทาง(s), แหล่งปลายทาง(t), และระหว่างทาง
- การไหล, f(a), ซึ่งต้องมีค่าไม่มากกว่า ความจุ c(a)

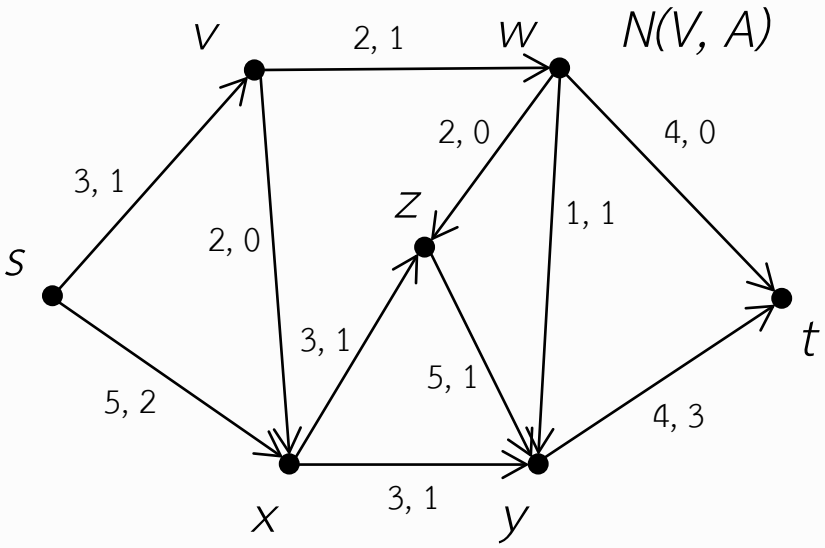
  ผลรวมการไหลออกของแหล่งต้นทาง = ผลรวมการไหลเข้าของแหล่งปลายทาง

  ผลรวมการไหลออกของจุดระหว่างทาง = ผลรวมการไหลเข้าของจุดระหว่างทาง
- ค่าการไหล $d = \sum_{a \in outdeg(s)} f(a) = \sum_{a \in indeg(t)} f(a)$ ค่านี้จะเป็นค่าการไหลของทั้งข่ายงาน



ทบทวนคาบที่แล้ว (Recap)

- ส่วนตัด (cut) จะแบ่งข่ายงานออกเป็นสองส่วน X และ $\bar{X}$; $\bar{X}$ คือ ส่วนเติมเต็มของ X; $\mathrm{A}(X, \bar{X})$ คือเซ็ตของเส้นทิศทางที่เชื่อมจาก X ไป $\bar{X}$; s ต้องอยู่ใน X เสมอ และ t ต้องอยู่ใน $\bar{X}$ เสมอ;
- ความจุส่วนตัด $\mathrm{c}(X, \bar{X})$ คือ ผลรวมความจุของเส้นเชื่อมขาออกระหว่างจุด X ไปยังจุด $\bar{X}$
- ค่าการไหลส่วนตัด $\mathrm{c}(X, \bar{X})$ คือ ผลรวมการไหลออกระหว่างจุด X ไปยังจุด $\bar{X}$ **ลบด้วย** ผลรวมการไหลออกจากจุด $\bar{X}$ ไปยังจุดใน X
- อัลกอริทึมฟอร์ดและฟูลเกอร์สัน (Ford and Fulkerson Algorithm)



เนื้อหาปลายภาค - Overview

- ต้นไม้, Tree
- กราฟทิศทาง, Directed graph
- ข่ายงาน และการประยุกต์ใช้ข่ายงาน, Network
- **การหาเส้นที่สั้นที่สุด Shortest path, Dijkstra's algorithm** TODAY
- การหาต้นไม้ทอดข้ามที่น้อยที่สุด Kruskal's algorithm & Prim's algorithm

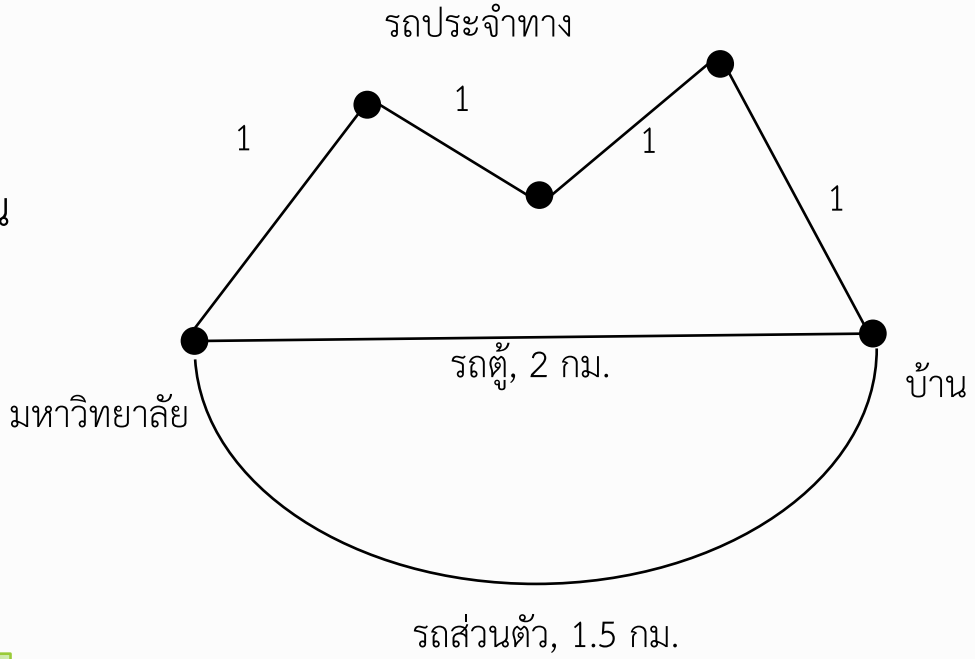
เส้นทางที่สั้นที่สุด, Shortest path

โดยปกติแล้วเมื่อเรามีการเก็บข้องมูลหรือการแสดงผลในรูปแบบกราฟ ผู้ใช้งานจะมีจุดมุ่งหมายหลายๆอย่างหรือวิธีการประยุกต์ใช้งานกราฟในทิศทางที่แตกต่างกันไป และสิ่งที่มักจะพบเจอเสมอๆเมื่อใช้งานกราฟคือ “**การหาเส้นทางที่สั้นที่สุด**”

ตัวอย่างด้านขวา เป็นกราฟแสดงการเดินทางจากมหาลัยไปถึงบ้านโดยมีทางเลือกอยู่ทั้งหมด 3 เส้นทางคือ

1. รถประจำทาง ระยะทาง 4 กม.
2. รถตู้ ระยะทาง 2 กม.
3. รถส่วนตัว ระยะทาง 1.5 กม.

จะสังเกตได้ว่ารถยนต์ส่วนตัวนั้นเป็นเส้นทางที่สั้นที่สุด



แต่ลองจินตนาการว่าถ้ากราฟนั้นมีขนาดใหญ่มากๆและเส้นเชื่อมนั้นมีความซับซ้อนมากๆ ไม่สามารถหาเส้นทางที่สั้นที่สุดได้ด้วยตาเปล่าได้ จะมีวิธีการอย่างไร?

องค์ประกอบกราฟ – น้ำหนัก, weight

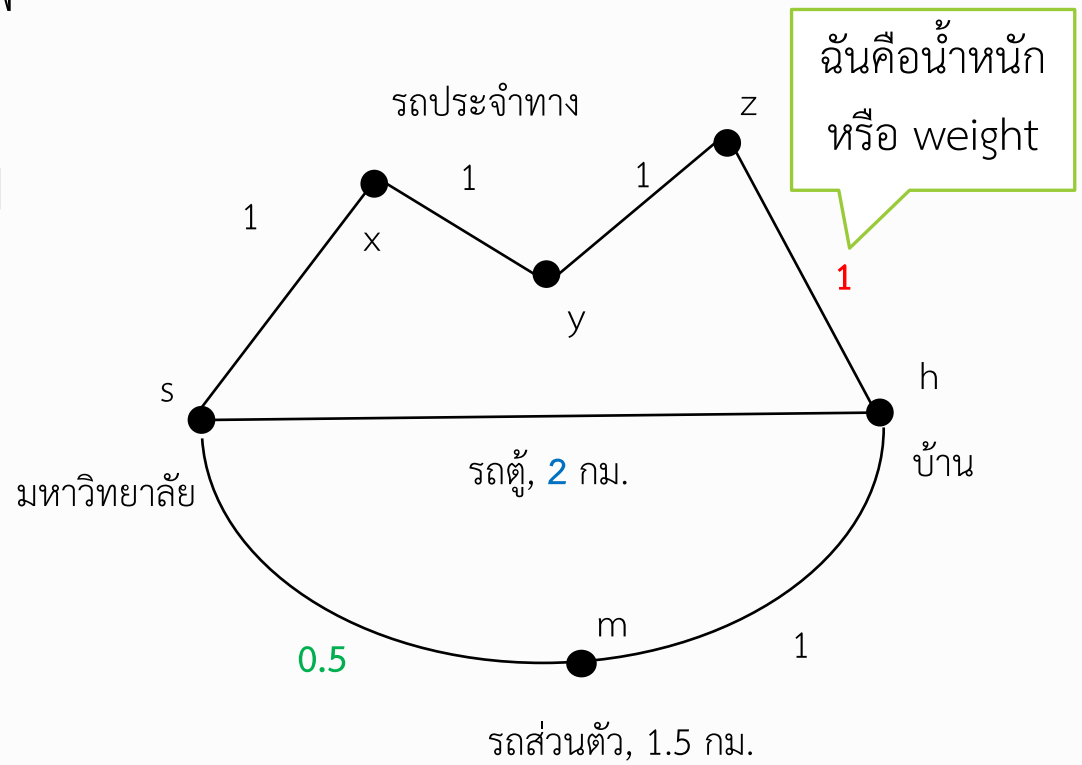
กราฟ G ต้องเป็นกราฟที่มีฟังก์ชันน้ำหนัก จะถูกเรียกว่ากราฟน้ำหนักหรือ weighted graph

ฟังก์ชันน้ำหนัก weight() คือ ฟังก์ชันที่ส่งเส้นเชื่อมของกราฟไปยังค่าจำนวนจริงที่ไม่ติดลบ ตัวอย่างเช่น

weight(s, h) = 2

weight(z, h) = 1

weight(s, m) = 0.5



องค์ประกอบกราฟ – ระยะทาง, distance

ฟังก์ชันระยะทาง dist(a) คือค่าน้ำหนักจากจุดยอดเริ่มต้น s ถึงจุดยอด a *แต่ค่าน้ำหนักนี้แปรผันไปตามเส้นทาง เช่น

1. เส้นทาง s, x, y, h จงหาระยะทาง dist(h)

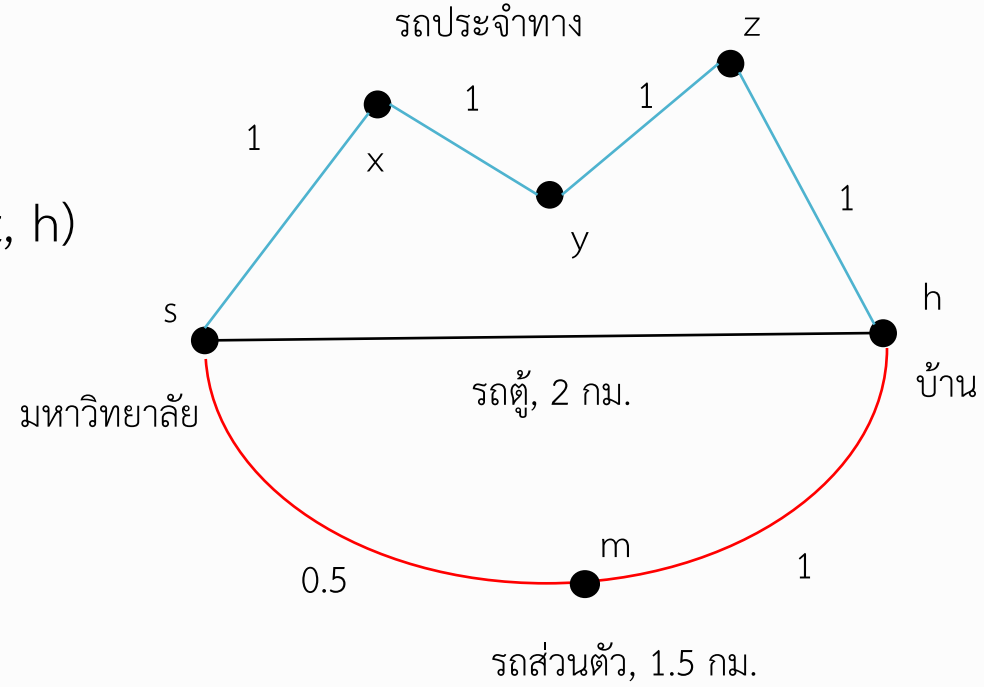
dist(h) = weight(s, x)+weight(x, y)+weight(y, z)+weight(z, h)

= 1+1+1+1 = 4

2. เส้นทาง s, m, h จงหาระยะทางdist(h)

dist(h) = weight(s, m)+weight(m, h)

= 0.5 + 1 = 4



องค์ประกอบกราฟ – จุดก่อนหน้า, previous

จุดยอดก่อนหน้า หรือ prev(a) คือจุดก่อนหน้าที่เชื่อมมา a
จุดยอดก่อนหน้านั้นแปรผันตามเส้นทางเช่นเดียวกับระยะทาง

1. เส้นทาง s, x, y, h

prev(x) = s

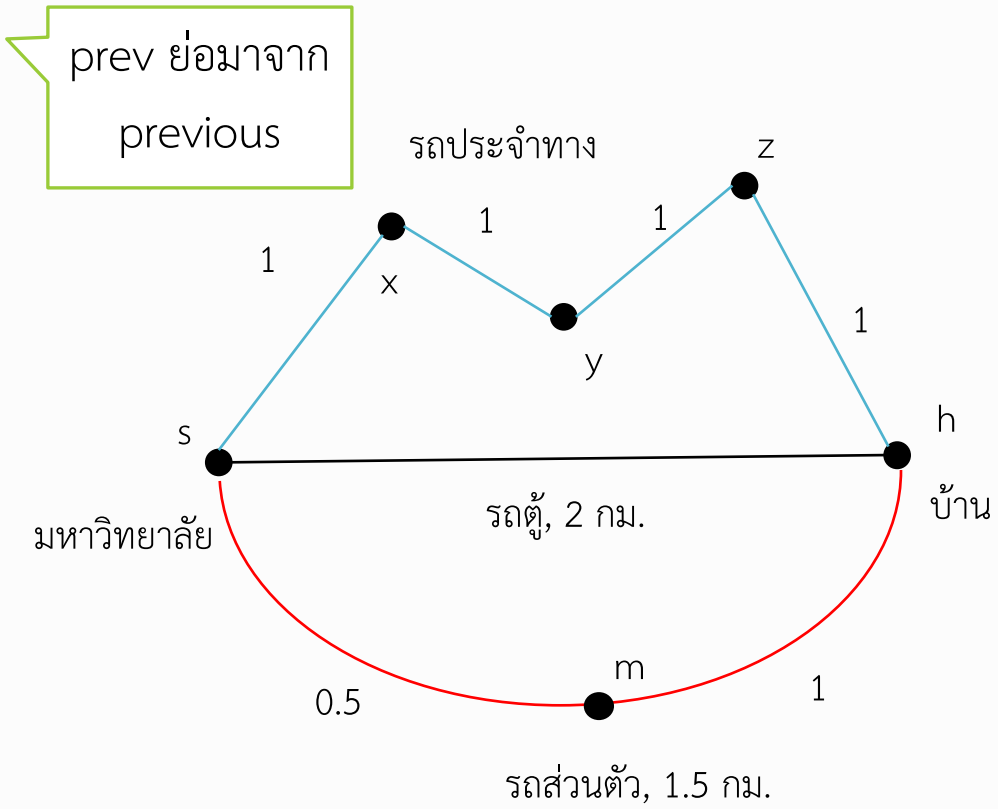
prev(h) = y

2. เส้นทาง s, m, h

prev(m) = s

prev(h) = m

prev(s) = *null*



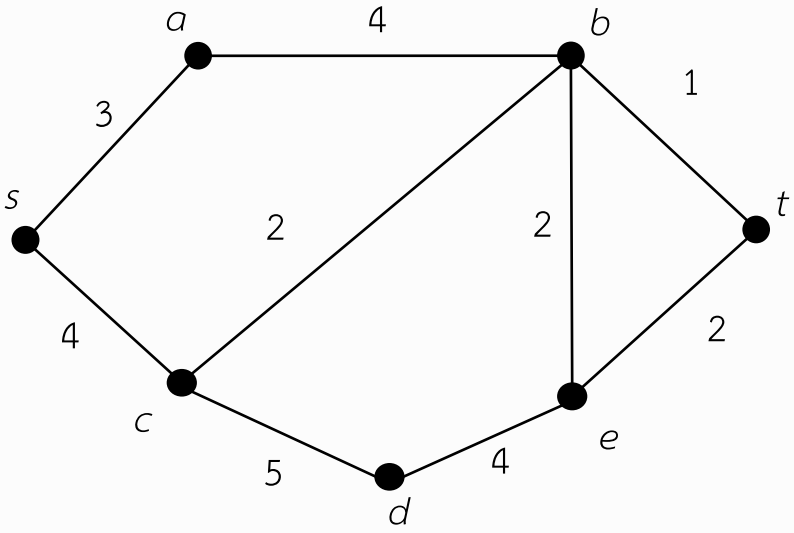
อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

อัลกอริทึมที่เป็นที่นิยมมากๆในการหาเส้นทางที่สั้นที่สุดภายในกราฟคือ อัลกอริทึมของไดส์ตร้า (Dijkstra’s Algorithm) ให้ G(V, E) เป็นกราฟ อัลกอริทึมนี้สามารถหาเส้นทางที่สั้นที่สุดจาก s ไป t ได้ การทำงานหลักๆของอัลกอริทึมนี้เป็นการทำงานวนซ้ำเป็นรอบๆ โดย

1. เริ่มจากจุดยอดเริ่มต้น s ในรอบแรกจะคำนวณหาหาค่า dist(u) โดยที่จุดยอด u คือจุดยอดอื่นๆที่มีเส้นเชื่อมมายัง s จนครบทุกจุด

2. จากนั้นเริ่มรอบถัดมาโดยให้จุดยอด v ที่มีค่า dist(v) น้อยที่สุดแล้วมาเป็นจุดเริ่มต้น หาจุดยอด u อื่นๆที่เชื่อมกับจุด v แล้วคำนวณหา dist(u) จนครบทุกจุดยอดทำซ้ำไปเรื่อยๆจนครบทุกจุดยอด

3. ถ้าจุดยอดใหม่ในข้อ 2 ที่เลือกมาเป็นจุดยอด t (ที่เป็นปลายทาง) อัลกอริทึมจะหยุดทำงาน



อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

ในทางคณิตศาสตร์แล้วรายละเอียดของอัลกอริทึมมีดังนี้

1. ให้ T เป็นเซ็ตของจุดยอดทั้งหมดหรือจุดยอดที่ยังไม่ถูกเยี่ยม (unvisited vertex)

2. กำหนดจุดยอดปัจจุบัน (current vertex) เป็นจุดยอด s และปรับ ระยะทางของ dist(s) = 0 และ dist(a) = ∞ โดยที่ a เป็นจุดยอดอื่นๆที่ a $\neq$ s

3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$ ถ้า $dist(v) + weight(v,u) < dist(u)$ ทำ 3.2 และ 3.3
   - 3.2 อัปเดต $dist(u) = dist(v) + weight(v,u)$
   - 3.3 $prev(u) = v$

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

เจอเส้นทางใหม่ที่สั้นกว่า และอัปเดตเส้นทางใหม่นั้น

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

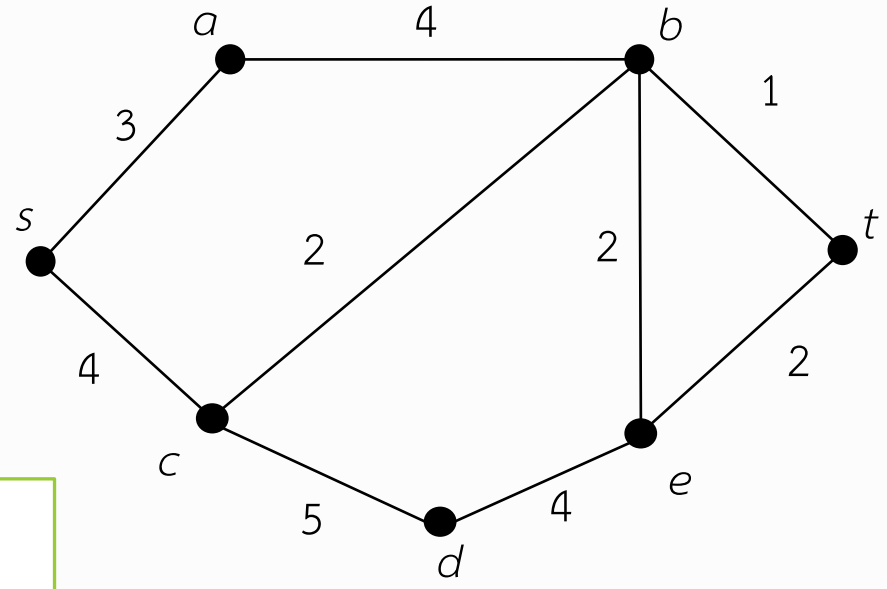
จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	∞	null
b	∞	null
c	∞	null
d	∞	null
e	∞	null
t	∞	null

(2)

$visited = \{\quad\}$

$T = \{s, a, b, c, d, e, t\}$ ← (1)

จุดยอดปัจจุบัน คือ ...



1. ให้ T เป็นเซ็ตของจุดยอดทั้งหมดหรือจุดยอดที่ยังไม่ถูกเยี่ยม (unvisited vertex)

2. กำหนดจุดยอดปัจจุบัน (current vertex) เป็นจุดยอด s และปรับ ระยะทางของ dist(s) = 0 และ dist(a) = ∞ โดยที่ a เป็นจุดยอดอื่นๆที่ a ≠ s

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0 ← (3)	null
a	∞	null
b	∞	null
c	∞	null
d	∞	null
e	∞	null
t	∞	null

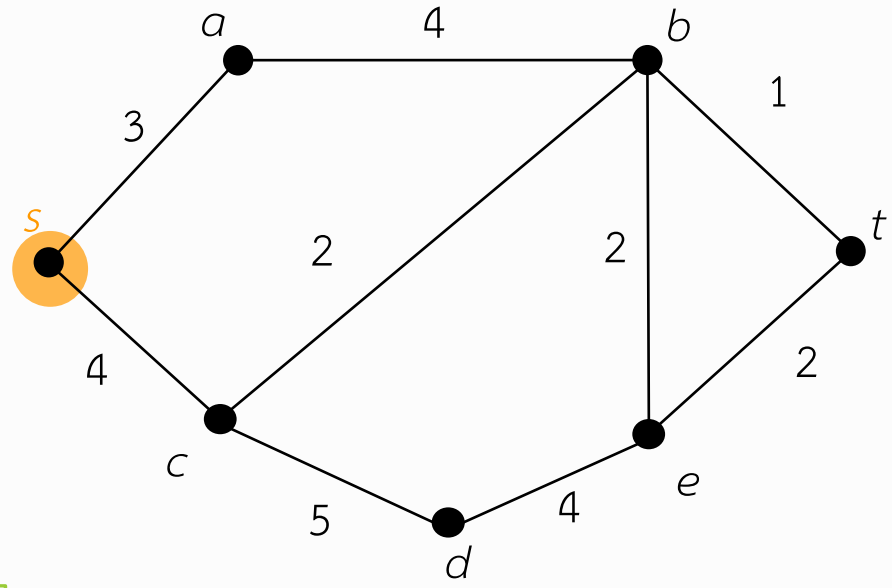
$visited = \{\quad\}$

$T = \{s, a, b, c, d, e, t\}$

จุดยอดปัจจุบัน คือ $v = s$ ← (3)
เพราะ s มี $dist(s)$ น้อยที่สุด

$u = a$ ← (3.1)
$dist(s) + weight(s, a) < dist(a)$?

$u = c$ ← (3.1)
$dist(s) + weight(s, c) < dist(c)$?



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$
     ถ้า $dist(v) + weight(v, u) < dist(u)$ ทำ 3.2 และ 3.3

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	∞ ← (3.1)	null
b	∞	null
c	∞ ← (3.1)	null
d	∞	null
e	∞	null
t	∞	null

$visited = \{\quad\}$

$T = \{s, a, b, c, d, e, t\};\ v = s$

$u = a$ ← (3.1)

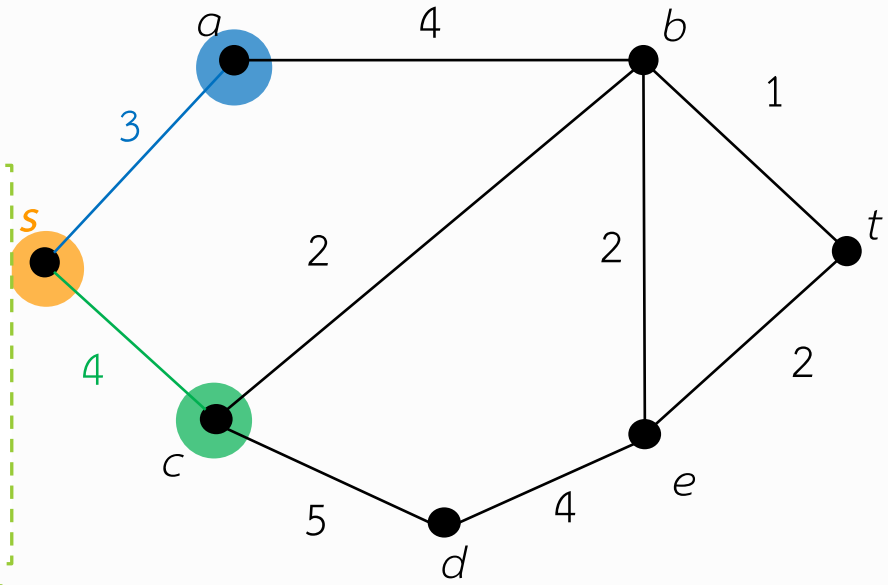
$dist(s) + weight(s, a) < dist(a)$?

$0 + 3 < \infty$?

$u = c$ ← (3.1)

$dist(s) + weight(s, c) < dist(c)$?

$0 + 4 < \infty$?



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$
     ถ้า $dist(v) + weight(v, u) < dist(u)$ ทำ 3.2 และ 3.3

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	∞ → 3 ← (3.2)	s ← (3.3)
b	∞	null
c	∞ → 4 ← (3.2)	s ← (3.3)
d	∞	null
e	∞	null
t	∞	null

$visited = \{s \quad\} \leftarrow (4)$

$T = \{s, a, b, c, d, e, t\};\ v = s$

$u = a$

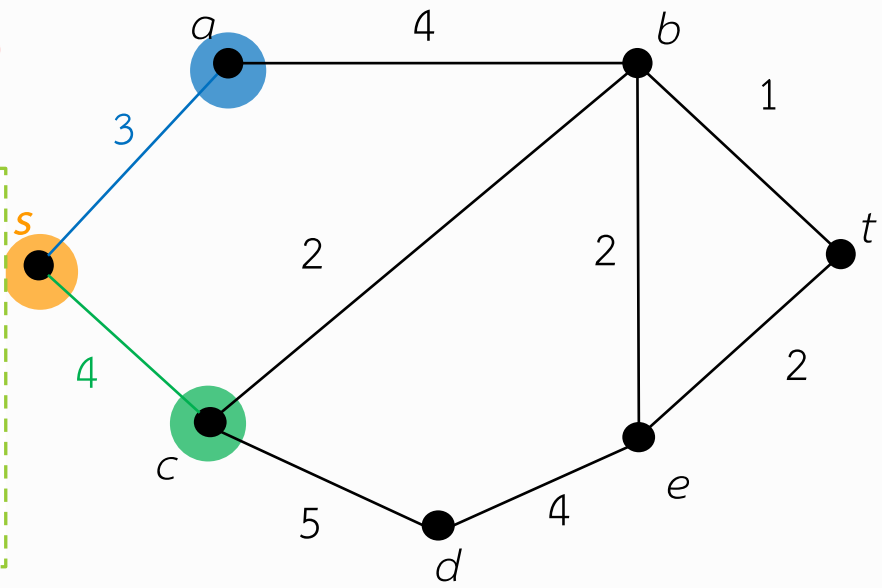
$dist(s) + weight(s,a) < dist(a)\ ?$

$0 + 3 < \infty\ ?$

$u = c$

$dist(s) + weight(s,c) < dist(c)\ ?$

$0 + 4 < \infty\ ?$



- 3.2 อัปเดต $dist(u) = dist(v) + weight(v,u)$
- 3.3 $prev(u) = v$

ขั้นตอนปัจจุบัน

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3 ← (3), (3.1)	s
b	∞	null
c	4	s
d	∞	null
e	∞	null
t	∞	null

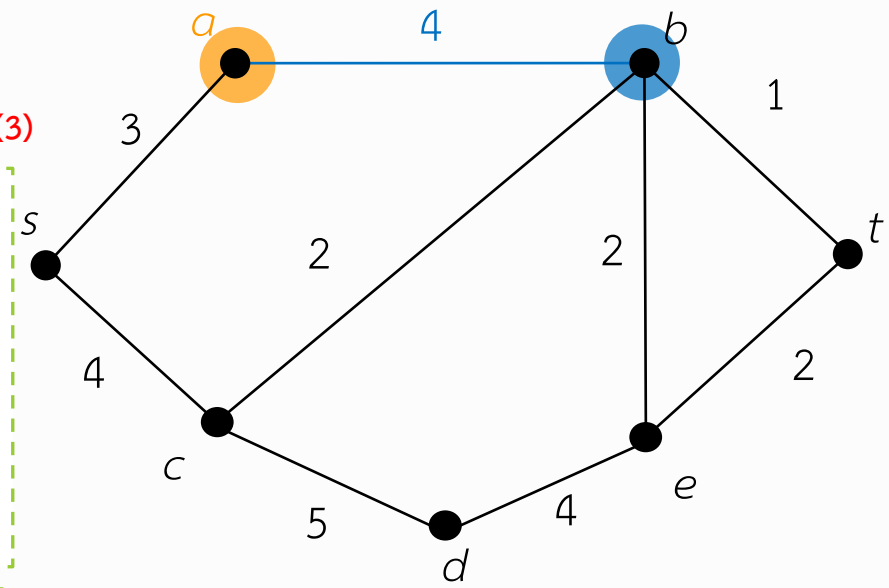
$visited = \{s \qquad\}$

$T = \{a, b, c, d, e, h\};\ v = a$ ← (3)

$u = b$ ← (3.1)

$dist(a) + weight(a, b) < dist(b)$?

$3 + \qquad 4 < \infty \qquad$?



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$
     ถ้า $dist(v) + weight(v, u) < dist(u)$ ทำ 3.2 และ 3.3

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	7 ← (3.2)	a ← (3.3)
c	4	s
d	∞	null
e	∞	null
t	∞	null

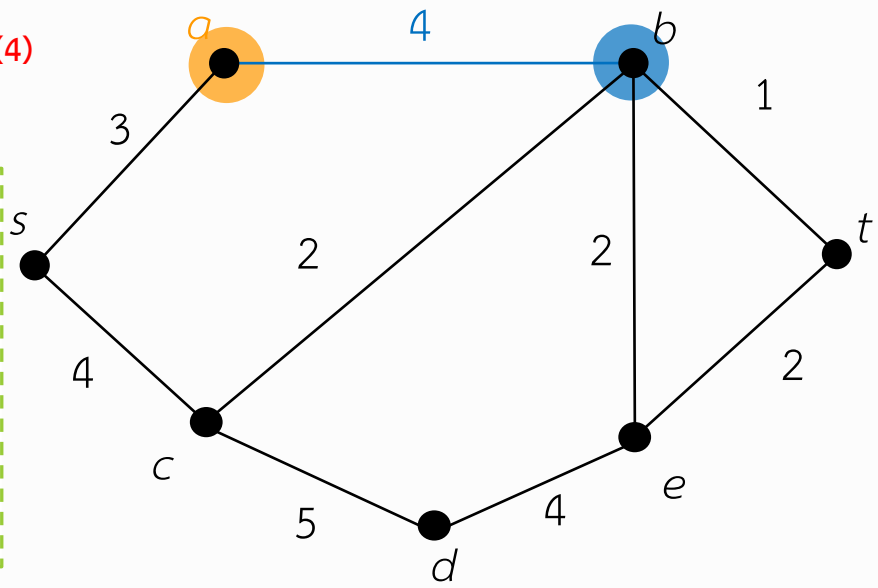
$visited = \{s, a \quad\} \leftarrow (4)$

$T = \{\cancel{a}, b, c, d, e, t\};\ v = a$

$u = b$

$dist(a) + weight(a, b) < dist(b)\ ?$

$3 + \quad 4 < \infty \quad ?$



- 3.2 อัปเดต $dist(u) = dist(v) + weight(v, u)$
- 3.3 $prev(u) = v$

ขั้นตอนปัจจุบัน

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	7	a
c	4 ← (3), (3.1)	s
d	∞	null
e	∞	null
t	∞	null

$visited = \{s, a \quad\}$

$T = \{b, c, d, e, t\};\ v = c$ ← (3)

$u = b$ ← (3.1)

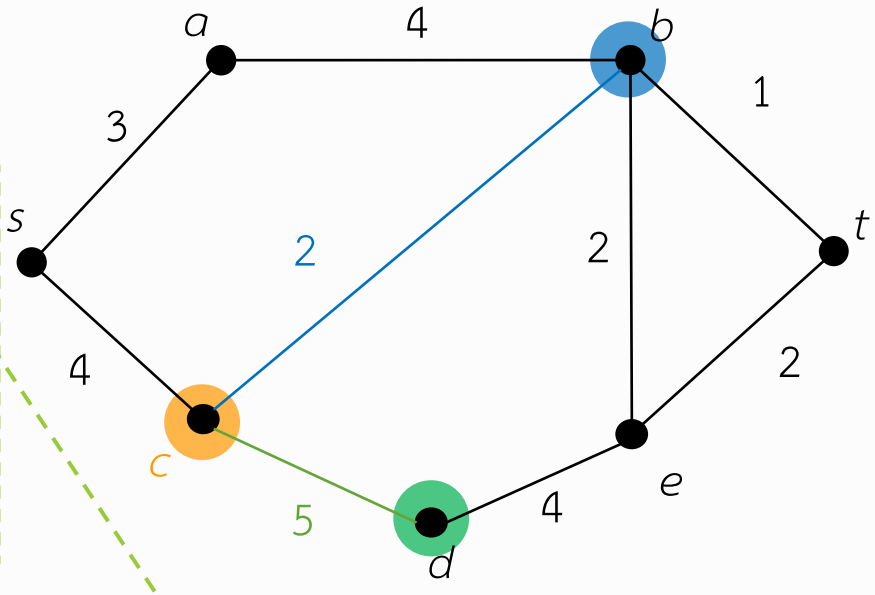
$dist(c) + weight(c, b) < dist(b)$?

$4 + 2 < 7$?

$u = d$ ← (3.1)

$dist(c) + weight(c, d) < dist(d)$?

$4 + 5 < \infty$?



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$
     ถ้า $dist(v) + weight(v, u) < dist(u)$ ทำ 3.2 และ 3.3

ขั้นตอนปัจจุบัน

เจอเส้นทางใหม่ที่สั้นกว่า และ
อัปเดตเส้นทางใหม่นั้น

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	~~7~~ → 6 ← (3.2)	~~a~~ → c ← (3.3)
c	4	s
d	9 ← (3.2)	c ← (3.3)
e	∞	null
t	∞	null

$visited = \{s, a, c \quad\} \leftarrow (4)$

$T = \{b, ~~c~~, d, e, t\};\ v = c$

$u = b$

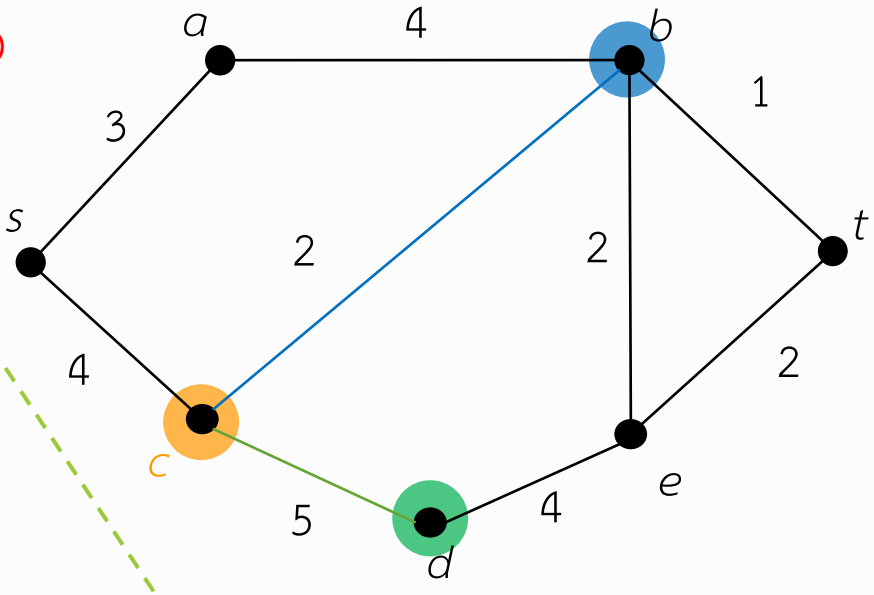
$dist(c) + weight(c, b) < dist(b)\ ?$

$4 + 2 < 7\ ?$

$u = d$

$dist(c) + weight(c, d) < dist(d)\ ?$

$4 + 5 < \infty\ ?$



- 3.2 อัปเดต $dist(u) = dist(v) + weight(v, u)$
- 3.3 $prev(u) = v$

ขั้นตอนปัจจุบัน

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

เจอเส้นทางใหม่ที่สั้นกว่า และ
อัปเดตเส้นทางใหม่นั้น

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	6 ← (3), (3.1)	c
c	4	s
d	9	c
e	∞ → 8 ← (3.2)	b ← (3.3)
t	∞ → 7 ← (3.2)	b ← (3.3)

$visited = \{s, a, c, b \quad\} \leftarrow (4)$

$T = \{\cancel{b}, d, e, t\};\ v = b \leftarrow (3)$

$u = e \leftarrow (3.1)$

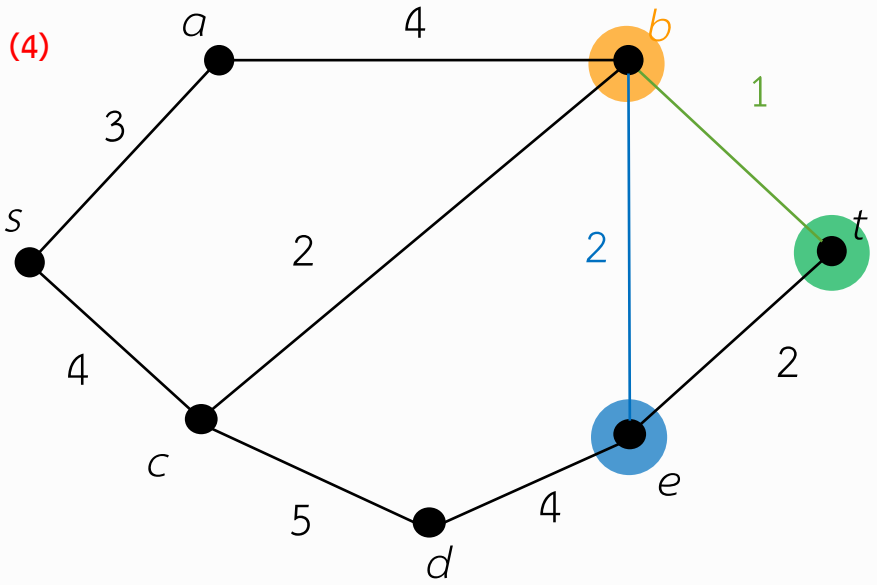
$dist(b) + weight(b,e) < dist(e)\ ?$

$6 + \quad 2 < \infty \quad ?$

$u = t \leftarrow (3.1)$

$dist(b) + weight(b,t) < dist(t)\ ?$

$6 + \quad 1 < \infty \quad ?$



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$ ถ้า $dist(v) + weight(v,u) < dist(u)$ ทำ 3.2 และ 3.3
   - 3.2 อัปเดต $dist(u) = dist(v) + weight(v,u)$
   - 3.3 $prev(u) = v$

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	6	c
c	4	s
d	9	c
e	8	b
t	7 ← (3), (3.1)	b

$visited = \{s, a, c, b, t \quad\}$ ← (4)

$T = \{d, e, \cancel{t}\}$; $v = t$ ← (3)

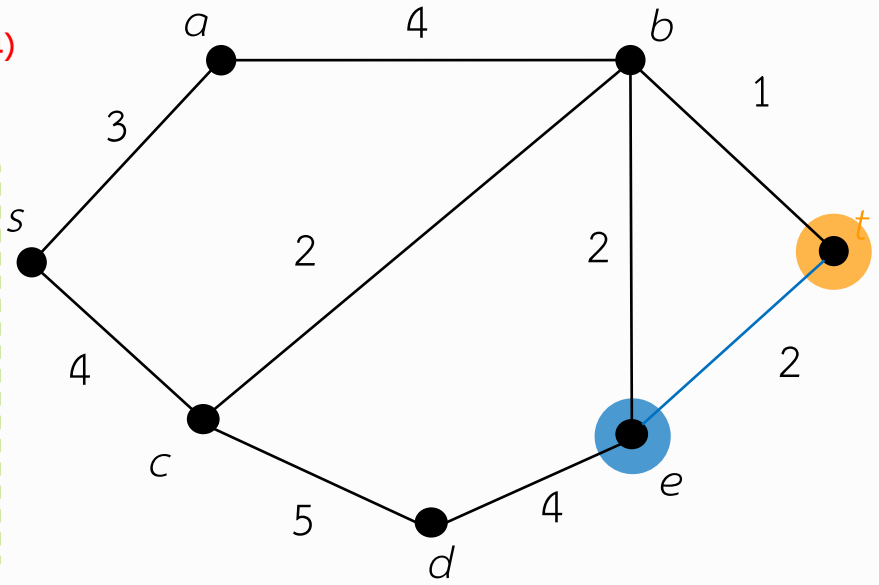
$u = e$ ← (3.1)

$dist(t) + weight(t, e) < dist(e)$?

$7 + \quad 2 < 8$?

$9 < 8$?

*ไม่ทำ 3.2 และ 3.3



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$ ถ้า $dist(v) + weight(v, u) < dist(u)$ ทำ 3.2 และ 3.3
   - ~~3.2 อัปเดต $dist(u) = dist(v) + weight(v, u)$~~
   - ~~3.3 $prev(u) = v$~~

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	6	c
c	4	s
d	9	c
e	8 ← (3), (3.1)	b
t	7	b

$visited = \{s, a, c, b, t, e \quad\}$ ← (4)

$T = \{d, \cancel{e}\};\ v = e$ ← (3)

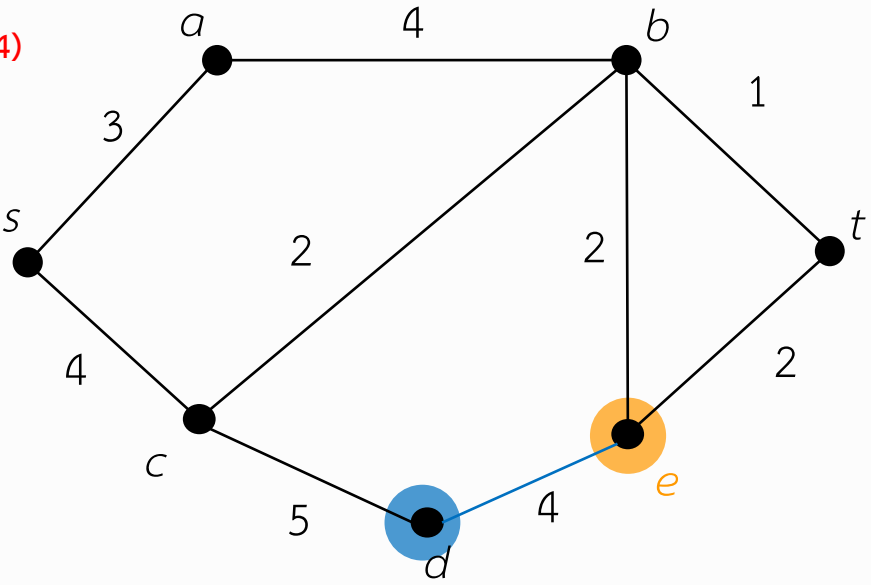
$u = d$ ← (3.1)

$dist(e) + weight(e,d) < dist(d)$?

$8 + \quad 4 < 9$?

$12 < 9$?

*ไม่ทำ 3.2 และ 3.3



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$ ถ้า $dist(v) + weight(v,u) < dist(u)$ ทำ 3.2 และ 3.3
   - ~~3.2 อัปเดต $dist(u) = dist(v) + weight(v,u)$~~
   - ~~3.3 $prev(u) = v$~~

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

ขั้นตอนปัจจุบัน

อัลกอริทึมของไดส์ตร้า, Dijkstra's algorithm

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
s	0	null
a	3	s
b	6	c
c	4	s
d	9 ← (3), (3.1)	c
e	8	b
t	7	b

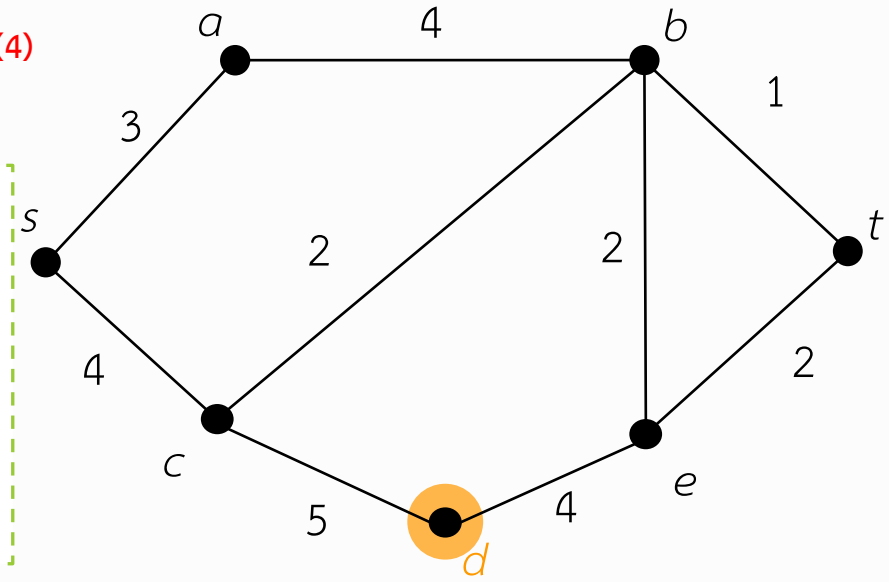
$visited = \{s, a, b, c, t, e, d\}$ ← (4)

$T = \{\cancel{d}\};\ v = d$ ← (3)

$u =$ ไม่มี ← (3.1)

เพราะไม่มีจุดยอดใดๆเป็นสมาชิกของ T

*ไม่ทำ 3.2 และ 3.3



3. หาจุดยอด $v \in T$ ที่ $dist(v)$ น้อยที่สุด, ให้ v เป็นจุดยอดปัจจุบัน
   - 3.1 สำหรับทุกจุดยอด u ที่ประชิดกับ v และ $u \in T$ ถ้า $dist(v) + weight(v,u) < dist(u)$ ทำ 3.2 และ 3.3
   - ~~3.2 อัปเดต $dist(u) = dist(v) + weight(v,u)$~~
   - ~~3.3 $prev(u) = v$~~

4. $T = T - \{v\}$ และกลับไปทำข้อ 3 ใหม่ จนกว่า T จะเป็นเซ็ตว่าง

ขั้นตอนปัจจุบัน

ตารางที่ได้คือเส้นทางที่สั้นที่สุดจากจุด s ไปยังจุดใดๆในกราฟ
โดยสามารถระบุเส้นทางเดินนั้นๆโดยการไล่ $prev()$ ย้อนกลับจนไปถึงจุดเริ่มต้น

แบบฝึกหัด 1, หาเส้นทางที่สั้นที่สุดจาก a ไป d

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
a	0	null
b		
c		
d		
e		
f		

$visited = \{$ $\}$

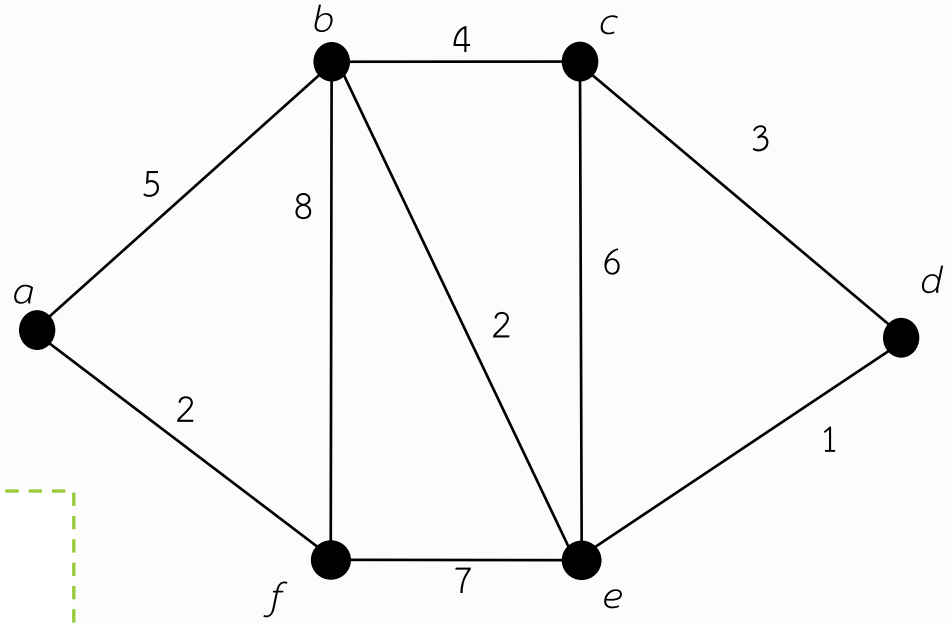
$T = \{$ $\}$

$v =$

$u =$

$u =$

$u =$



แบบฝึกหัด 2, หาเส้นทางที่สั้นที่สุดจาก a ไป d

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
a	0	null
b		
c		
d		
e		
f		
g		
h		

$visited = \{$ $\}$

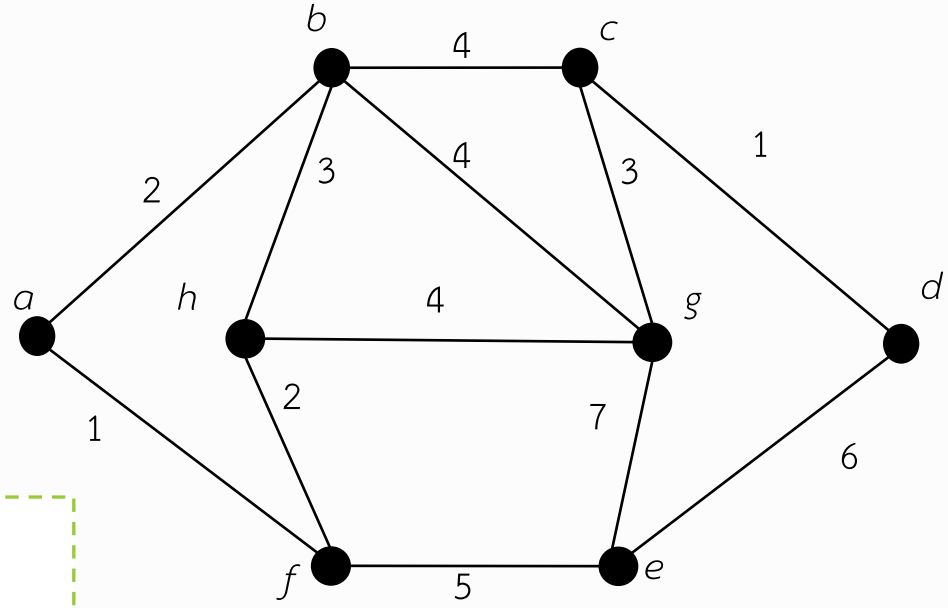
$T = \{$ $\}$

$v =$

$u =$

$u =$

$u =$



แบบฝึกหัด 3, หาเส้นทางที่สั้นที่สุดจาก a ไป d

จุดยอด v	ระยะทาง, $dist(v)$	จุดยอดก่อนหน้า, $prev(v)$
a	0	null
b		
c		
d		
e		
f		
g		
h		

$visited = \{$ $\}$

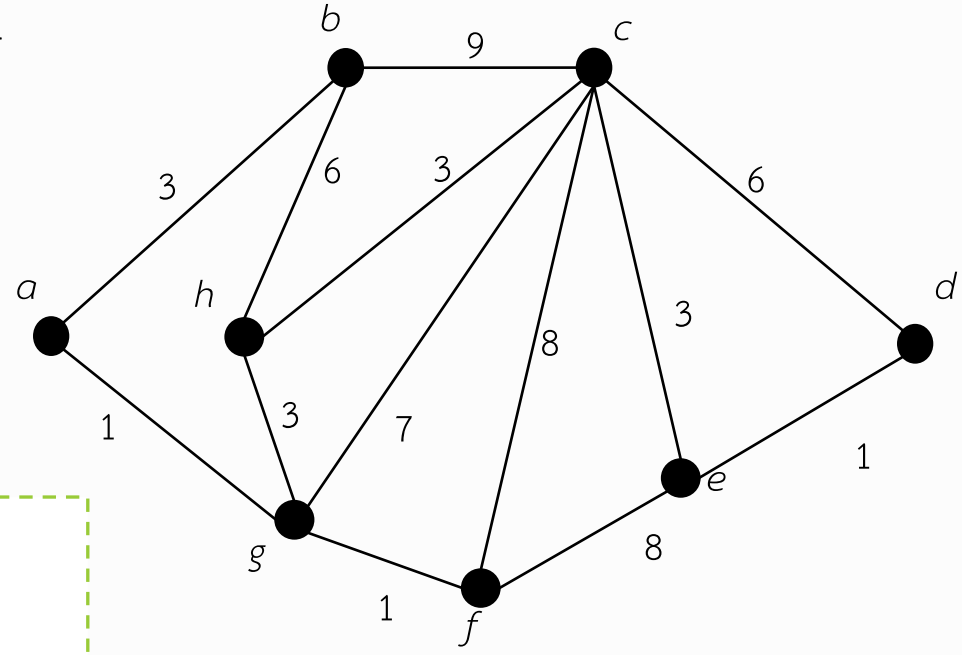
$T = \{$ $\}$

$v =$

$u =$

$u =$

$u =$



สรุป

อัลกอริทึมของไดส์ตร้า Dijkstra’s algorithm

องค์ประกอบที่จำเป็น: น้ำหนัก $weight(v,u)$, ระยะทาง $dist(v,u)$, จุดยอดก่อนหน้า $prev(v)$

การทำซ้ำหลายๆรอบของอัลกอริทึมไดส์ตร้า

1. เลือกจุดยอดปัจจุบัน v
2. คำนวณระยะทาง v ไป u เช็คว่าน้อยกว่าหรือไม่?
3. อัปเดตระยะทาง และ จุดยอดก่อนหน้า
4. วนไปเรื่อยๆจนจุดที่ยังไม่ได้เยี่ยมนั้นหมด