JOI春季セミナー
上級2日目　最短路問題

目標

- グラフの最短路問題のうち、単一始点最短路問題を解くアルゴリズムを使えるようになる
  - ベルマン・フォード法
  - ダイクストラ法

最短路問題

最短路問題とは

- グラフ上で長さが最小の路（walk）を求める問題
  - （復習）路：グラフ G 上の 2 頂点 $s, t \in V$ について、s から t へと隣接する頂点を辿って到達できるとき、その経路を s-t 路という
- 単一始点最短路問題
  - ある頂点 s から他のすべての頂点に至る最短路を求める問題
- この講義においては特に言及がなければグラフ G は有向グラフとする

DAGにおける最短路問題

グラフが DAG （有向非巡回グラフ）のとき ⇒ 実は昨日やった（！？）

DAGにおける最短路問題

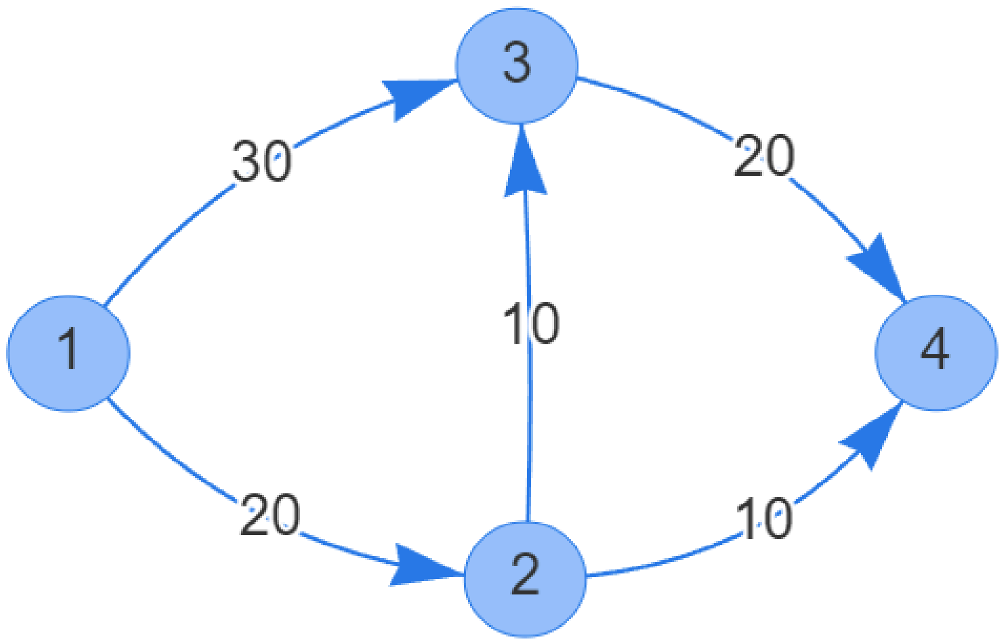
Frog 1 は DAG における単一始点最短路問題と捉えることができる

- 足場 1 に辿り着く最小コスト、足場 2 に辿り着く最小コスト、...というように、足場に辿り着く最小コストの求まる順番が定まっている
- 足場 i を頂点 i 、足場間のジャンプを頂点 i から頂点 j への重み $|h_i - h_j|$ の有向辺と見ると、頂点 1 から頂点 N への単一始点最短路問題

DAGにおける最短路問題

Frog 1 の場合

$N = 4, h_i = (10, 30, 40, 20)$ のとき



DAGにおける最短路問題

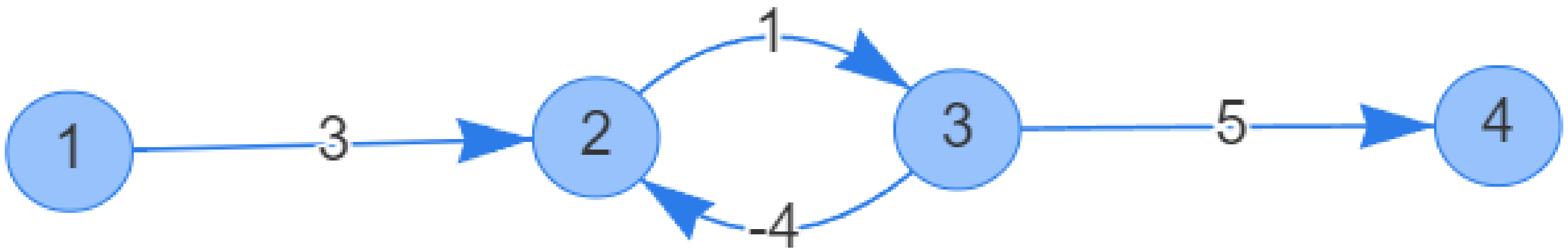
- 実際にはどの頂点から最短路が求まるかを調べるために、トポロジカルソートをする必要がある
- トポロジカルソートにかかる計算量は $O(|V| + |E|)$
- 最短路を求めるためにかかる計算量は $O(|V| + |E|)$

ベルマン・フォード法

グラフ G が負閉路を含む場合

最短路が定まらない可能性がある

- 例えば下のグラフで頂点 1 から頂点 4 に行きたいとき、頂点 2 と頂点 3 を往復することでいくらでも最短距離を小さくできる

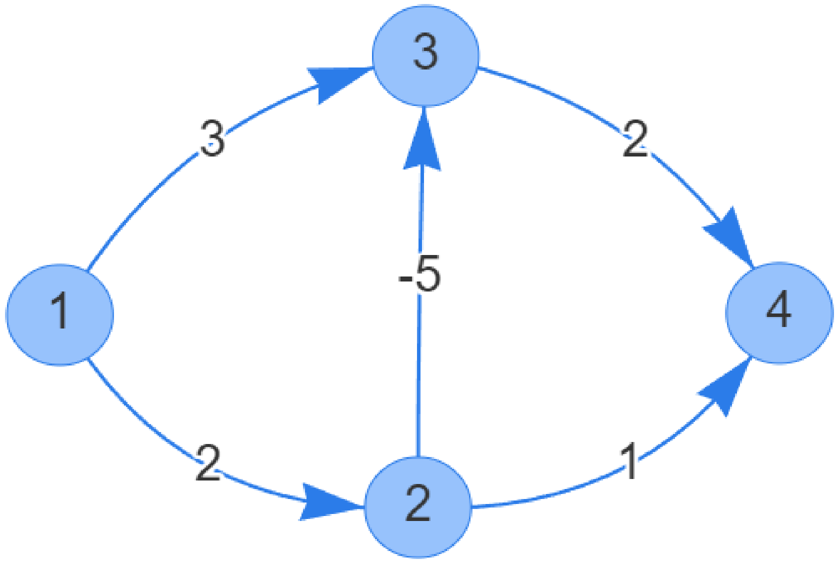


ベルマン・フォード法

グラフ G が負閉路を含む場合

対策 ⇒ すべての辺について緩和する操作を繰り返す

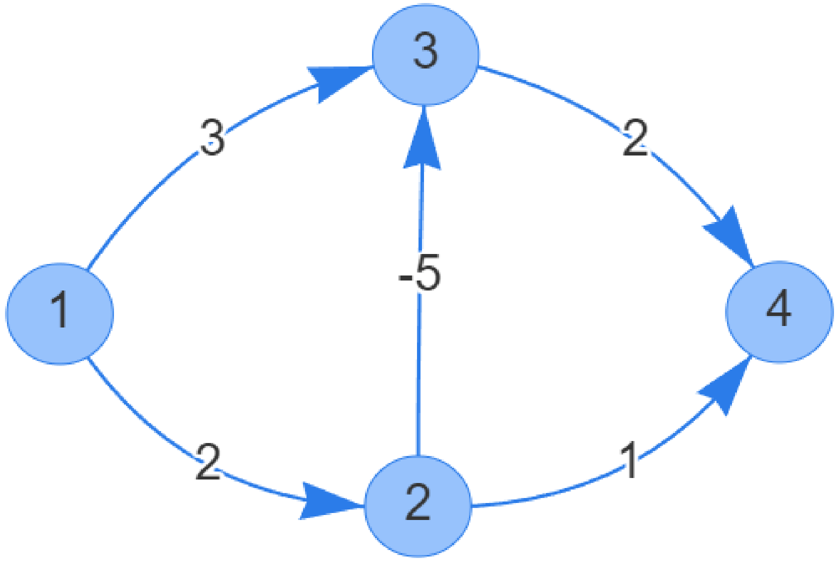
以下のグラフで考えてみる（以降、頂点 i から頂点 j へつながる有向辺を $e_{i,j}$ と表記）



ベルマン・フォード法

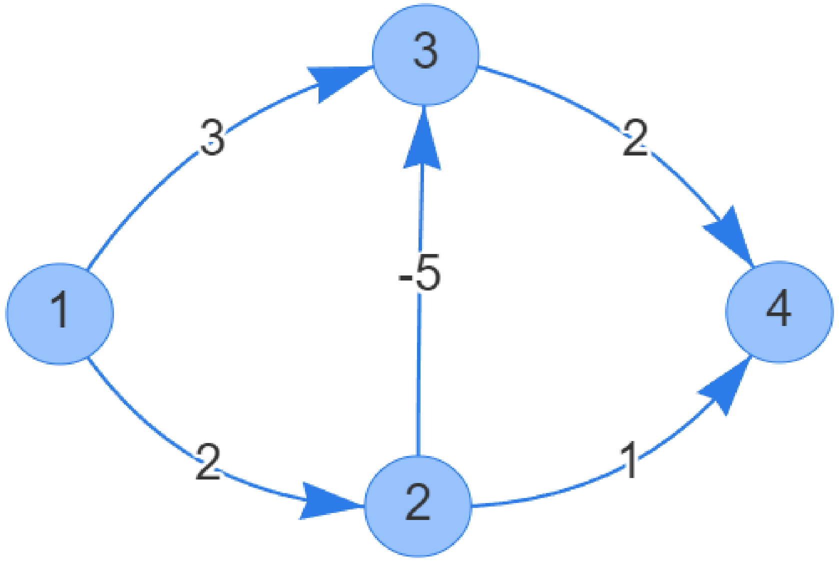
d_i を「頂点 i に辿り着く最短距離」として、
頂点 1 から頂点 4 に対する最短距離を求める

$d_i = (0, \infty, \infty, \infty)$ を初期状態とする



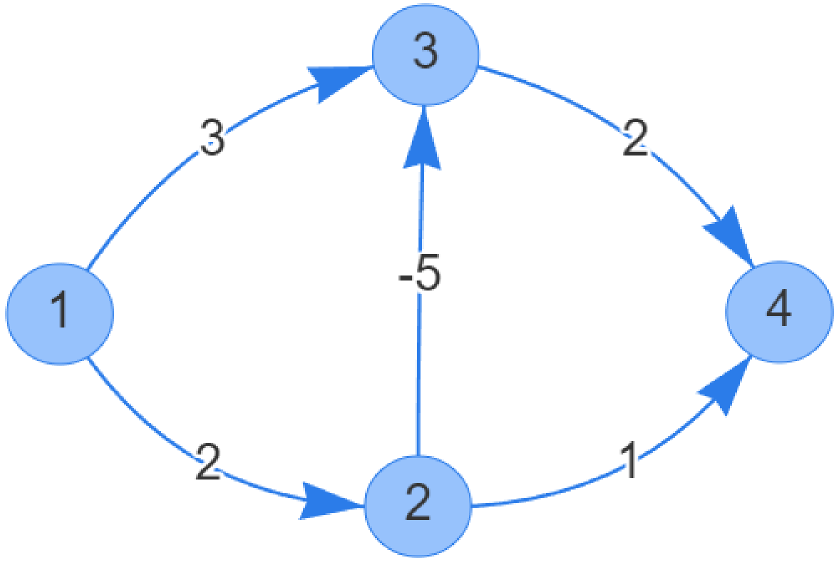
ベルマン・フォード法

- 1 回目の操作では $e_{1,2}, e_{1,3}$ について緩和が行われる
- $d_i = (0, 2, 3, \infty)$



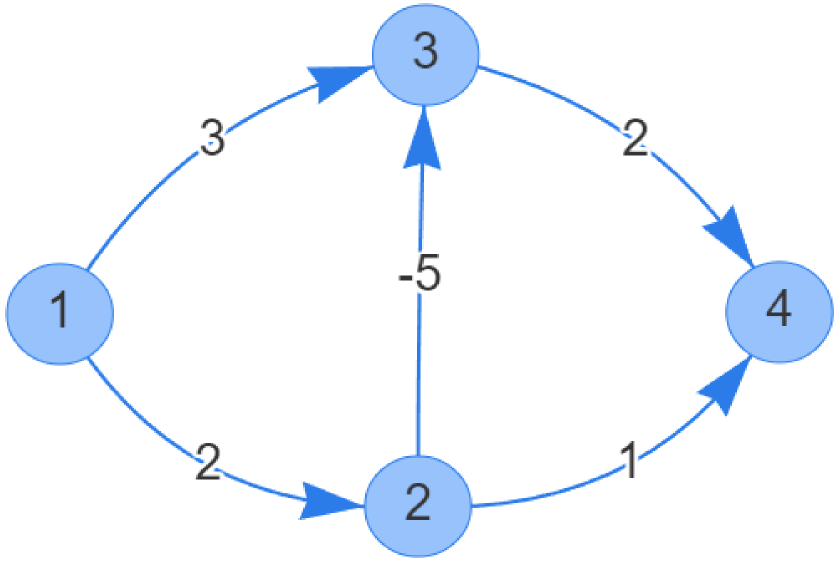
ベルマン・フォード法

- 2 回目の操作では $e_{2,3}, e_{2,4}, e_{3,4}$ について緩和が行われる
- $d_i = (0, 2, -3, 3)$



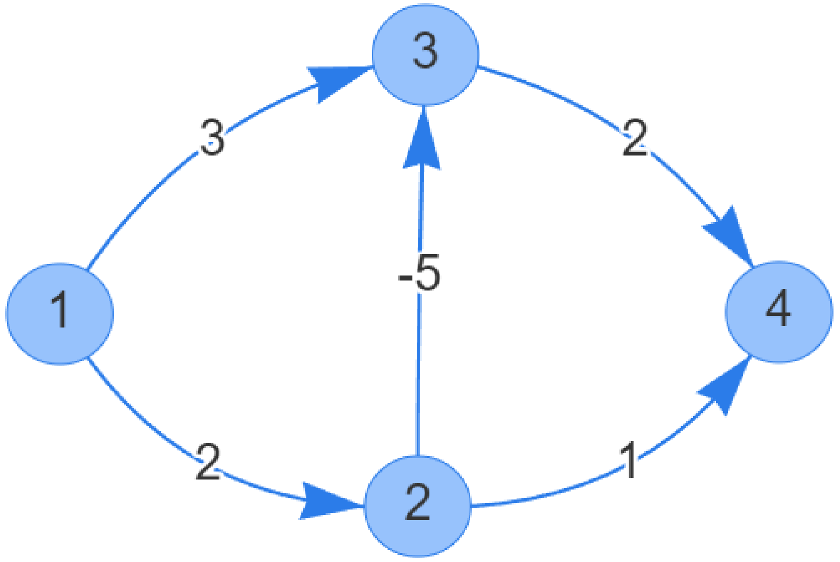
ベルマン・フォード法

- 3 回目の操作では $e_{3,4}$ について緩和が行われる
- $d_i = (0, 2, -3, -1)$



ベルマン・フォード法

- 4 回目以降の操作では緩和が行われないため、操作を終了する
- 最短距離は -1
- $d_i = (0, 2, -3, -1)$



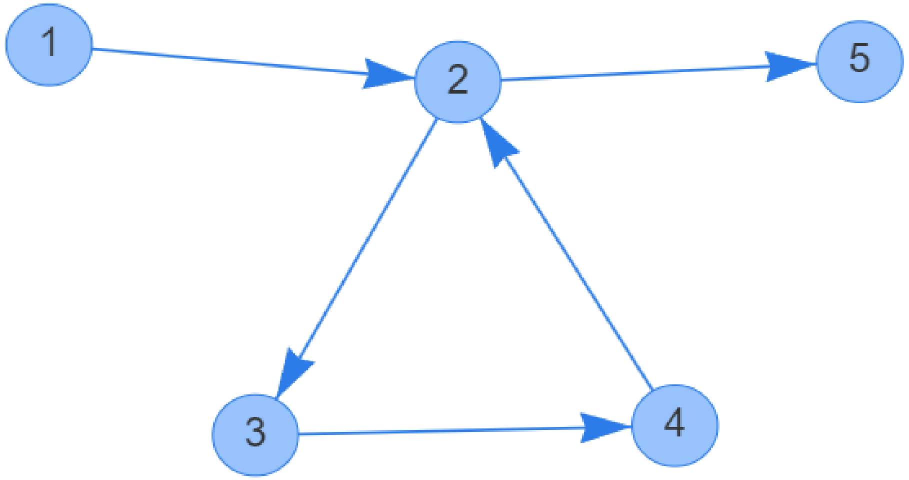
ベルマン・フォード法

- この操作は何回行えばいいのか
  - 結論から言うと、負閉路がない場合は $|V| - 1$ 回でいい
  - 逆に、$|V|$ 回目の操作で緩和が行われたなら、G に負閉路が含まれると言える

ベルマン・フォード法

$|V|-1$ 回で十分な理由（発展）

- G が負閉路を含まないとき、最短**路**問題ではなく最短**道**問題と考えて良い
  - （復習）道：同じ頂点を 2 回以上通らないような路(walk)
  - 同じ頂点を通ると必ず無駄になる
    - 下のグラフで、道 $(1, 2, 5)$ は明らかに路 $(1, 2, 3, 4, 2, 5)$ より短い



ベルマン・フォード法

$|V| - 1$ 回で十分な理由（発展）

- G が負閉路を含まないとき、最短**路**問題ではなく最短**道**問題と考えて良い
  - 最短道に含まれる辺の本数は $|V| - 1$ 本以下
  - 緩和が行われる回数は高々 $|V| - 1$ 回になるので、操作は $|V| - 1$ 回で十分

ベルマン・フォード法

負閉路があるとき $|V|$ 回目の操作で緩和が行われる理由（発展）

- 到達可能な負閉路に含まれる頂点列を $P = (v_1, v_2, \ldots, v_k, v_{k+1} = v_1)$ とする
- $|V|$ 回目の操作で更新が起こらないと仮定する
  - このとき、すべての i $(1 \leq i \leq k)$ について次が成り立つ

    （ただし $l(e_{i,j})$ は辺 $e_{i,j}$ の重み）

    $d_{v_i} + l(e_{v_i,v_{i+1}}) \geq d_{v_{i+1}}$

ベルマン・フォード法

負閉路があるとき $|V|$ 回目の操作で緩和が行われる理由（発展）

- $d_{v_i} + l(e_{v_i,v_{i+1}}) \geq d_{v_{i+1}}$ が成り立つことから、次が成立

$$l(P) := \sum_{i=1}^{k} l(e_{v_i,v_{i+1}}) \geq \sum_{i=1}^{k} (d_{v_{i+1}} - d_{v_i}) = 0$$

- しかしこれは P は負閉路であることと矛盾
  - したがって $|V|$ 回目の操作で更新が起こる

ベルマン・フォード法

計算量

- 各辺を緩和する操作の計算量は $O(|E|)$
- この操作を $|V|-1$ 回繰り返すため、全体の計算量は $O(|V||E|)$

ベルマン・フォード法

演習

単一始点最短経路（負の重みをもつ辺を含む）を解いてみましょう

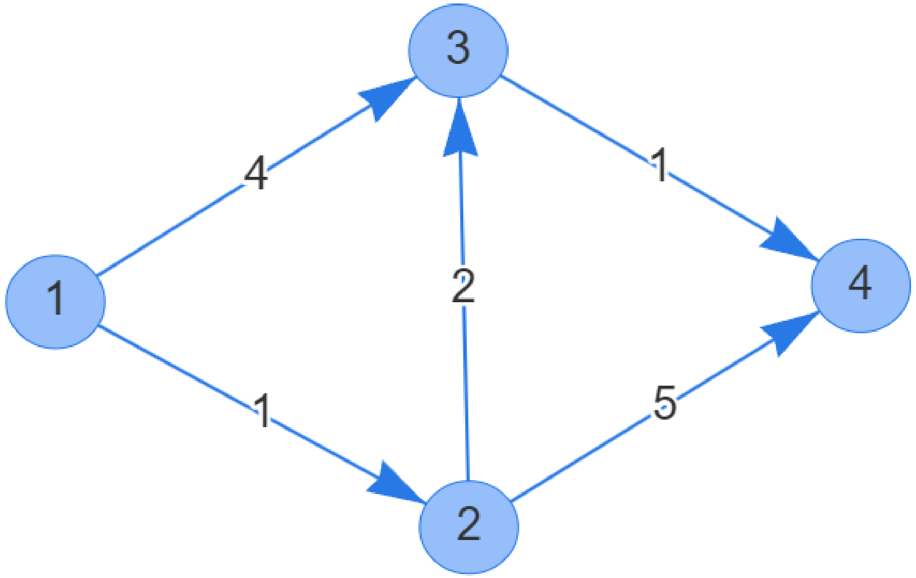
ベルマン・フォード法

演習問題の実装例

- C++での実装 ⇒ https://wandbox.org/permlink/s1dY3PfdlbIYUWou
- Pythonでの実装 ⇒ https://wandbox.org/permlink/snxGdtb5xlghYmO1

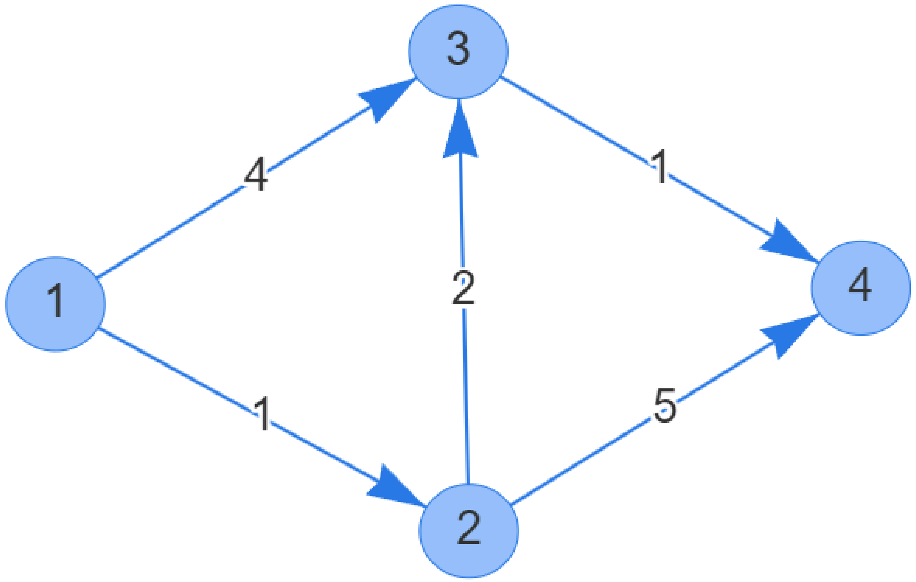
ダイクストラ法

- 以降では G の辺の重みがすべて非負として考える
- 負の辺がないとき、緩和の過程で最短路が確定する頂点が分かる
  - そのような頂点の集合を S とする
- 以下のグラフで確かめる（頂点 1 から頂点 4 への最短距離を求める）



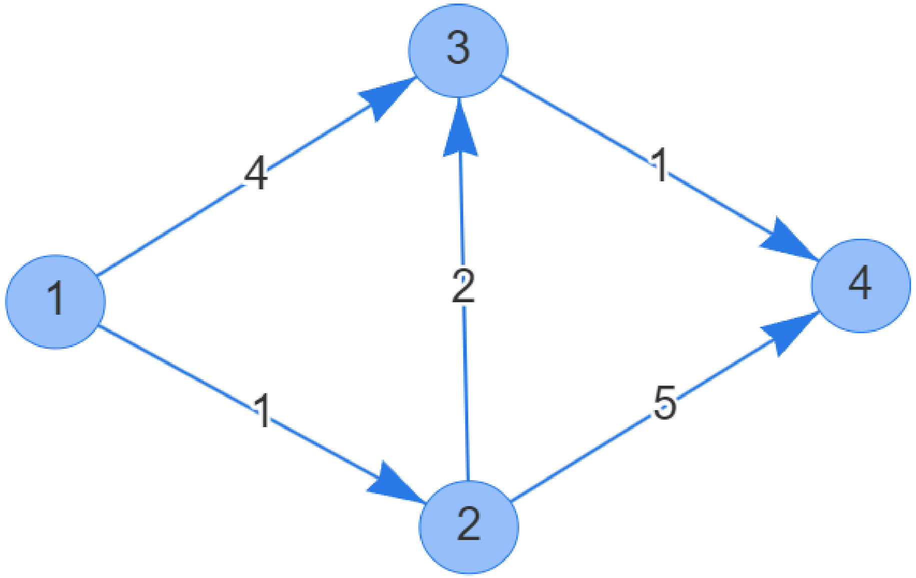
ダイクストラ法

- $d_i = (0, \infty, \infty, \infty), S = \{1\}$ で初期化



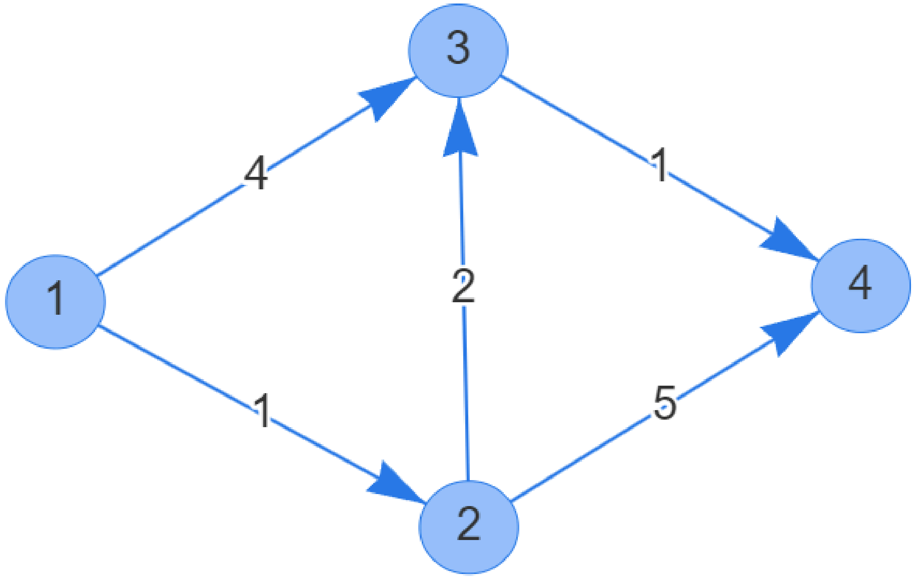
ダイクストラ法

- 次のような緩和が行われ、頂点 2 への最短距離が確定する
  - $d_1 + l(e_{1,2}) = 1 < \infty = d_2$ より頂点 2 への最短距離 d_2 が更新
  - $d_1 + l(e_{1,3}) = 4 < \infty = d_3$ より頂点 3 への最短距離 d_3 が更新
- $d_i = (0, 1, 4, \infty), S = \{1, 2\}$



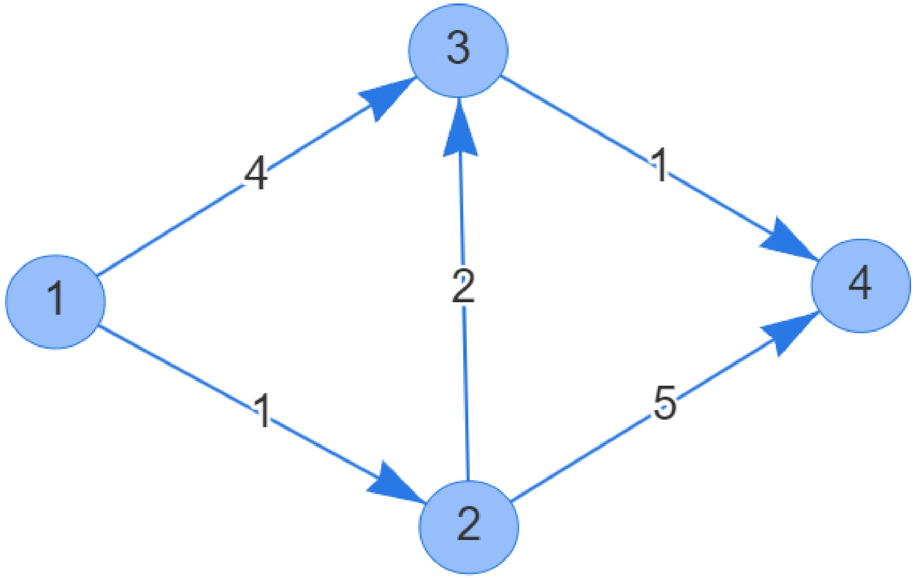
ダイクストラ法

- 次のような緩和が行われ、頂点 3 への最短距離が確定する
  - $d_2 + l(e_{2,3}) = 3 < 4 = d_3$ より頂点 3 への最短距離 d_3 が更新
  - $d_2 + l(e_{2,4}) = 6 < \infty = d_4$ より頂点 4 への最短距離 d_4 が更新
- $d_i = (0, 1, 3, 6), S = \{1, 2, 3\}$



ダイクストラ法

- 次のような緩和が行われ、頂点 4 への最短距離が確定する
  - $d_3 + l(e_{3,4}) = 4 < 6 = d_4$ より頂点 4 への最短距離 d_4 が更新
- $d_i = (0, 1, 3, 4), S = \{1, 2, 3, 4\}$
- すべての頂点に対する最短距離が求まったので、終了



ダイクストラ法

正当性の証明（発展）

- 帰納法で証明する
- 使用済みの頂点についてはすべて最短距離が求まっていると仮定する
- このとき使用済みでない頂点で、d_v が最小である頂点 v について $d_v = d_v^*$ が成り立つことを示す
- 始点 s から頂点 v への最短路の一つを P とし、P において v の直前の頂点を u とする

ダイクストラ法

正当性の証明（発展）

- u が使用済みの場合
  - すでに $d_v = \min\left(d_v, d_u^* + l(e_{u,v})\right)$ という緩和が行われているので、
    $d_v = d_v^*$ が成立
- u が使用済みでない場合
  - P において、s から順に辿って最初の使用済み出ない頂点を x とすると、
    $d_x = d_x^*$ が成立
  - G の各辺の重みは非負なので $d_x^* \leq d_v^*$ が成立
  - v は使用済みでない頂点のうち d_v が最小となる頂点だから $d_v \leq d_x$ が成立
  - まとめると $d_v \leq d_x = d_x^* \leq d_v^*$ が成立

ダイクストラ法

計算量

- 愚直に実装すると $O(|V|^2)$
  - 「使用済みでない頂点の中で d_i が最小である頂点」を検索するのに $O(|V|)$ 要しているため
- ヒープを使うと $O(|E| \log |V|)$ に計算量を落とすことができる
  - C++ならば `priority_queue`
  - Pythonならば `heapq`

ダイクストラ法

演習

単一始点最短経路 を解いてみましょう

ダイクストラ法

演習問題の実装例

- C++ ⇒ https://wandbox.org/permlink/6mzMiOk63vS4KjvR
- Python ⇒ https://wandbox.org/permlink/uTpwT2IXaYu9ymQm

演習問題（From AtCoder）

- 鉄則本 A64 - Shortest Path 2
- ABC 340 D - Super Takahashi Bros.
- ABC 061 D - Score Attack
- ABC 137 E - Coins Respawn
- ABC 070 D - Transit Tree Path
- ABC 325 E - Our clients, please wait a moment
- ABC 252 E - Road Reduction
- ABC 342 E - Last Train

演習問題（From JOI / JOIG）

- JOI 2008年 予選 F - 船旅 (難易度5)
- JOI 2014年 予選 E - タクシー (Taxis) (難易度7)
- JOI 2016年 予選 E - ゾンビ島 (Zombie Island) (難易度7)
- JOIG 2022年 本選 F - タクシー2 (Taxis 2) (難易度 9)

おわり

おつかれさまでした！