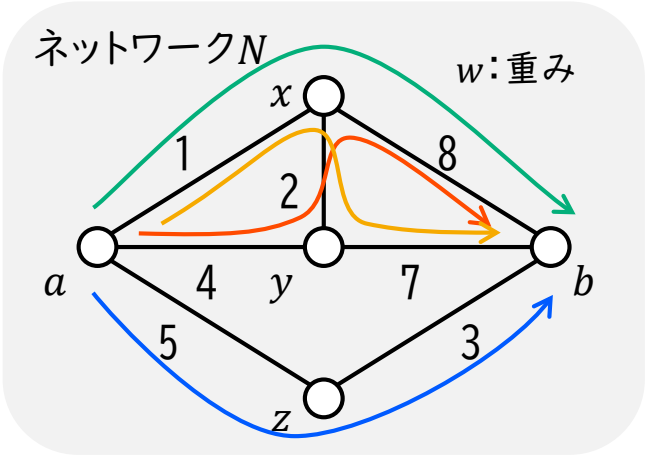
離散構造とアルゴリズム
(3-2) 最短路アルゴリズム

高橋篤司
東京工業大学 工学院 情報通信系

3-2（1）最短路アルゴリズム

■ 最短路問題（shortest-path problem）

- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$　← 重みは非負
  - 始点a $(\in V)$，終点b $(\in V)$
- 質問：ネットワーク$N = (G, w)$の最短(a, b)-路を一つ示せ

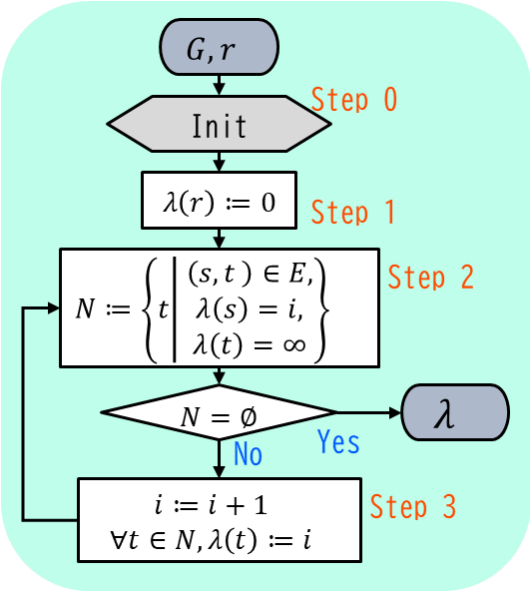


(a,b)-路	重み	
→	9	(=1+8)
→	10	(=1+2+7)
→	14	(=4+2+8)
→	8	(=5+3)

最短路アルゴリズム（BFS利用）

■ 最短路重み問題
- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 質問：ネットワーク$N = (G, w)$の最短(a, b)-路の重みを示せ

■ 距離ラベル付けアルゴリズムを利用したら？
- 辺重みが整数の場合
- グラフに変換してBFS利用



最短路アルゴリズム（BFS利用）

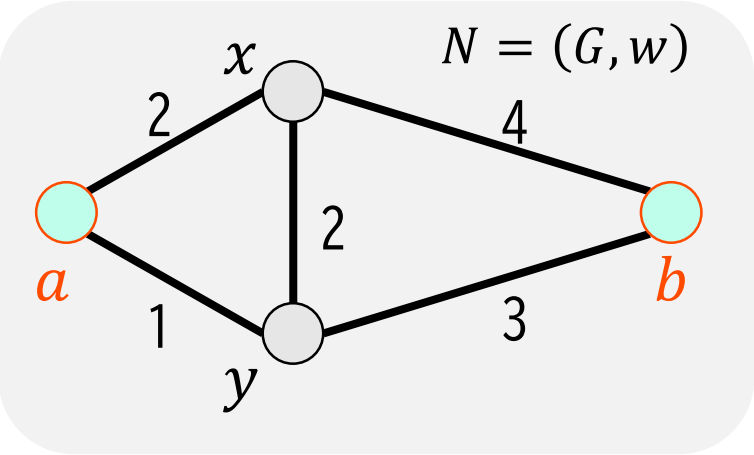
■ Algorithm 3.7g (BFS)

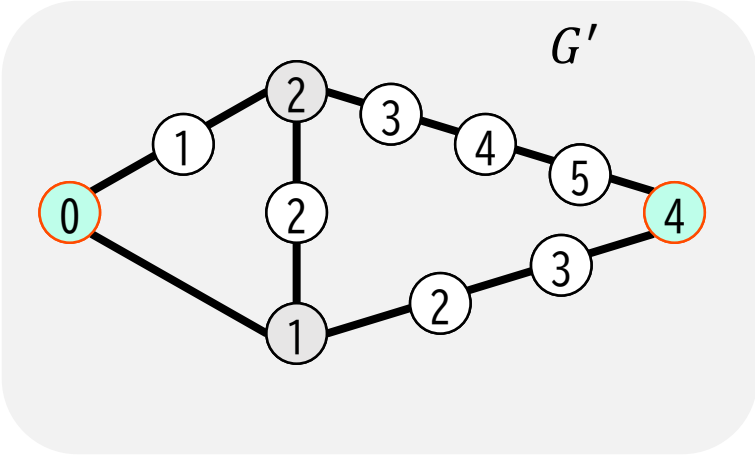
$n = |V|, m = |E|$

- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N = (G, w)$の最短(a, b)-路の重み

Step 0: Construct a graph G'
by replacing each edge e with a path of length $w(e)$

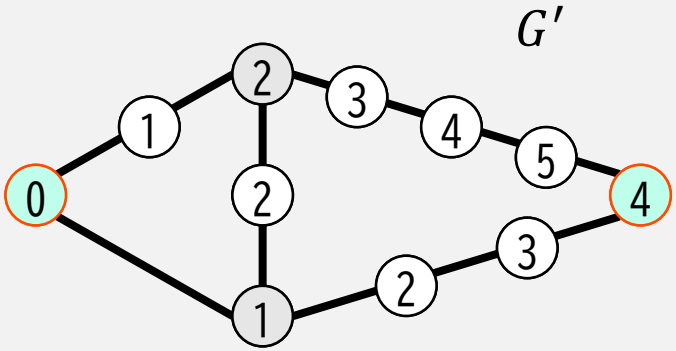
Step 1: Apply Algorithm 3.6 (graph-distance) on G' as $r = a$,
and output $\lambda(b)$





最短路アルゴリズム（BFS利用）

- Algorithm 3.7gは多項式時間アルゴリズムか?
  - NO!
- なぜ?
  - 辺重み最大値が2^nの場合
    - $\max\{w(e)|e \in E\} = 2^n$
      - 各辺重みはnビットで表現
  - 入力サイズ
    - $|G| = \Theta(n+m), |w| = \Theta(mn)$
    - $|(G,w)| = \Theta(mn)$
  - $G' = (V', E'), |V'| \geq 2^n$
- Algorithm 3.7gの時間計算量は $\Omega(2^n)$



入力サイズに対して
指数時間

- Algorithm 3.7g (BFS)

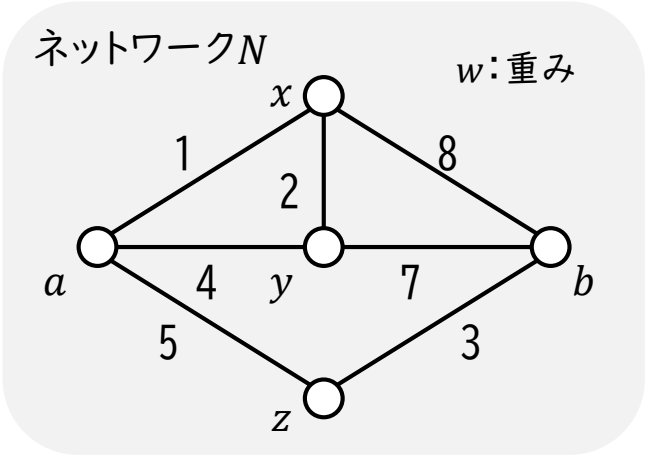
Step 0: Construct a graph G'
by replacing each edge e with a path of length $w(e)$

Step 1: Apply Algorithm 3.6 (graph-distance) on G' as $r = a$,
and output $\lambda(b)$

最短路アルゴリズム（重み行列利用）

■ 最短路重み問題
- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 質問：ネットワーク$N = (G, w)$の最短(a, b)-路の重みを示せ

■ 重み行列を利用したら？
- 辺数$1, 2, \ldots, \ell, \ldots, n-1$の最短$(u, v)$-ウォークの重みを求める
- 最短(a, b)-路の重みはその中にある

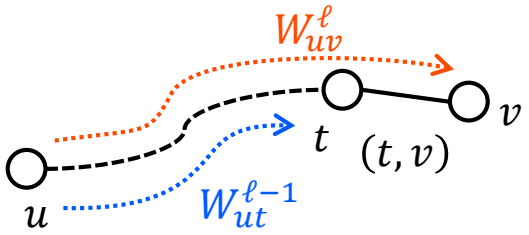


$$W = \begin{array}{c} \begin{matrix} a & x & y & z & b \end{matrix} \\ \begin{bmatrix} \infty & 1 & 4 & 5 & \infty \\ 1 & \infty & 2 & \infty & 8 \\ 4 & 2 & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty & 3 \\ \infty & 8 & 7 & 3 & \infty \end{bmatrix} \end{array} \begin{matrix} a \\ x \\ y \\ z \\ b \end{matrix}$$

$w_{i,j}$：辺数1の最短(i, j)-ウォークの重み

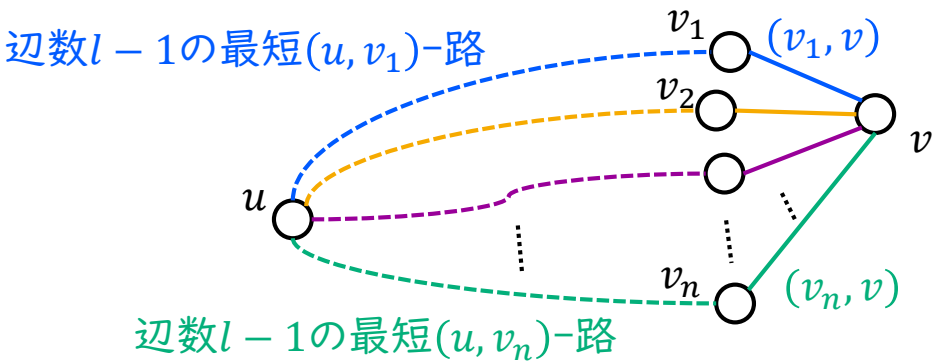
最短路アルゴリズム（重み行列利用）

- 辺数ℓの最短(u,v)-ウォークW_{uv}^{ℓ}（重みd_{uv}^{ℓ}）
  - 辺数$\ell-1$の最短(u,t)-ウォーク$W_{ut}^{\ell-1}$と辺(t,v)に分解できる



$$w(W_{uv}^{\ell}) = w(W_{ut}^{\ell-1}) + w(t,v)$$
$$d_{uv}^{\ell} = d_{ut}^{\ell-1} + w(t,v)$$

  - 辺数$\ell-1$の最短(u,v_i)-ウォーク+辺$(v_i,v) \in E(G)$の中にある
    - $d_{uv}^{\ell} = \min_{v_i \in V}\{d_{uv_i}^{\ell-1} + w(v_i,v)\}$



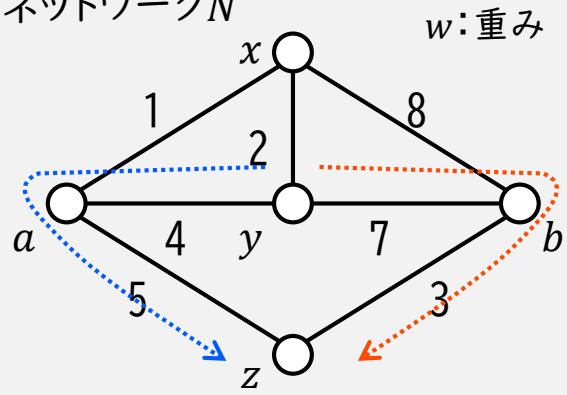
最短路アルゴリズム (重み行列利用)

■ $D_\ell = [d_{ij}^\ell]$

- (i,j)要素d_{ij}^ℓ：辺数ℓの最短(i,j)-ウォークの重み

- $D_0 = \begin{bmatrix} 0 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \begin{matrix} a \\ x \\ y \\ z \\ b \end{matrix}$

- $D_1 = \begin{bmatrix} \infty & 1 & 4 & 5 & \infty \\ 1 & \infty & 2 & \infty & 8 \\ 4 & 2 & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty & 3 \\ \infty & 8 & 7 & 3 & \infty \end{bmatrix} = W$



$$d_{ij}^\ell = \min_k \{d_{ik}^{\ell-1} + d_{kj}^1\} \ (\ell \geq 1)$$

- $D_2 = \begin{bmatrix} 2 & 6 & 3 & \infty & 8 \\ 6 & 2 & 5 & 6 & 9 \\ 3 & 5 & 4 & 9 & 10 \\ \infty & 6 & 9 & 6 & \infty \\ 8 & 9 & 10 & \infty & 6 \end{bmatrix} = \underbrace{\begin{bmatrix} \infty & 1 & 4 & 5 & \infty \\ 1 & \infty & 2 & \infty & 8 \\ 4 & 2 & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty & 3 \\ \infty & 8 & 7 & 3 & \infty \end{bmatrix}}_{D_1} \otimes \underbrace{\begin{bmatrix} \infty & 1 & 4 & 5 & \infty \\ 1 & \infty & 2 & \infty & 8 \\ 4 & 2 & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty & 3 \\ \infty & 8 & 7 & 3 & \infty \end{bmatrix}}_{D_1} \begin{matrix} a \\ x \\ y \\ z \\ b \end{matrix}$

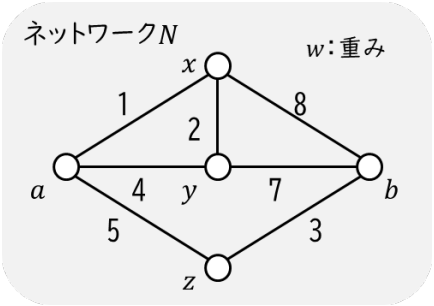
■ $d_{yz}^2 = 9 = \min\{4+5,\ 2+\infty,\ \infty+\infty,\ \infty+\infty,\ 7+3\}$

a経由　x経由　y経由　z経由　b経由

最短路アルゴリズム (重み行列利用)

- $D_\ell = [d_{ij}^\ell]$
  - (i,j)要素d_{ij}^ℓ：辺数ℓの最短(i,j)-ウォークの重み
  - $$D_3 = \begin{bmatrix} 7 & 3 & 6 & 7 & 10 \\ 3 & 7 & 4 & 11 & 9 \\ 6 & 4 & 7 & 8 & 11 \\ 7 & 11 & 8 & \infty & 9 \\ 10 & 9 & 11 & 9 & 17 \end{bmatrix} = \begin{bmatrix} 2 & 6 & 3 & \infty & 8 \\ 6 & 2 & 5 & 6 & 9 \\ 3 & 5 & 4 & 9 & 10 \\ \infty & 6 & 9 & 6 & \infty \\ 8 & 9 & 10 & \infty & 6 \end{bmatrix} \otimes \begin{bmatrix} \infty & 1 & 4 & 5 & \infty \\ 1 & \infty & 2 & \infty & 8 \\ 4 & 2 & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty & 3 \\ \infty & 8 & 7 & 3 & \infty \end{bmatrix} \begin{matrix} a \\ x \\ y \\ z \\ b \end{matrix}$$
    - $d_{ay}^3 = 6 = \min\{2+4,\ 6+2,\ 3+\infty,\ \infty+\infty,\ 8+7\}$

      a経由　x経由　y経由　z経由　b経由



- Algorithm 3.7w (DP)
  - 入力
    - 連結グラフ $G=(V,E)$，重み関数 $w: E \to \mathbb{R}^+$
    - 始点 $a\,(\in V)$，終点 $b\,(\in V)$
  - 出力：ネットワーク$N=(G,w)$の最短(a,b)-路の重み

$n = |V|, m = |E|$

Step 0: Set D_0 and D_1

Step 1: For $i = 2$ to $n-1$, $D_i = D_{i-1} \otimes D_1$

Step 2: Output $\min\{D_0(a,b), D_1(a,b), \ldots, D_{n-1}(a,b)\}$

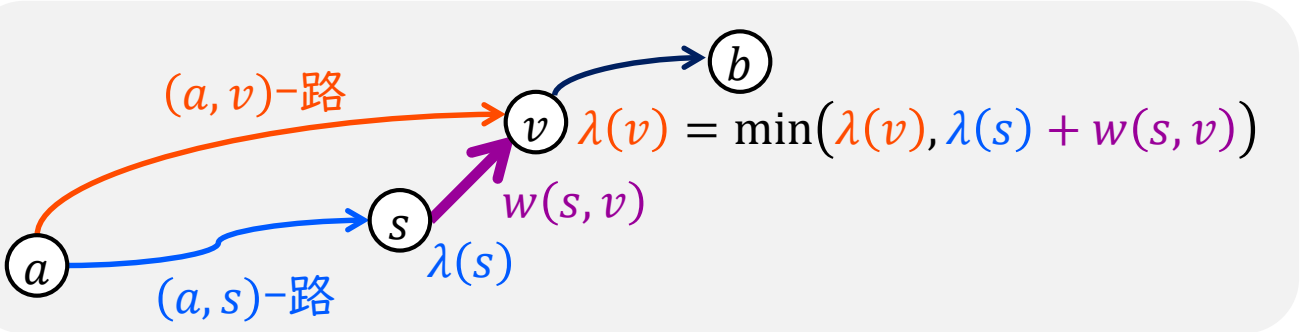
$O(n) \times n^2 \times (n-3)$

- 定理: Algorithm 3.7wの時間計算量は$O(n^4)$

最短路アルゴリズム（Bellman-Ford）

■ 最短路重み問題
- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 質問：ネットワーク$N = (G, w)$の最短(a, b)-路の重みを示せ

■ 路を探索したら？
- 始点から各点までの路を探索
- 辺を経由した方が短い路ならば情報を更新



最短路アルゴリズム（Bellman-Ford）

■ Algorithm 3.7u（Bellman-Ford）

$n = |V|, m = |E|$

- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N = (G, w)$の最短(a, b)-路の重み

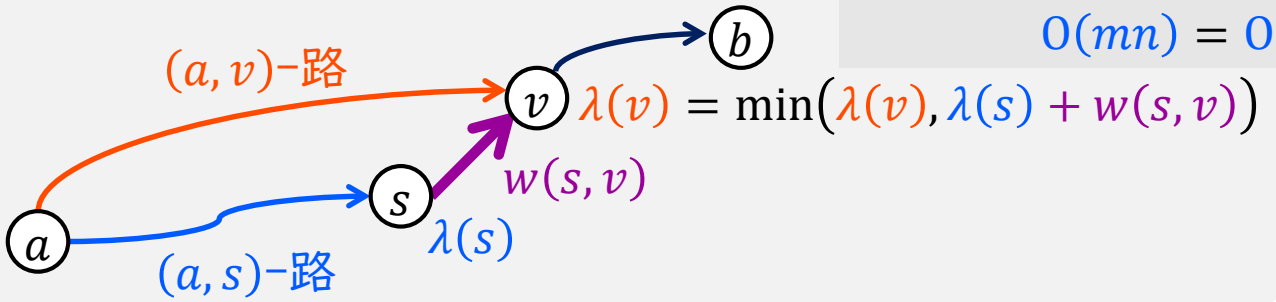
Step 0: Set $\lambda(v) = \infty, \forall v \in V$, and set $\lambda(a) = 0$ and Insert(Q, a)

Step 1: If $Q = \emptyset$ then output $\lambda(b)$

Step 2: Set $s =$ Delete(Q), and $\forall (s, v) \in E$, if $\lambda(v) > \lambda(s) + w(s, v)$ then set $\lambda(v) = \lambda(s) + w(s, v)$ and Insert(Q, v)

Step 3: Return to Step 2

1ラウンド：各辺高々1回評価$O(m)$
高々$n - 1$ラウンド
$O(mn) = O(n^3)$

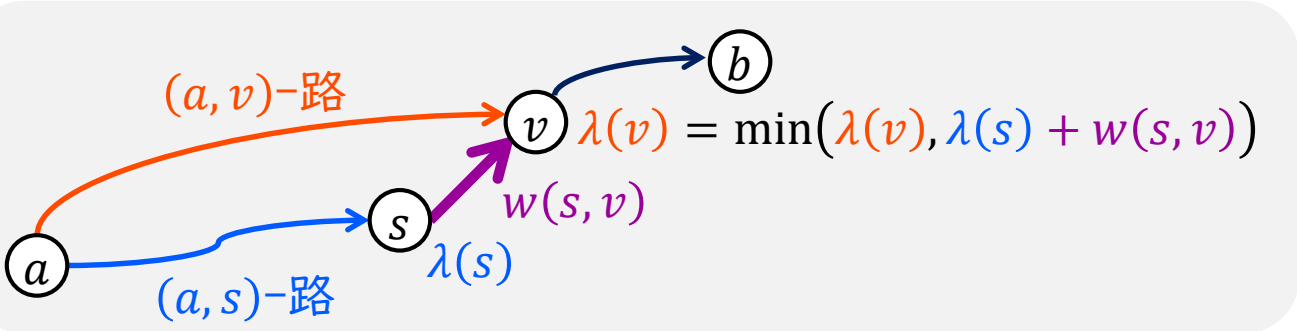


■ 定理：Algorithm 3.7uの時間計算量は$O(n^3)$

最短路アルゴリズム（Dijkstra）

■ 最短路問題（shortest-path problem）
- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a ($\in V$)，終点b ($\in V$)
- 質問：ネットワーク$N = (G, w)$の最短(a, b)-路を一つを示せ

■ 始点から各点までの路を探索
- 辺を経由した方が短い路ならば情報を更新
- 辺の評価の順序を工夫（始点に近い点に接続する辺から）



最短路アルゴリズム（Dijkstra）

■ Algorithm 3.7 (Dijkstra)

$n = |V|, m = |E|$

- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N = (G, w)$の最短(a, b)-路

Step 0: Set $\lambda(v) = \infty$, $p(v) = v$ $(\forall v \in V(G))$, $X = \emptyset$, $Y = V(G)$

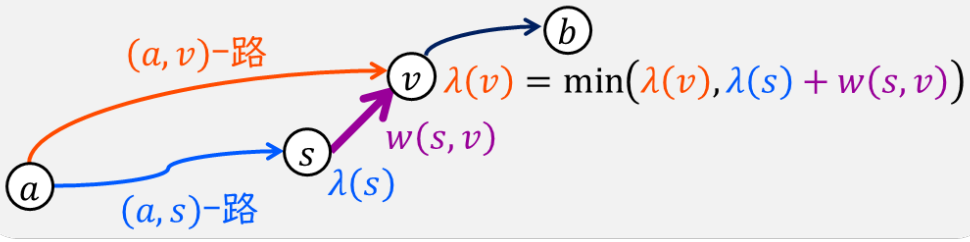
Step 1: Set $\lambda(a) = 0$ and $s = a$

Step 2: Set $X = X \cup \{s\}$, $Y = Y \setminus \{s\}$

Step 3: If $s = b$, then output $(a, \ldots, p(p(b)), p(b), b)$, and halt

Step 4: Set $\lambda(v) = \min\{\lambda(v), \lambda(s) + w(s, v)\}$ for every $v \in Y$.
Set $p(v) = s$ if the second term is smaller

Step 5: Select s such that $\lambda(s) = \min\{\lambda(v) | v \in Y\}$,
and return to Step 2



最短路アルゴリズム（Dijkstra）

■ Dijkstraアルゴリズム

入力

- 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
- 始点a $(\in V)$，終点b $(\in V)$

出力 ： ネットワーク$N=(G,w)$の最短(a,b)-路

$n=|V|, m=|E|$

⇨ 0. 初期設定
- 距離ラベル$\lambda(v) := \infty$, 前点ラベル$p(v) := v$
- 確定点集合 $X := \emptyset$, 未確定点集合$Y := V$

⇨ 1. 始点設定
- $\lambda(a) := 0, s := a$

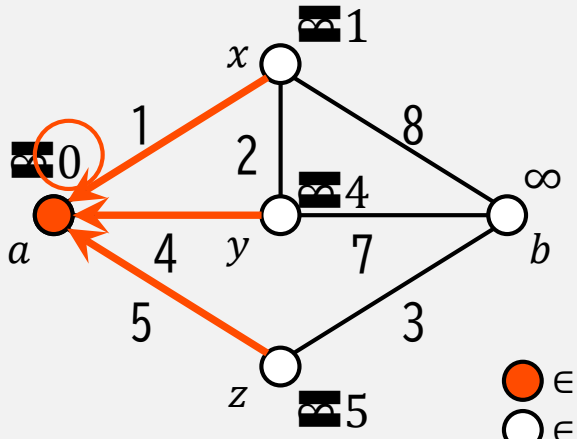
⇨ 2. 確定点集合更新（sの距離確定）
- $X := X \cup \{s\}, Y := Y \setminus \{s\}$

3. 終了判定
- $s=b \implies (a,\ldots,p(p(b)),\ p(b),\ b)$出力し終了

⇨ 4. ラベル更新
- $\forall v \in Y,\ \lambda(v) := \min(\lambda(v),\ \lambda(s)+w(s,v))$
  - 第2項が小さければ$p(v) := s$

5. 確定点選択
- s :=「Yの中でλが最小の点」
- Step 2へ戻る

ネットワークN



Step	a	x	y	z	b
0.（初期）	∞	∞	∞	∞	∞
1.（始点）	0	∞	∞	∞	∞
4.（更新）		1	4	5	∞

最短路アルゴリズム（Dijkstra）

■ Dijkstraアルゴリズム

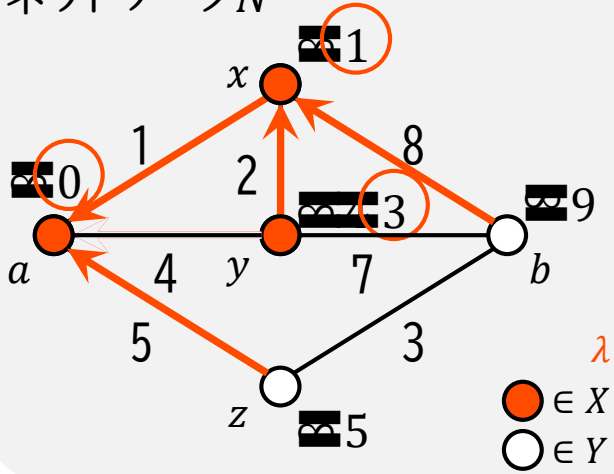
入力

- 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
- 始点a $(\in V)$，終点b $(\in V)$

出力 ： ネットワーク$N=(G,w)$の最短(a,b)-路

$n=|V|, m=|E|$

0. 初期設定
   - 距離ラベル$\lambda(v) := \infty$, 前点ラベル$p(v) := v$
   - 確定点集合 $X := \emptyset$, 未確定点集合$Y := V$
1. 始点設定
   - $\lambda(a) := 0, s := a$
2. 確定点集合更新（sの距離確定）
   - $X := X \cup \{s\}, Y := Y \setminus \{s\}$
3. 終了判定
   - $s=b \implies (a,\ldots,p(p(b)),\ p(b),\ b)$出力し終了
4. ラベル更新
   - $\forall v \in Y,\ \lambda(v) := \min(\lambda(v),\ \lambda(s)+w(s,v))$
     - 第2項が小さければ$p(v) := s$
5. 確定点選択
   - s :=「Yの中でλが最小の点」
   - Step 2へ戻る



Step	a	x	y	z	b
0.（初期）	∞	∞	∞	∞	∞
1.（始点）	0	∞	∞	∞	∞
4.（更新）		1	4	5	∞
4.（更新）			3	5	9
4.（更新）				5	9

最短路アルゴリズム（Dijkstra）

■ Dijkstraアルゴリズム

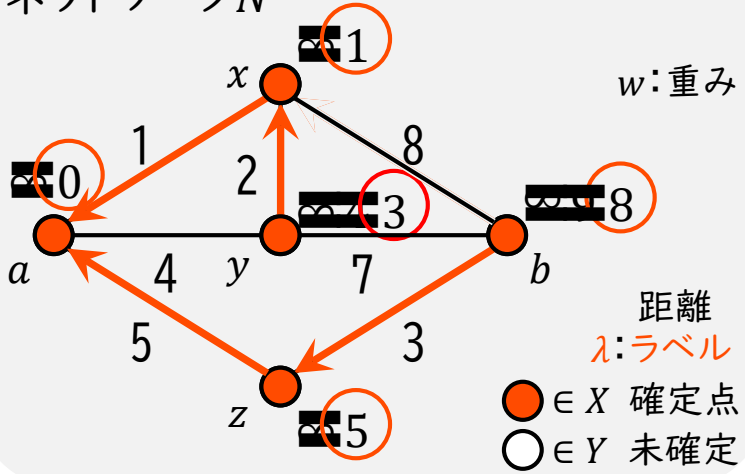
入力

- 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
- 始点a $(\in V)$，終点b $(\in V)$

出力 ： ネットワーク$N=(G,w)$の最短(a,b)-路

$n=|V|, m=|E|$

0. 初期設定
   - 距離ラベル$\lambda(v) := \infty$, 前点ラベル$p(v) := v$
   - 確定点集合 $X := \emptyset$, 未確定点集合$Y := V$
1. 始点設定
   - $\lambda(a) := 0, s := a$
2. 確定点集合更新（sの距離確定）
   - $X := X \cup \{s\}, Y := Y \setminus \{s\}$
3. 終了判定
   - $s=b \implies (a,\dots,p(p(b)),\ p(b),\ b)$出力し終了
4. ラベル更新
   - $\forall v \in Y,\ \lambda(v) := \min(\lambda(v),\ \lambda(s)+w(s,v))$
     - 第2項が小さければ$p(v) := s$
5. 確定点選択
   - s := 「Yの中でλが最小の点」
   - Step 2へ戻る



Step	a	x	y	z	b
0.（初期）	∞	∞	∞	∞	∞
1.（始点）	0	∞	∞	∞	∞
4.（更新）		1	4	5	∞
4.（更新）			3	5	9
4.（更新）				5	9
4.（更新）					8

最短路アルゴリズム（Dijkstra）

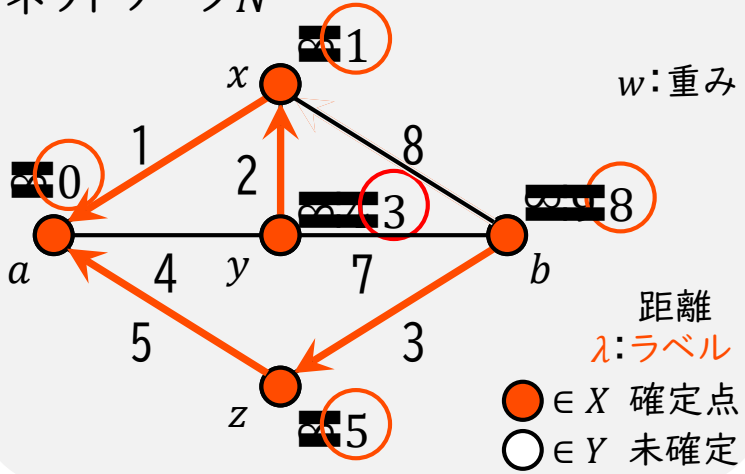
■ Dijkstraアルゴリズム $O(n^2)$

入力

- 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
- 始点a $(\in V)$，終点b $(\in V)$

出力 ： ネットワーク$N=(G,w)$の最短(a,b)-路

0. 初期設定 $n=|V|, m=|E|$
   - 距離ラベル$\lambda(v) := \infty$, 前点ラベル$p(v) := v$ $O(n)$
   - 確定点集合 $X := \emptyset$, 未確定点集合$Y := V$
1. 始点設定 $O(1)$
   - $\lambda(a) := 0, s := a$
2. 確定点集合更新（sの距離確定）
   - $X := X \cup \{s\}, Y := Y \setminus \{s\}$ $O(1) \times O(n) = O(n)$
3. 終了判定 $O(1) \times O(n) + O(n) = O(n)$
   - $s = b \implies (a, \ldots, p(p(b)),\ p(b),\ b)$出力し終了
4. ラベル更新 $O(m)$ or $O(n) \times O(n) = O(n^2)$
   - $\forall v \in Y,\ \lambda(v) := \min(\lambda(v),\ \lambda(s) + w(s,v))$
     - 第2項が小さければ$p(v) := s$
5. 確定点選択 $O(n) \times O(n) = O(n^2)$
   - $s :=$「Yの中でλが最小の点」
   - Step 2へ戻る



Step	a	x	y	z	b
0.（初期）	∞	∞	∞	∞	∞
1.（始点）	0	∞	∞	∞	∞
4.（更新）		1	4	5	∞
4.（更新）			3	5	9
4.（更新）				5	9
4.（更新）					8

最短路アルゴリズム（Dijkstra）

■ 定理（Theorem 3.8）：Algorithm 3.7(Dijkstra)は
最短路問題を$O(n^2)$で解く

□ 証明

➢ 時間計算量：$O(n^2)$
- Step 1~3：全体で$O(n)$　　各辺は高々1回評価
- Step 4：$|Y| = i$のとき，比較・加算・代入はi回，全体で$O(n^2)$ or $O(m)$
- Step 5：$|Y| = i$のとき，比較は$i-1$回，代入1回，全体で$O(n^2)$

➢ 正当性
- $v \in X$のλは更新なし
  ⇒ vをXに追加するとき$\lambda(v) = \mathbf{dis}_N(a,v)$であることを示す

– 数学的帰納法

◆ 初期段階 $X = \{a\}$
- $\lambda(a) = \mathbf{dis}_N(a,a) = 0$

■ Algorithm 3.7 (Dijkstra)
- 入力
  - 連結グラフ$G = (V,E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N = (G,w)$の最短(a,b)-路

Step 0: Set $\lambda(v) = \infty$, $p(v) = v$ $(\forall v \in V(G))$, $X = \emptyset$, $Y = V(G)$
Step 1: Set $\lambda(a) = 0$ and $s = a$
Step 2: Set $X = X \cup \{s\}$, $Y = Y \setminus \{s\}$
Step 3: If $s = b$, then output $(a, \ldots, p(p(b)), p(b), b)$, and halt
Step 4: Set $\lambda(v) = \min\{\lambda(v), \lambda(s) + w(s,v)\}$ for every $v \in Y$.
Set $p(v) = s$ if the second term is smaller
Step 5: Select s such that $\lambda(s) = \min\{\lambda(v) | v \in Y\}$,
and return to Step 2

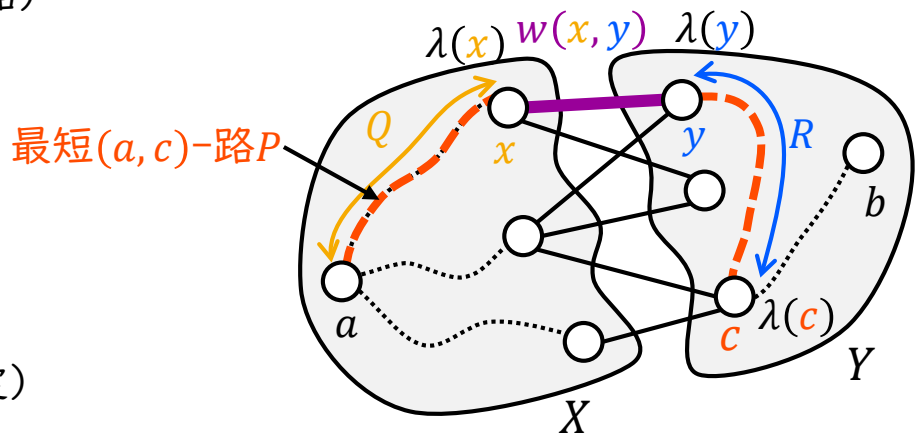
最短路アルゴリズム（Dijkstra）

証明(cont.)

◆ 帰納段階 $X \coloneqq X \cup \{c\}$ $(c \in Y)$

➢ $\lambda(c) = \mathbf{dis}_N(a, c)$ を示す

- $\mathbf{dis}_N(a, c) \leq \lambda(c)$ （∵長さ$\lambda(c)$ の (a, c)-路）
- $\lambda(c) \leq \lambda(u), \forall u \in Y$ （∵確定点選択）



最短(a, c)-路P

- $\exists (x, y) \in P$ s.t. $x \in X, y \in Y$
  - $\mathbf{dis}_N(a, c) = l(P) = l(Q) + w(x, y) + l(R)$

✓ $\lambda(x) = \mathbf{dis}_N(a, x)$ （∵帰納段階の仮定）

✓ $\lambda(y) \leq \lambda(x) + w(x, y)$ （∵ラベル更新）

✓ $l(R) \geq 0$ （∵重みは非負）

- $\lambda(y) \leq \lambda(x) + w(x, y) = \mathbf{dis}_N(a, x) + w(x, y) = l(Q) + w(x, y)$

- $\lambda(c) \leq \lambda(y) \leq l(Q) + w(x, y) \leq l(Q) + w(x, y) + l(R) = \mathbf{dis}_N(a, c) \leq \lambda(c)$

✓ $\lambda(c) = \mathbf{dis}_N(a, c)$ ■

最短路アルゴリズム (Dijkstra)

■ 定理 (Theorem 3.8): Algorithm 3.7(Dijkstra)は
最短路問題を$O(n^2)$で解く

➢ 計算量を改善できないか？
- Step 4: $O(m)$ ⇒ これは限界
- Step 5: $O(n^2)$
  - 最小値の選択を効率よくできるか？

■ Algorithm 3.7 (Dijkstra)
- 入力
  - 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N=(G,w)$の最短(a,b)-路

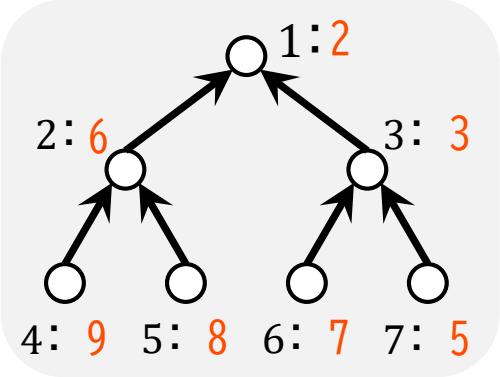
Step 0: Set $\lambda(v)=\infty$, $p(v)=v$ $(\forall v \in V(G))$, $X=\emptyset$, $Y=V(G)$
Step 1: Set $\lambda(a)=0$ and $s=a$
Step 2: Set $X=X\cup\{s\}$, $Y=Y\setminus\{s\}$
Step 3: If $s=b$, then output $(a,\ldots,p(p(b)),p(b),b)$, and halt
Step 4: Set $\lambda(v)=\min\{\lambda(v),\lambda(s)+w(s,v)\}$ for every $v\in Y$.
Set $p(v)=s$ if the second term is smaller
Step 5: Select s such that $\lambda(s)=\min\{\lambda(v)|v\in Y\}$,
and return to Step 2

ヒープ(heap)構造

■ 二分木

- 「親の値」は「子の値」より小さいか等しい
  - ✓ 根の値は最小値
- 配列で実現可能
  - ✓ 親アドレスa ⇒ 子アドレス$2a$, $2a+1$

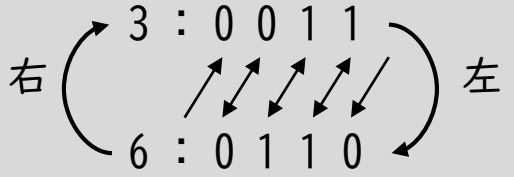
データ数n
⇒ 木の高さ $O(\log n)$

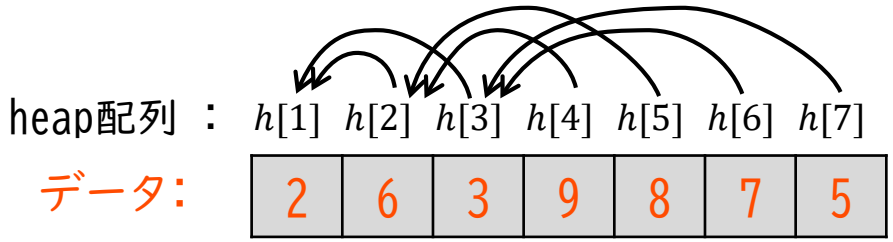


親アドレスから子アドレス計算
- 2倍 ⇒ 左1ビットシフト
- +1 ⇒ 加算 ⇒ 最下位ビットを1に

子アドレスから親アドレス計算
- 除算(½倍)
  ⇒ 右1ビットシフト

シフト演算



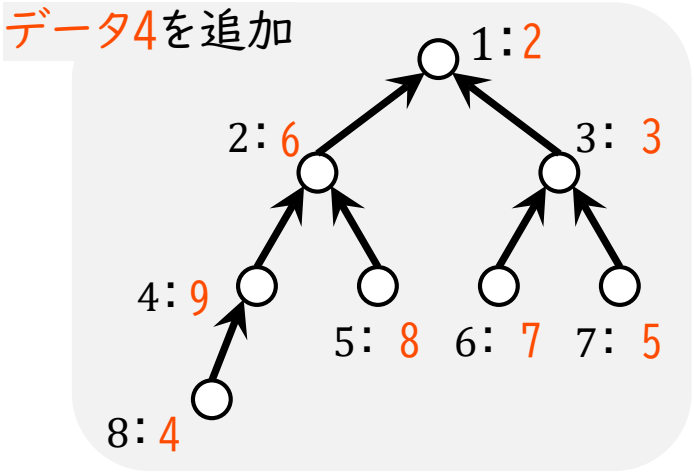


ヒープ(heap)構造

■ データの追加 ： $O(\log n)$

- $h[n+1]$にデータ追加
- 親の値より小さければ交換 ⇒ 根に向かって繰り返す

データ数n
⇒ 木の高さ $O(\log n)$



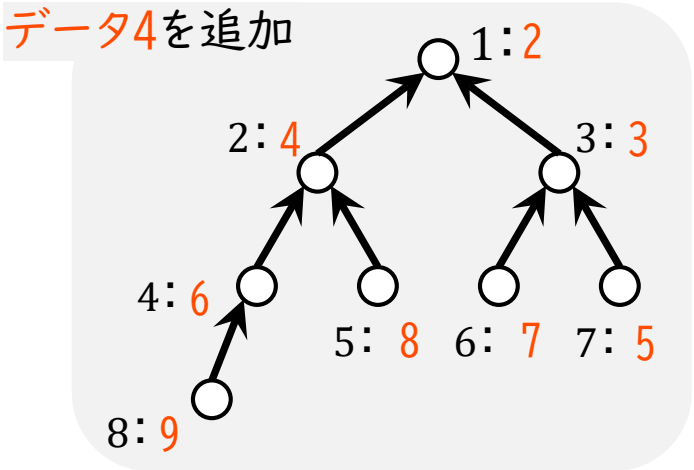
heap配列 ： $h[1]$ $h[2]$ $h[3]$ $h[4]$ $h[5]$ $h[6]$ $h[7]$ $h[8]$

データ：	2	4	3	6	8	7	5	9

ヒープ(heap)構造

- データの追加 ： $O(\log n)$
  - $h[n+1]$にデータ追加
  - 親の値より小さければ交換 ⇒ 根に向かって繰り返す
- 根の削除 ： $O(\log n)$
  - 最後のデータを根に移動
  - 小さい子より大きければ交換 ⇒ 交換したら葉に向かって繰り返す

データ数n
⇒ 木の高さ $O(\log n)$



heap配列 ： $h[1]$ $h[2]$ $h[3]$ $h[4]$ $h[5]$ $h[6]$ $h[7]$ $h[8]$

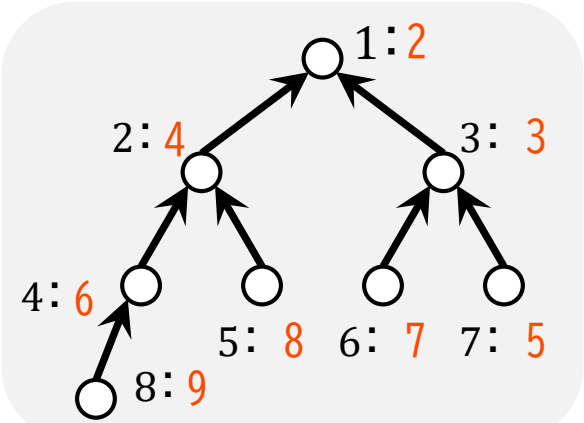
データ：	3	4	5	6	8	7	9	9

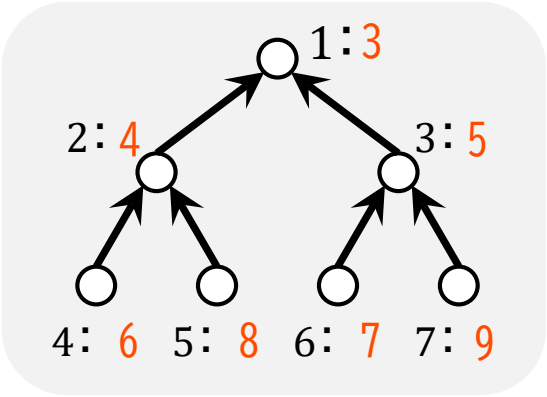
ヒープ(heap)構造

- heapソート $O(n \log n)$
- Dijkstraでのheapの利用
  - ヒープ利用なし $O(n^2)$
  - ヒープ利用 $O(m \log n)$
    - ✓ step 4.で点ラベル更新(減少)が$O(m)$回発生
      - 更新一回当たり$O(\log n)$で再構築可 ⇒ 最小値情報維持$O(m \log n)$
    - ■ 辺が多いグラフ $m = O(n^2)$ ⇒ $O(n^2 \log n)$ ∴ 遅い
    - ■ 辺が少ないグラフ $m = O(n)$ ⇒ $O(n \log n)$ ∴ 早い

データ数n
⇒ 木の高さ $O(\log n)$

$n = |V|, m = |E|$





$h[1]$	$h[2]$	$h[3]$	$h[4]$	$h[5]$	$h[6]$	$h[7]$	$h[8]$
2	4	3	6	8	7	5	9

$h[1]$	$h[2]$	$h[3]$	$h[4]$	$h[5]$	$h[6]$	$h[7]$
3	4	5	6	8	7	9

最短路の性質

■ 最適性の原理(principal of optimality)が成立

✓ 最短路の部分路は最短路

➢ 最短路Pの部分路Qは最短路である

□ 証明：背理法

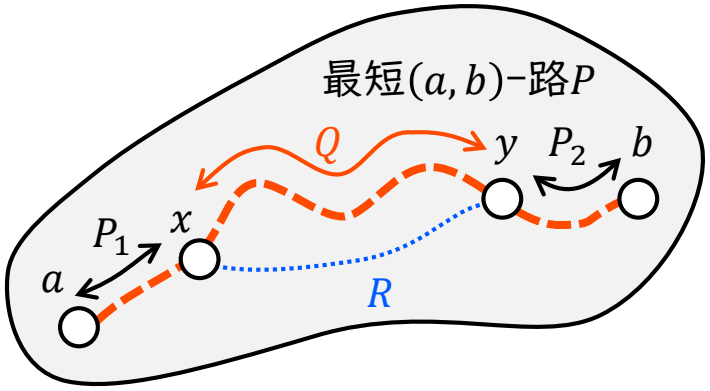
➢ Qは最短(x,y)−路でないと仮定

⇒ Qより短い(x,y)−路Rが存在　$\therefore l(Q) > l(R)$

⇒ Pより短い(a,b)−路$(P_1 + R + P_2)$が存在

⇒ Pが最短路であることに矛盾

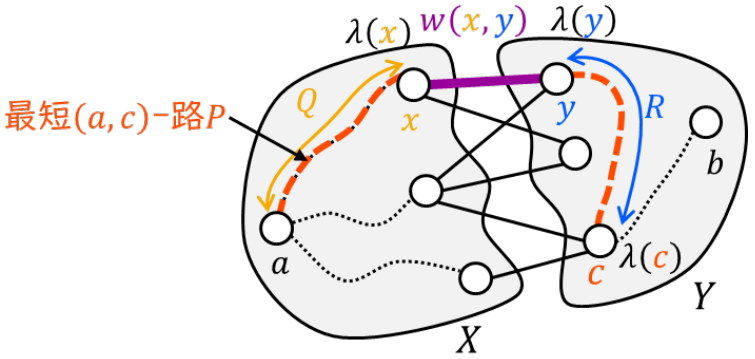
✓ Qは最短(x,y)−路 ■

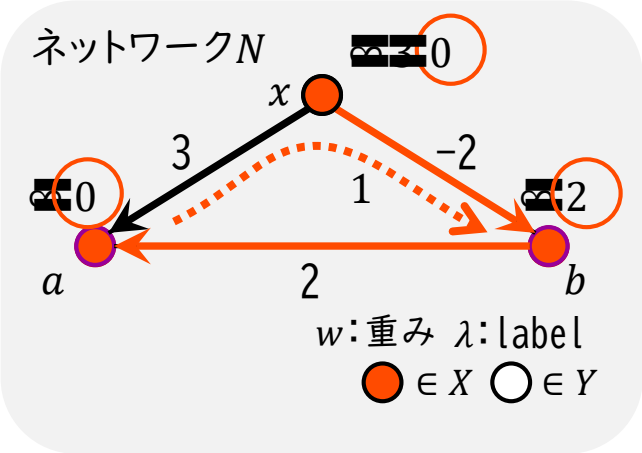


$$l(P) = l(P_1 + Q + P_2) > l(P_1 + R + P_2)$$

最短路アルゴリズム

- 負重み辺が存在する場合
  - 最適性の原理は不成立
  - Algorithm 3.7 (Dijkstra)
    - 最短(a,b)-路を出力するとは限らない
  - Algorithm 3.7u (Bellman-Ford)
    - 負閉路が存在しない場合（有向グラフ）
      - 最短(a,b)-路を出力





■ Algorithm 3.7 (Dijkstra)

$w: E \to \mathbb{R}$

- 入力
  - 連結グラフ$G=(V,E)$，重み関数$w: E \to \mathbb{R}^+$
  - 始点a $(\in V)$，終点b $(\in V)$
- 出力：ネットワーク$N=(G,w)$の最短(a,b)-路

Step 0: Set $\lambda(v)=\infty$, $p(v)=v$ $(\forall v \in V(G))$, $X=\emptyset$, $Y=V(G)$

Step 1: Set $\lambda(a)=0$ and $s=a$

Step 2: Set $X=X\cup\{s\}$, $Y=Y\setminus\{s\}$

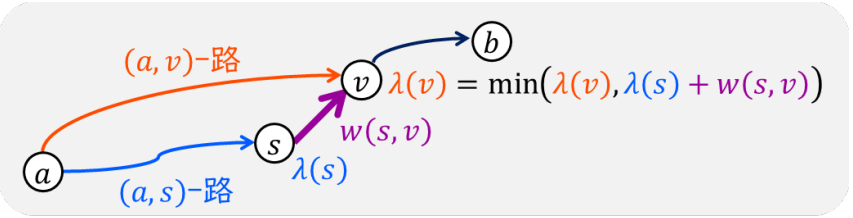
Step 3: If $s=b$, then output $(a,\ldots,p(p(b)),p(b),b)$, and halt

Step 4: Set $\lambda(v)=\min\{\lambda(v),\lambda(s)+w(s,v)\}$ for every $v\in Y$.
Set $p(v)=s$ if the second term is smaller

Step 5: Select s such that $\lambda(s)=\min\{\lambda(v)|v\in Y\}$,
and return to Step 2

最短路アルゴリズム

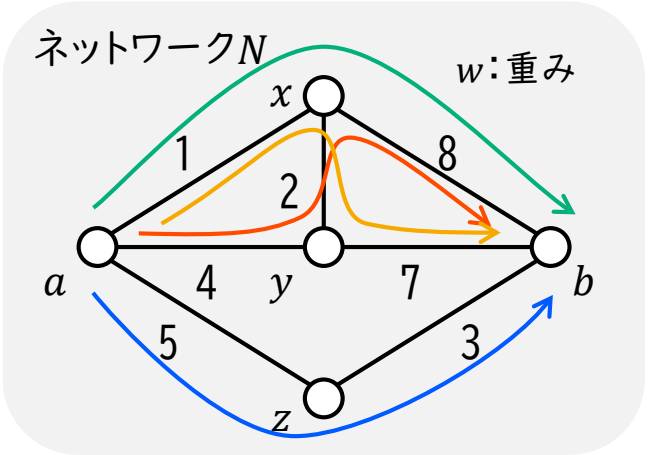
- 最短路問題
  - 入力
    - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$
    - 始点a $(\in V)$，終点b $(\in V)$
  - 質問：ネットワーク$N = (G, w)$の最短(a, b)-路を一つ示せ
- 最短路アルゴリズム ： 辺評価の繰り返しが主流
  - Dijkstra
    - 始点に近い点に接続する辺から（各辺高々1回）
  - BFS（迷路法，Wavefront），DFS（右手法）
    - 辺を次々に処理．平面や空間に対応するなど単一辺重みの場合
  - Bellman-Ford
    - 辺の端点のラベルが更新されたら
  - A^*アルゴリズム
    - 終点までの予測道程が小さい辺優先（平面や空間に対応する場合など）
  - ネットワークフロー利用
  - ...



3-2（2）最長路問題

■ 最長路問題（shortest-path problem）

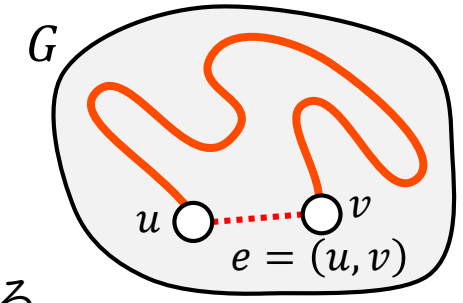
- 入力
  - 連結グラフ$G = (V, E)$，重み関数$w: E \to \mathbb{R}^+$　← 重みは非負
  - 始点a $(\in V)$，終点b $(\in V)$
- 質問：ネットワーク$N = (G, w)$の最長(a, b)-路を一つ示せ



(a,b)-路	重み	
→	9	(=1+8)
→	10	(=1+2+7)
→	14	(=4+2+8)
→	8	(=5+3)

最長路問題とハミルトングラフ判定問題

■ ハミルトングラフ判定問題



- グラフGがハミルトングラフ
  - ハミルトン閉路$C \subseteq E(G)$あり
    - $\forall e \in C$, eの両端点を結ぶハミルトン路が$G - \{e\}$にある
- グラフGがハミルトングラフであるための必要十分条件
  - $\exists e \in E(G)$, eの両端点を結ぶハミルトン路が$G - \{e\}$にある

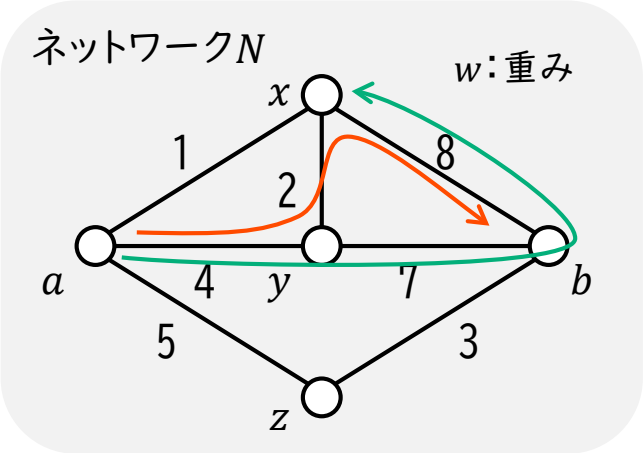
✓ 最長路問題を$E(G)$回用いて解くことができる

- $\forall e \in E(G)$, $G - \{e\}$でeの両端点間の最長路を求める
  - 得られた最長路の中に，辺数$|V(G)| - 1$の最長路（ハミルトン路）
    - あり ⇒ Gはハミルトングラフ
    - なし ⇒ Gはハミルトングラフでない

最長路問題の難しさ

■ 最適性の原理(principal of optimality)が不成立 ⇒ 難しい

✓ 最長路の部分路は最長路とは限らない



最長(a,b)−路 → 14 (=4+2+8)

最長(a,x)−路 → 19 (=4+7+8)

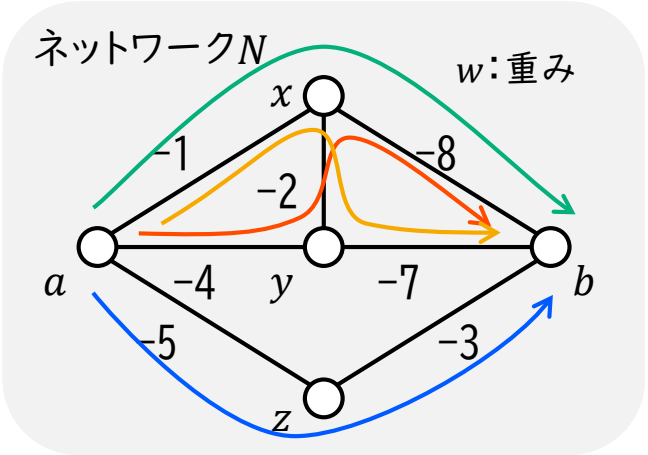
最長路問題の難しさ

■ 最適性の原理(principal of optimality)が不成立 ⇒ 難しい

✓ 最長路の部分路は最長路とは限らない

➢ 簡単な場合もあり

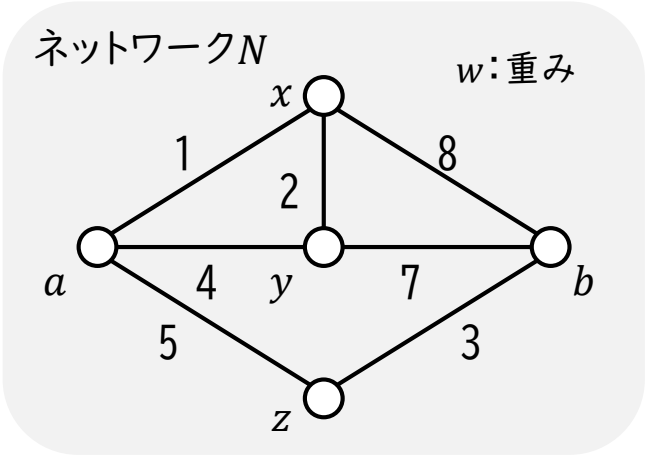
- 重みがすべて非正
  - 重みの正負反転で最短路問題に
- 木（路は唯一）
- 有向無閉路グラフ(directed acyclic graph, DAG)
  - 最適性の原理が成立



(a,b)−路	重み	
→	−9	(=−1−8)
→	−10	(=−1−2−7)
→	−14	(=−4−2−8) 最[redacted]短
→	−8	(=−5−3) 最[redacted]長

最短路問題と最長路問題

- 最短路問題 (shortest-path problem)
- 最長路問題 (longest-path problem)
  - グラフや辺重みの特徴，要求に応じて様々なアルゴリズムあり



辺重み	最短路	最長路
正（非負）	易	難
正負	難	難
負（非正）	難	易