アルゴリズムとデータ構造

第14回　最小全域木(2)、最短路

今日の内容

- 最小全域木を求めるクラスカルのアルゴリズム 前回の復習
- 最小全域木を求めるプリムのアルゴリズム
  - プリムのアルゴリズムの考え方
  - 時間計算量 $O(m \log n)$
- 最短路問題
  - 最短路、最短路木
  - ダイクストラのアルゴリズム
  - アルゴリズムの直観的な理解

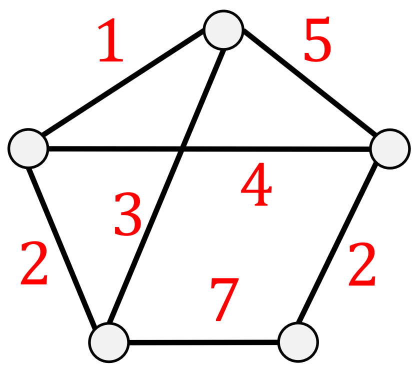
最小全域木 (minimum spanning tree)

前回の復習

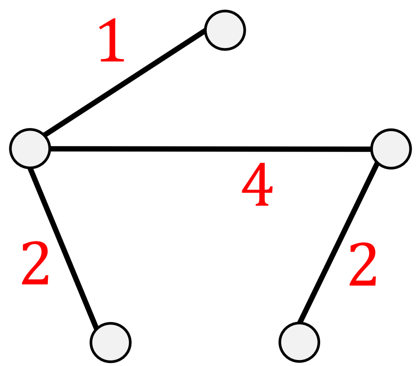
- ネットワーク (重み付きグラフ) $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$
- **定義**: G_1 が G_2 の**最小全域木** (**最小スパニング木**) である
  ⇔ G_1 は、G_2 の全域木で、
  含まれる**辺の重みの総和が最小のもの**

補足: 1つのネットワークに、
最小全域木が複数ある場合
もある (辺の重みの総和は同じ)

ネットワーク G_1



G_1 の最小全域木



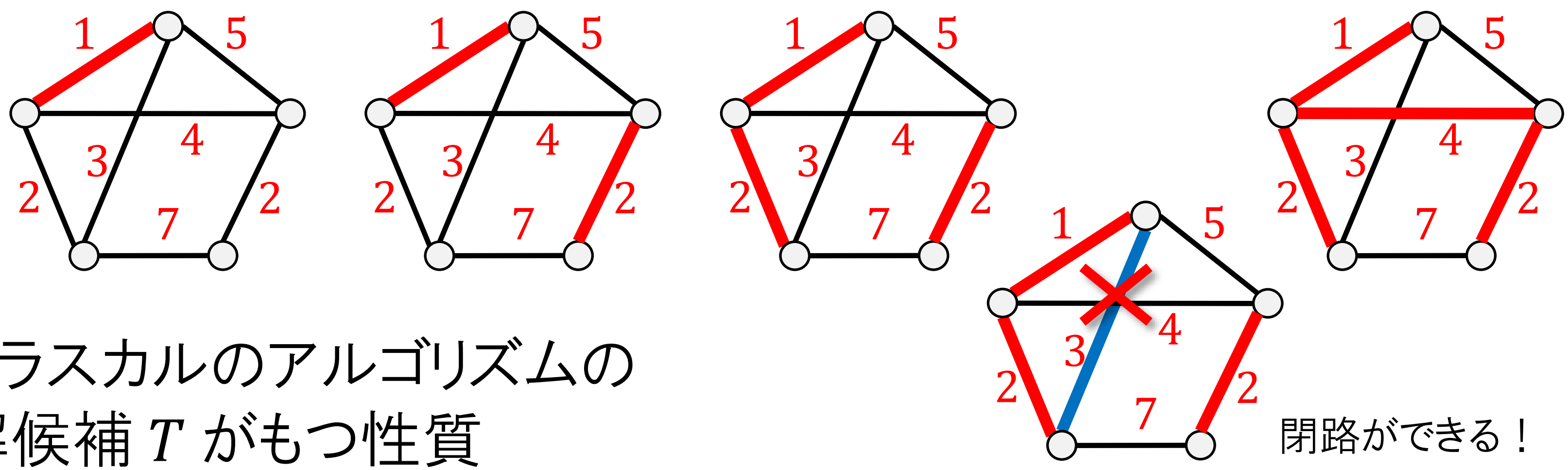
G_1 の辺が配管の候補で、この中
から、全体をつないで(連結)、
コスト(辺の重み)の総和が最小、
というものを見つけたい

応用例として, 通信網の設計, ガス･水道の配管設計などがある

クラスカルのアルゴリズム

前回の復習

［考え方］解候補Tを**空集合から始めて**，**重みが小さい辺から選択**し，現在の解候補Tに加えていく．ただし，選択することにより**閉路ができる場合には，その辺は選択しない**．



クラスカルのアルゴリズムの
解候補 T がもつ性質

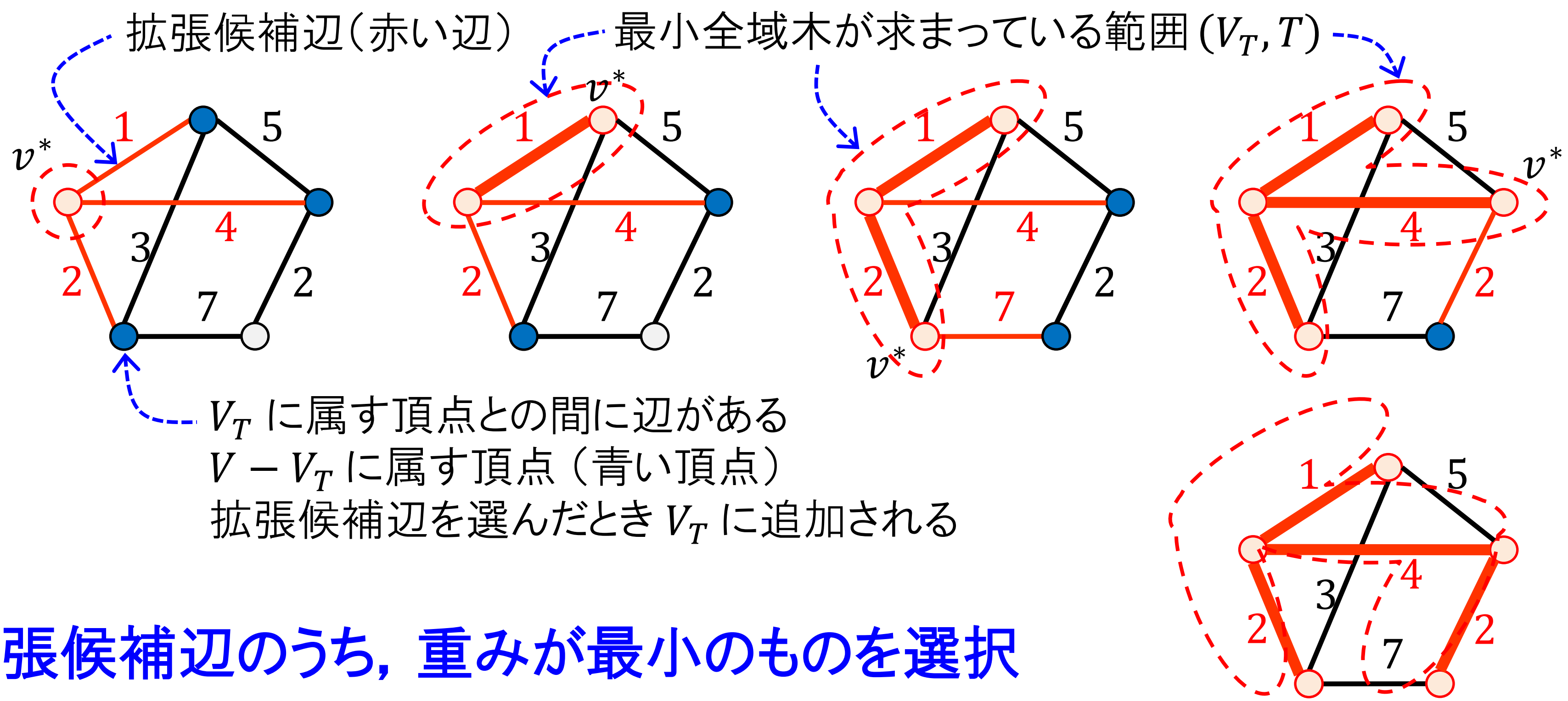
- T は閉路を含まない
- つまり，T は **森** になっている（必ずしも連結であるとは限らない）

※ グラフ G が森 ⇔ G は閉路のないグラフ

補足： 重みが同じ辺は、どの辺から選択しても ok

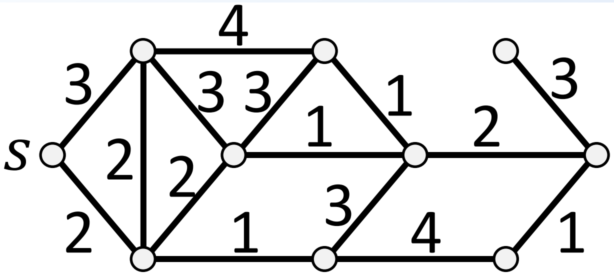
最小全域木を求めるプリムのアルゴリズム

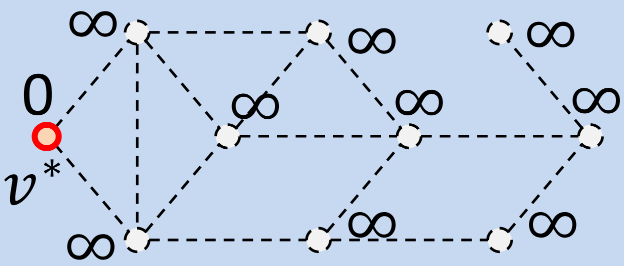
[考え方] 解候補 (V_T, T) を, **ある頂点 v_0 のみ**からなる 木
（v_0からなる部分グラフに対する最小全域木）から始める.
そこから, 最小全域木 (V_T, T) の**範囲を徐々に広げていく**.



拡張候補辺のうち, 重みが最小のものを選択
その先の頂点を加えたら, 拡張候補辺を更新

プリムのアルゴリズムの動作例

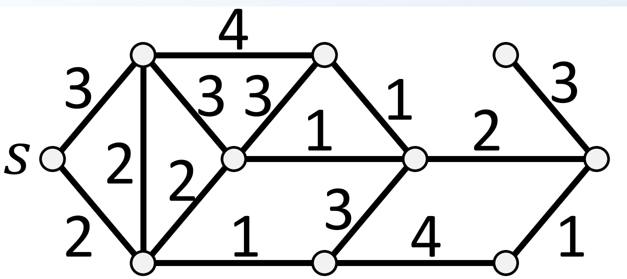


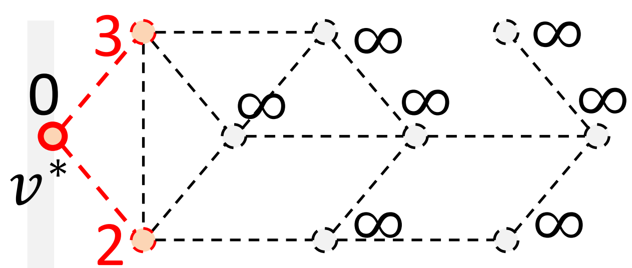


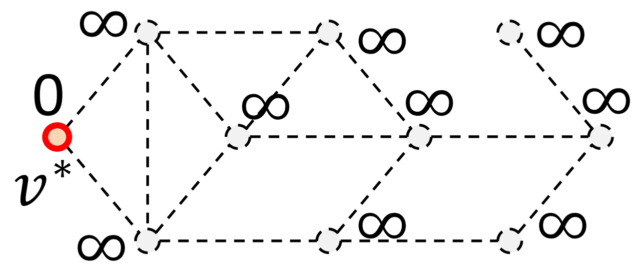
起点 s を v^* に（v^* を全域木に入れる）
$d(s) \leftarrow 0;$
他の頂点 v は、すべて $d(v) = \infty$

v^*：追加された頂点
赤字：更新された値
- - - ：拡張候補辺
：解候補

プリムのアルゴリズムの動作例





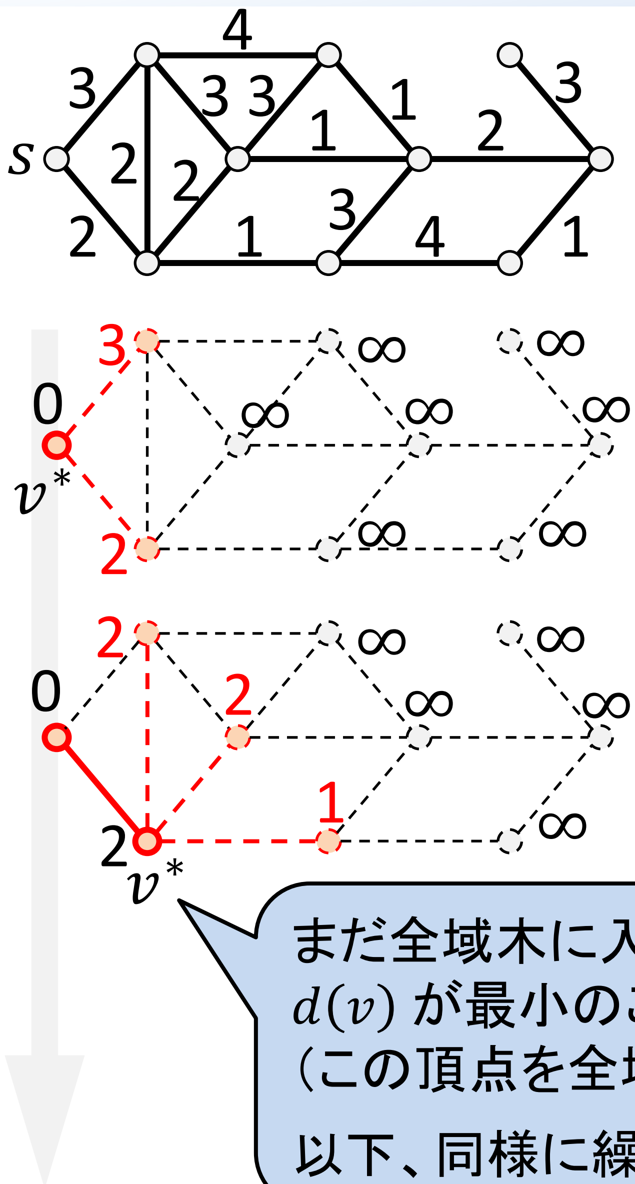


起点 s を v^* に（v^* を全域木に入れる）
$d(s) \leftarrow 0;$
他の頂点 v は、すべて $d(v) = \infty$

v^* に隣接するすべての頂点 v に対して
- $d(v)$ を更新
- 辺 (v^*, v) を拡張候補辺として記憶

v^*： 追加された頂点
赤字： 更新された値
：拡張候補辺
：解候補

プリムのアルゴリズムの動作例

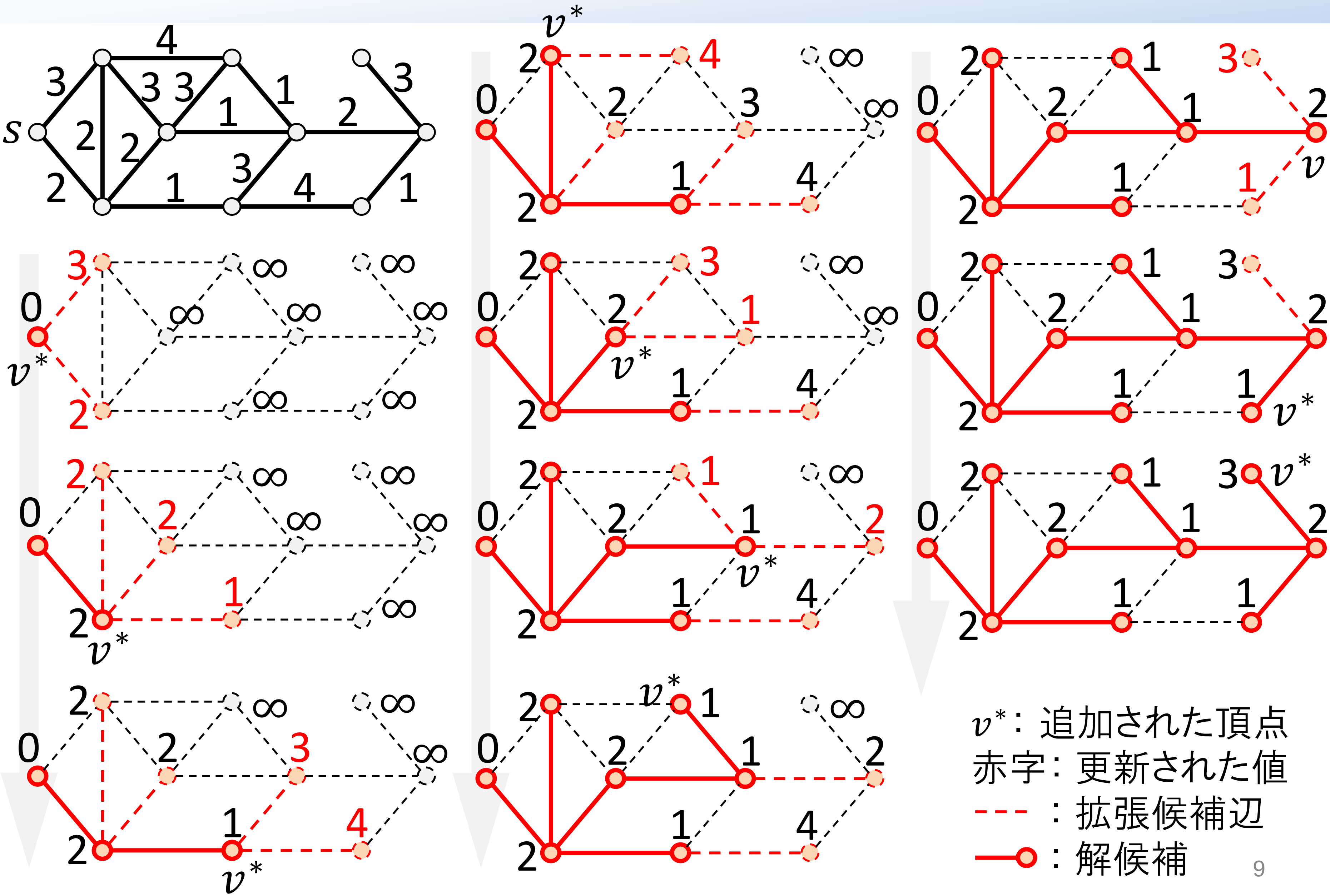


まだ全域木に入っていない頂点で
$d(v)$ が最小のこの頂点を v^* とする
（この頂点を全域木に入れる）

以下、同様に繰り返す

v^*：追加された頂点
赤字：更新された値
：拡張候補辺
：解候補

プリムのアルゴリズムの動作例



プリムのアルゴリズム（疑似コード）

Procedure MST-Prim($G=(V,E)$: グラフ, w: 重み, s: 起点)

$v^* \leftarrow s;\ T \leftarrow \emptyset;\ V_T \leftarrow \{s\}; d(s) \leftarrow 0;$

for all $v \in V - V_T$ do $d(v) = \infty;$

while ($V_T \neq V$) do

  for each (頂点v^*と隣接する頂点$v \in V - V_T$) do

    if ($w(v^*, v) < d(v)$) then

      $d(v) \leftarrow w(v^*, v),\ e(v) \leftarrow (v^*, v);$

    end if

  end for

  $v^* \leftarrow \arg\min_{v \in V - V_T} d(v);$

  $V_T \leftarrow V_T \cup \{v^*\},\ T \leftarrow T \cup \{e(v^*)\};$

end while

最小全域木Tを出力する;

$d(v)$：vを端点とする拡張候補辺の重み（動作例の各頂点の数字）

$e(v)$：vを端点とする拡張候補辺（赤色の辺）

$V - V_T$ の頂点で $d(v)$ が最小となる頂点を v^* とする

全域木に、v^* を頂点として $e(v)$ を辺として加える

最悪時間計算量が $\mathrm{O}(m \log n)$ の証明

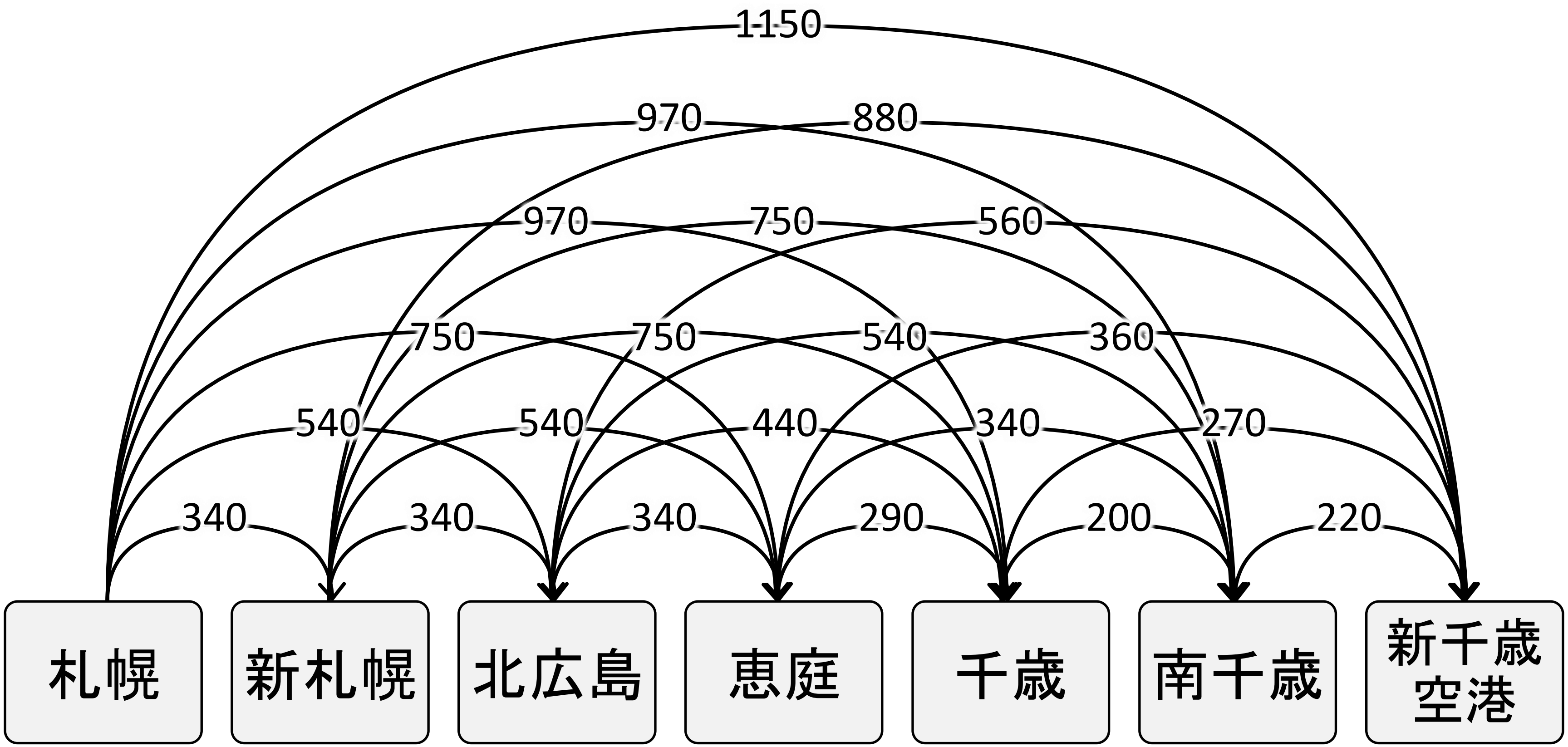
発展

（証明）1行目は定数時間で，2行目は明らかに$\mathrm{O}(n)$時間かかる．
3行目のwhileのチェックはVとV_Tのサイズ比較で行えるため，毎回の実行は定数時間である．
4行目のfor eachのループは新たなv^*毎に実行される．v^*は結局はすべての$v \in V$が一度ずつなるので，for eachループの中身はwhileループ全体で$2m$回実行される．for eachループの中身は数値比較と代入だけなので毎回の実行は定数時間であり，グラフが隣接リストで与えられている場合，全体で$\mathrm{O}(m)$時間で実行できる．
9行目のarg minの実行は，単純に行うとk回目のループのとき$\mathrm{O}(n-k)$時間がかかる．しかし，ヒープを使えば1回あたり$\mathrm{O}(\log n)$時間でできる．10行目は明らかに定数時間でできるので，以上から，whileループ全体では$\mathrm{O}(m + n \log n)$時間かかることになる．
グラフの連結性より$m \geq n-1$なので，よって全体を$\mathrm{O}(m \log n)$時間で抑えることができる．【証明終わり】

休憩

- ここで、少し休憩しましょう。
- 深呼吸したり、肩の力を抜いてから、
  次のビデオに進んでください。

Quiz: 札幌から新千歳空港への最安乗り換えは？



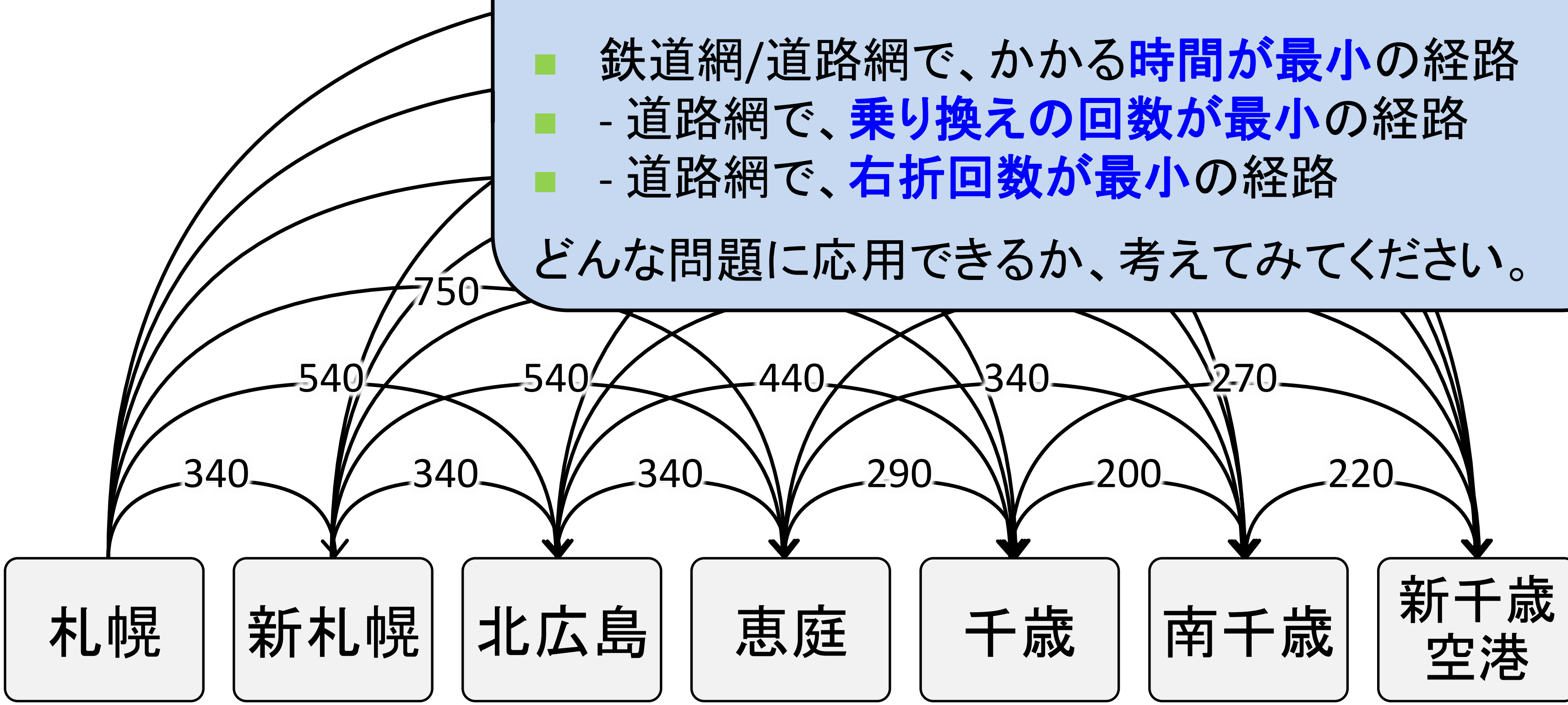
※ JR北海道のウェブサイト, 2023年6月調べ
快速エアポート（自由席）利用の場合

Quiz: 札幌から新千歳空港への最安乗り換えは？

この Quiz では、**費用を最小に**していますが、
他にも色々な場面で応用できます。

- 鉄道網/道路網で、かかる**時間が最小**の経路
- - 道路網で、**乗り換えの回数が最小**の経路
- - 道路網で、**右折回数が最小**の経路

どんな問題に応用できるか、考えてみてください。

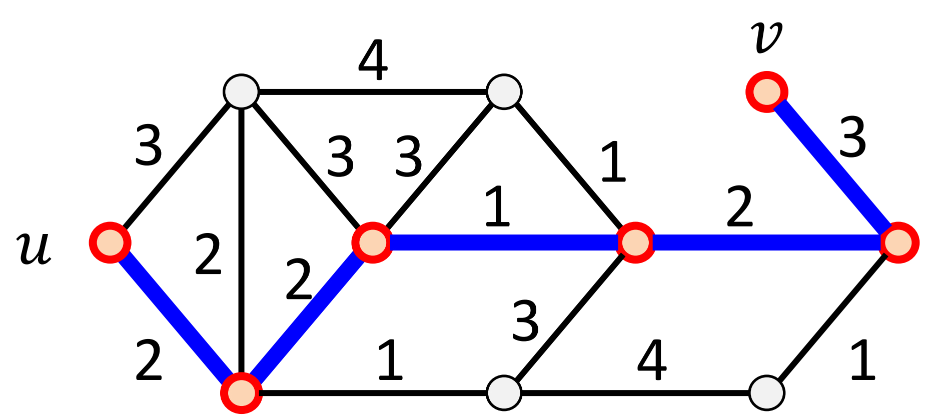


※ JR北海道のウェブサイト, 2023年6月調べ
快速エアポート(自由席)利用の場合

最短路 (shortest path)

定義：

無向ネットワーク$G = (V, E)$において，頂点$u \in V$から頂点$v \in V$への最短路（shortest path）は，u から v への路（path）のうち，**辺の重みの総和が最小のもの**である



uからvへの最短路

※ 負の重みを許すか否かで問題の難しさが変わる
本講義では，重みはすべて非負の値をとると仮定する

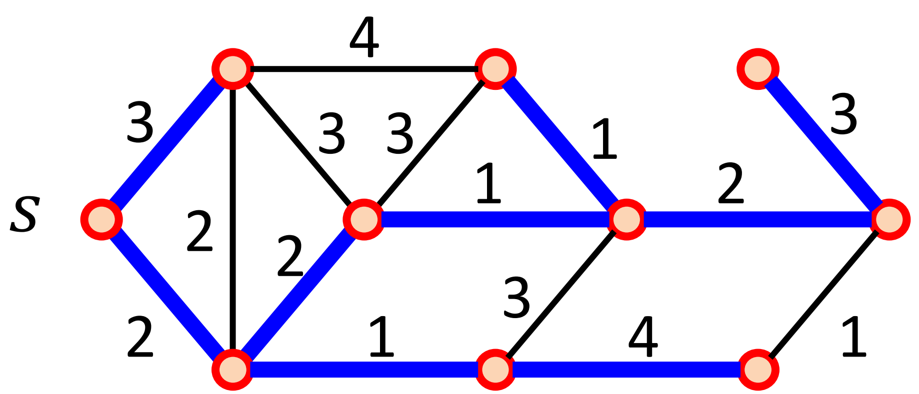
最短路木 (shortest path tree)

定義：

辺の重みが非負である連結な無向ネットワーク$G = (V, E)$に対し，
グラフ G' は頂点 $s \in V$ からの最短路木（shortest path tree）である

$$\Updownarrow$$

G' は **s を根とする G の全域木**で，s から各頂点への路が**最短路になっている**

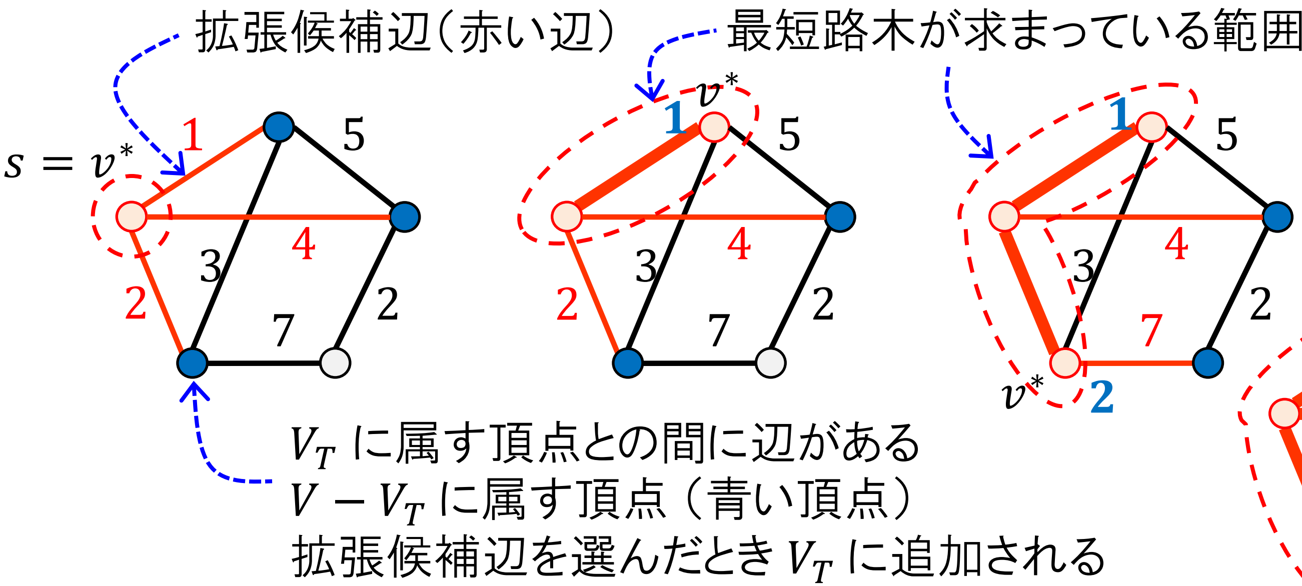


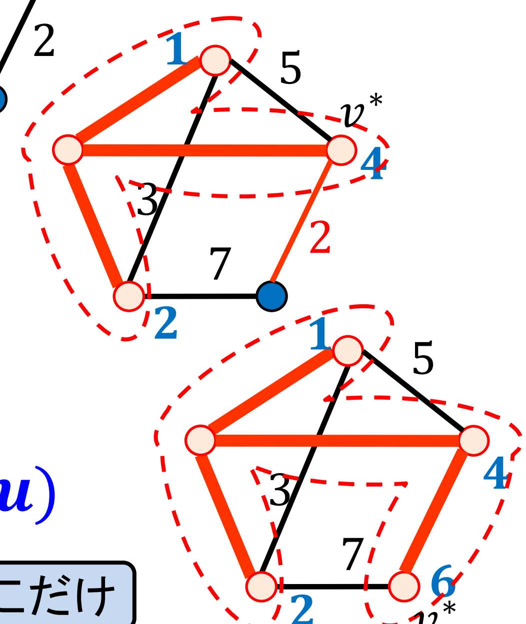
s を根とする最短路木

最短路木を求めるダイクストラ アルゴリズム

［考え方］解候補 (V_T, T) を，**ある頂点 s のみ**からなる木（s からなる部分グラフに対する最短路木）から始める．そこから，最短路木 (V_T, T) の**範囲を徐々に広げていく**．

Edsger W. Dijkstra
(1930–2002)オランダ
1972年チューリング賞

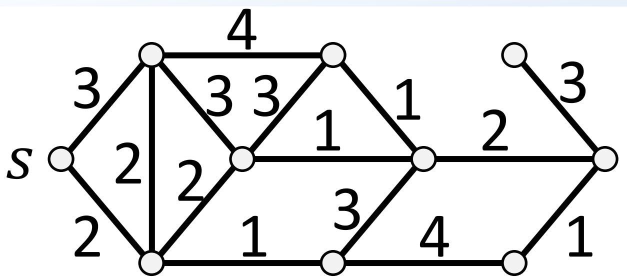


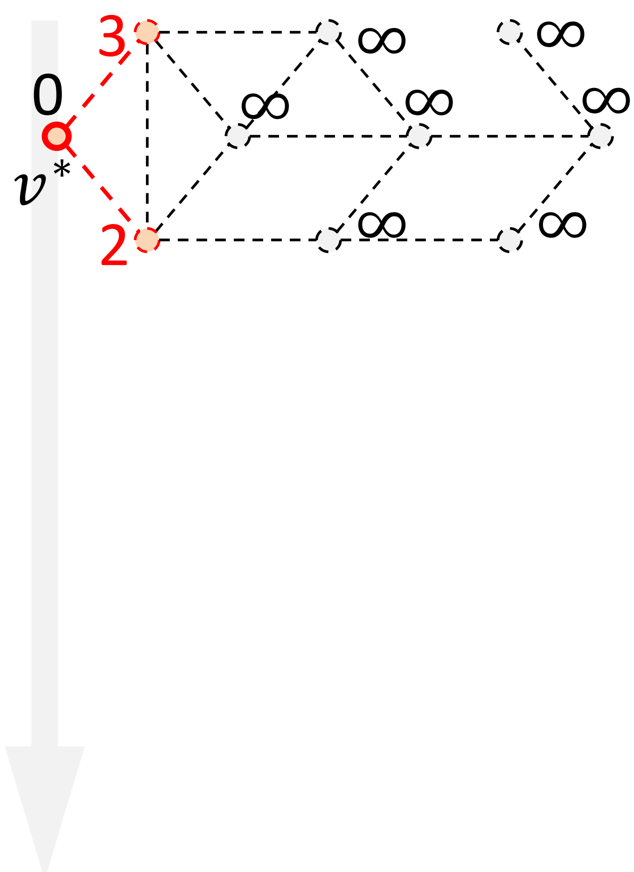


拡張候補辺は，$u \in V_T$ と $v \in V - V_T$ との間の辺 $e = (u, v)$ のうち，s から u への**最短路長 $d(u)$ と $w(e)$ の和が最小のもの**

プリム法との違いは，ここだけ

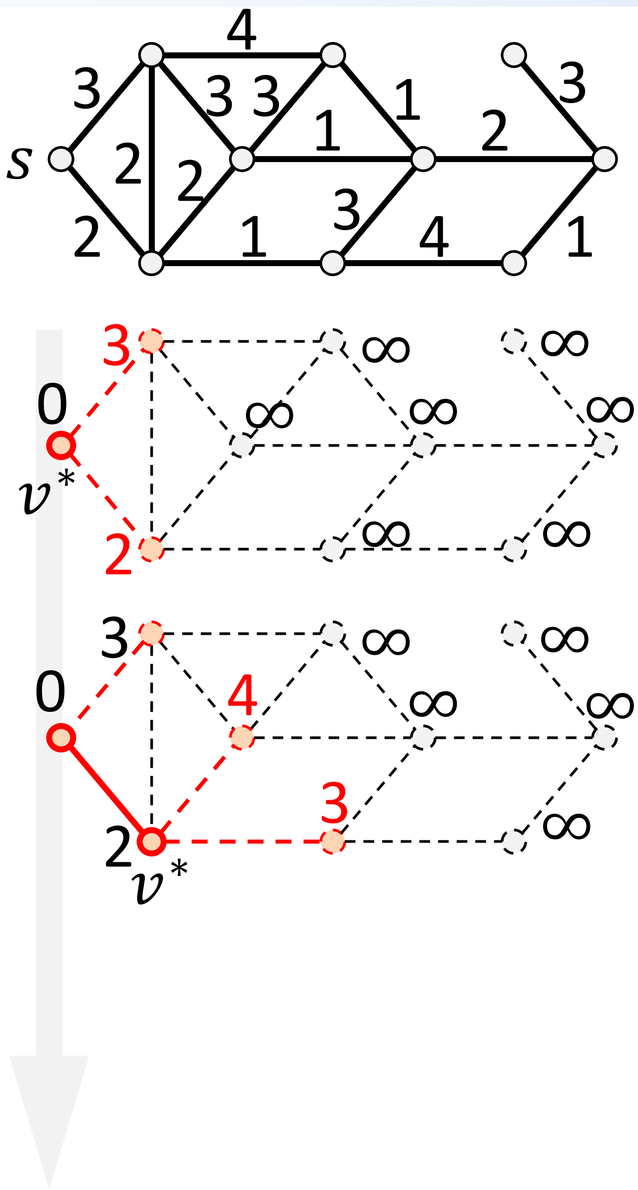
ダイクストラ アルゴリズムの動作例





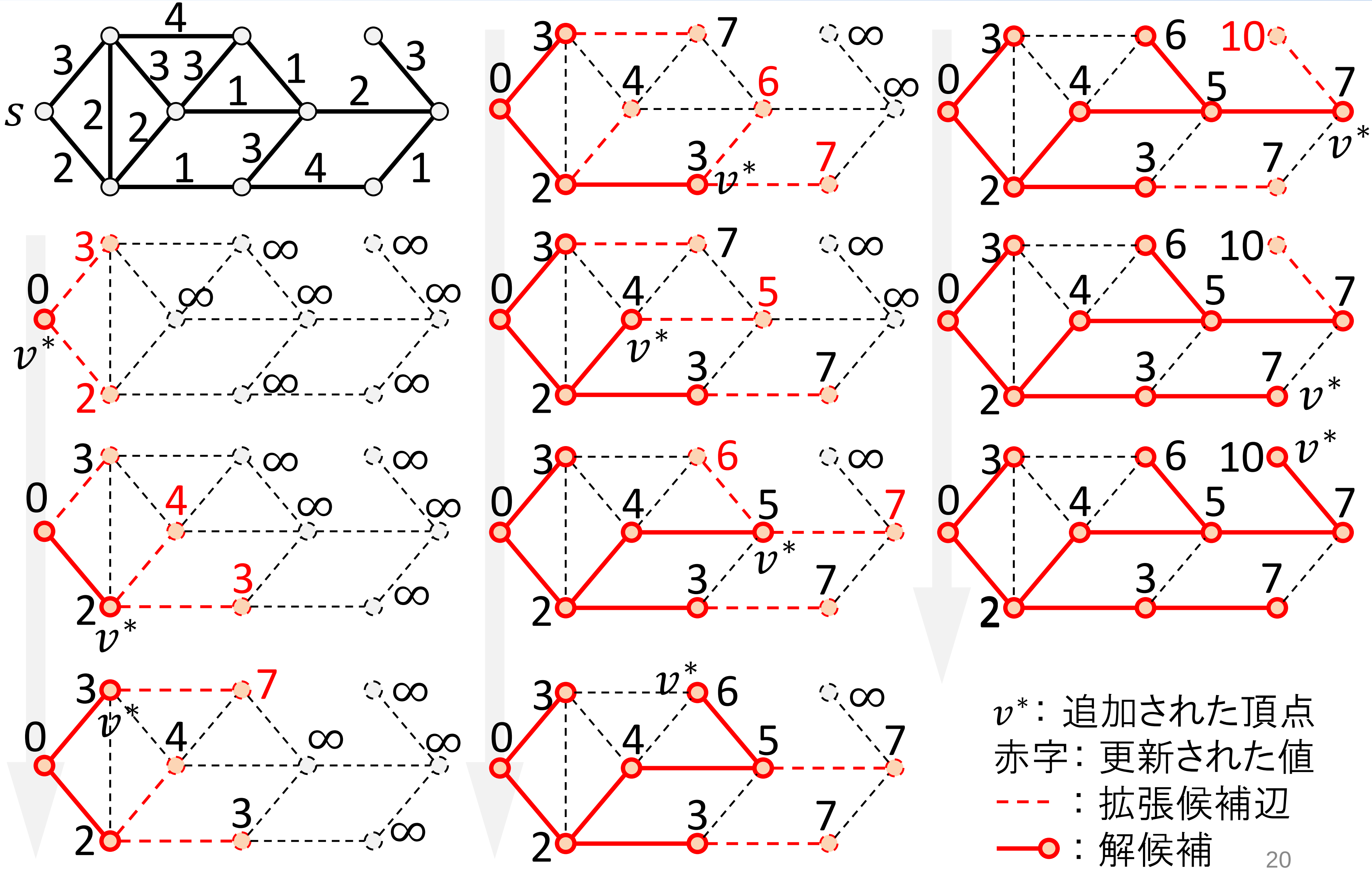
v^*：追加された頂点
赤字：更新された値
：拡張候補辺
：解候補

ダイクストラ アルゴリズムの動作例



v^*：追加された頂点
赤字：更新された値
：拡張候補辺
：解候補

ダイクストラ アルゴリズムの動作例



ダイクストラ アルゴリズム（疑似コード）

Procedure SPT-Dijkstra($G = (V, E)$: グラフ, w: 重み, s: 起点)

$v^* \leftarrow s;\ T \leftarrow \emptyset;\ V_T \leftarrow \{s\}; d(s) \leftarrow 0;$

for all $v \in V - V_T$ do $d(v) = \infty;$

while ($V_T \neq V$) do

for each (頂点v^*と隣接する頂点$v \in V - V_T$) do

if ($w(v^*, v) + \boldsymbol{d(v^*)} < d(v)$) then

$d(v) \leftarrow w(v^*, v) + \boldsymbol{d(v^*)},\ e(v) \leftarrow (v^*, v);$

end if

end for

$v^* \leftarrow \arg\min_{v \in V - V_T} d(v);$

$V_T \leftarrow V_T \cup \{v^*\},\ T \leftarrow T \cup \{e(v^*)\};$

end while

最短路木Tを出力する;

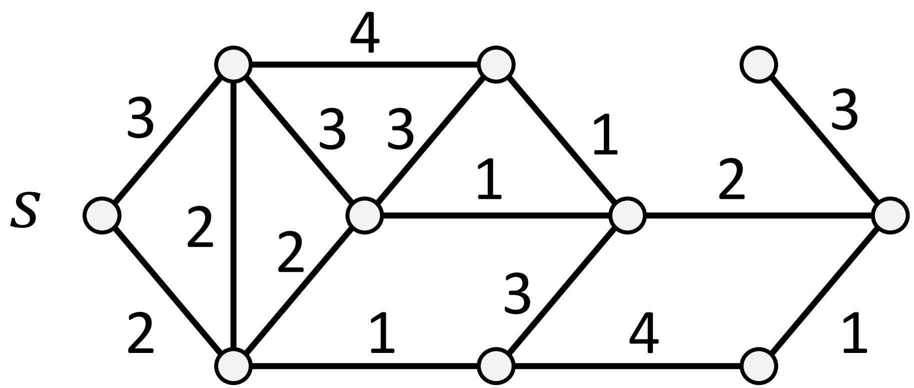
$d(v)$：vを端点とする拡張候補辺の重み（動作例の各頂点の数字）

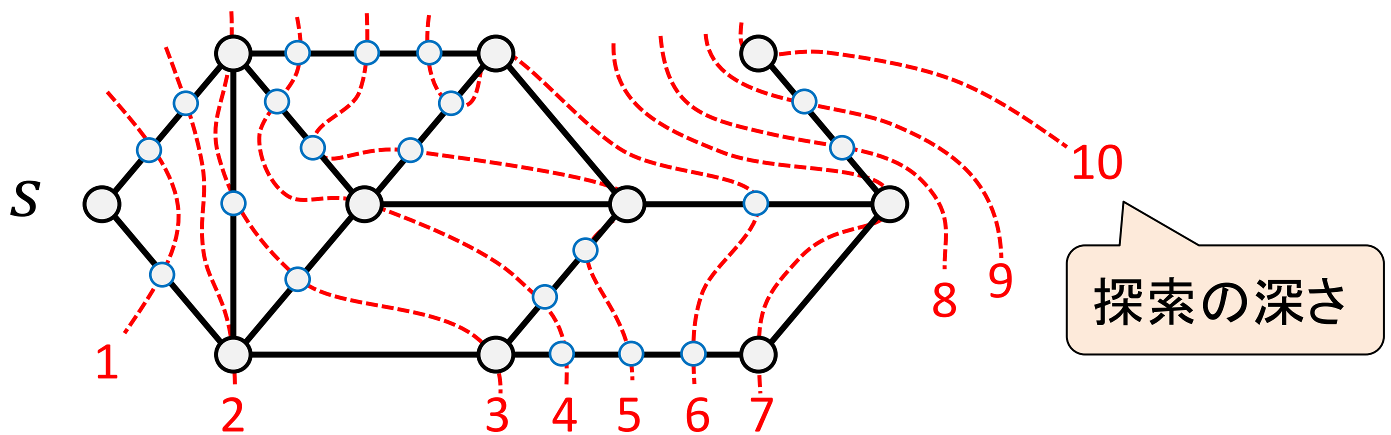
$e(v)$：vを端点とする拡張候補辺（赤色の辺）

Primのアルゴリズムとの違いは $+\boldsymbol{d(v^*)}$ だけ！

時間計算量は同じ $\boldsymbol{O(m \log n)}$

ダイクストラ アルゴリズムの直観的理解



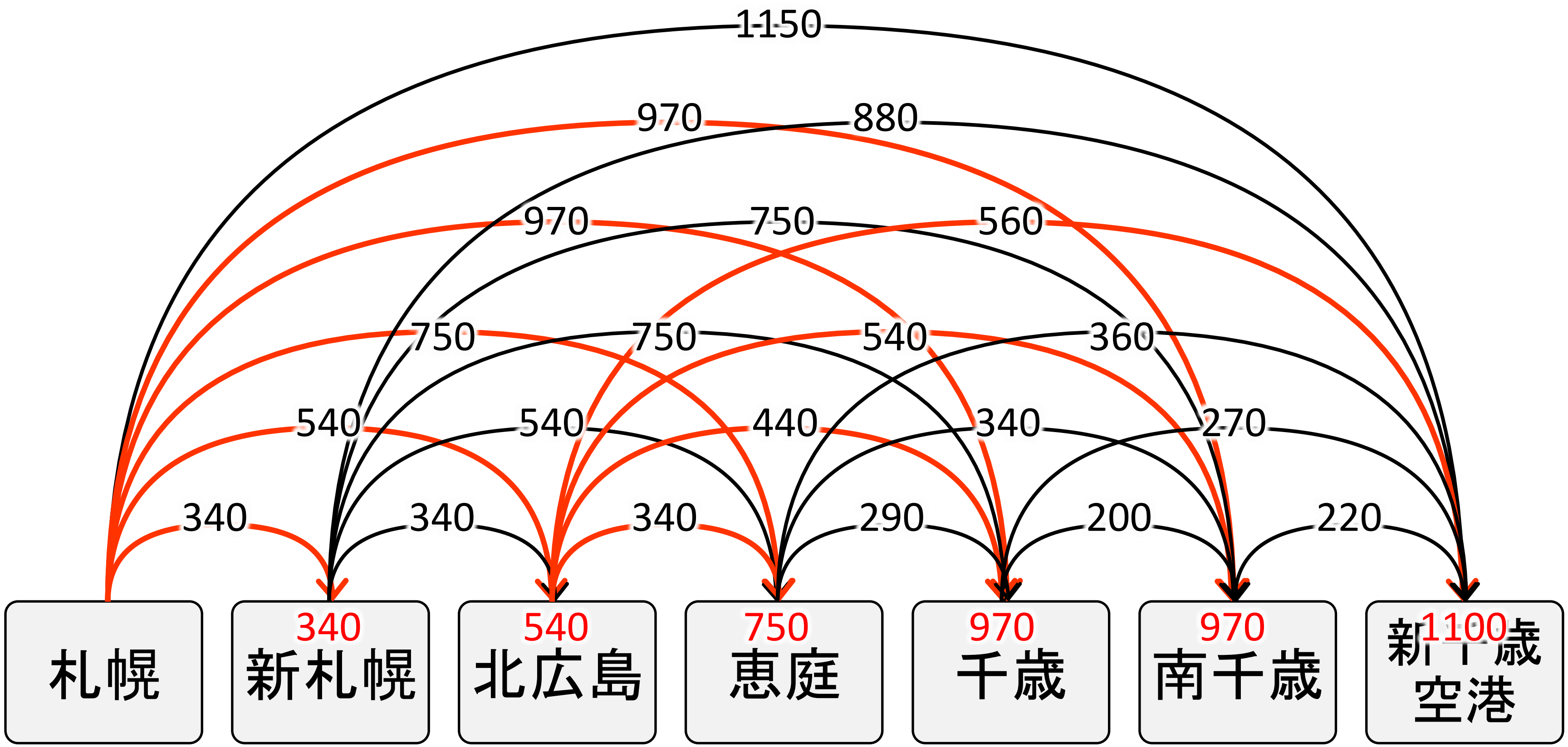


- 辺を細かく分けて，すべての辺の重みが等しく（1に）なるように分割
- 起点 s から**幅優先探索**を行い，到達した順に頂点を解に加える

おまけ：最短路に関する発展的な話題

- 辺の重みが非負整数の場合には，平均時間計算量を $\mathbf{O}(\boldsymbol{m})$ まで改善できる（ただし，ランダムアルゴリズムによる）
  [U. Meyar, Single-source shortest-paths on arbitrary directed graphs in linear average-case time, ACM-SIAM Symposium. Discrete Algorithms, 2001, pp. 797-806.]
- ベルマン-フォード法（Bellman-Ford algorithm）
  - 正負の重みを扱える
  - 最悪時計算量は $O(mn)$
- ワーシャル-フロイド法（Warshall-Floyd algorithm）
  - 与えられたグラフについて，すべての2頂点間の距離を求める
  - 最悪時間計算量は $O(n^3)$

Quiz: 札幌から新千歳空港への最安乗り換えは？



※ JR北海道のウェブサイト，2023年6月調べ
快速エアポート（自由席）利用の場合

今日のまとめ

- 最小全域木を求めるクラスカルのアルゴリズム 前回の復習
- 最小全域木を求めるプリムのアルゴリズム
  - プリムのアルゴリズムの考え方
  - 時間計算量 $O(m \log n)$
- 最短路問題
  - 最短路、最短路木
  - ダイクストラのアルゴリズム
  - アルゴリズムの直観的な理解