

# データ構造とアルゴリズム

## 第11週

掛下 哲郎

kake@is.saga-u.ac.jp

# 前回のまとめ

## グラフアルゴリズム

- グラフの定義
  - 無向グラフ, 有向グラフ, 重み付きグラフ
- グラフの表現
  - 隣接行列, 隣接リスト
- グラフの探索
  - 幅優先探索 (BFS), 深さ優先探索 (DFS)
- 探索の応用
  - 探索木
  - 前置記法, 中置記法, 後置記法

# 深さ優先探索 vs 幅優先探索

- 幅優先探索
  - 完全性: 解が存在するならば, 必ず発見できる.
  - 最適性: 長さが最も短い経路を返す
  - 規模の大きな探索においては効率が悪い
- 深さ優先探索
  - 完全性や最適性は保証されない
  - 平均的なケースでは, 幅優先探索と比較して記憶領域の使用量が少ない
  - 再帰アルゴリズムにより, 簡単に記述できる
  - 後置記法を求める場合にも使用

# 講義スケジュール

週	講義計画
1-2	導入
3	探索問題
4-5	基本的なデータ構造
6	動的探索問題とデータ構造
7	アルゴリズム演習(第1回)
8-9	データの整列
10-11	グラフアルゴリズム
12	文字列照合のアルゴリズム
13	アルゴリズム演習(第2回)
14	アルゴリズムの設計手法
15	計算困難な問題への対応



# 今日の内容

- 探索問題(続き)
  - Minimax法
- 最短経路問題(shortest path problem)
- ネットワークフロー(network flow)

# コンピュータ対戦型リバーシ

The screenshot shows a Reversi game window titled "Reversi". On the left is an 8x8 green board with a 2x2 cluster of pieces in the center: a white piece at (4,4), a black piece at (4,5), a black piece at (5,4), and a white piece at (5,5). On the right is a settings panel titled "コンピュータの設定". The panel includes a "レベル" dropdown set to "01", a "評価方法" dropdown set to "コマ数", and a "手番" section with radio buttons for "白" (selected) and "黒". Below these are checkboxes for "コマ数評価への切り替え" (set to 0) and "手前に切り替える". A "初期化" button is also present. At the bottom of the panel are two input fields for "白" and "黒" (both set to 2), a checkbox for "待った" (checked), and a "2手前に戻る" button. Annotations with arrows point to various elements: "コンピュータの手番を設定" points to the "手番" radio buttons; "何手先まで読むかを設定" points to the "レベル" dropdown; "局面の評価方法を設定" points to the "評価方法" dropdown; "途中で評価方法を「コマ数評価」へ切り替える場合は、最後から何手前に切り替えるかを入力" points to the "コマ数評価への切り替え" input field; "盤面、設定をリセット" points to the "初期化" button; "局面に応じてメッセージを表示" points to the message area below the settings; "各プレイヤーのコマ数" points to the "白" and "黒" input fields; "現在の手番を示す" points to the "待った" checkbox; "2手前に戻る" points to the "2手前に戻る" button; and "コマを打つ" points to the game board.

コンピュータの手番を設定

何手先まで読むかを設定

局面の評価方法を設定

途中で評価方法を「コマ数評価」へ切り替える場合は、最後から何手前に切り替えるかを入力

盤面、設定をリセット

局面に応じてメッセージを表示

各プレイヤーのコマ数

現在の手番を示す

2手前に戻る

コマを打つ

# 局面の評価方法

コマ数による評価	コマの位置による 重み付け評価	着手可能数による評価
自分のコマ数から敵のコマ数を引いた値を評価値とする。	盤面の各マスに評価値を設け、自分のコマがあればその評価値を加算し、敵のコマがあれば減算します。そしてその合計値を評価値とする。	自分のコマが置けるマスの数を評価値とする。

同じ局面でも、評価方法を変えると、評価値は大きく変化

30	-12	0	-1	-1	0	-12	30
-12	-15	-3	-3	-3	-3	-15	-12
0	-3	0	-1	-1	0	-3	1
-1	-3	-1	-1	-1	-1	-3	-1
-1	-3	-1	-1	-1	-1	-3	-1
0	-3	0	-1	-1	0	-3	1
-12	-15	-3	-3	-3	-3	-15	-12
30	-12	0	-1	-1	0	-12	30

# Minimax法による最善手の探索

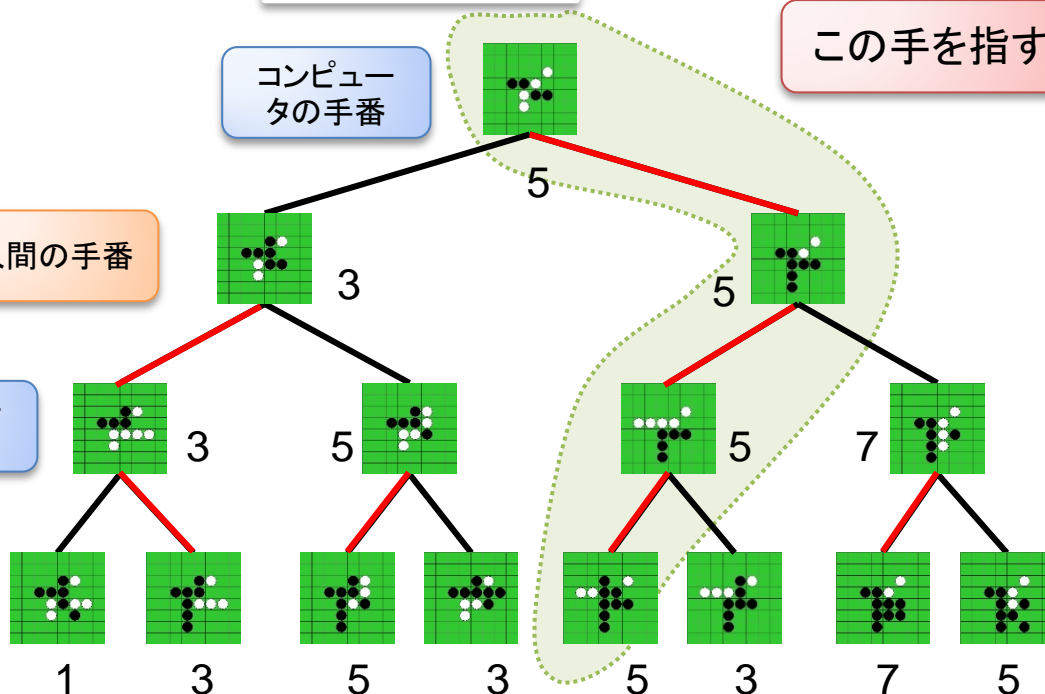
現在の盤面  
さてどこに打つ？

コンピュー  
タの手番

この手を指す

人間の手番

コンピュー  
タの手番



4. 自分にとって一番有利な  
手を採用！

⇒ 評価値が最大の手

3. 相手は自分にとって一番  
不利な手を採用

⇒ 評価値が最小の手

2. 自分にとって一番有利  
な手を採用

⇒ 評価値が最大の手

1. 3手先の評価値を求める

ミニマックス法 (minimax)

- 自分の手番の時は最大値を採用
- 相手の手番の時は最小値を採用

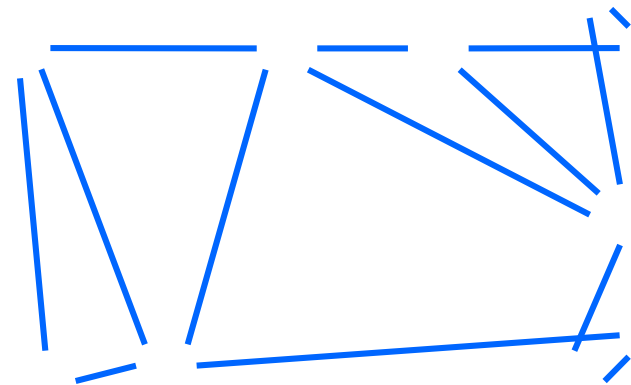
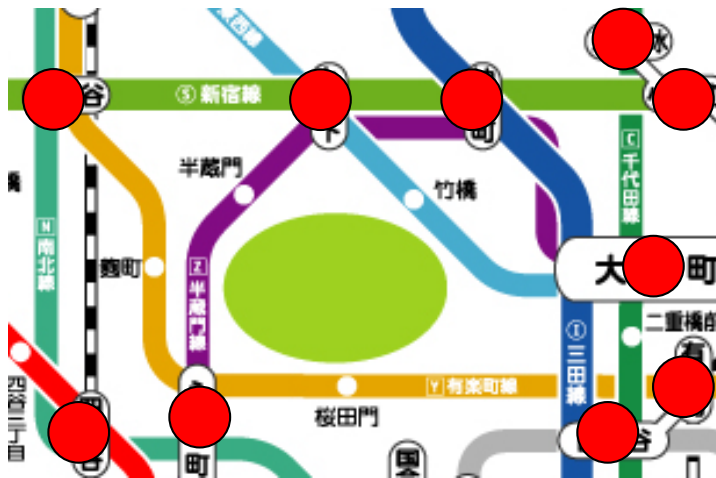
コンピュー  
タの読み筋

要書き  
込み



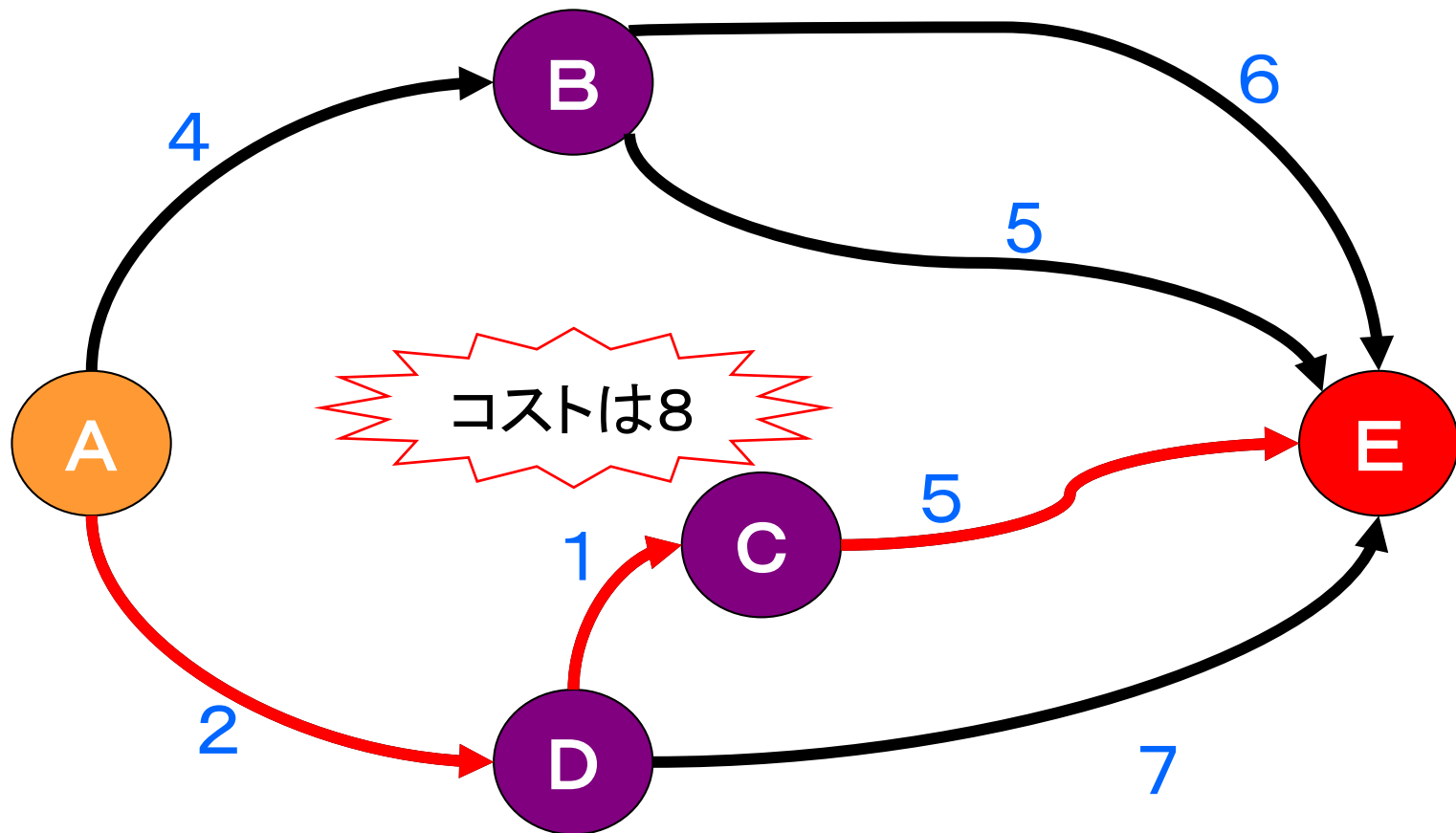
# 最短経路問題

- 最もコスト(運賃, 時間, 距離など)の少ないルートを見つける



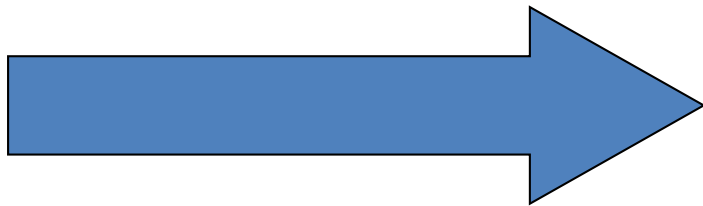
# 重み付き有向グラフで考える

Q1: 都市Aから都市Eへの, 最短経路は?



# グラフが複雑になると...

- 目で見ても分からない
- 全てのルートを調べると、非常に時間がかかる



Dijkstraの方法

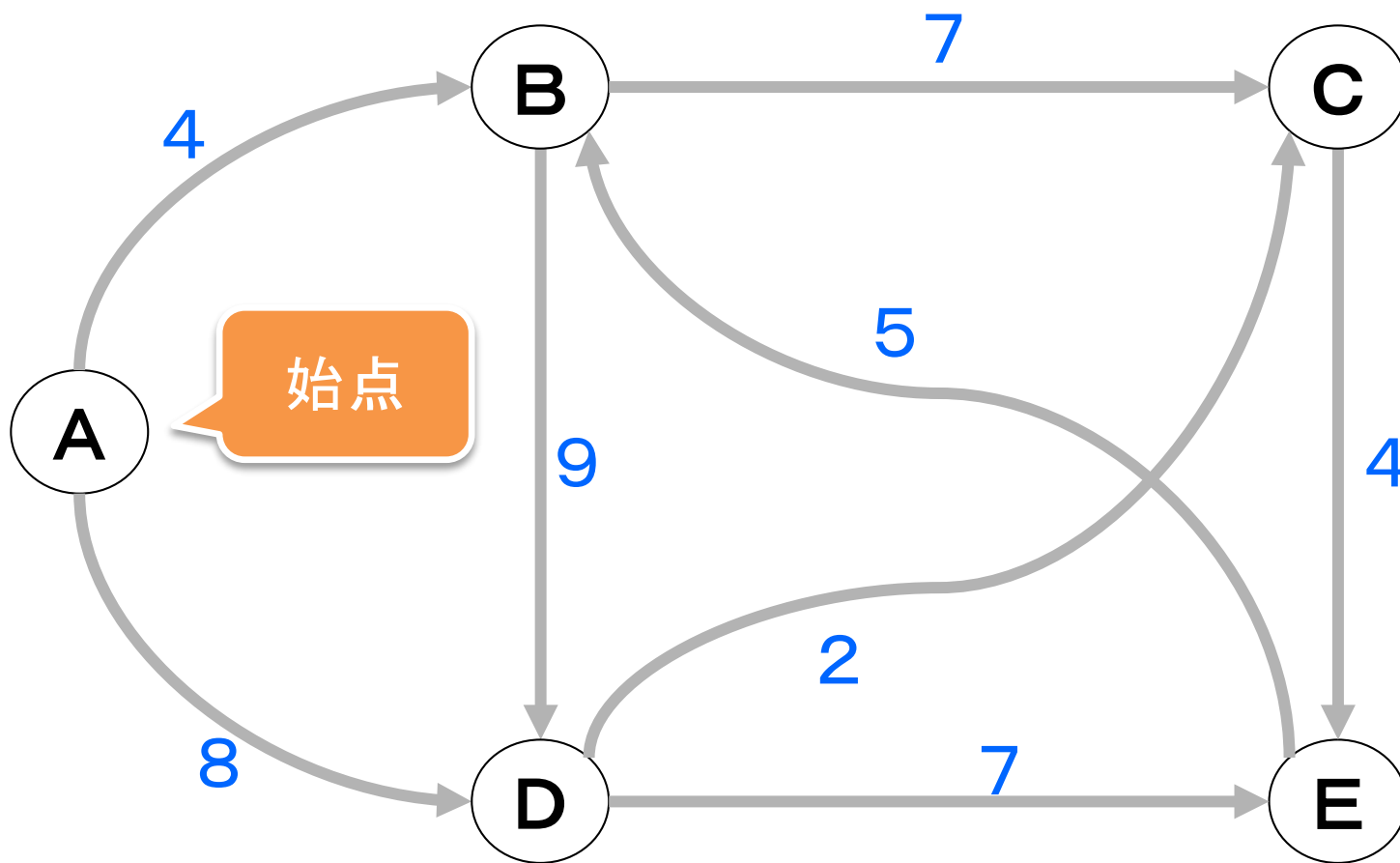


# 最短経路問題

- 入力: 重み付き有向グラフ  $G=(V, E, c)$ 
  - 頂点数 $n$ , 辺数 $m$
- 出力: 2つの頂点間の最小距離
- いくつかのバリエーション
  - 「ある1頂点」から「他の1頂点」への距離
  - 「ある1頂点」から「他の全ての頂点」への距離
  - 「全ての頂点」から「全ての頂点」への距離

# ダイクストラ法

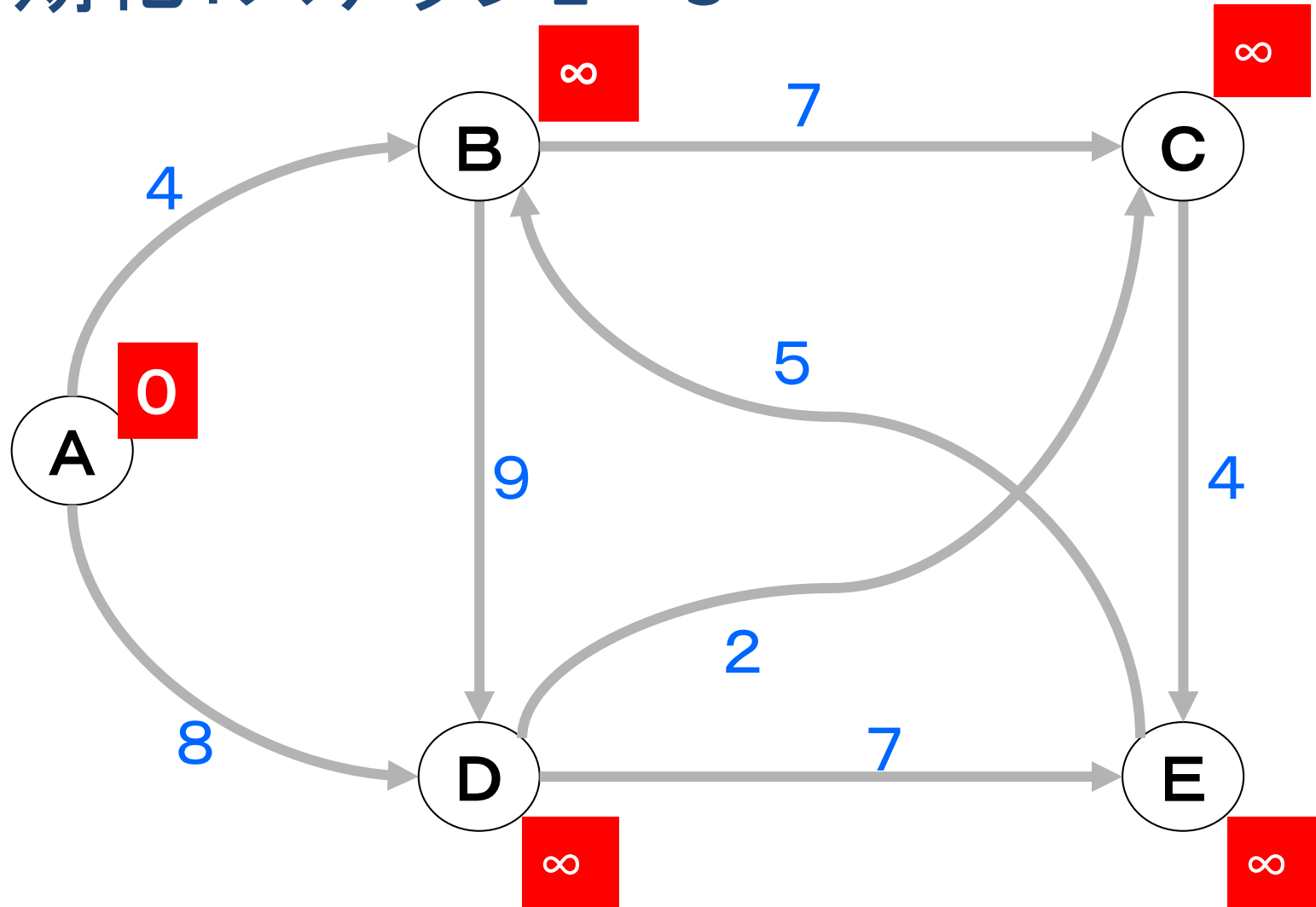
始点から、調べる範囲を  
ジワジワと広げながら距  
離を求めていく



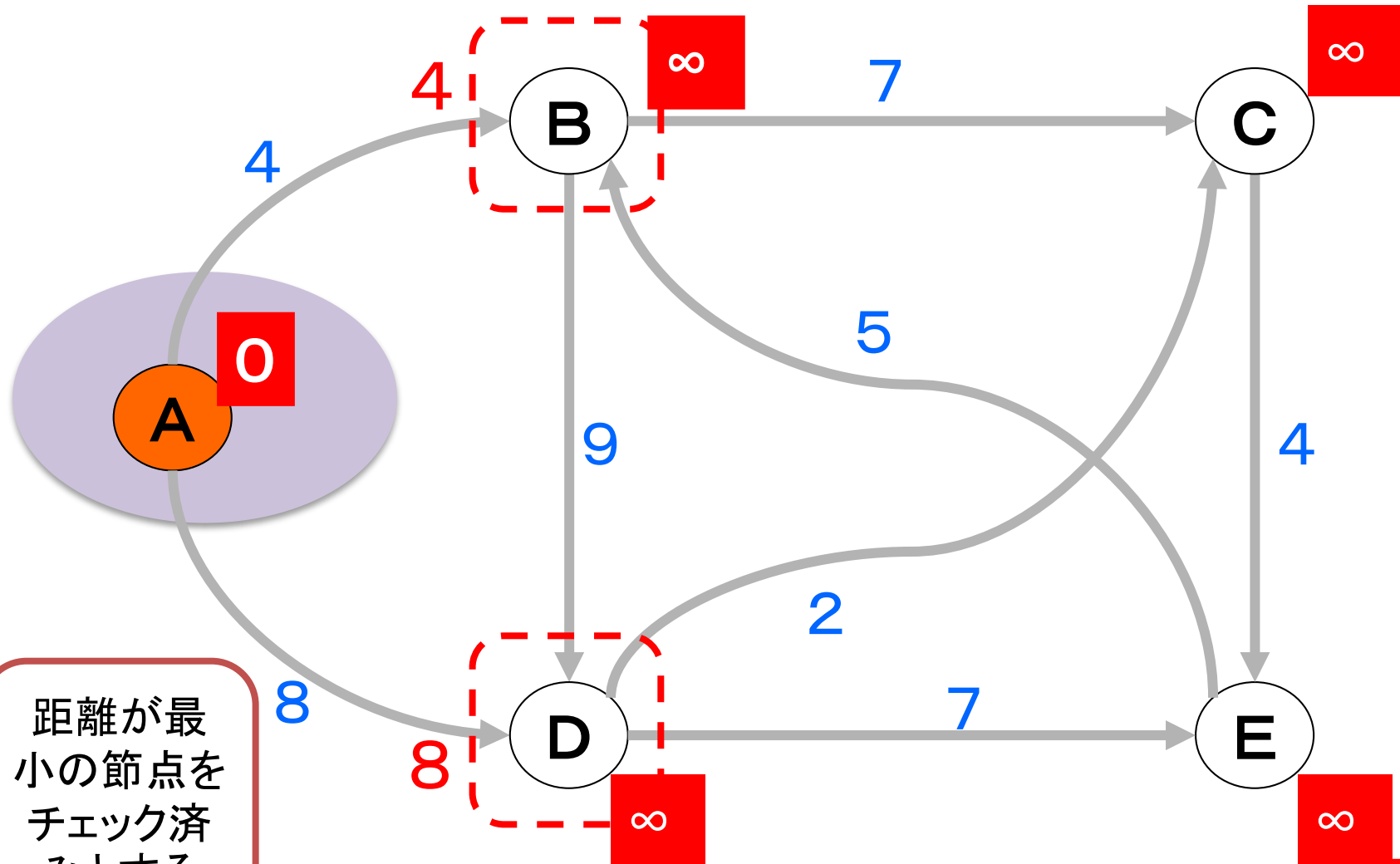
# ダイクストラの最短経路アルゴリズム

1.  $G$ の各頂点 $v$ について,  $v$ が保持する距離を $\infty$ とする.
2. 始点が保持する距離を0とする.
3. チェック済みの頂点集合 $X$ を空とする.
4. すべての頂点がチェック済みになるまで, 以下の処理を繰り返す.
  - 4-1 チェック済みでない頂点の中で, **保持する距離が最小の頂点**を $u$ とする.
  - 4-2 頂点 $u$ をチェック済みとし,  $X$ に追加する.
  - 4-3 頂点 $u$ から未チェック頂点 $v$ に向かう各辺について,  **$u$ が保持する距離 + 辺の重みが,  $v$ が保持する距離よりも小さいならば,  $v$ が保持する距離を更新する.**

# 初期化: ステップ1~3



# ループ1回目 : Aをチェック済みとする

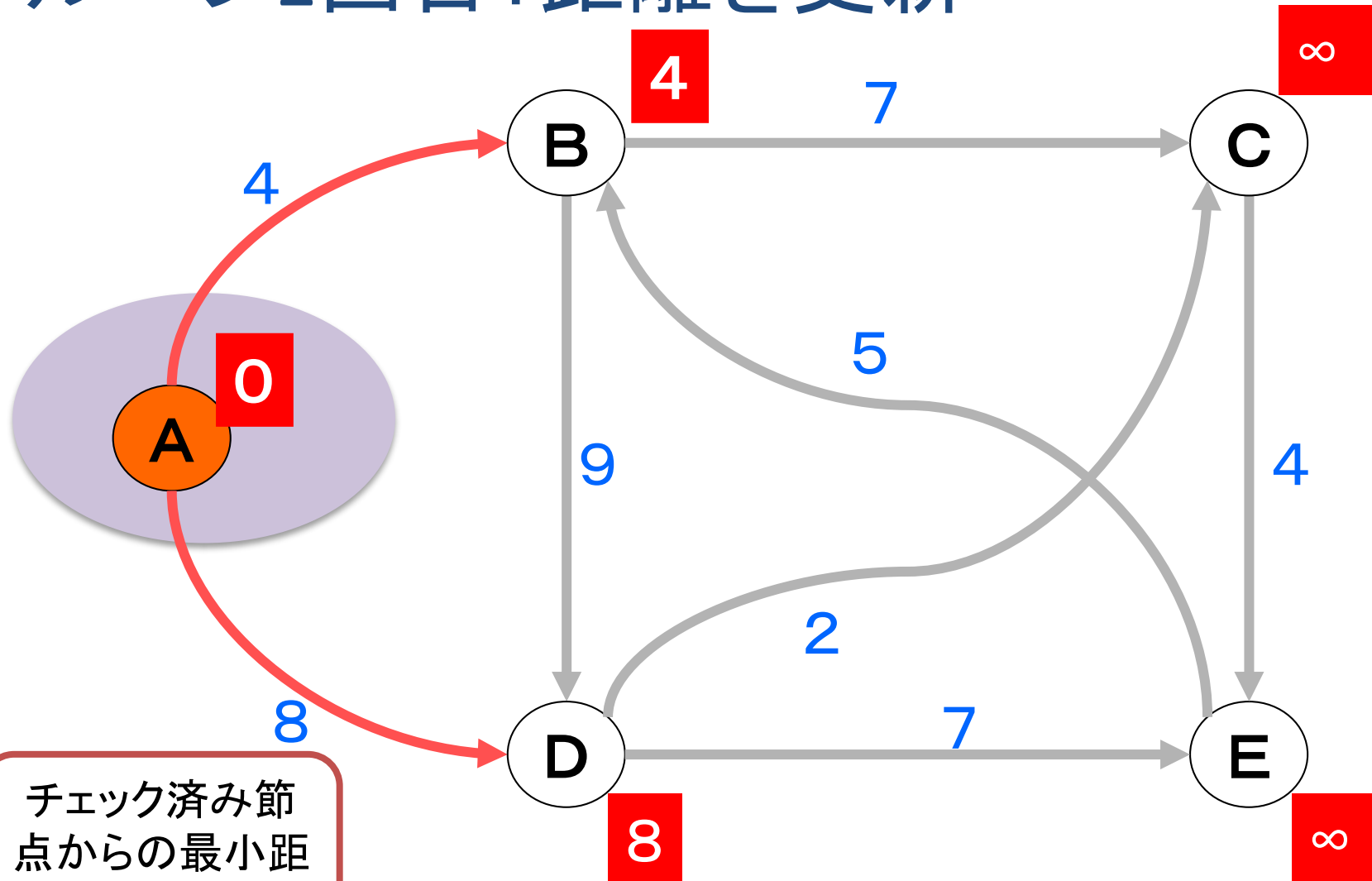


距離が最小の節点を  
チェック済みとする

要書き込み



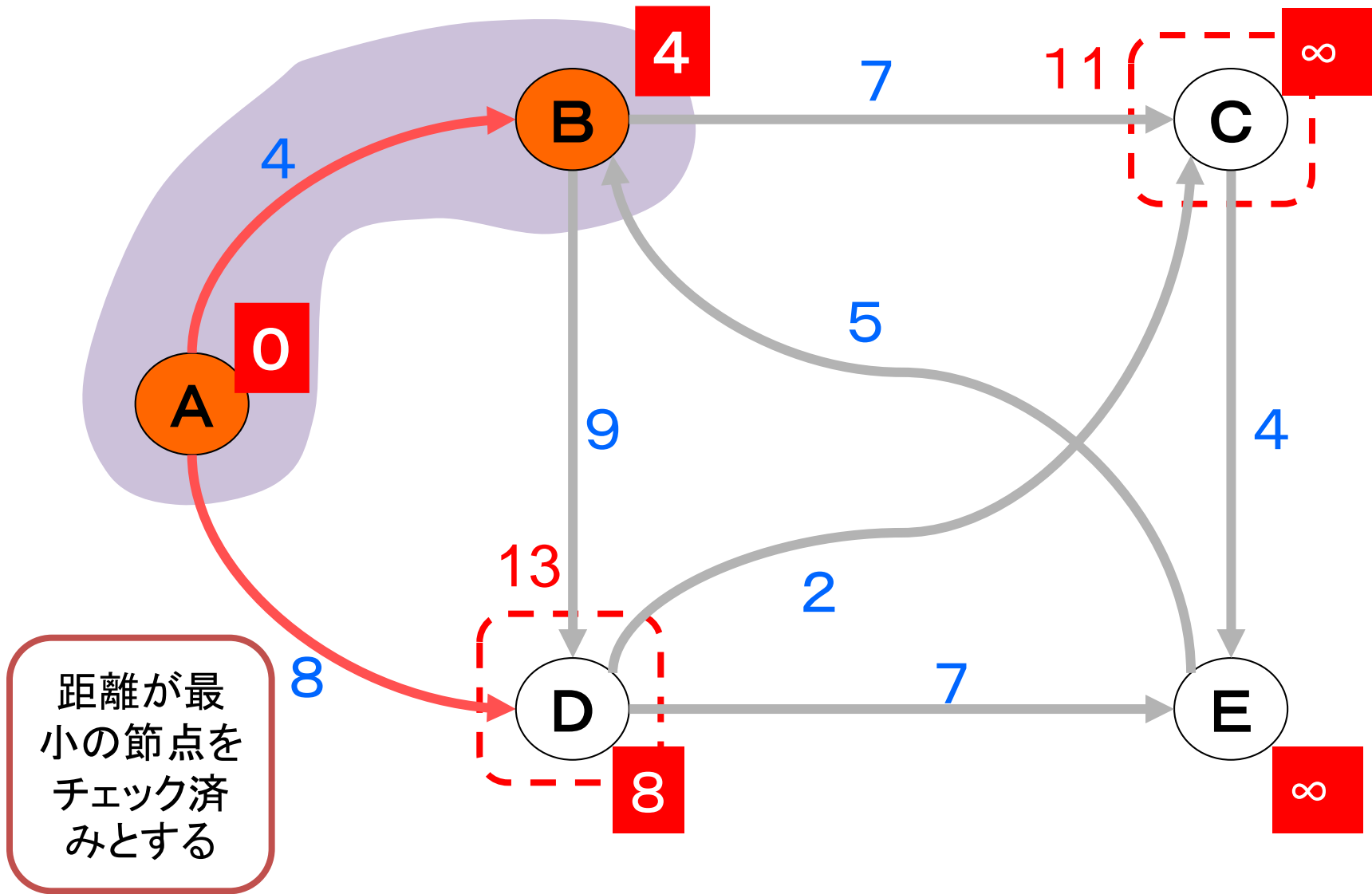
# ループ1回目：距離を更新



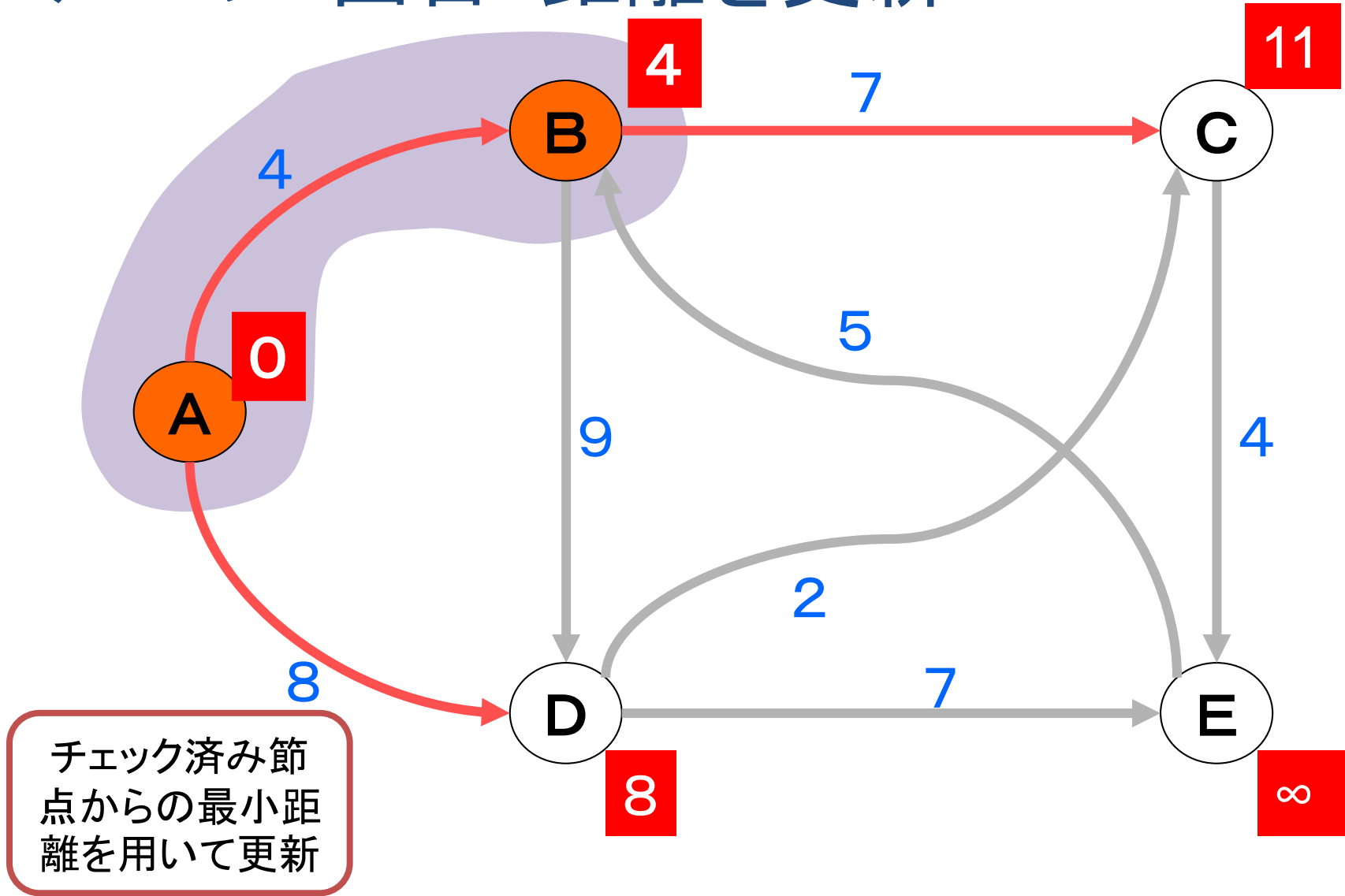
チェック済み節  
点からの最小距  
離を用いて更新

要書き  
込み

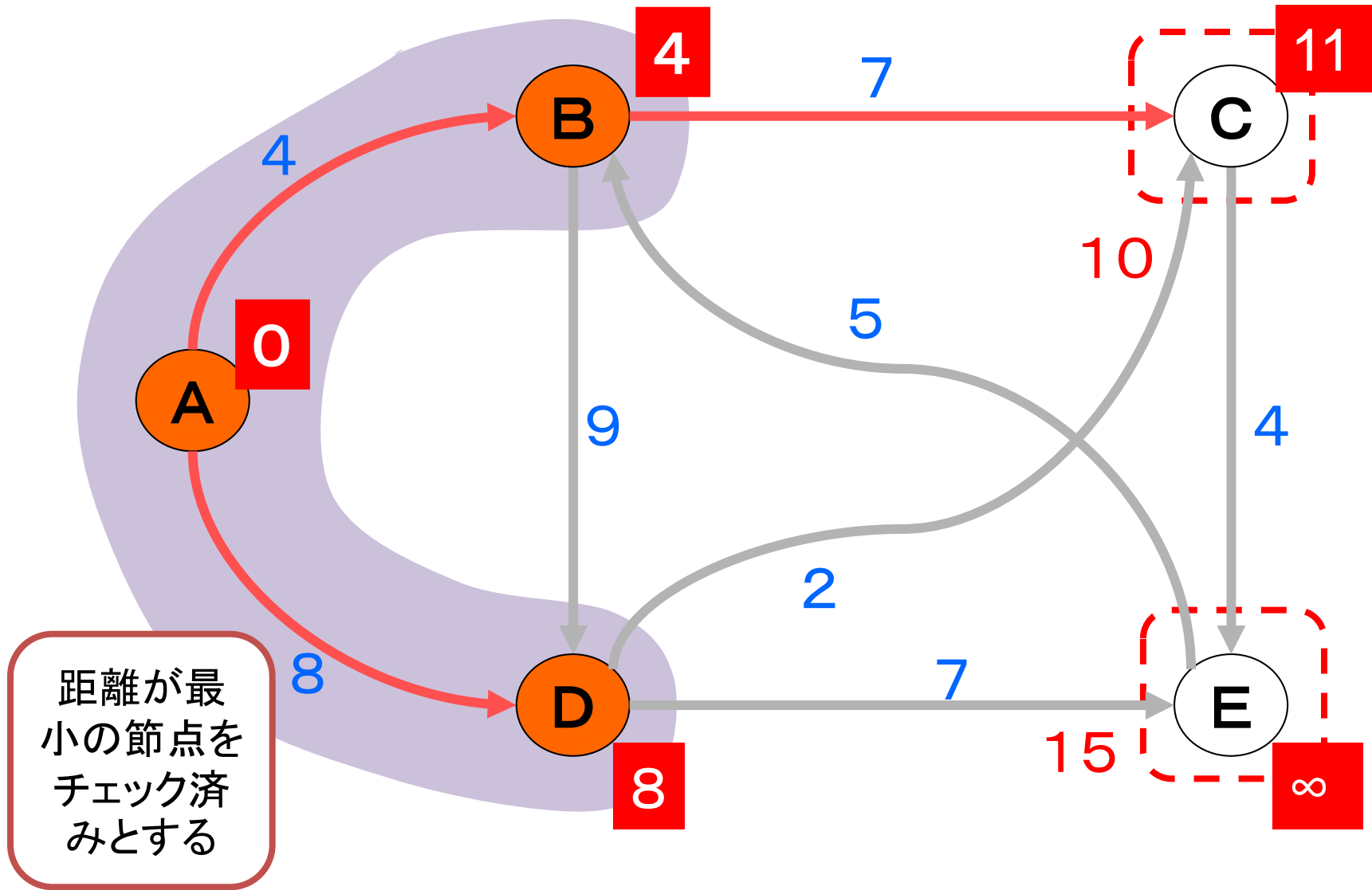
# ループ2回目 : Bをチェック済みとする



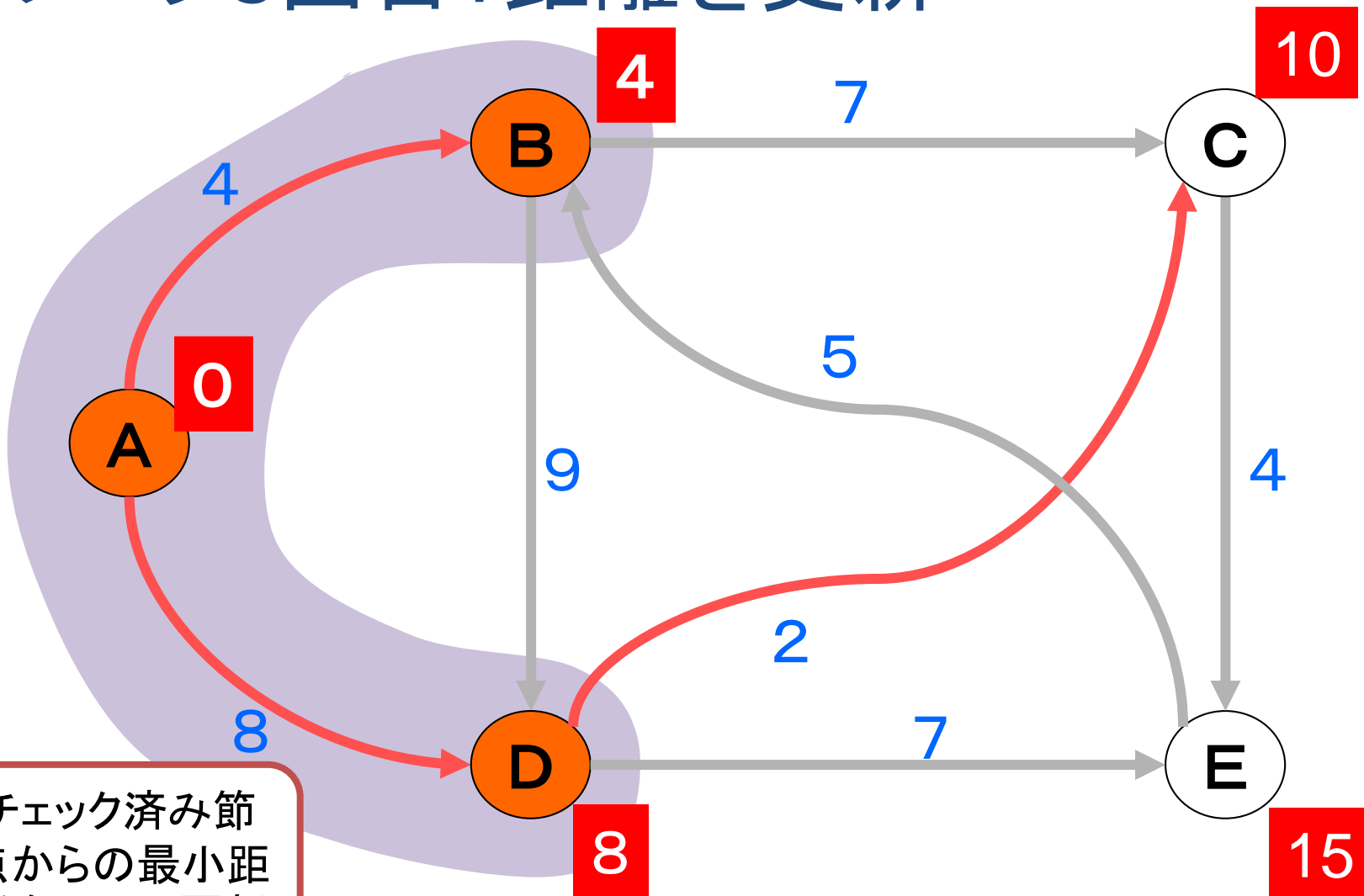
# ループ2回目：距離を更新



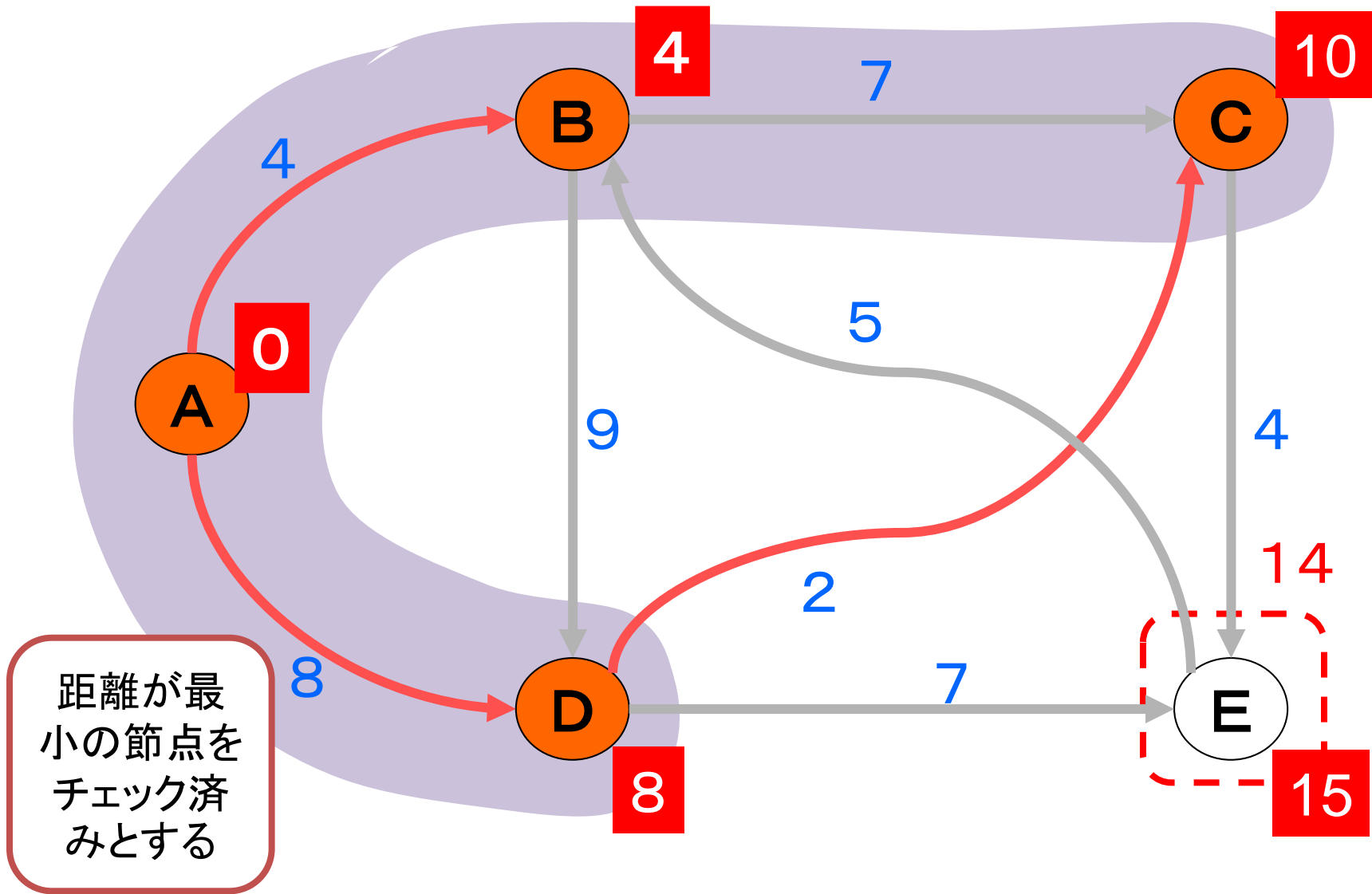
# ループ3回目 : Dをチェック済みとする



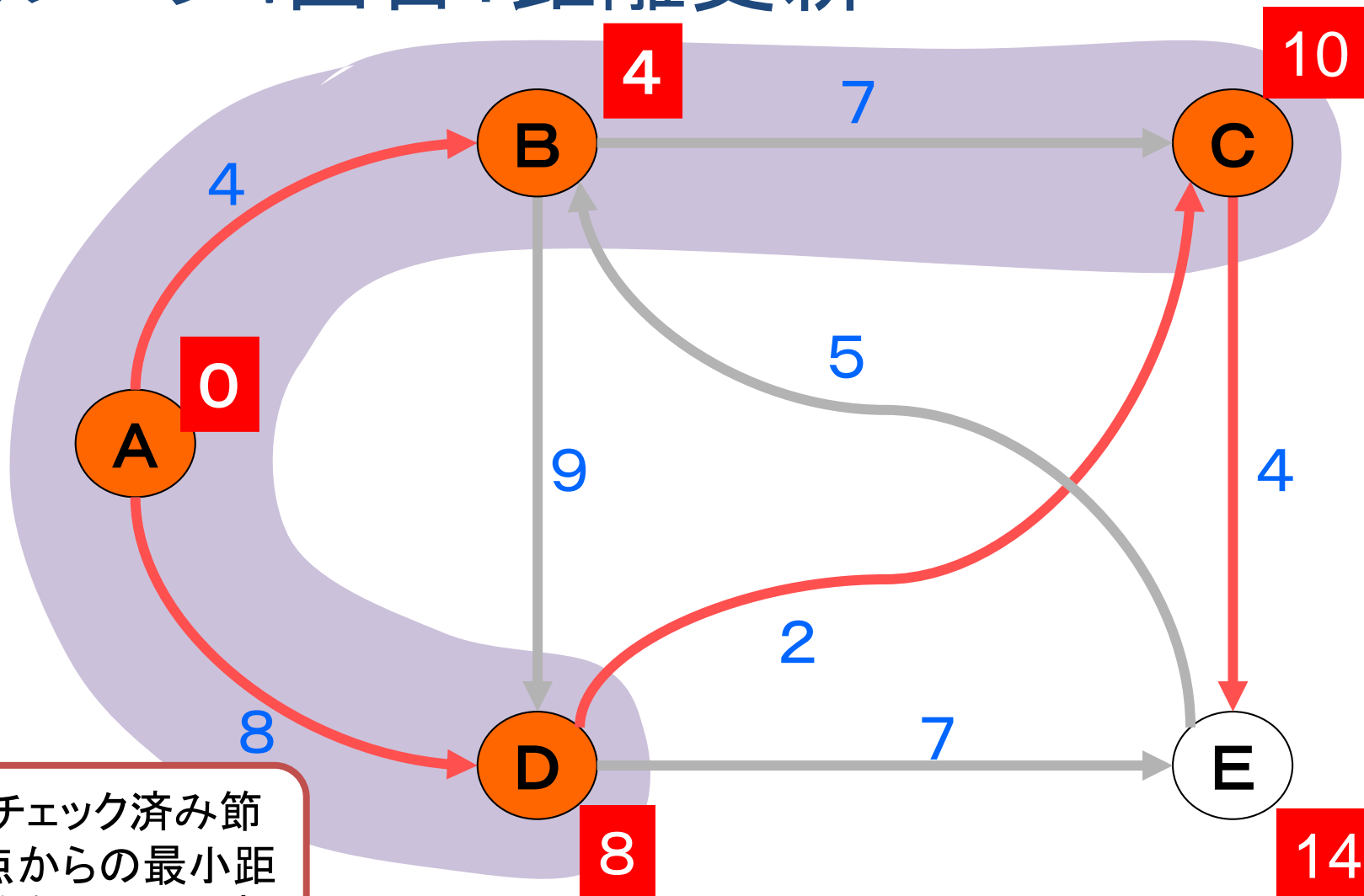
## ループ3回目：距離を更新



# ループ4回目 : Cをチェック済みとする

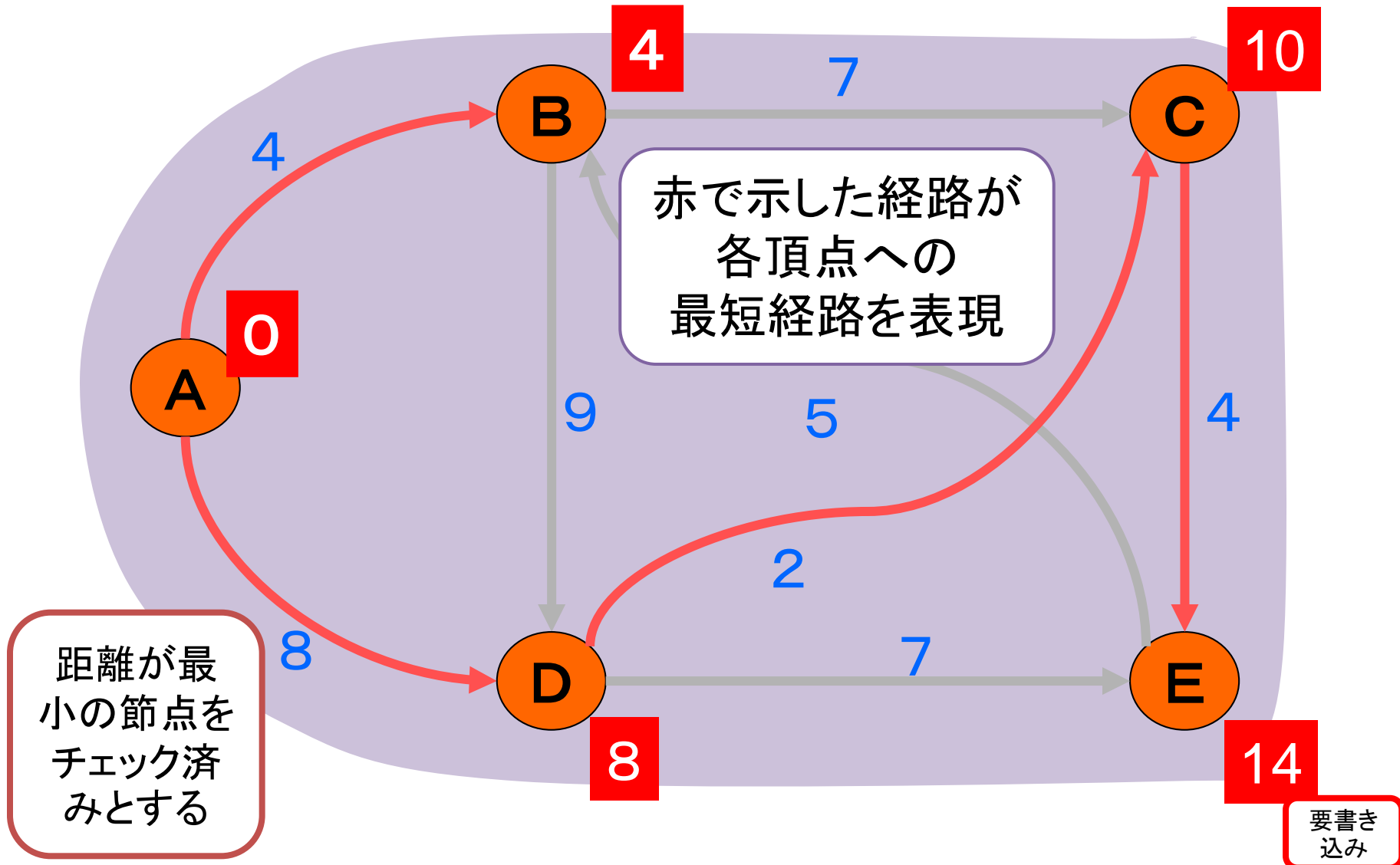


# ループ4回目: 距離更新



チェック済み節  
点からの最小距  
離を用いて更新

# ループ5回目:Eをチェック済みとする



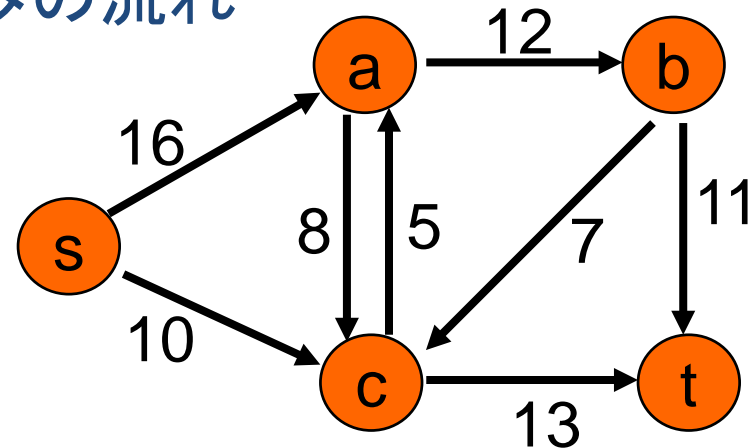


# 整理

- ダイクストラの最短経路アルゴリズム
    - 命題
      - ・ チェック済み頂点は、始点からの最短距離を保持する.
    - 各頂点への最短経路も同時に計算できる.
    - 計算時間
      - ・ ステップ1, 3:  $O(n)$
      - ・ ステップ2:  $O(1)$
      - ・ ステップ4: 繰り返し回数  $n$  回
      - ・ ステップ4-1: 総実行回数  $O(n^2)$ ,  $O(m \log n)$
      - ・ ステップ4-3: 総実行回数  $O(m)$
    - 総計算時間  $O(n^2+m)$  または  $O(m \log n)$
- 隣接リストを用いた場合
- ヒープを用いた場合
- 用いるデータ構造に依存

# ネットワークフロー

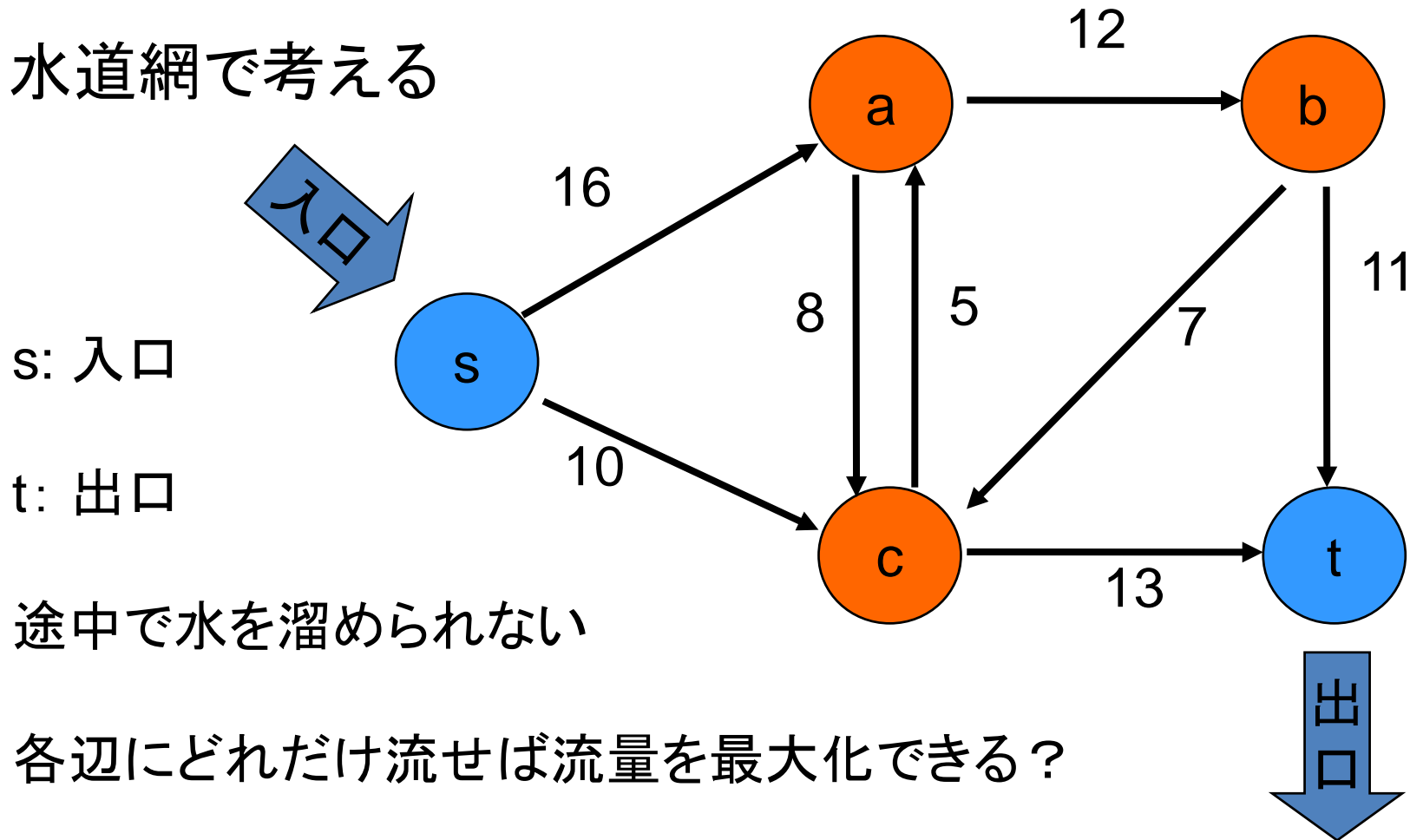
- ネットワーク上の流れ(flow)をモデル化
  - 水道網における実際の水の流れ
  - インターネットにおけるデータの流れ
  - 道路網における交通の流れ



- 最大流問題: Maximum Flow Problem
  - s から t まで流すことができる流量の最大値と, その最大の流量を与える流し方 f を求める
  - s: ソース, t: シンク

# ネットワークフロー

- 水道網で考える



# フロー $f$ が満たす性質

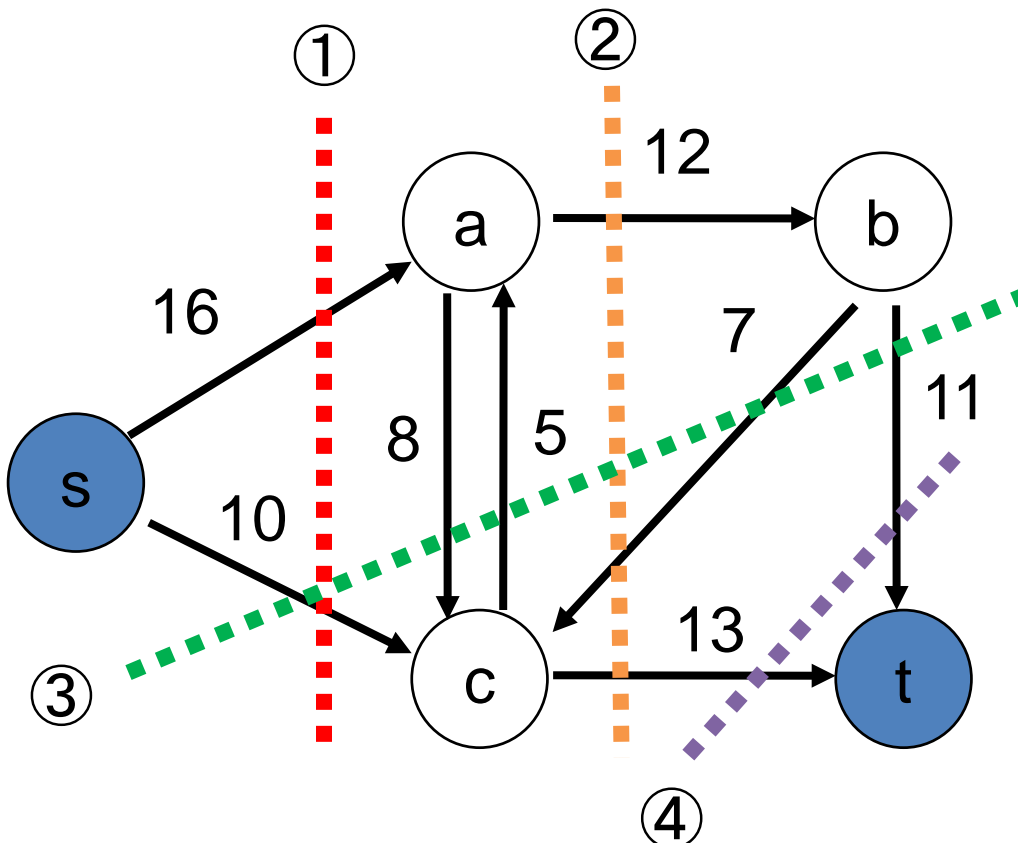
- $f(u, v)$ : 辺  $(u, v)$  に流す水の量(フロー)
- 歪(ひずみ)対称性: 任意の頂点  $u, v$  に対し、  
 $f(u, v) = -f(v, u)$   
※ 逆方向のフローはマイナスとする
- フロー保存則:  $s, t$  以外の任意の頂点  $u$  に対し、  
 $\sum_{v \in V} f(u, v) = 0$   
※  $u$  に流れ込む量 =  $u$  から出る量
- 容量制約: 任意の頂点  $u, v$  に対し、  
 $f(u, v) \leq c(u, v)$   
※ 各辺の容量  $c(u, v)$  を超えない

# カット(cut)

- s,t-カット: sとtとを分離する線

## カットの容量

カット上をs側から  
t側に流せる流量  
の最大値



- ①  $16 + 10 = 26$
- ②  $12 + 13 = 25$
- ③  $10 + 8 + 7 + 11 = 36$
- ④  $11 + 13 = 24$

# 最大流量-最小カット定理

- 任意のカット $C$ 、任意のフロー $f$ に対し、

$$f \text{ の流量 } \leq \text{ カット } C \text{ の容量}$$

が成立。また、等号が成り立つようなカットとフローが存在する。

- 「最大流量 = カットの容量の最小値」がいえる

# 最大流アルゴリズム

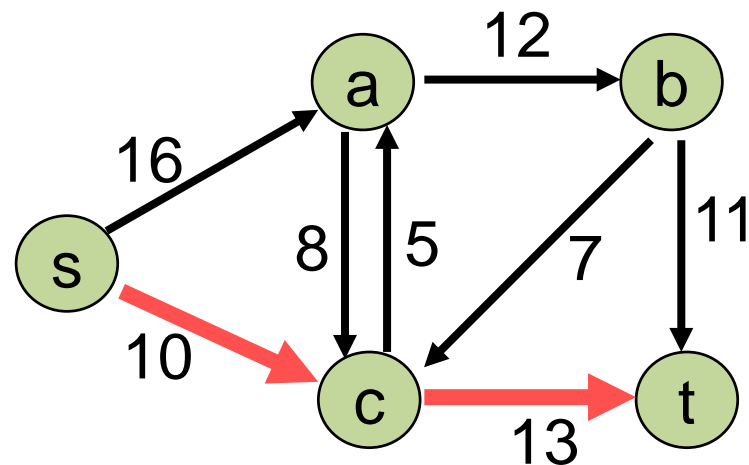
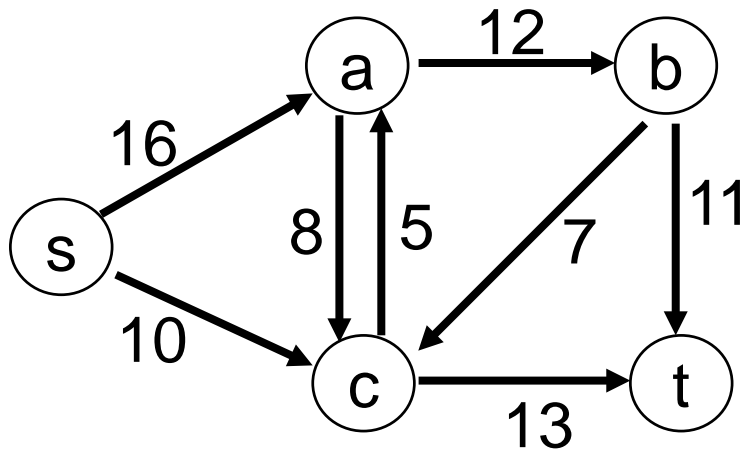
- 適当なフローからスタートし、「追加できる」流量を順次追加していくやり方
- 以下を計算しながら、フロー  $f$  を順次更新
  - **残余容量**  $r(u,v) = c(u,v) - f(u,v)$   
現時点でのフローと、容量との差。この分だけまだ余裕がある。
  - **残余グラフ**: 残余容量を重みにしたグラフ
  - **拡張可能経路**: 残余グラフにおける  $s$  から  $t$  への経路

# 最大流アルゴリズム

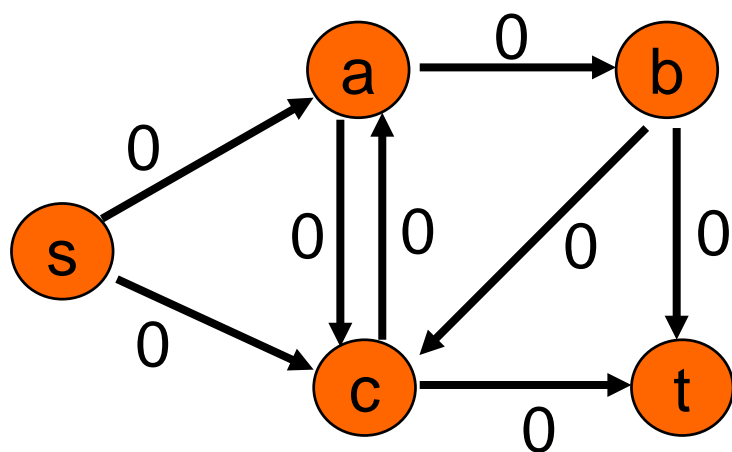
1.  $f$ をゼロフローとする.
2.  $f$ に対する残余グラフを $G_f$ とする.
3.  $G_f$ に拡張可能経路がある限り, 以下の処理を繰り返す.
  - 3-1  $G_f$ において拡張可能経路 $p$ を求める.
  - 3-2  $p$ 上の辺の最小残余容量の分だけフロー $f$ を増やす.
  - 3-3 残余グラフ $G_f$ を計算し直す.
4.  $f$ を最大フローとして出力する.



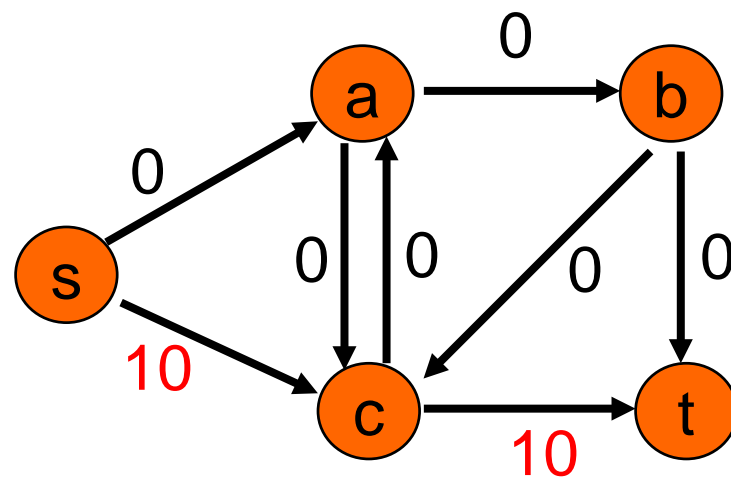
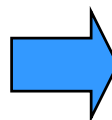
# ループ1回目



残余グラフと拡張可能経路

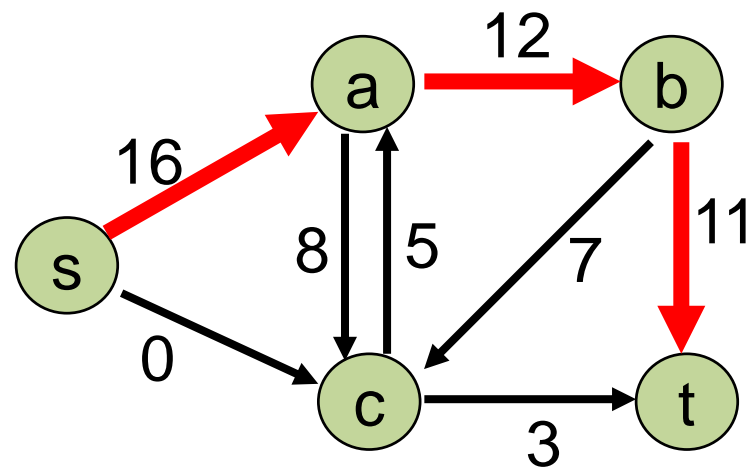
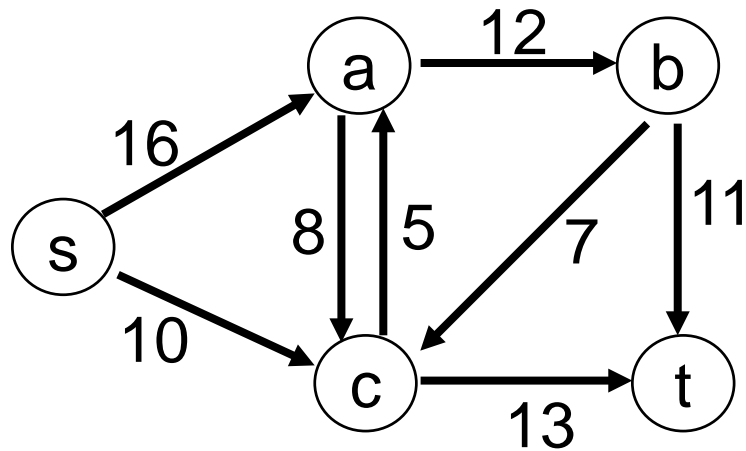


フロー $f$ (流量0)

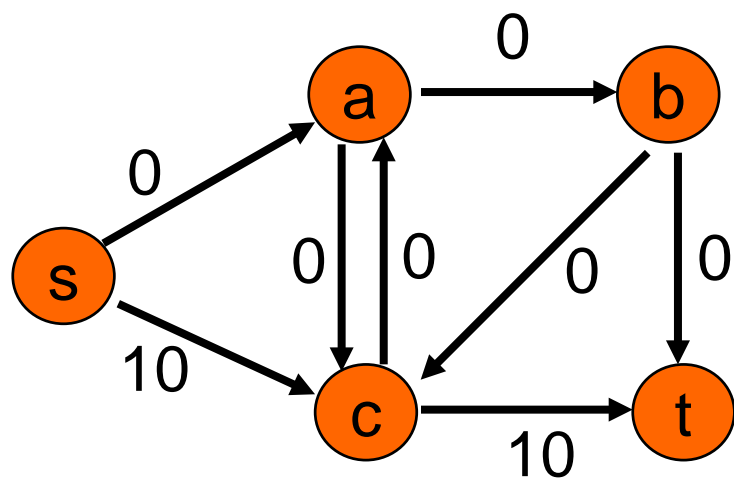


フロー $f$ (流量10)

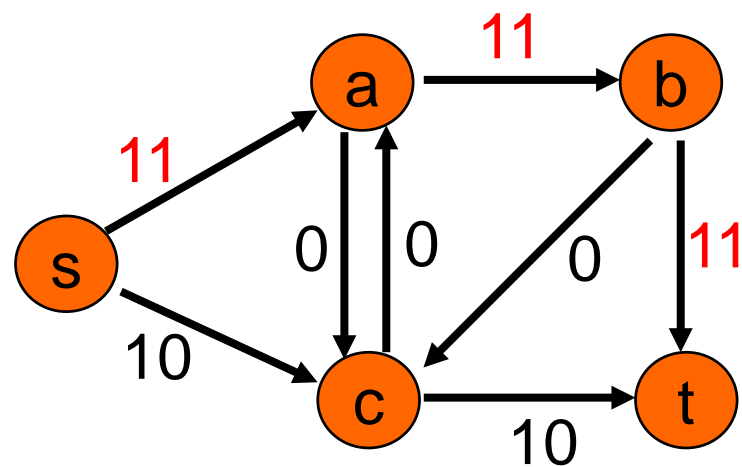
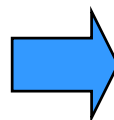
## ループ2回目



残余グラフと拡張可能経路

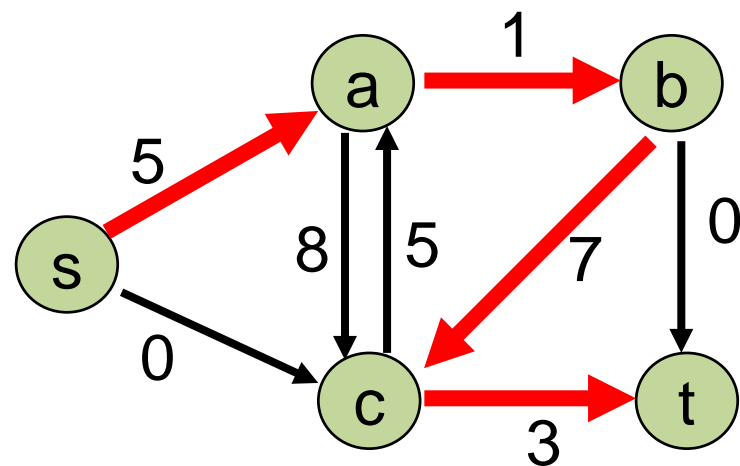
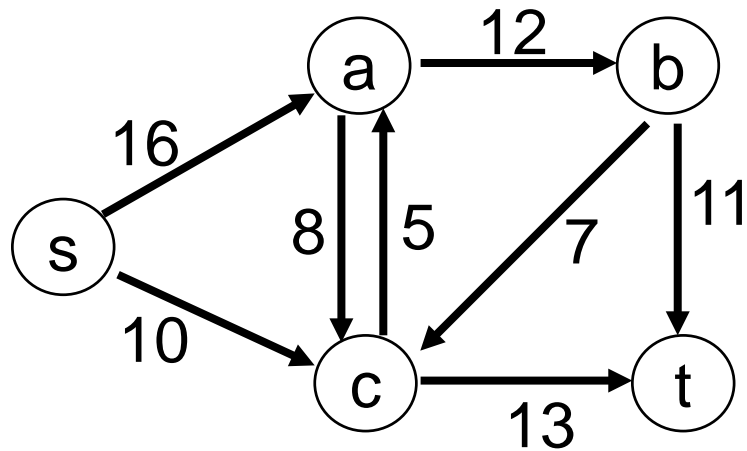


フローf(流量10)

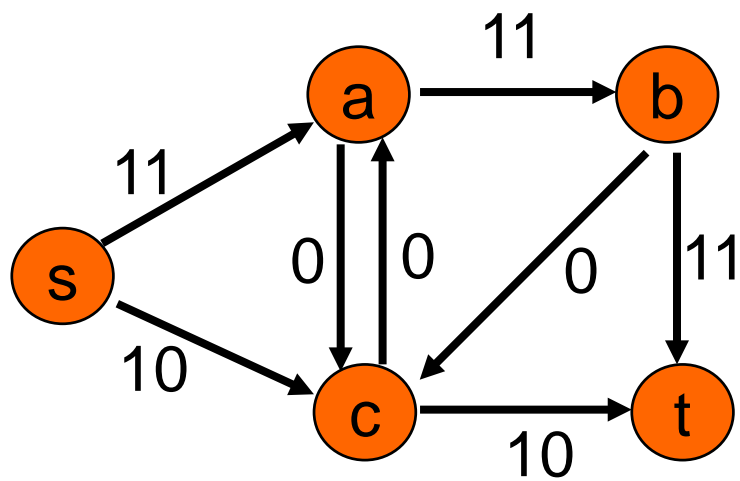


フローf'(流量21)

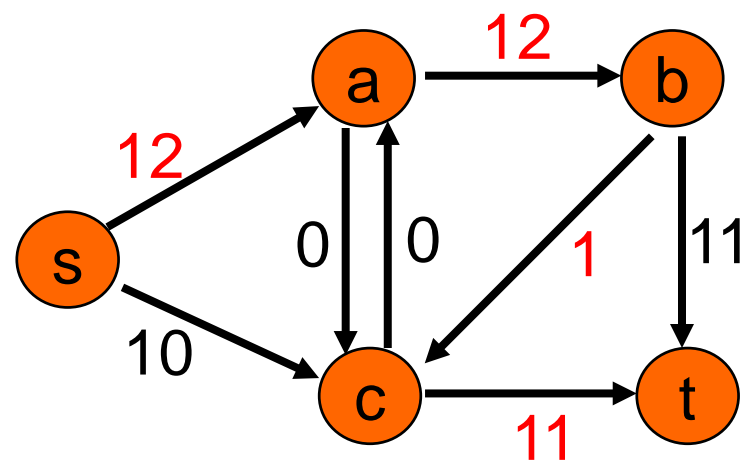
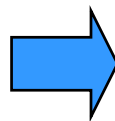
# ループ3回目



残余グラフと拡張可能経路

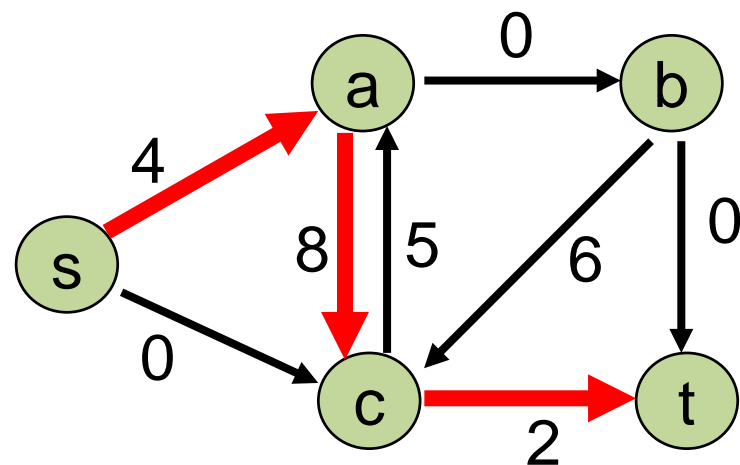
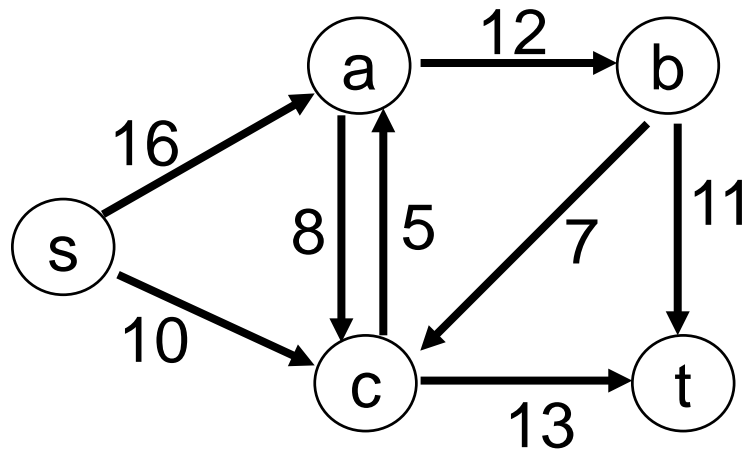


フロー $f$ (流量21)

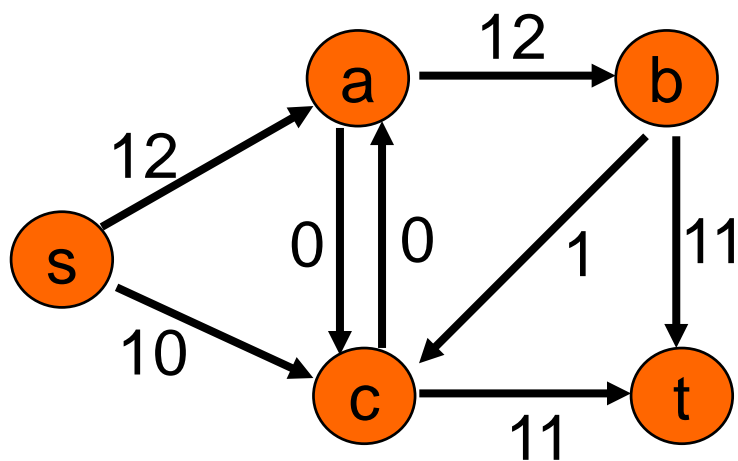


フロー $f'$ (流量22)

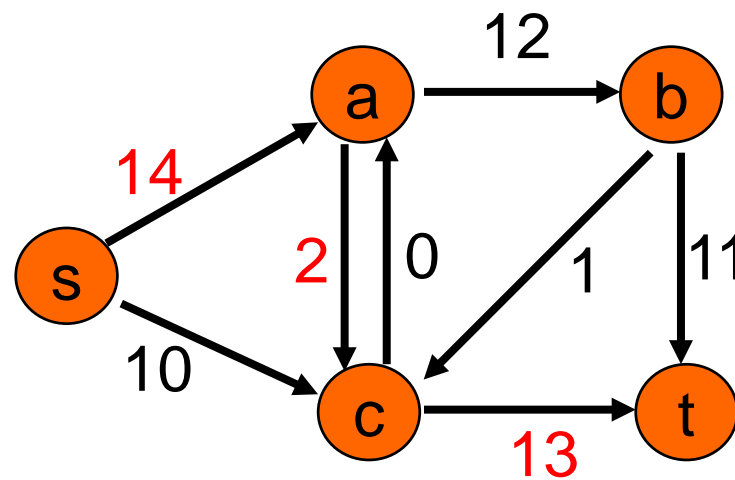
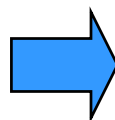
# ループ4回目



残余グラフと拡張可能経路

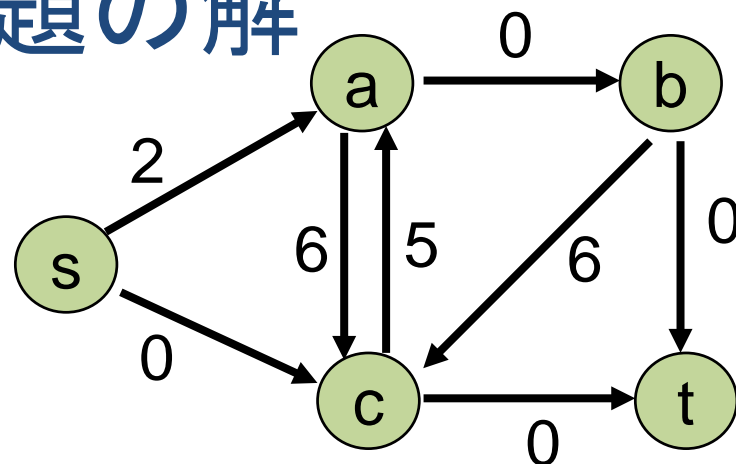
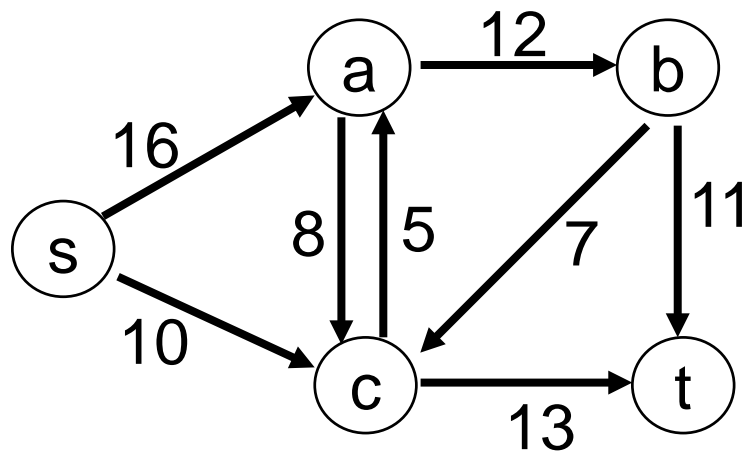


フロー $f$ (流量22)

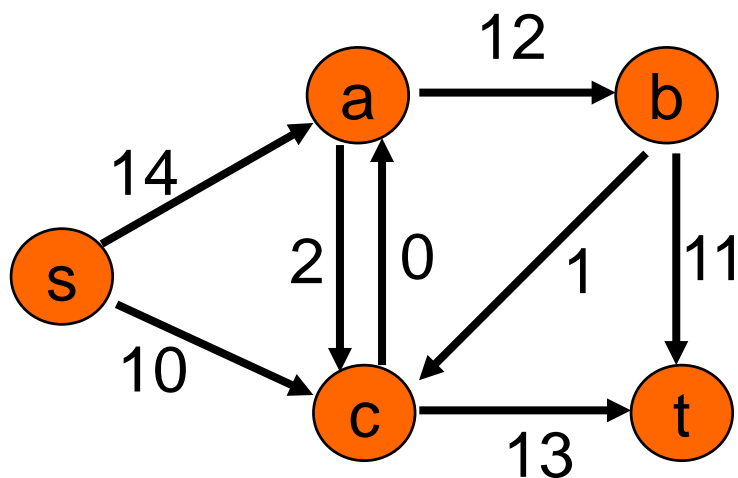


フロー $f$ (流量24)

# ネットワークフロー問題の解



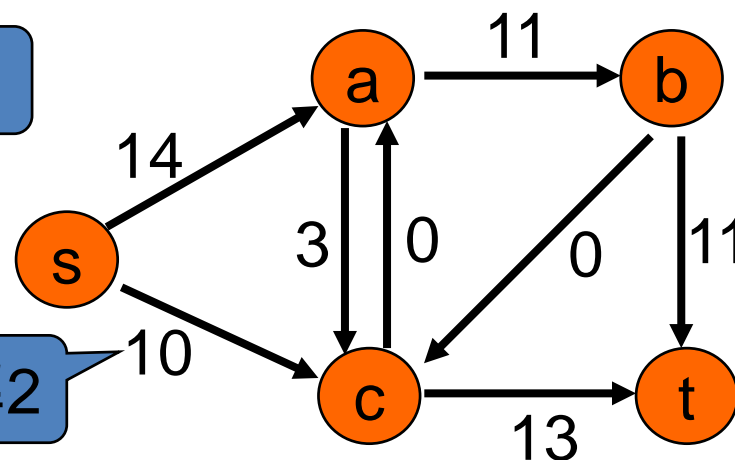
残余グラフと拡張可能経路



フロー $f$ (流量24)

解1

解2



フロー $f$ (流量24)

# まとめ

- ダイクストラ法による最短経路アルゴリズム
  - チェック済み頂点は、始点からの最短距離を保持する.
  - 各頂点への最短経路も同時に計算できる.
  - 計算時間  $O(n^2+m)$  または  $O(m \log n)$ 
    - 頂点数 $n$ , 辺数 $m$
- ネットワークフローの計算時間
  - ネットワークの容量が整数値の場合:  $O(m \times f_{\max})$ 
    - ネットワークの容量が整数値の場合
    - 最大流の値  $f_{\max}$
  - 一般の場合:  $O(n^3)$ ,  $O(nm^2)$

用いるデータ  
構造に依存

# 確認テスト(第11回)

- Minimax法
- ダイクストラの最短経路問題
- ネットワークフロー