アルゴリズムとデータ構造

第13回　最小全域木 (1)

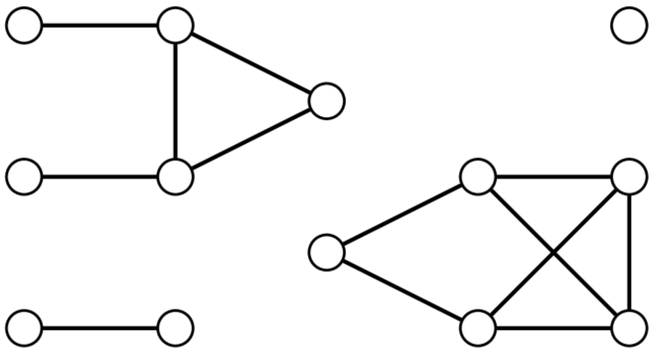
今日の内容

- さまざまな グラフ
  - 木、森
  - 全域木（スパニング木）
  - 最小全域木（最小スパニング木）
- 最小全域木を求めるアルゴリズム
  - クラスカルのアルゴリズム
  - アルゴリズムの高速化
  - アルゴリズムの正当性の証明
  - 時間計算量 $O(m \log n)$

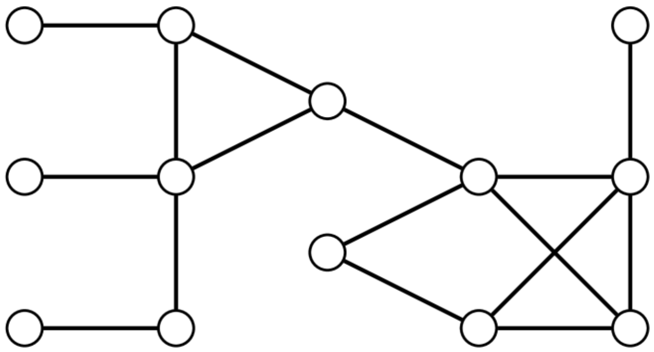
連結 / 非連結 (connected/disconnected)

前回の復習

- 無向グラフ $G = (V, E)$
- 定義： G が**連結**である
  $\Leftrightarrow$ V の任意の 2 つの頂点の間にパスが存在する
- G が連結でないとき、G は**非連結**であるという



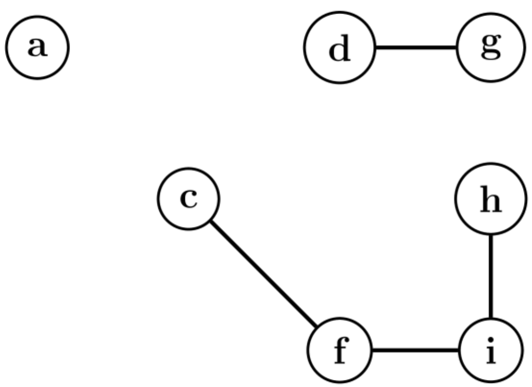
非連結



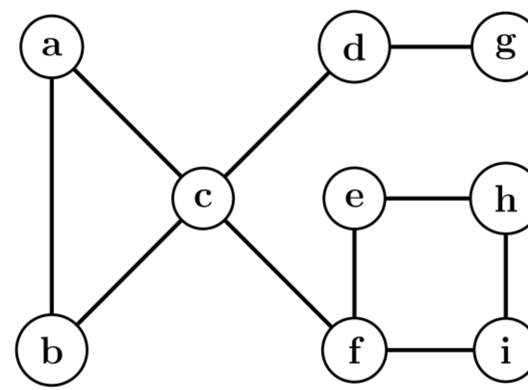
連結

部分グラフ (subgraph)

- 無向グラフ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$
- 定義： G_1 が G_2 の**部分グラフ**
  $\Leftrightarrow V_1 \subseteq V_2$ かつ $E_1 \subseteq E_2$



G_1

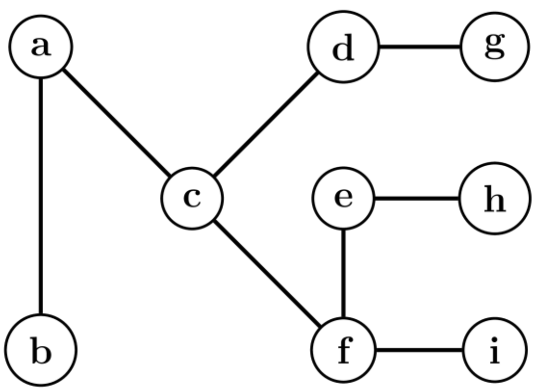


G_2

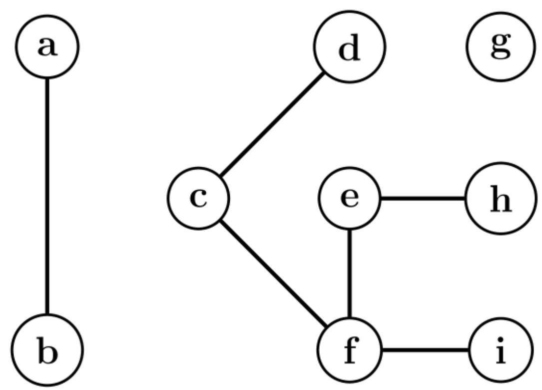
- 有向グラフについても、同様に定義される

木 (tree)

- 無向グラフ $G = (V, E)$
- 定義： 以下の 2 つの条件を満たす
  - G は連結である
  - G は閉路を部分グラフとして含まない

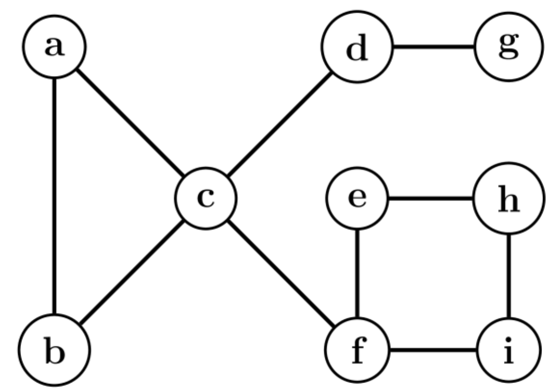


木



木でない

連結でない



木でない

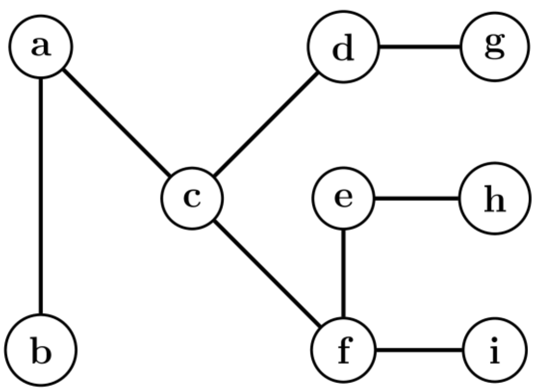
閉路を含む

Quiz： 頂点1つのグラフは？

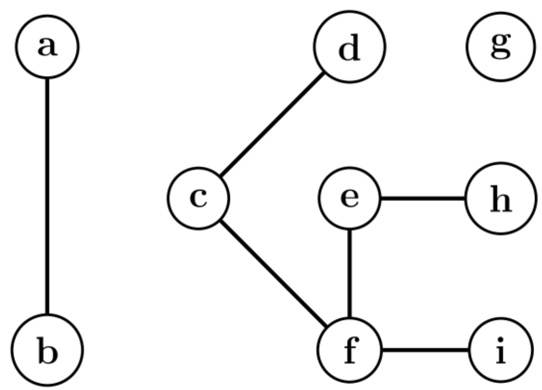
森 (forest)

- 無向グラフ $G = (V, E)$
- 定義： G は閉路を部分グラフとして含まない

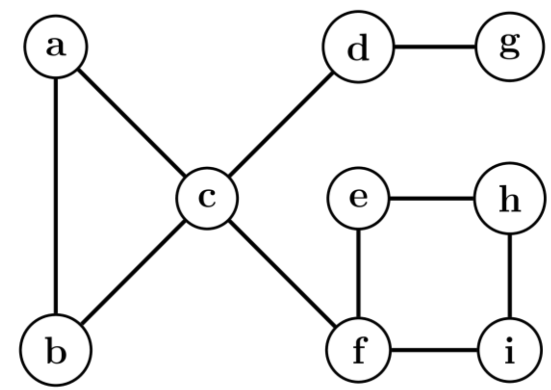
・連結でなくても ok
・直観的には、木が1つ以上集まったもの



森



森



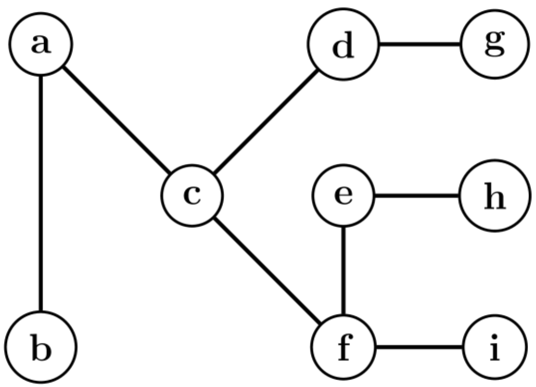
森でない

閉路を含む

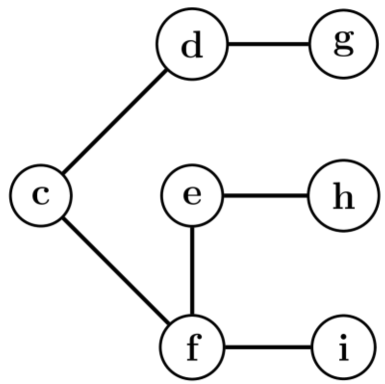
グラフ G に対する全域木 (spanning tree)

- 無向グラフ $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
- **定義**： G_1 が G_2 の**全域木 (スパニング木)** である
  $\Leftrightarrow$ G_1 は G_2 の全頂点を含む部分グラフで、木である

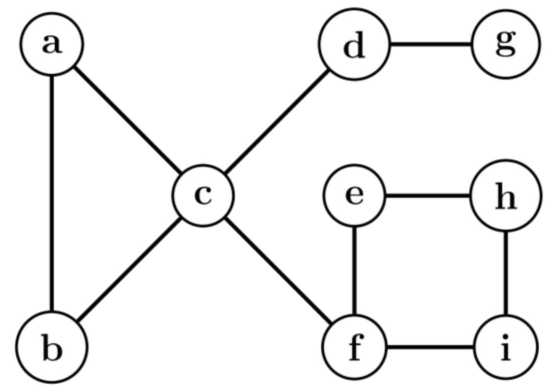
$V_1 = V_2$ かつ $E_1 \subseteq E_2$



G_1



G_1'



G_2

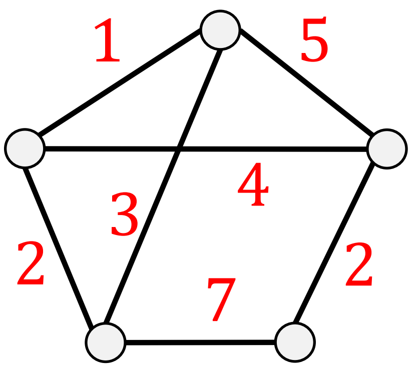
- G_1 は G_2 の全域木である
- G_1' は G_2 の全域木ではない

ネットワークGの最小全域木 (minimum spanning tree)

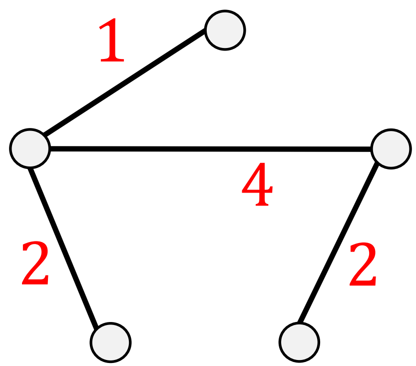
- ネットワーク (重み付きグラフ) $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$
- **定義**： G_1 が G_2 の**最小全域木** (**最小スパニング木**) である
  ⇔ G_1 は、G_2 の全域木で、
  含まれる**辺の重みの総和が最小のもの**

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

休憩

- ここで、少し休憩しましょう。

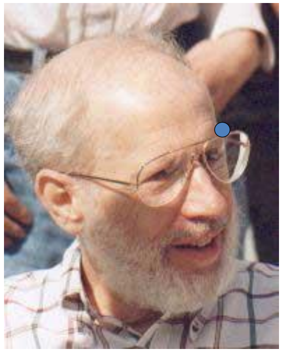
- 深呼吸したり、肩の力を抜いてから、
次のビデオに進んでください。

最小全域木 (MST) を求めるアルゴリズム

代表的なアルゴリズム

- **クラスカル** (Kruskal) のアルゴリズム

  1956年に Joseph Kruskal（ジョセフ･クラスカル）によって提案された。**重みが小さい辺から選択**し、**現在の解に追加**していくという方針で求める。



Joseph B. Kruskal
1928-2010, USA

- **プリム** (Prim) のアルゴリズム

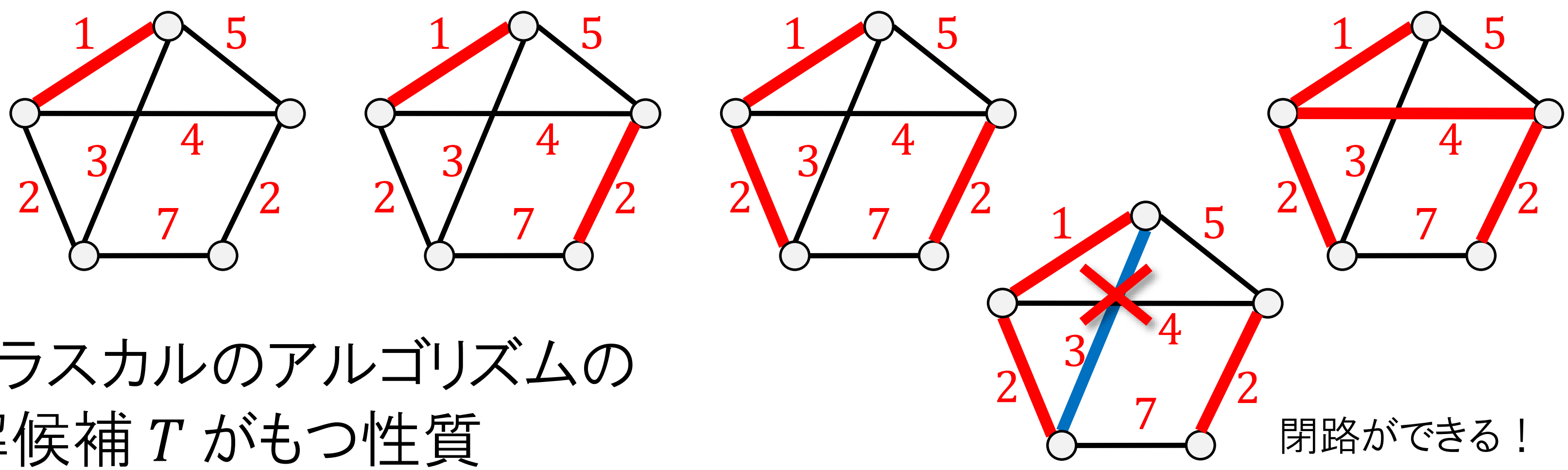
  1930年に Vojtěch Jarník（ヤルニーク）が開発。それとは独立に1957年に Prim により開発された。また、1959年には Dijkstra により再発見された。DJP 法、Jarník 法、Jarník-Prim 法とも呼ばれる。**最小全域木**が構成されている**連結成分の範囲を徐々に広げていく**という方針で求める。

Robert Clay Prim
1921-, USA

写真は https://alchetron.com/Joseph-Kruskal https://www.ithistory.org/honor-roll/dr-robert-clay-prim より

クラスカルのアルゴリズム

［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

クラスカルのアルゴリズム（疑似コード）

```
Procedure MST-Kruskal(G: グラフ, w: 重み)
  T ← φ;   // 解候補 T は、最初は空集合
  辺の配列 e を重みの小さい順 e[0], … , e[m − 1] に整列する;
  for i ← 0, 1, … , m − 1 do
      if (T ∪ {e[i]} が閉路を含まない) then
          T ← T ∪ {e[i]};
      end if
  end for
  最小全域木 T を出力する;
```

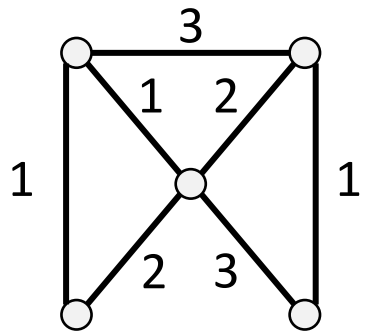
このチェックは
意外に難しい！

単純にやると、毎回 $O(n)$ 時間かかる
（for ループ全体では $O(mn)$ 時間）
（ヒント： $e[i]$の一方の頂点から
T を探索をする）

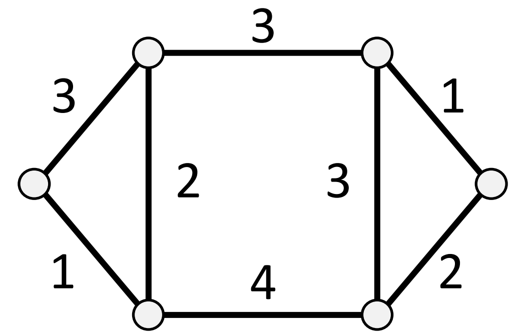
練習問題: 最小全域木を求めてみよう！

補足: 重みが同じ辺は、どの辺から選択しても ok

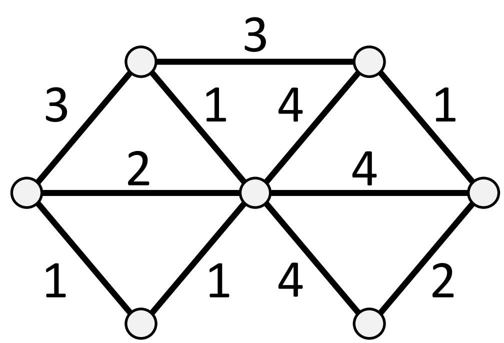
問1



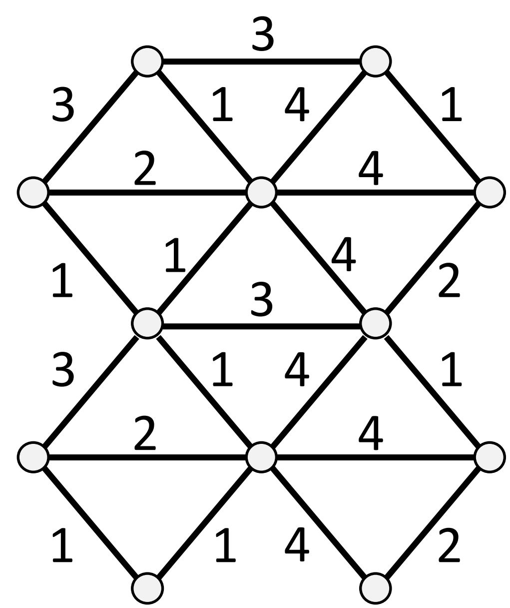
問2



問3



問4

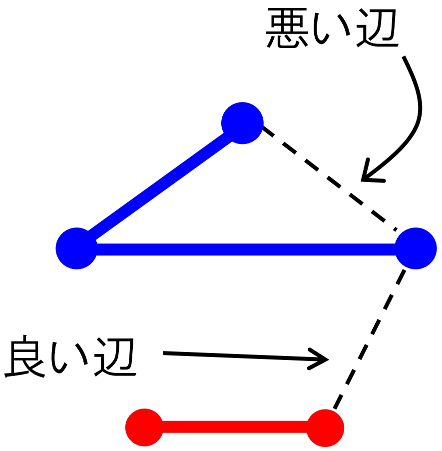


クラスカルのアルゴリズムの高速化

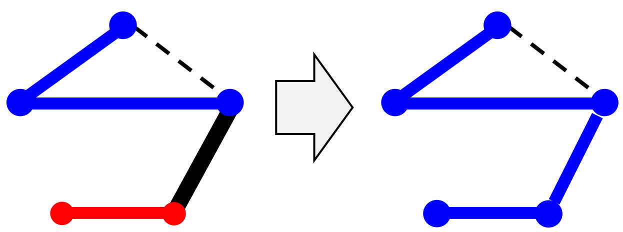
「$T \cup \{e[i]\}$ が閉路を含まない」のチェック

解決法

T の連結成分ごとに，頂点に異なる「色」を塗っておいて，新しく加えた辺 $e[i]$ の両端の色を比べることでチェックできる！

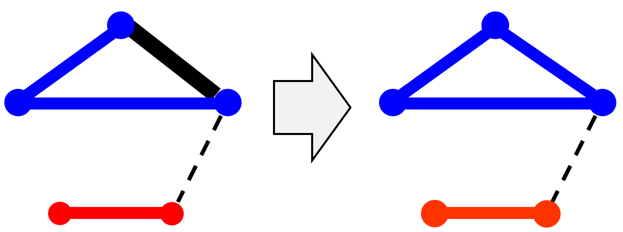


両端が異なる色のときは，辺を加えても閉路はできない



辺を加えたら色を塗り直す

両端が同じ色のときは，辺を加えると閉路ができる



単純に塗り直すと毎回 $O(n)$ 時間かかる

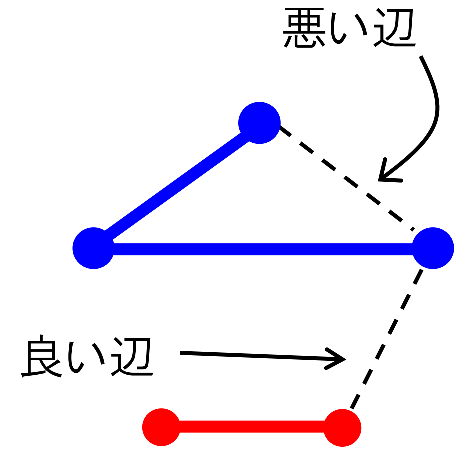
クラスカルのアルゴリズムの高速化

「$T \cup \{e[i]\}$ が閉路を含まない」のチェック

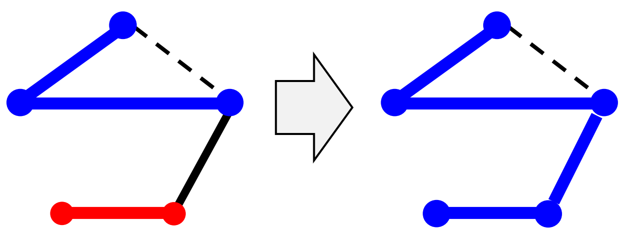
解決法

T の連結成分ごとに，頂点に異なる「色」を塗っておいて，新しく加えた辺 $e[i]$ の両端の色を比べることでチェックできる！

「色」のグループを表すのには，**Union-Findデータ構造**を使う

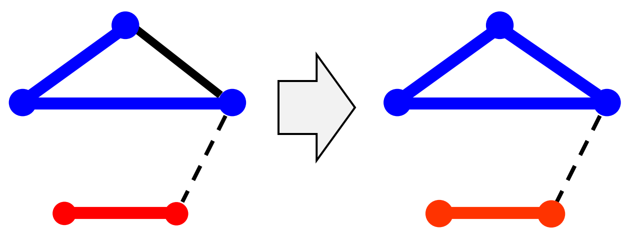


両端が異なる色のときは，辺を加えても閉路はできない



辺を加えたら色を塗り直す

両端が同じ色のときは，辺を加えると閉路ができる



単純に塗り直すと毎回 $O(n)$ 時間かかる

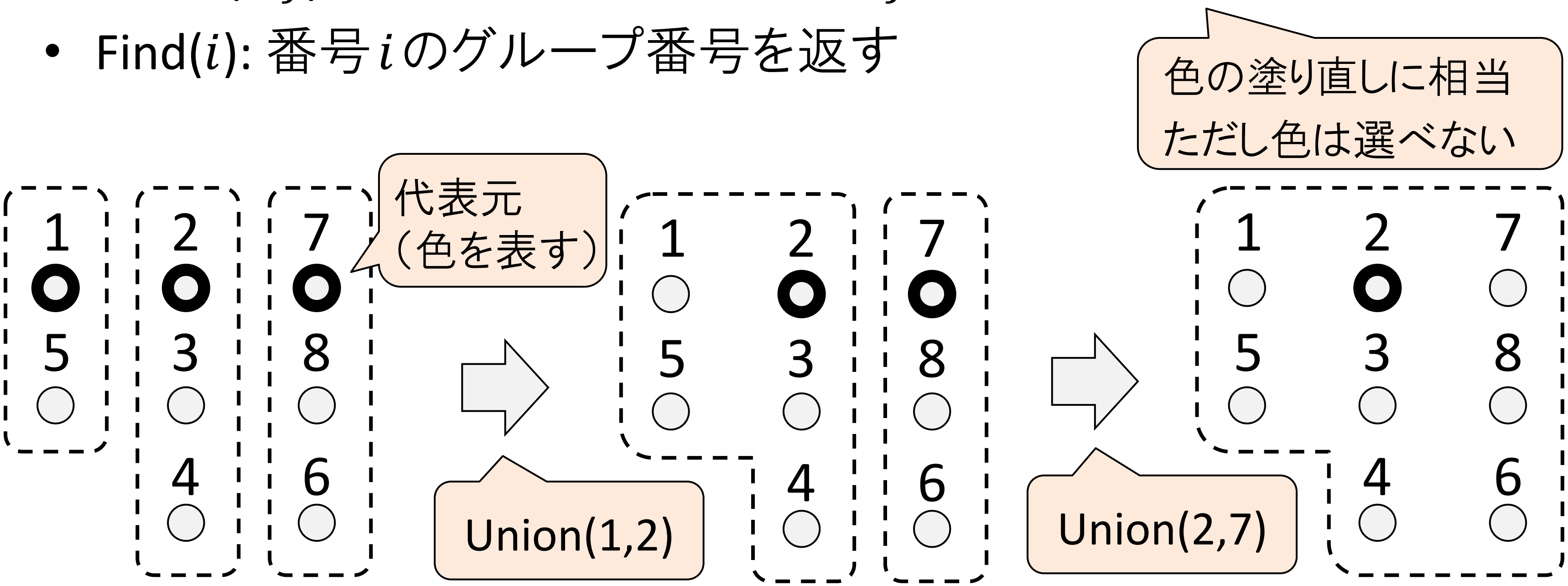
Union-Findで $O(\log n)$ 時間に

Union-Find データ構造

「**色**」(**グループ番号**)を持つグループ分けを管理するデータ構造

次の二つの演算を毎回 $O(\log n)$ 時間で実行できる

- Union(i, j): 番号 i のグループと番号 j のグループを合併する
- Find(i): 番号 i のグループ番号を返す



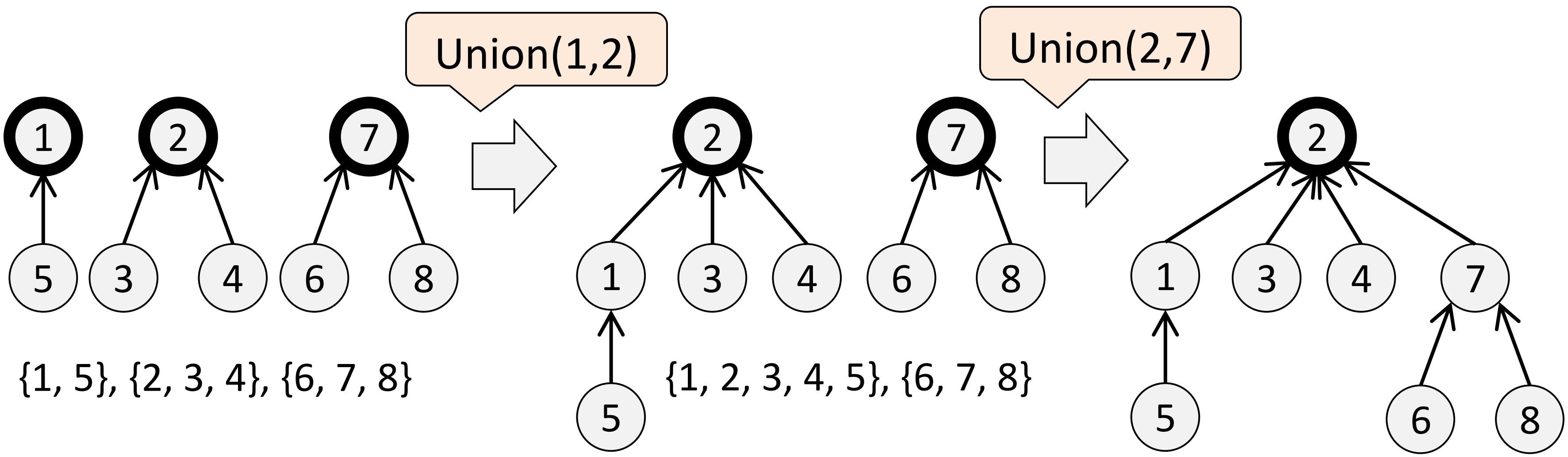
色 : 1, 2, 7	色 : 2, 7	色 : 2
Find(1) → 1	Find(1) → 2	Find(1) → 2
Find(4) → 2	Find(4) → 2	Find(4) → 2
Find(6) → 7	Find(6) → 7	Find(6) → 2

Union-Find データ構造の実現方法

集合のグループ分けを，**木の集まり(森)で表現**する

各グループは，要素番号を頂点ラベルとする一つの木に対応する

グループの色は，対応する木の根がもつ要素番号とする



Find(i):　番号 i の頂点から親をたどり，根の番号を返す

Union(i,j):　番号 i と番号 j が属す木をマージする．**ただし，高さが低い方の木の根を，高い方の木の根の子とする**（高さが同じ場合はどちらでもよい）

どちらも $O(\log n)$ 時間（nは要素数）

※ Find(i)の証明は，それほど自明ではない

クラスカルのアルゴリズム（疑似コード）

再掲
+ α

Procedure MST-Kruskal(G: グラフ, w: 重み)

$\boldsymbol{T \leftarrow \phi}$;　// 解候補 T は、最初は空集合

辺の配列 e を重みの小さい順$e[0], \ldots, e[m-1]$に整列する;

for $i \leftarrow 0, 1, \ldots, m-1$ do

　　if ($\boldsymbol{T \cup \{e[i]\}}$**が閉路を含まない**) then

　　　　$T \leftarrow T \cup \{e[i]\}$;

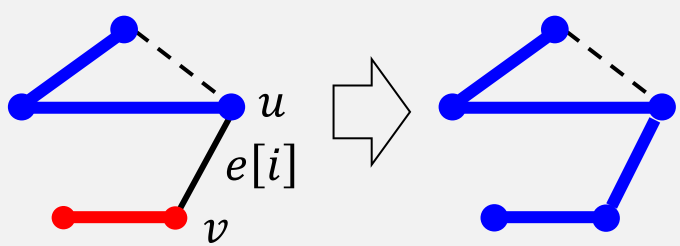
　　end if

end for

最小全域木 Tを出力する;

Union(u,v)

$e[i]$ の両端の頂点 u, v に対して Find(u) と Find(v) が異なる



クラスカルのアルゴリズム（疑似コード）

再掲
＋α

頂点数 n, 辺の本数 m のグラフ G に対して $\mathbf{O}(\boldsymbol{m}\log\boldsymbol{n})$ **時間**

$m \le n^2$ より
$\mathrm{O}(\log m) = \mathrm{O}(\log n)$

$\mathrm{O}(m\log m)$
$= \mathrm{O}(m\log n)$ 時間

$\mathrm{O}(1)$ 時間

MST-Kruskal(G: グラフ, w: 重み)

$\boldsymbol{T} \leftarrow \boldsymbol{\phi}$;   // 解候補 T は、最初は空集合

辺の配列 e を重みの小さい順$e[0], \dots, e[m-1]$に整列する;

for $i \leftarrow 0, 1, \dots, m-1$ do

　　if ($\boldsymbol{T} \cup \{\boldsymbol{e}[\boldsymbol{i}]\}$**が閉路を含まない**) then

　　　　$T \leftarrow T \cup \{e[i]\}$;

　　end if

end for

最小全域木 Tを出力する;

$e[i]$ の両端の頂点 u, v に対して
Find(u) と Find(v) が異なる

Union(u,v)

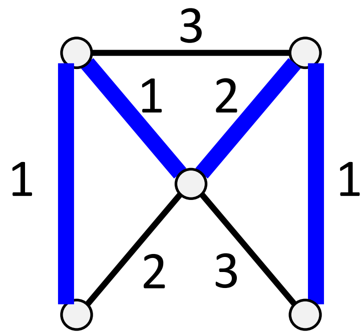
$\mathrm{O}(\log n)$ 時間

$\mathrm{O}(m\log n)$ 時間

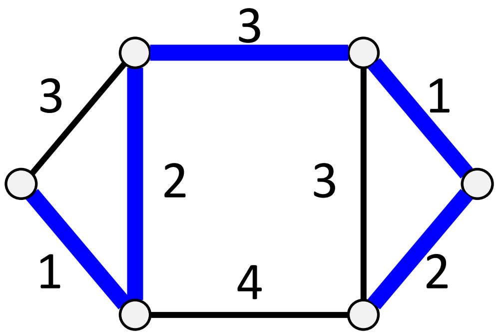
練習問題: 最小全域木を求めてみよう！

補足： 重みが同じ辺は、どの辺から選択しても ok

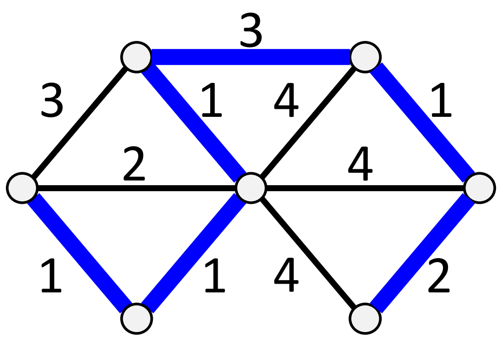
問1



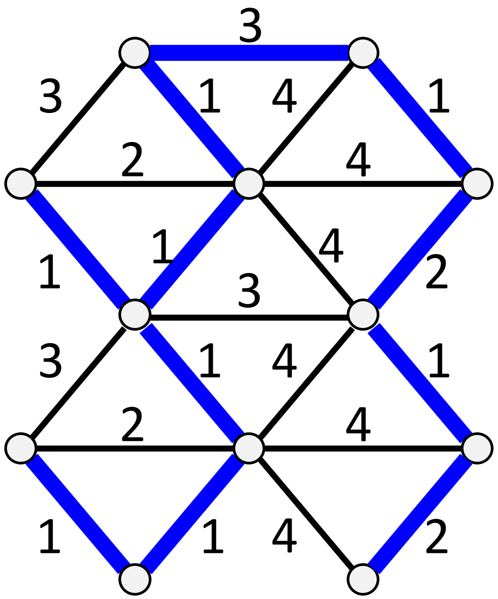
問2



問3



問4



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。

クラスカルのアルゴリズムのベースとなる補題

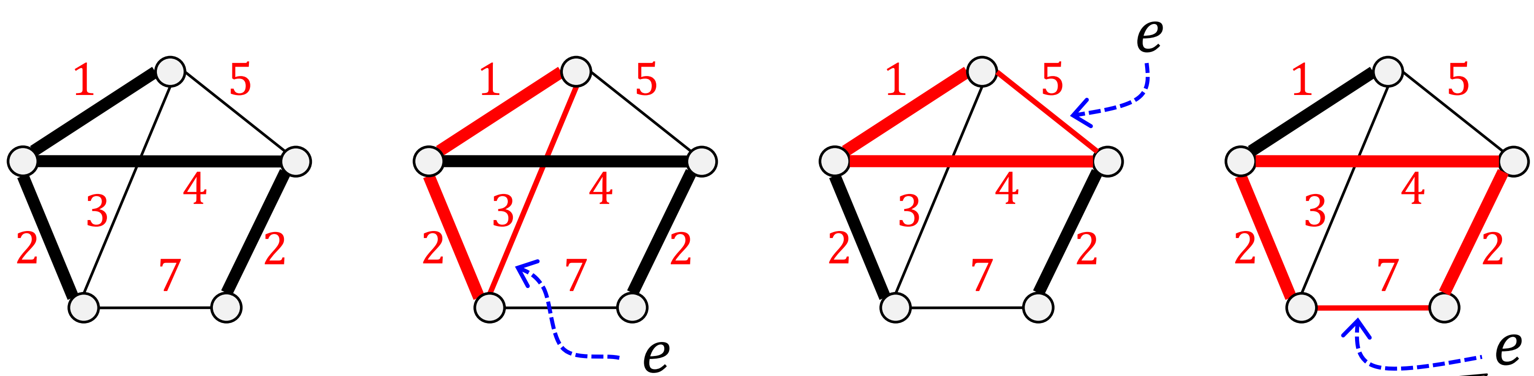
【補題】 (V,E)を連結無向グラフとし，$S \subseteq E$とする．このとき，以下の条件は，部分グラフ(V,S)が(V,E)の最小全域木であるための必要十分条件である．

条件： 部分グラフ(V,S)は(V,E)の全域木で，すべての$e \in E - S$に対して，

$$w(e) \geq w(e') \text{ for all } e' \in C_S(e)$$

$E - S$は差集合
（$E \setminus S$とも書く）

ただし，$w(e)$は辺eの重みを表し，$C_S(e)$は辺eとSの辺によって作られる閉路に含まれるSの辺集合を表す．



※ 太線：S，　細線：$E - S$，　赤色太線：$C_S(e)$

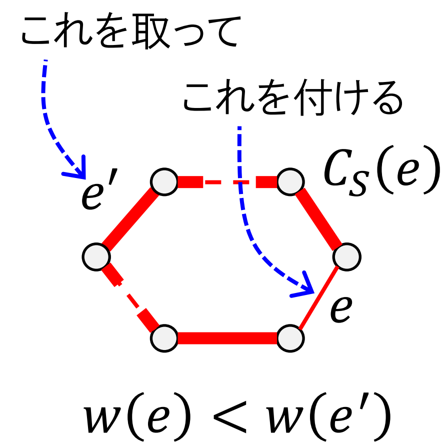
全域木に選ばれなかった辺 e は $C_S(e)$ の辺の中で最も重い

補題の証明（前半：⇒の証明）

(V,S) は最小全域木 ⇒ 任意の $e \in E - S$ に対し
「$w(e) \geq w(e')$ for all $e' \in C_S(e)$」

(V,S)が**最小全域木なのに条件を満たさないと仮定**する.

すると，仮定より，ある $e \in E - S$ が存在し，
ある $e' \in C_S(e)$ に対して $w(e) < w(e')$ が成り立つ.



$(V, S \cup \{e\} - \{e'\})$ は (V,E) の全域木であり，
(V,S) よりも辺の重みの和が小さい.
これは (V,S) が**最小全域木であることに矛盾**する.

よって，(V,S)が最小全域木であれば必ず条件を満たす.

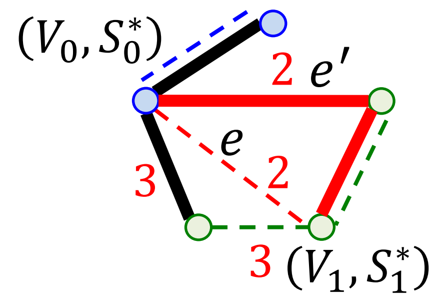
はいりほ～

補題の証明（後半：$\Leftarrow$の証明）

発展

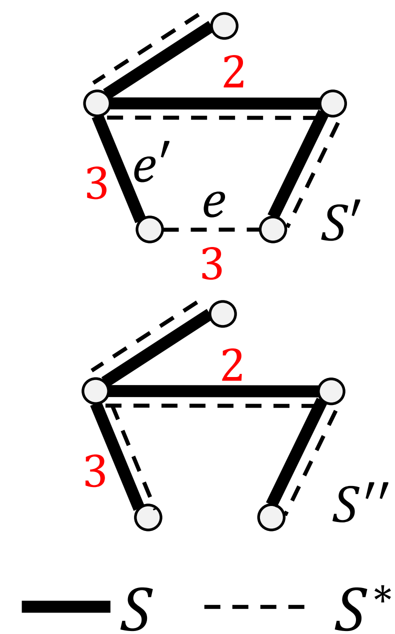
「(V,S)は最小全域木」$\Leftarrow$「$w(e) \geq w(e')$ for all $e' \in C_S(e)$」

最小全域木の一つを(V,S^*)とする. **条件を満たす(V,S)の辺の重みの和は (V,S^*) の辺の重みの和に等しい**ことを示す.



$e \in S^* - S$ が存在したとする. e により分かれる二つの連結成分を (V_0,S_0^*), (V_1,S_1^*) とすると, $e' \in C_S(e)$ で二つの連結成分 (V_0,S_0^*), (V_1,S_1^*) をまたぐ辺 $e' \in S$ が存在する.

(V,S) は条件を満たしており, $w(e) \geq w(e')$ が成り立つので, $\boldsymbol{S' = S^* \cup \{e'\} - \{e\}}$ とすれば, S' の重みの総和は S^* の重みの総和以下になる. S^* は最小全域木なので, **S' も S^* と重みの総和が等しい最小全域木であり, $|S^* - S| > |S' - S|$ を満たす**.
ただし, $|S|$ は集合 S の要素数を表すものとする.



したがって, S' を S^* として同じ操作を繰り返すと, 最終的には $S' = S$ となり, S も S^* と重みの総和が等しい最小全域木であることが示される. 【証明終わり】

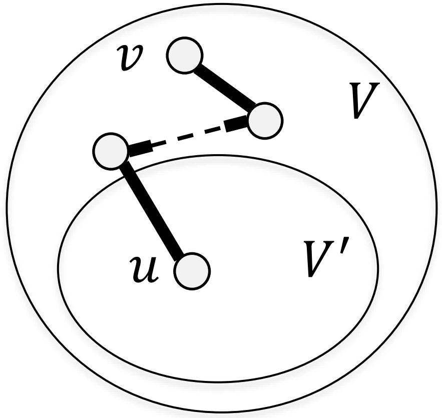
クラスカルのアルゴリズムの正当性

発展

（証明） クラスカルのアルゴリズムにより求めた辺の集合をS_0とする.
このとき，補題の条件が成り立っていることを示せばよい.

まず，**(V, S_0)が全域木であることを示す**.
アルゴリズムの動作より (V, S_0) は明らかに閉路を持たない.

いま，S_0 に含まれる辺の端点の集合を V' とし，
$V' = V$ を示す. **$V' \neq V$ と仮定すると**，ある頂点 $v \in V - V'$が存在する. (V, E) は連結であるので，ある頂点 $u \in V'$ が存在して，V' の頂点を経由しないで v へ到達する路が存在する. その路に属する辺を S_0 に加えても閉路はできない. これはアルゴリズムの「閉路ができない限り辺を加える」という動作に**矛盾する**.



よって $V' = V$ となり，(V, S_0) は全域木であることが示された.

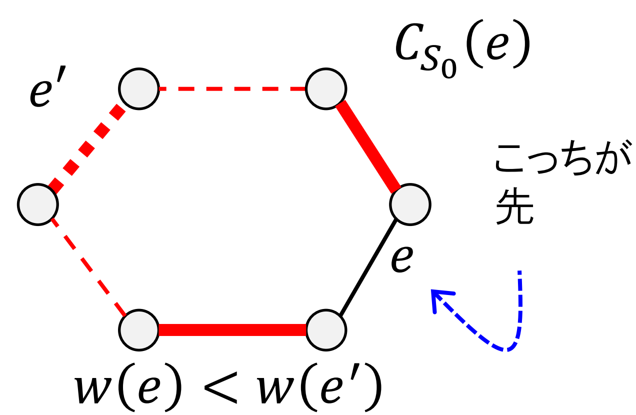
証明つづき

発展

次に，すべての $e \in E - S_0$，すべての $e' \in C_{S_0}(e)$ に対して $w(e) \geq w(e')$ が成り立つことを示す．

ある $e \in E - S_0$ が存在して，ある $e' \in C_{S_0}(e)$ に対して $w(e) < w(e')$ が成り立ったと仮定する．すると，MST-Kruskal アルゴリズムの４行目において $S \cup \{e\}$ が閉路を持つか否かチェックするときには，S にはまだ e' は含まれていないので，$S \cup \{e\}$ は閉路を持たないことになる．これは，S_0 が e を含んでいないことに**矛盾する**．

よって，補題の条件が成り立つ．したがって，(V, S_0) は最小全域木である．【証明終わり】



今日のまとめ

- さまざまな グラフ
  - 木、森
  - 全域木（スパニング木）
  - 最小全域木（最小スパニング木）
- 最小全域木を求めるアルゴリズム
  - クラスカルのアルゴリズム
  - アルゴリズムの高速化
  - アルゴリズムの正当性の証明
  - 時間計算量 $O(m \log n)$