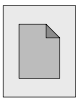
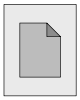
データ構造とアルゴリズムII

第9回
単一始点最短路 (I)

412

小テスト内容

413

24.単一始点最短路問題

414

第24章の構成

- 単一始点最短路問題とは
- 単一始点最短路問題の考え方
- 単一始点最短路問題を解く3つのアルゴリズム
  - ベルマン･フォードのアルゴリズム
  - トポロジカル･ソートによる解法
  - ダイクストラのアルゴリズム

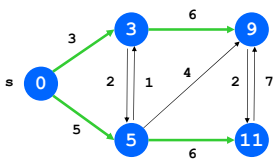
415

単一始点最短路問題とは

416

単一始点最短路問題とは

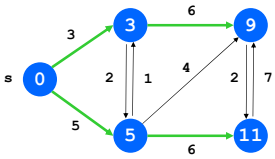
- 前提: 重み付き有向グラフ
- 特定の開始頂点 s から任意の頂点 v までの最短路を求める問題
  - 例: シカゴからボストンまでの最短経路
  - ※最短 ･･･ 重み最小 (経路本数最小ではない)



417

最短路重み δ(u, v)

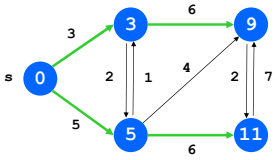
- w(u, v) ･･･ u → v の重み
- δ(u, v) ･･･ u → v の最短路重み
  - u → v の経路がないとき δ(u, v) = ∞



418

単一始点最短路問題で得られる情報

- (1) 始点 s から任意の頂点 v までの最短路重み δ(s, v)
- (2) 任意の頂点 v までの経路
  - 先行点 π[v] ･･･ v の前の頂点
  - π[v] は最短路木を構成



各頂点内の数字が δ(s, v)

緑の辺の集合が最短路木

「始点から特定の頂点への経路や最短路重み」ではなく，「始点から各頂点へのそれ」を求めているという点に注意

419

派生問題

- 単一始点最短路問題 (1 to N) が解けると以下の問題も解ける
  - 単一目的地最短路問題 (N to 1)
    - 1 to N を N to 1 に変更すれば OK
  - 単一点対最短路問題 (1 to 1)
    - 単一始点最短路から自明
    - 一見，1 to 1 なので単一始点最短路を求めるよりも良い方法がありそうだが，最悪の場合に漸近的に速く実行できる方法は知られていない
  - 全点対最短路問題 (N to N)
    - これはもっと良い方法がある → 第25章

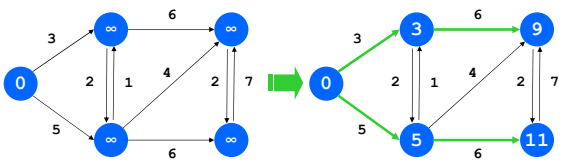
420

単一始点最短路問題の考え方

421

ざっくりとした解き方の説明

- 各頂点に始点 s からの重みの和を記録．最初は全部 ∞
- 適当なアルゴリズムで各辺を調べて，頂点に記録している重みが最短路のそれになるよう調整



422

複数のアルゴリズム

- 前提条件により最適なアルゴリズムが変わる
  - 負の重み，閉路のありなし (後述)
- アルゴリズムの違い
  - 各辺を調べる(緩和する，後述) 回数，調べる順番に相違がある
  - 当然，調べる回数/順番が多い方が，より汎用的な目的に使えるアルゴリズムになる

423

各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

424

各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

425

1. 最短路の部分構造最適性

- 最短路の部分経路は最短路
  - 証明: 補題 24.1
- 部分構造最適性
  - 動的計画法，貪欲アルゴリズムが適用できる可能性
    - ダイクストラ法は貪欲戦略を採っている

426

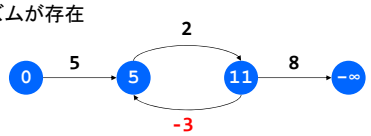
各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

427

2. 負の重みを持つ辺の扱い

- 全体として負の重みを持つ閉路，が問題
  - その閉路を巡回すると最短路重みを無限に小さくできる
    - $s \to v$ に至る経路上に負の重みの閉路が存在するなら $\delta(s, v) = -\infty$ とする
    - 最短路が定義不可能
- (閉路にならない) 負の重みの経路
  - 扱えるアルゴリズム/扱えないアルゴリズム
  - 負の重みの閉路があれば，それを発見して終了するアルゴリズムが存在



428

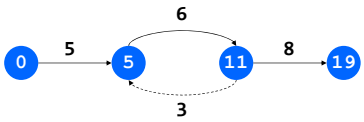
各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

429

3. 閉路の扱い

- (前述通り) 負の重みを持つ閉路が経路にあると最短路が定義できない
- 最短路は正の重みを持つ閉路を含まない
  - その閉路を取り除くと同一の始点と目的地を持つより小さな重みを持つ経路が生じるから



430

各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

431

4. 最短路の表現

- 頂点v に対して別の頂点か NIL を値とする先行点 π[v]
  - π[v] = u ⋯ u → v が最短路に含まれる
  - π[u] = x ⋯ x → u が最短路に含まれる
  - π[x] = s ⋯ s → x が最短路に含まれる
  - s → x → v → u が最短路
- 第22.2節の PRINT-PATH(G, s, v) で最短路を出力できる

432

各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

433

5. 緩和 (RELAX) #1

- 頂点に保存する重み ⋯ 最短路推定値 d[v] とする（最短路の重みの上界）
  - ※d[v] と δ(s, v) を混同しないこと
- 緩和
  - (u, v) に対する緩和操作 ⋯ u を経由することで v を改善できるなら d[v] およびπ[v] を更新する
  - 緩和により d[v] が減少し, π[v] が更新される
- 緩和を適当な順序でグラフに施していくことで, 最短路木を得る
- 上界を厳しくしていく操作を“緩和”と呼ぶのは奇妙なのだが, 伝統的にこの用語が使われる

434

5. 緩和 (RELAX) #2

RELAX(u, v, w)

一つ前の頂点 (先行点) から緩和するところがポイント

RELAX(u, v, w)

緩和の結果何も更新されないこともある

435

5. 緩和#3 擬似コード

```
RELAX(u, v, w)
    if d[v] > d[u] + w(u, v)
        then d[v] ← d[u] + w(u, v)
            π[v] ← u
```

436

ついでに，初期化の擬似コード

```
INITIALIZE-SINGLE-SOURCE(G, s)
    for 各頂点 v ∈ V[G]
        do d[v] ← ∞
            π[v] ← NIL
    d[s] ← 0
```

437

各アルゴリズムの前に知っておくべきこと

1. 最短路の部分構造最適性
2. 負の重みを持つ辺の扱い
3. 閉路の扱い
4. 最短路の表現
5. 緩和
6. 最短路と緩和の性質

438

6. 最短路と緩和の性質

- 本章のアルゴリズムの正当性を証明するための，最短路と緩和に関する諸性質
  1. 三角不等式
  2. 上界性
  3. 無経路性
  4. 収束性
  5. 経路緩和性
  6. 先行点グラフの性質

本章の各アルゴリズムは，なぜそれで最短路木，最短路重みが得られるのかそれほど直感的ではないので，上記の諸条件から理詰めで正当性を考えると良い

439

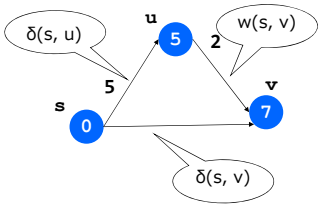
6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

440

三角不等式

任意の辺 $(u, v) \in E$ に対して $\delta(s, v) \leq \delta(s, u) + w(s, v)$ が成立する



441

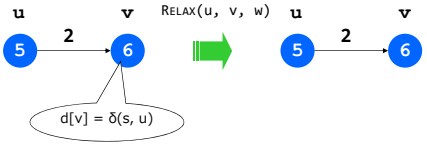
6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

442

上界性

すべての v ∈ V に対して，d[v] ≧ δ(s, v) が成立する．ひとたび d[v] が値 δ(s, v) を取ると，その後は決して変化しない



443

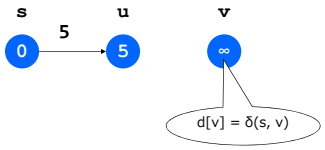
6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

444

無経路性

頂点 s から v に至る経路がない場合，d[v] = δ(s, v) = ∞ が成立する



初期化ですべての d[v] は ∞になっていて，d[v] が更新されるのは緩和操作時だけ．緩和操作は先行点から行われるが，孤立した頂点は先行点がないので∞から更新されることがない

445

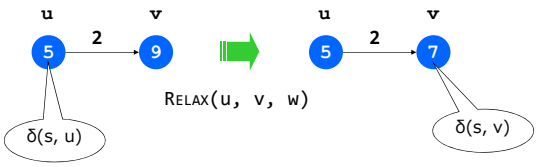
6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

446

収束性

ある u, v ∈ V に対して，s ～> u → v を G の最短路と仮定する．辺 (u, v) に対して緩和を実行する前に d[u] = δ(s, u) が成立した時点があったとすると緩和実行後は常に d[v] = δ(s, v) が成立する



447

6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

経路緩和性

$p = \langle v_0, v_1, \cdots, v_k \rangle$ が $s = v_0$ から v_k に至る最短路で，p の辺が $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ の順序で緩和されたとき，
$d[v_k] = \delta(s, v_k)$ が成立する．

この性質は他の任意の緩和操作とは無関係に成立する．たとえこれらの緩和操作の実行が p の緩和操作の実行とシャッフルされた順序で実行されたとしてもこの性質は成立する．

6. 最短路と緩和の性質

1. 三角不等式
2. 上界性
3. 無経路性
4. 収束性
5. 経路緩和性
6. 先行点部分グラフの性質

先行点部分グラフの性質

すべての $v \in V$ に対して $d[v] = \delta(s, v)$ が成立するとき，先行点部分グラフは s を根とする最短路木である

この性質から，すべての頂点を緩和で $\delta(s, v)$ にすることができれば目的を達成したことになる，と言える

先の経路緩和性から，定められた順序で緩和していけば $d[v_k] = \delta(s, v_k)$ が得られることは分かっている．よって

順番に緩和 → 全部の頂点が δ → 最短路が得られる

というのがアルゴリズムの基本方針となる

3つのアルゴリズム

- ベルマン･フォードのアルゴリズム
  - O(VE)
  - 負の重みOK, (負の重みは持たない) 閉路OK
- トポロジカル･ソート順序の緩和
  - Θ(V + E)
  - 負の重み OK, 閉路なし
- ダイクストラのアルゴリズム
  - O(VlgV + E) or O((V+E)lgV)
  - 負の重みなし

ベルマン･フォードのアルゴリズム

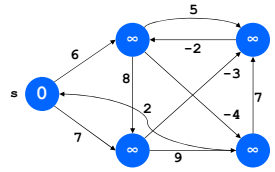
ベルマン・フォードのアルゴリズム

- 一般の単一始点最短路問題を解く
  - 負の重みを持つ辺を含んでも OK
  - 負の重みを持つ閉路の存在をチェックすることができる
- BELLMAN-FORD(G, w, s)
  - 負の重みを持つ閉路がない ･･･ 返値 TRUE
    - d[v] と π[v] も想定通りに埋まる
  - 負の重みを持つ閉路がある ･･･ 返値 FALSE

ベルマン・フォードのアルゴリズムの方針

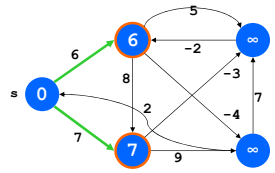
- |V| - 1 回，すべての辺を緩和するとすべての v に対して d[v] = δ(s, v) になる
  - すべての v に対して... → 先行点部分グラフの性質から，グラフは最短路木

ベルマン・フォードのアルゴリズムの動き



|V| - 1 回ループし，ループ毎に各辺を緩和する

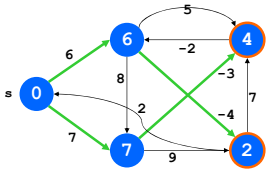
左はループ開始直前



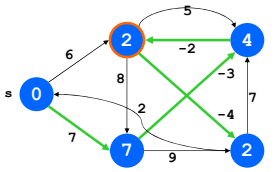
ループ1回目．始点 s 以外の頂点は d[v] = ∞ なので，始点 s の出辺だけ d が更新される

(緑の辺は先行点の値)

ベルマン・フォードのアルゴリズムの動き

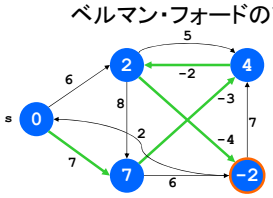


ループ2回目．1回目のループで d が減少した頂点からの出辺の緩和により2頂点が更新



ループ3回目．2回目のループで d が減少した頂点からの出辺の緩和により更新

ベルマン・フォードのアルゴリズムの動き



ループ4回目 (最後)．3回目のループで d[v] が減少した頂点からの出辺の緩和により更新

ループを抜けた時点での d と π の値が最終的な値

擬似コード

```
BELLMAN-FORD(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    for i ← 1 to |V[G]| - 1
        do for 各辺(u, v) ∈ E[G]
               do RELAX(u, v, w)
    for 各辺 (u, v) ∈ E[G]
        do if d[v] > d[u] + w(u, v)
               then return FALSE
    return TRUE
```

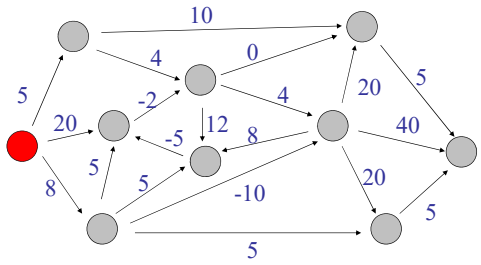
各辺の緩和操作

負の重みを持つ閉路の存在確認
(あったら FALSE を返す)

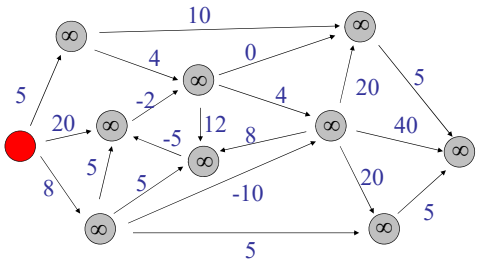
正当性は補題 24.4

演習

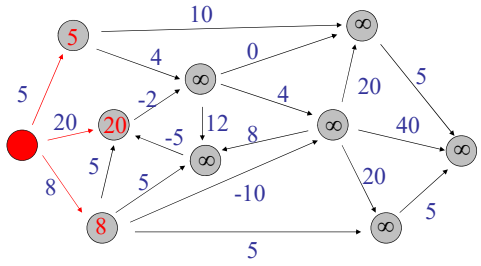
- このネットワークの赤の頂点からの最短距離をベルマン・フォードのアルゴリズムで求めよ



演習

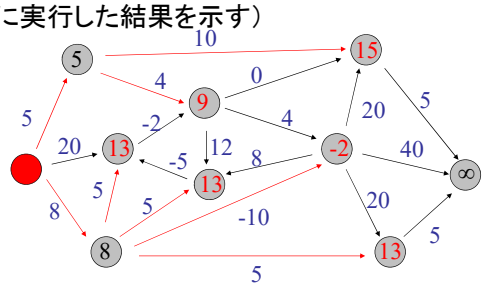


演習

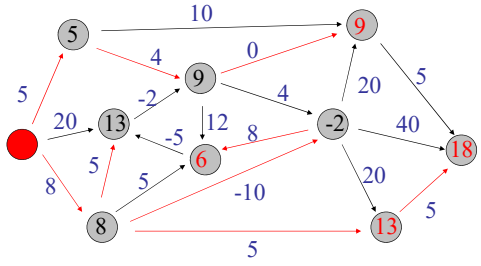


演習

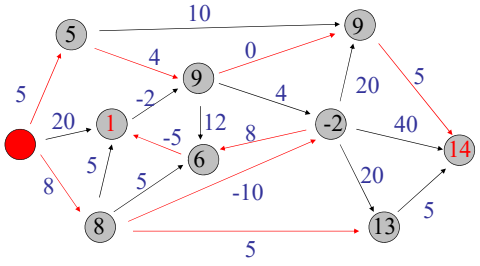
- 緩和操作は，各ノードに関して逐次的に実行しても，並列に実行しても良い（ここでは並列に実行した結果を示す）



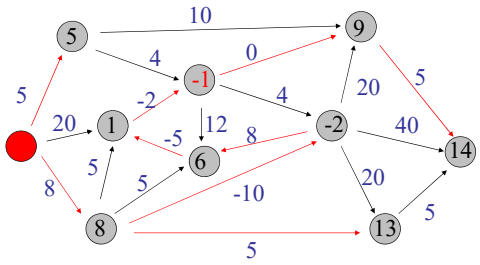
演習



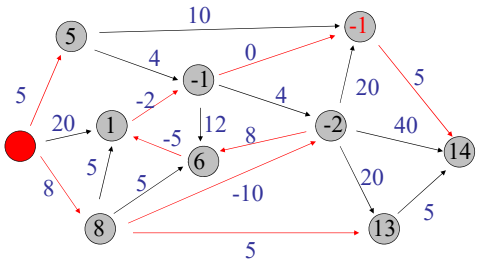
演習



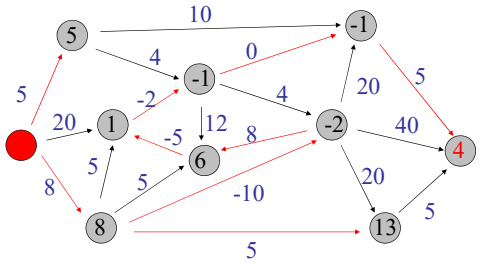
演習



演習



演習



演習：アルゴリズムの正当性

- ベルマン・フォートアルゴリズムで最短路が得られることを証明せよ
  - 証明すべきこと ･･･ "|V| - 1 回繰り返した後，すべての頂点 v に対して d[v] = δ(s, v) になっている"
    - これが分かれば先行点部分グラフの性質から最短路木が得られることが証明できる
  - ヒント：経路緩和性を使う

469

演習：アルゴリズムの正当性

- 経路緩和性による証明 (補題 24.2)
  - 各辺はループ毎に必ず緩和される → 経路緩和性の順序に沿って緩和が行われたと考えることができる，という点がポイント
    - $s \to v$ の経路 $p = \langle v_0, v_1 \ldots v_k \rangle$ としたとき，経路 p は高々 |V| - 1 個の辺を持つ．$\therefore k \leqq |V| - 1$
    - $i = 1, 2 \ldots k$ に対して (v_{i-1}, v_i) は i 回目の繰り返しで緩和される辺のひとつ → 経路緩和性が成立する
    - $v_0 = s$, $V_k = v$ であり，経路緩和性から $d[v] = d[v_k] = \delta(s, v_k) = \delta(s, v)$

470

計算量

- O(VE)
  - 初期化 O(V)
  - |V| - 1 のループ一回あたり O(E) → O(VE)
  - 負の閉路発見の for ループの実行時間 O(E)

471