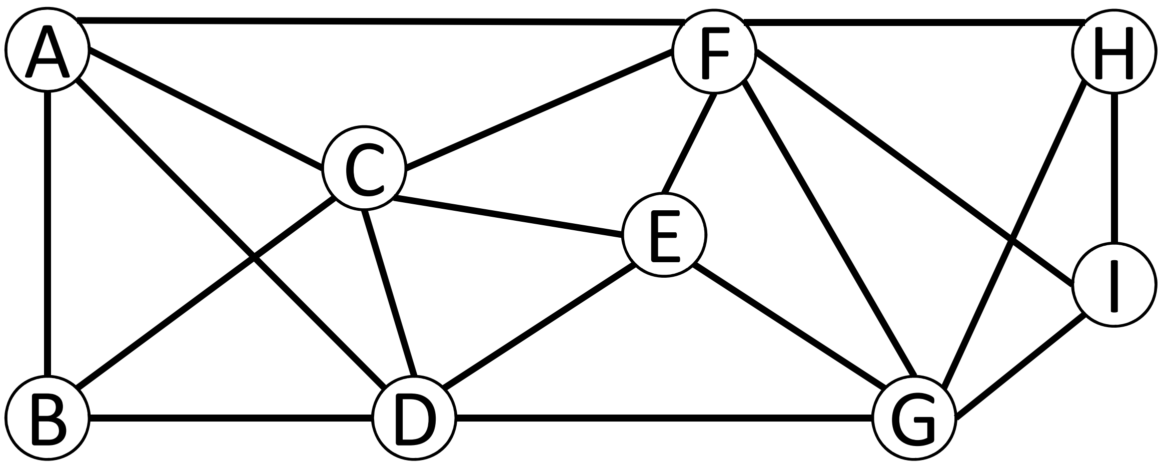
最適化基礎
第8回
最小全域木問題

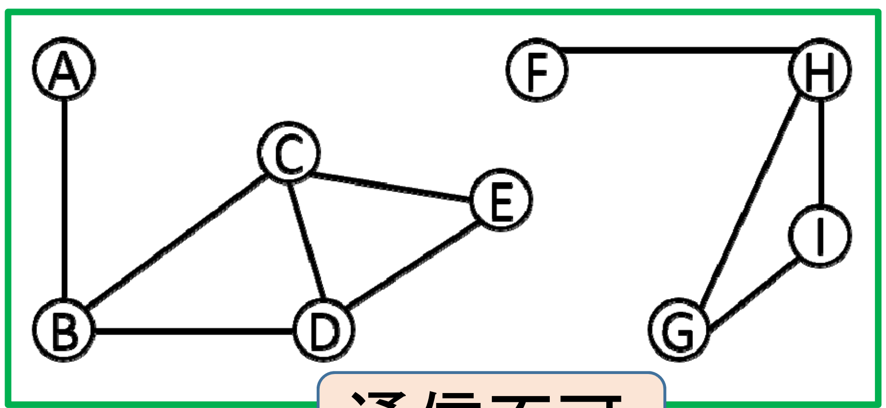
塩浦昭義
東京工業大学 社会工学専攻 准教授
shioura.a.aa@m.titech.ac.jp
http://www.soc.titech.ac.jp/~shioura/teaching

通信ネットワークの構築

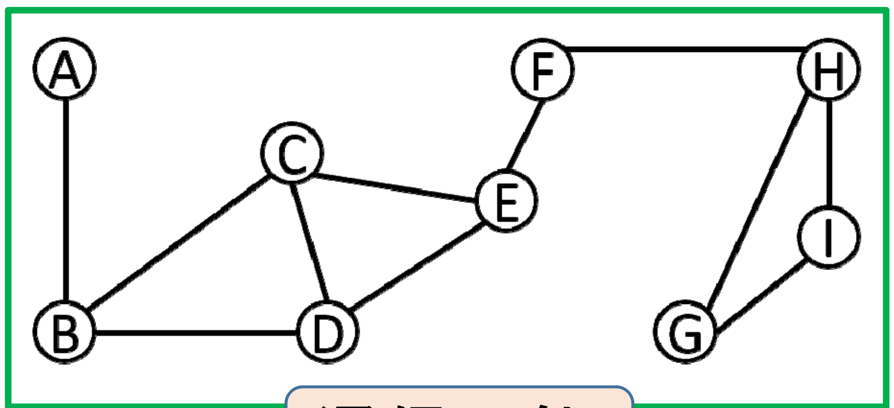
- 大学内の通信用ネットワークを構築したい
  - 地点 A, B, …, I をケーブルで接続，互いに通信したい
  - 直接ケーブルで接続できるところ，できないところがある
  - 複数のケーブルで繋がっていても可

接続可能な地点





通信不可

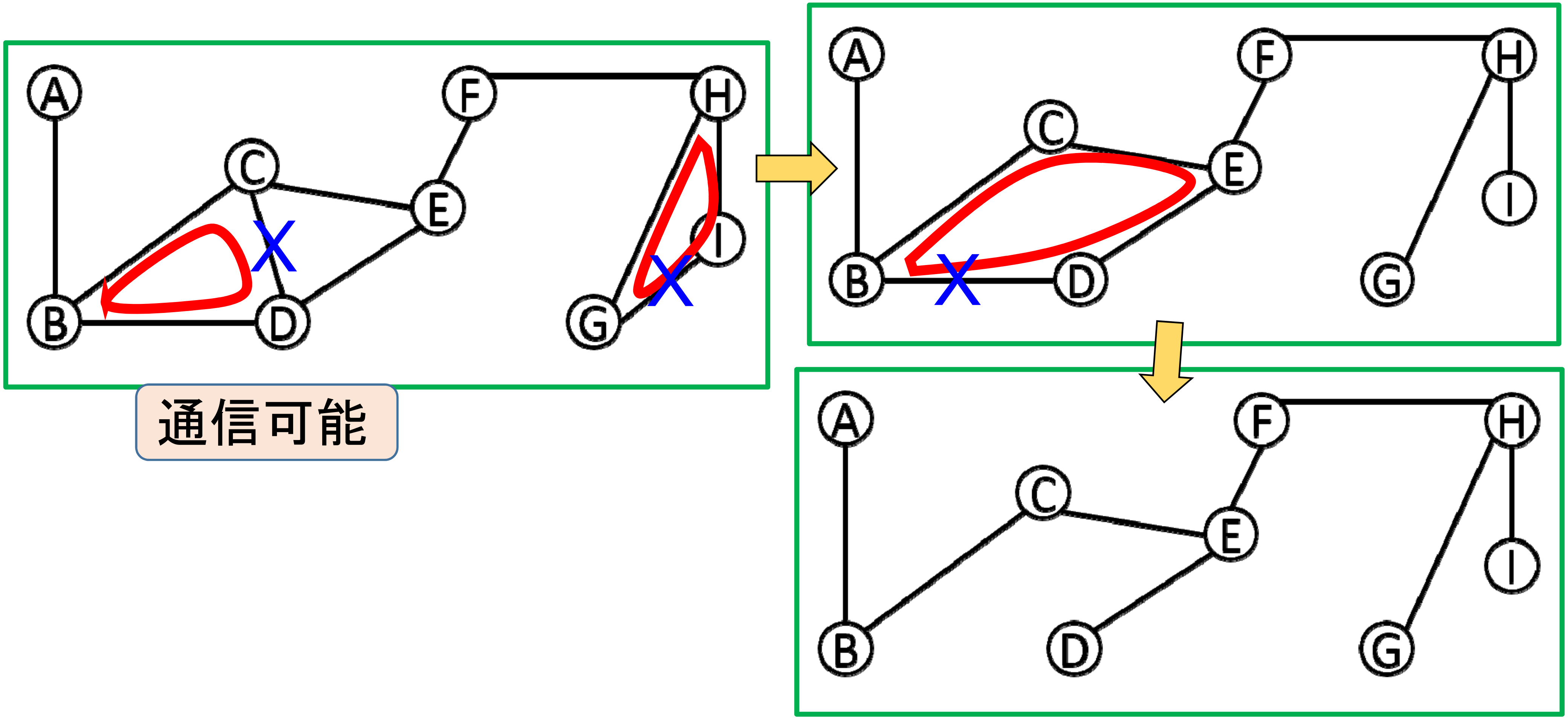


通信可能

無駄を省いたネットワーク

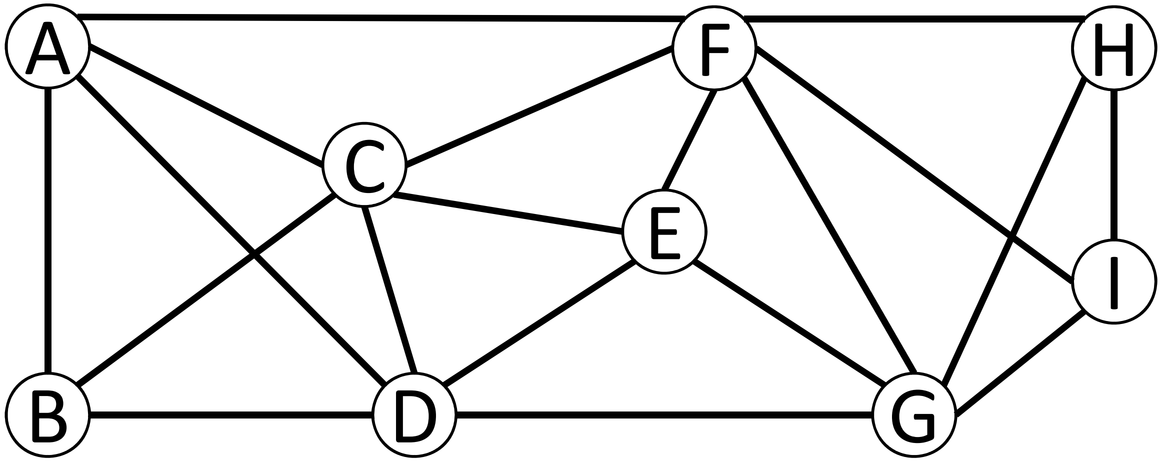
- 通信できればOK
  - 無駄なケーブルはできる限り省く
  - ケーブルで一周できる➔周上の各地点に2系統の通信路

    ➔ケーブルをひとつ削除しても通信可



グラフ

- 定義：グラフ＝「丸」を「線」で結んだ図
  - 頂点＝「丸」，枝＝「線」



グラフの例

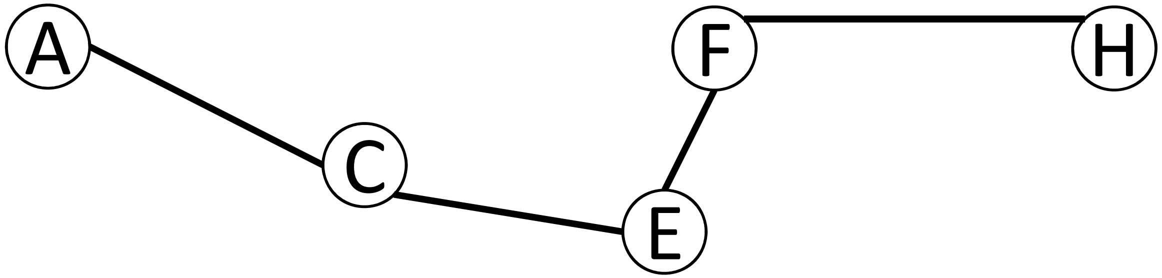
- 友人関係の図
- 鉄道路線図，道路網
- 組織図，家系図

※数学的には，
グラフ$G = (V, E)$は全頂点の集合Vと全枝の集合Eの対として表現
各枝$(u, v) \in E$は頂点の対として表現

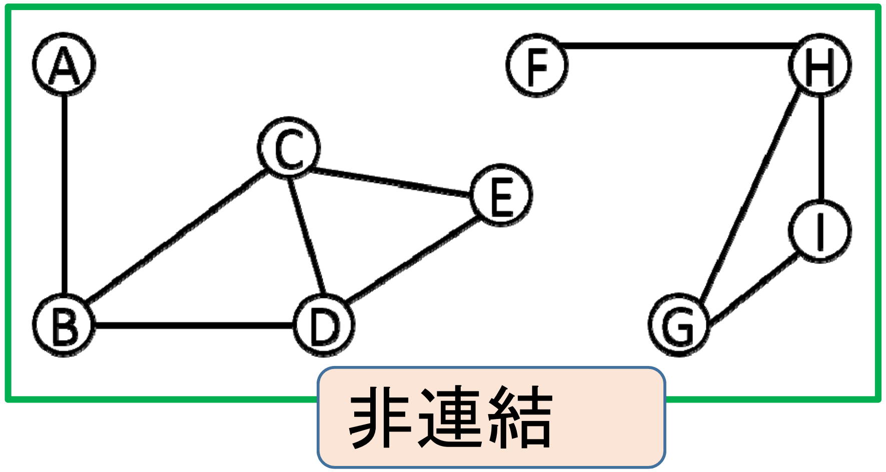
上記のグラフの場合，
すべての頂点の集合$V = \{A, B, C, \ldots, H, I\}$
すべての枝の集合$E = \{(A, B), (B, C), (C, D), (B, D) \ldots\}$

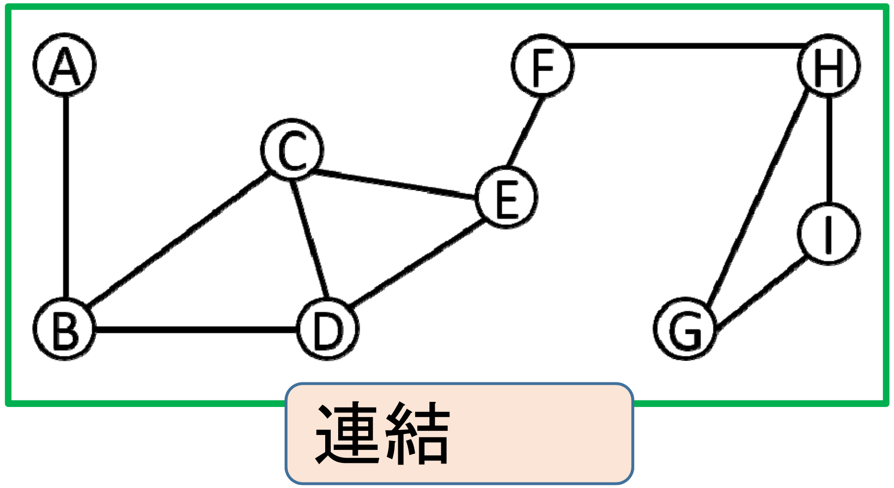
グラフの路と閉路

- 定義：路（みち）（path, パス）＝複数の枝が1つにつながったもの



- 定義：グラフが連結＝すべての頂点対の間に路が存在





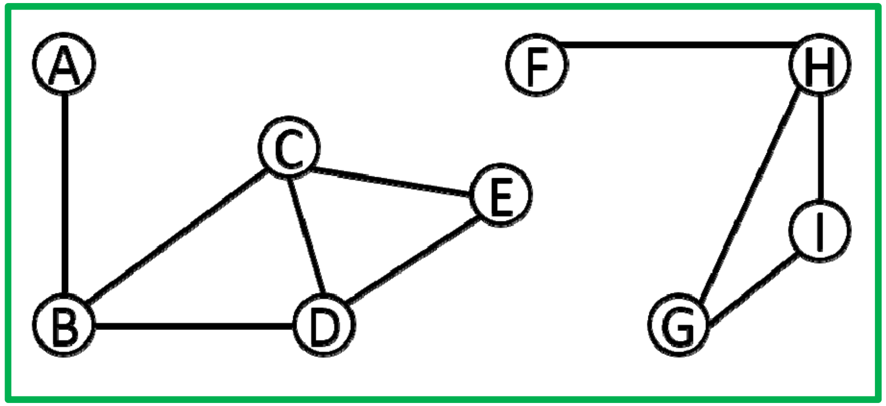
- 定義：閉路（cycle, サイクル）＝複数の枝が1つの輪になったもの

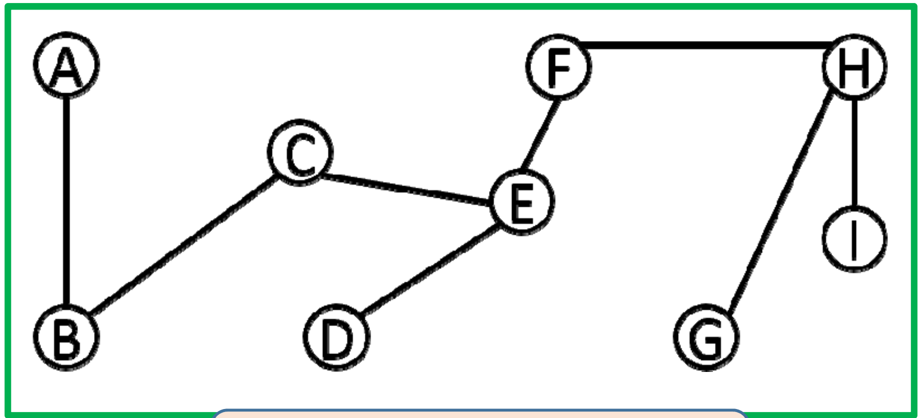
グラフの全域木

- グラフの全域木 --- 無駄のないネットワークのこと
- 定義 : 全域木＝次の条件を満たす枝の集合

  (i) 枝集合が連結

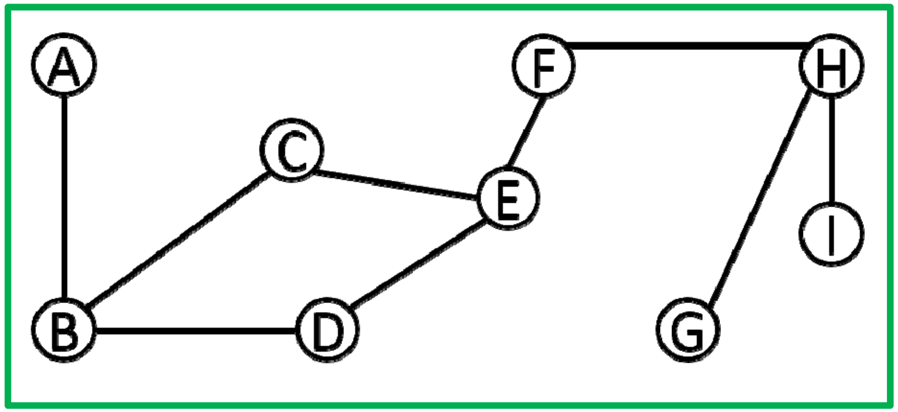
  (ii) 枝集合が閉路を含まない

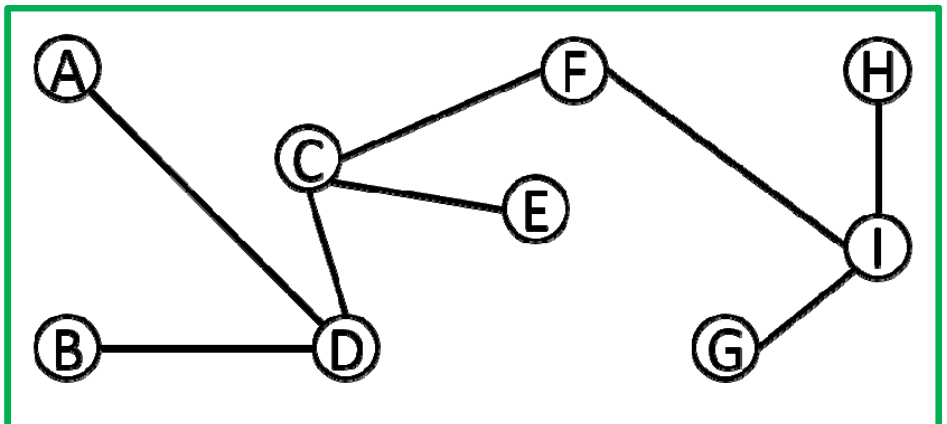




全域木ではない

全域木である

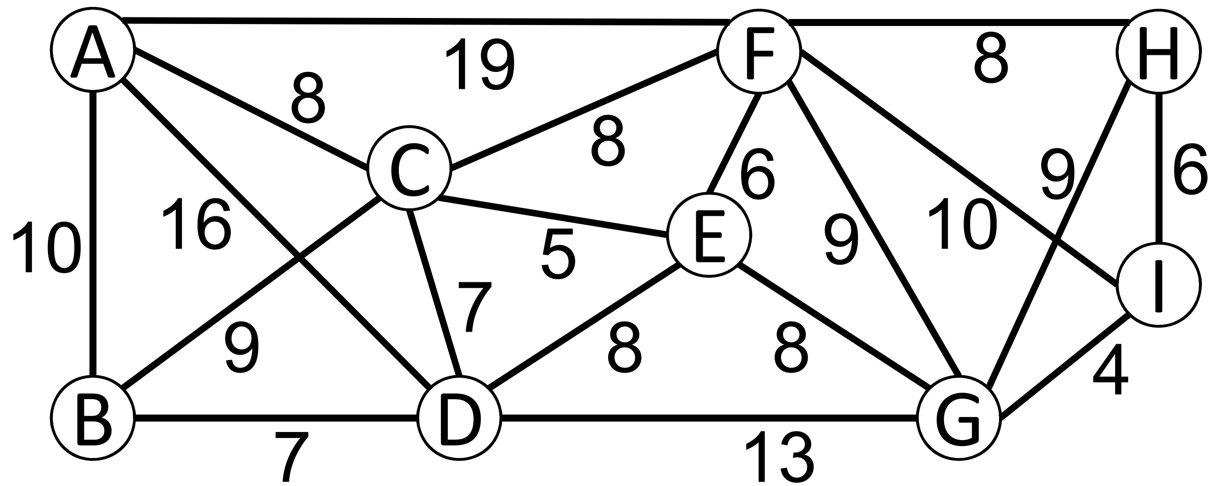




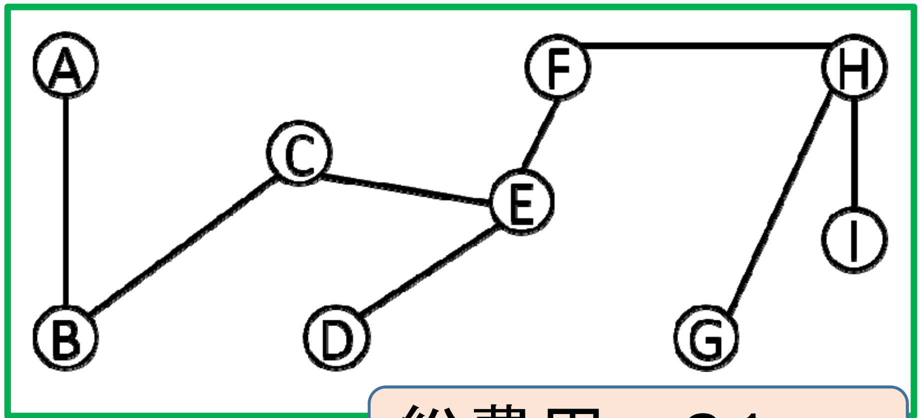
最小全域木問題

- 大学内の通信用ネットワークを構築したい
  - ケーブルの設置には費用が必要
  - 通信可能なネットワークを，出来るだけ最小費用でつくりたい

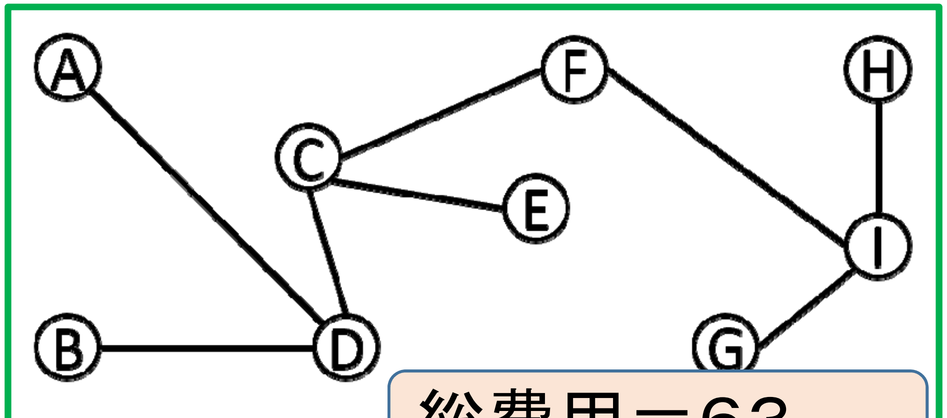
➔費用の和が最小の全域木を求める問題（最小全域木問題）



各ケーブルの設置費用



総費用＝61



総費用＝63

最小全域木を求めるアルゴリズム

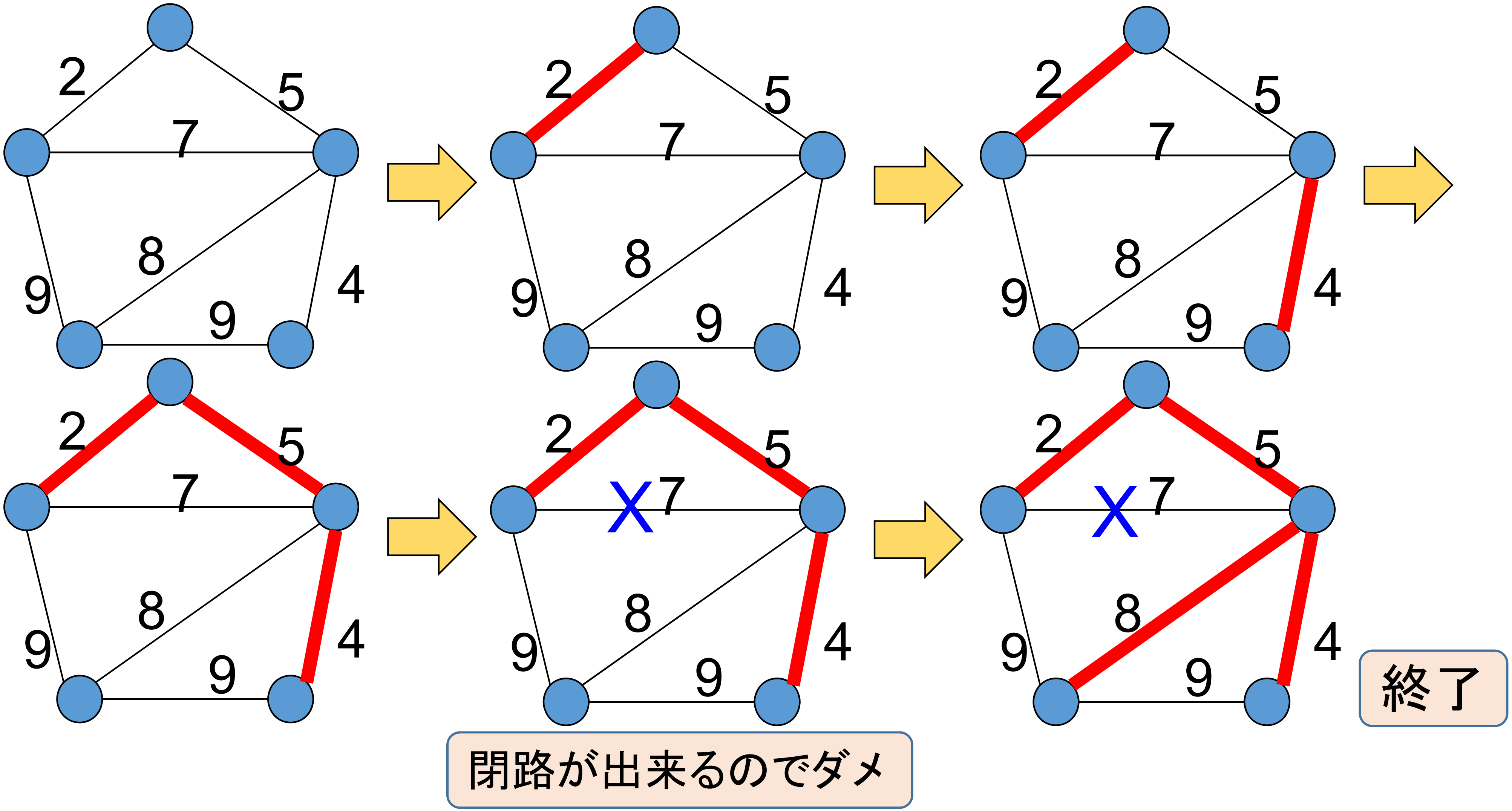
- 以下の2つが有名（それぞれ，提案した人の名前でよばれる）

- クラスカルのアルゴリズム
  - 費用の小さい順に枝を追加
  - ただし，閉路が出来る場合は追加しない

- プリムのアルゴリズム
  - 適当な頂点s を選び，固定
  - s を根として，木を徐々に成長させる.
  - その際，できるだけ費用の小さい枝を追加する

クラスカルのアルゴリズム

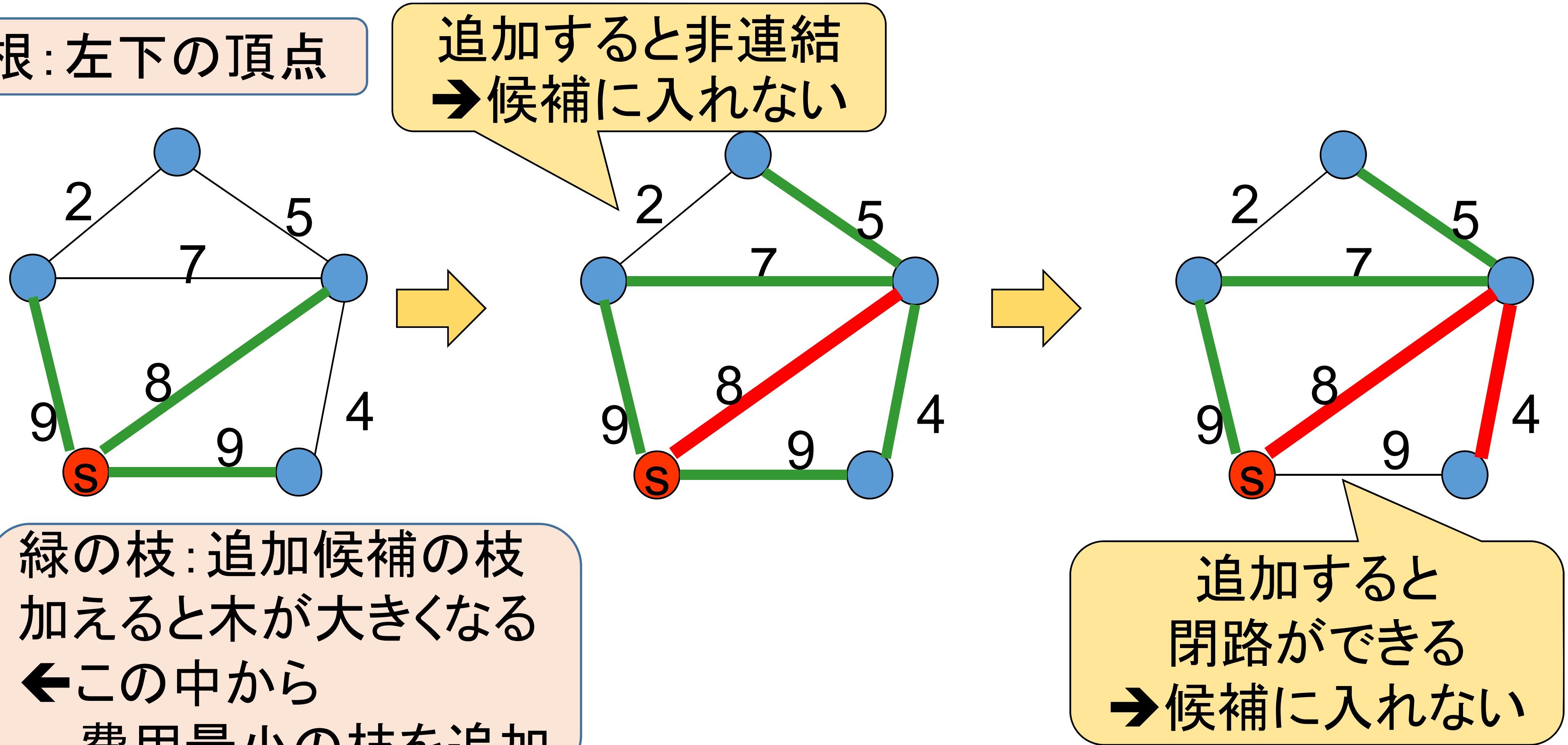
- 費用の小さい順に枝を追加
- ただし，閉路が出来る場合は追加しない



プリムのアルゴリズム

- 適当な頂点s を選び，固定
- s を根として，木を徐々に成長させる.
- その際，できるだけ費用の小さい枝を追加

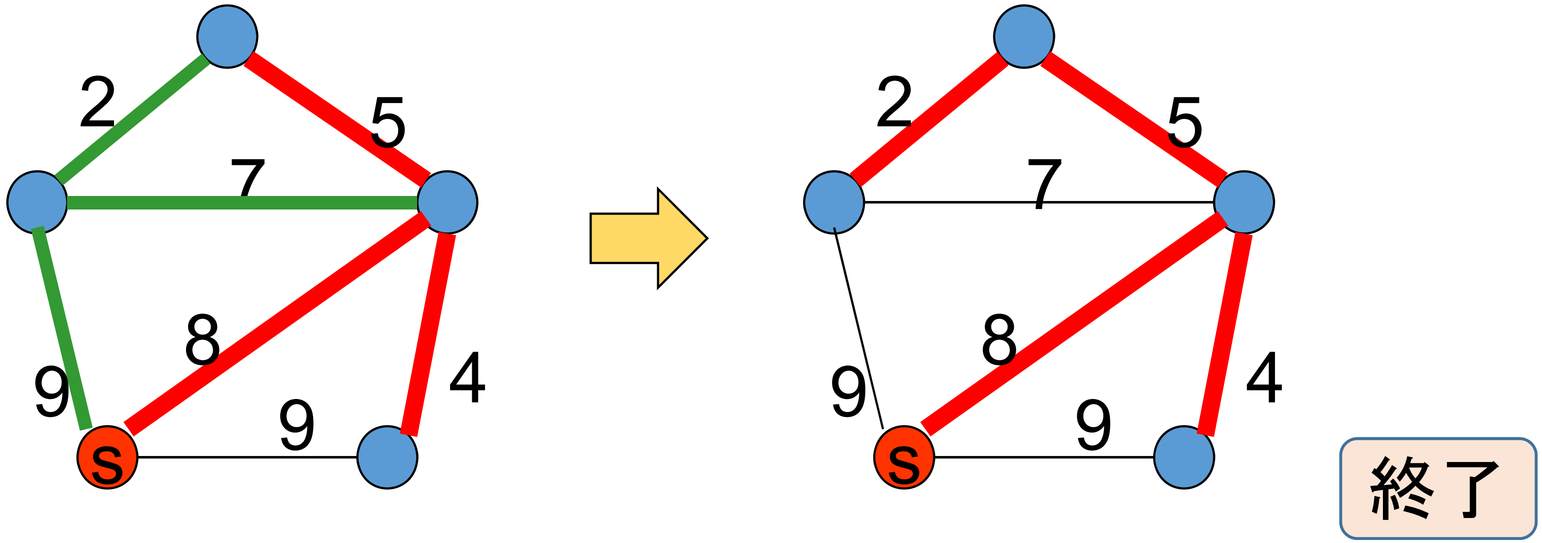
根：左下の頂点



緑の枝：追加候補の枝
加えると木が大きくなる
🡐この中から
費用最小の枝を追加

プリムのアルゴリズム

- 適当な頂点s を選び，固定
- s を根として，木を徐々に成長させる.
- その際，できるだけ費用の小さい枝を追加



アルゴリズムの正当性の証明

アルゴリズムの出力（最終的に求めた解）Ｔに対し，
以下を証明すればよい

- すべての全域木の中で，Ｔの費用が最小であること
- Ｔが全域木になること（🡸🡺閉路を含まない，連結）

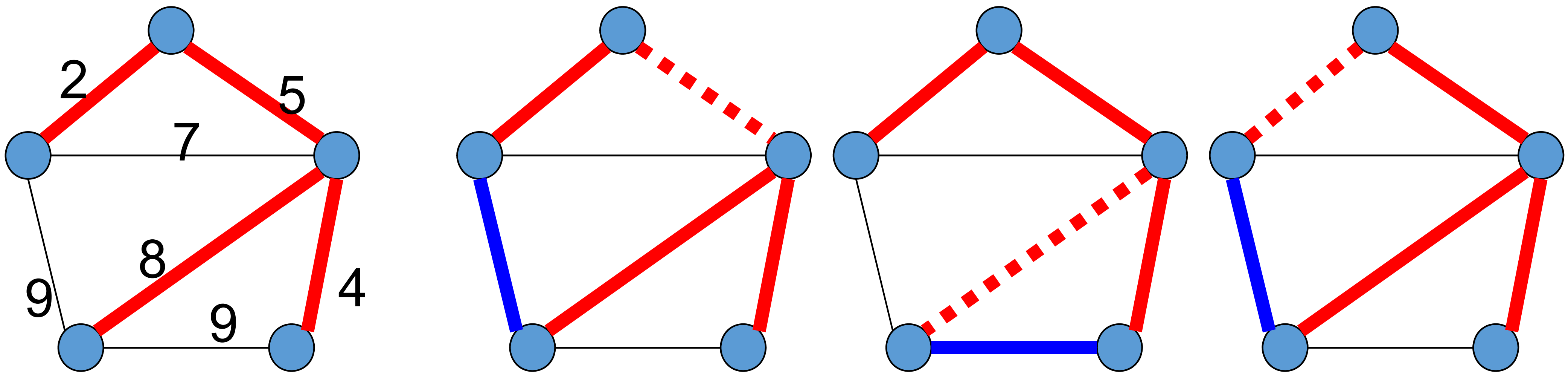
以下，クラスカルのアルゴリズムの正当性のみ証明

最小全域木の最適性条件

定理1:
全域木　T　は費用が最小(任意の全域木 T’ の費用≧Ｔの費用)
⇔Ｔの枝を一つだけ交換して得られる，任意の全域木 T’ の費用
≧　Ｔの費用

証明は後で

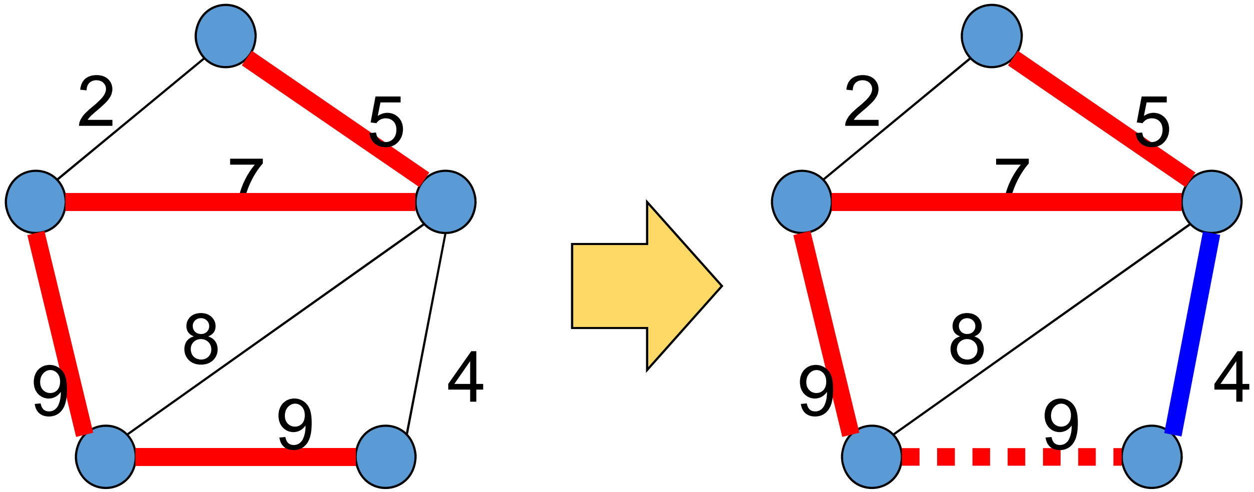
- 全域木の最適性をチェックするとき，
  すべての全域木と比較する必要なし(全域木の総数は最悪指数個)
- 枝を一つだけ交換して得られる全域木の総数
  ≦(全域木の枝数)×(全域木に含まれない枝数)



最小全域木の最適性条件

- 最適でないことのチェックも容易

定理1の対偶:
全域木　T　は費用が最小ではない
⟺ T の枝を一つだけ交換して得られる，ある全域木 T' の費用
＜　T の費用



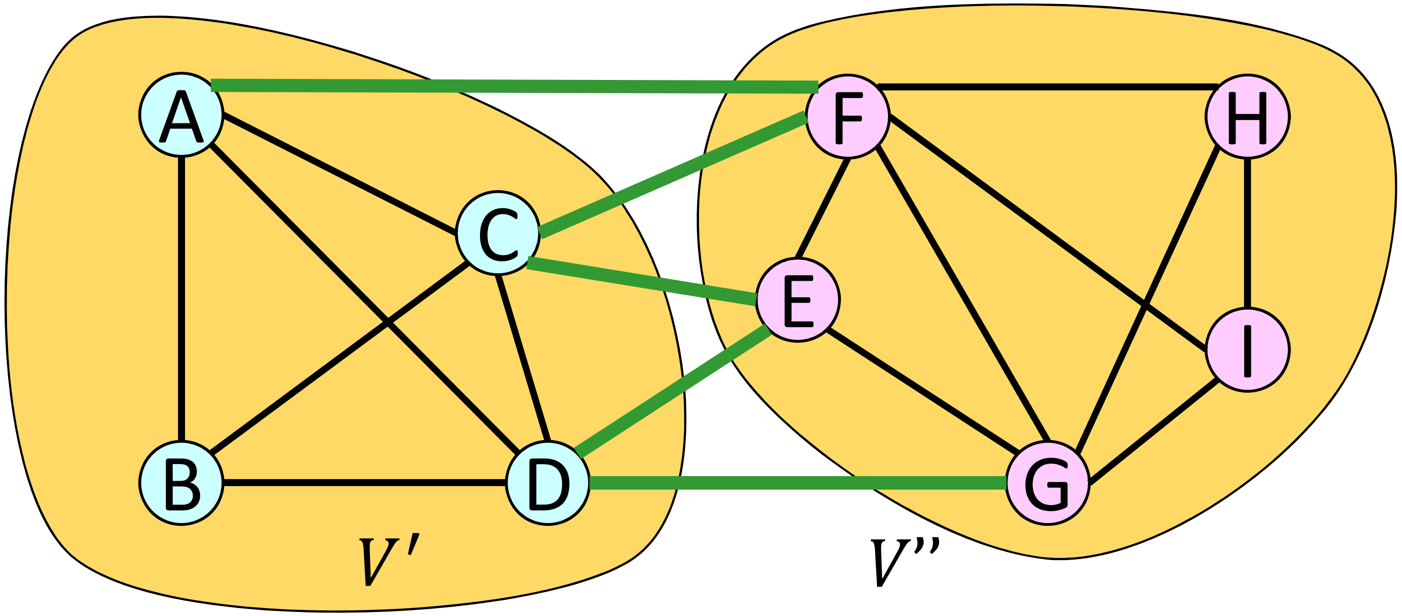
最小でない
全域木

枝を一つだけ交換
➔費用の小さい，新しい全域木が得られる

グラフのカットセット

- V', V'': グラフの頂点集合の分割(全ての頂点を2つに分けたもの)
- 定義 : V', V'' に関する**カットセット**$E(V', V'')$

　　　　$\Leftrightarrow$ V', V''を結ぶ枝全ての集合



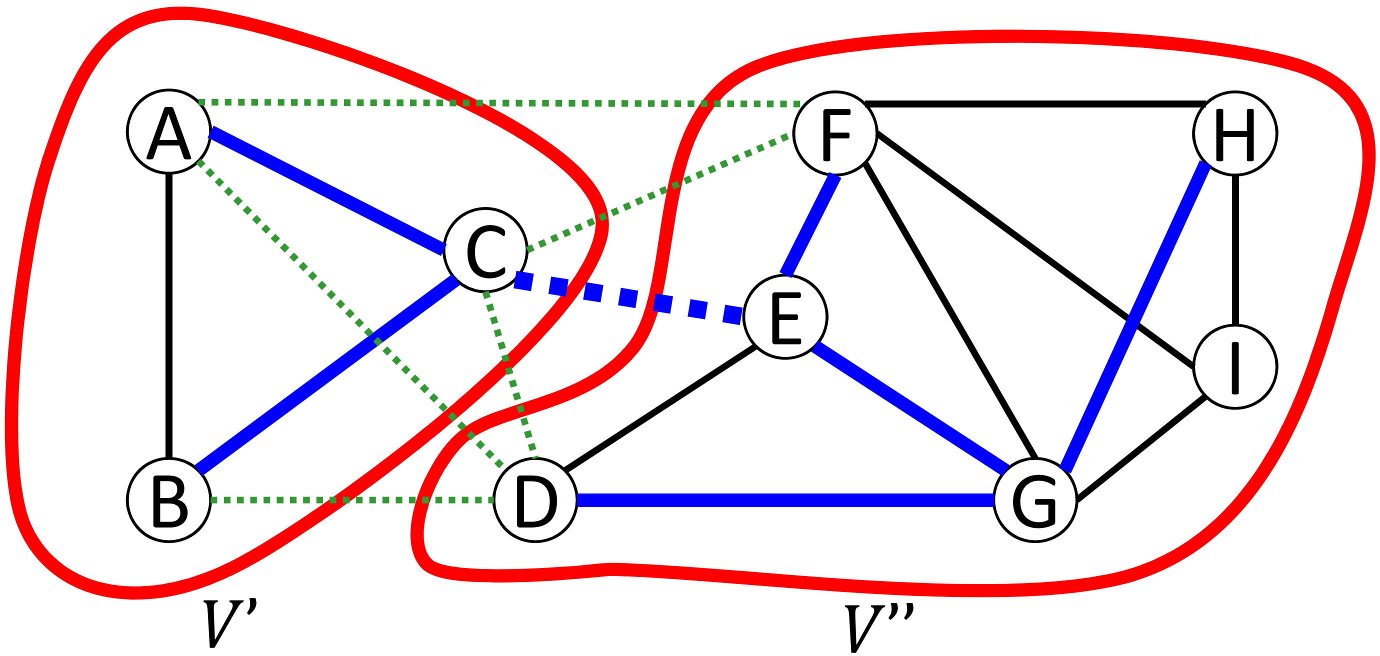
基本カットセット

- 全域木 T から，Tの枝 (u,v) をひとつ取り除く

➔全域木が2つに分かれる

➔対応する頂点集合V', V'' から，カットセットが得られる

（T と (u,v) に関する基本カットセット）



青い全域木（太い枝）と枝(C,E)に関する基本カットセット
＝点線の枝集合

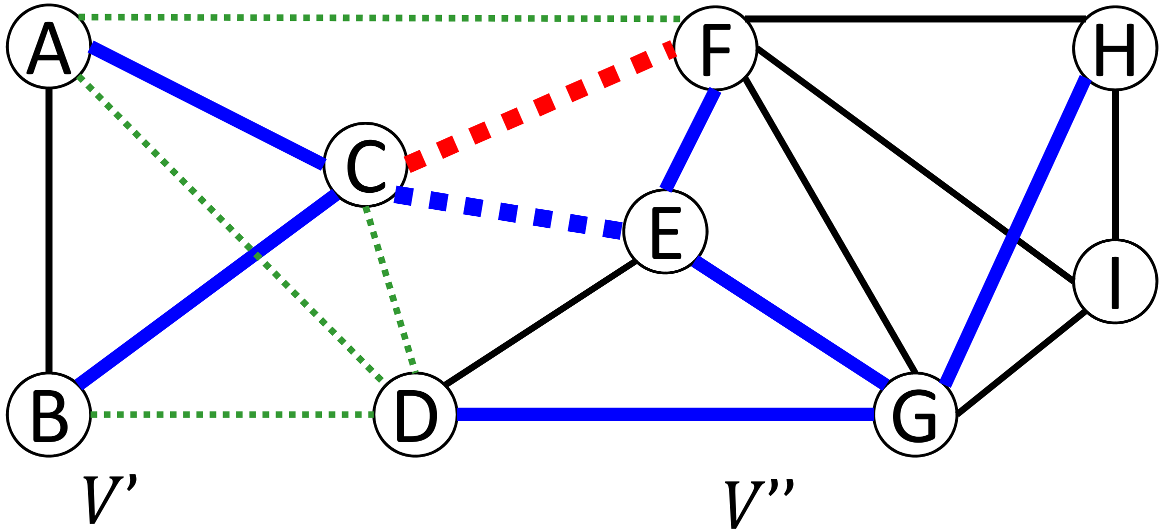
基本カットセットと全域木

性質：

全域木 T から，Tの枝 (u,v) をひとつ取り除く

➔T と (u,v) に関する基本カットセットの枝を一つ追加

➔新しい全域木が得られる



青い全域木（太い枝）から枝(C,E)を削除，(C,F)を追加

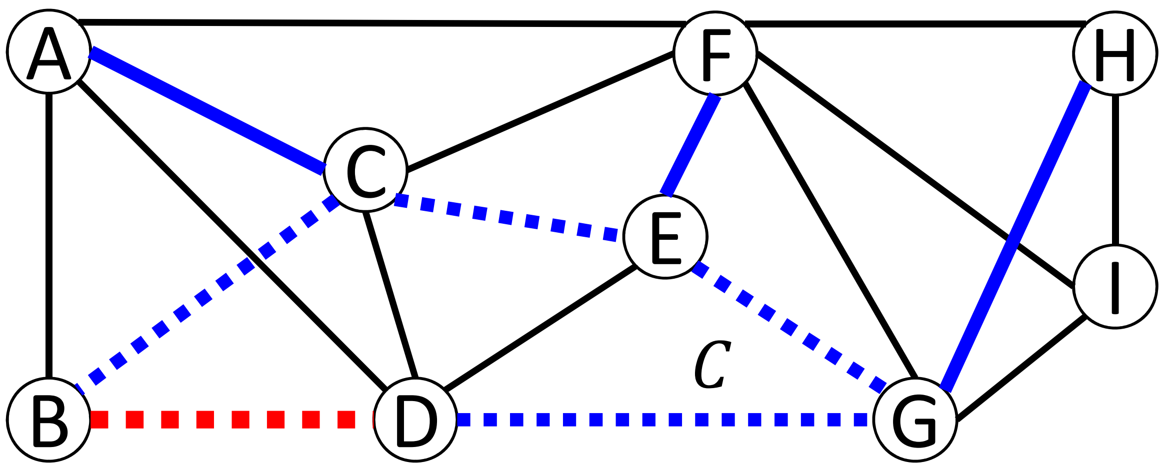
➔ 新しい全域木

基本閉路

- 全域木 T へ，Tに含まれない枝 (u,v) をひとつ追加

➔u と v の間には路が存在するので，閉路ができる

（T と (u,v) に関する基本閉路）



青い全域木（太い枝）と枝(B,D)に関する基本閉路

＝点線の枝集合

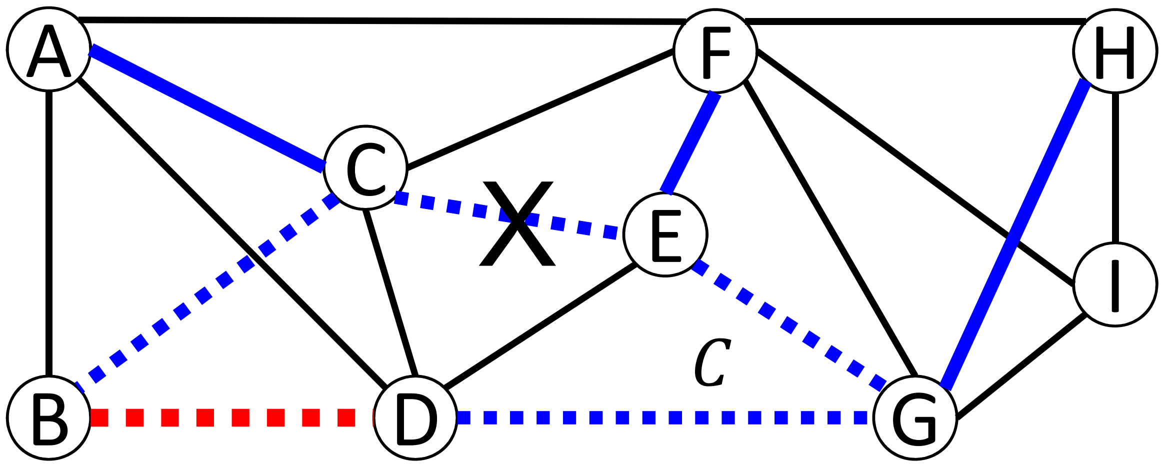
基本閉路と全域木

性質:

全域木 T へ，Tに含まれない枝 (u,v) をひとつ追加

➔ T と (u,v) に関する基本閉路の枝を一つ削除

➔ 新しい全域木が得られる



青い全域木（太い枝）に枝(B,D)を追加，(C,E)を削除

➔　新しい全域木

最小全域木の最適性条件の書き換え

- 最適性条件を，基本カットセットおよび基本閉路を使って書き換え

定理1：
全域木　T　は費用が最小（任意の全域木 T’ の費用≧Ｔの費用）
⇔ Ｔの枝を一つだけ交換して得られる，任意の全域木 T’ の費用
≧　Ｔの費用
⇔ [基本カットセットを使った条件]
Ｔの任意の枝 (u,v) と,
Ｔと(u,v)に関する基本カットセットの任意の枝 (u, v) に対し,
全域木 T – (u,v) + (s,t) の費用 ≧ Ｔの費用
⇔ [基本閉路を使った条件]
Ｔに含まれない任意の枝 (s,t) と,
Ｔと(s,t)に関する基本閉路の任意の枝 (s, t) に対し,
全域木 T + (s,t) – (u,v) の費用 ≧ Ｔの費用

クラスカルのアルゴリズムの正当性1：全域木の最適性

> 補題2：
> クラスカルのアルゴリズムが出力する全域木Ｔは費用が最小

証明　最適性条件件を利用

「Ｔに含まれない任意の枝 (s,t) と,

Ｔと(s,t)に関する基本閉路任意の枝 (u, v) に対し,

枝 (s,t) の費用 ≧ 枝 (u,v) の費用」　を示せば良い

- (s,t) はＴに含まれない➔ (s,t)を加えようとしたら閉路ができたから

（これが基本閉路）

- Ｔと(s,t)に関する基本閉路:

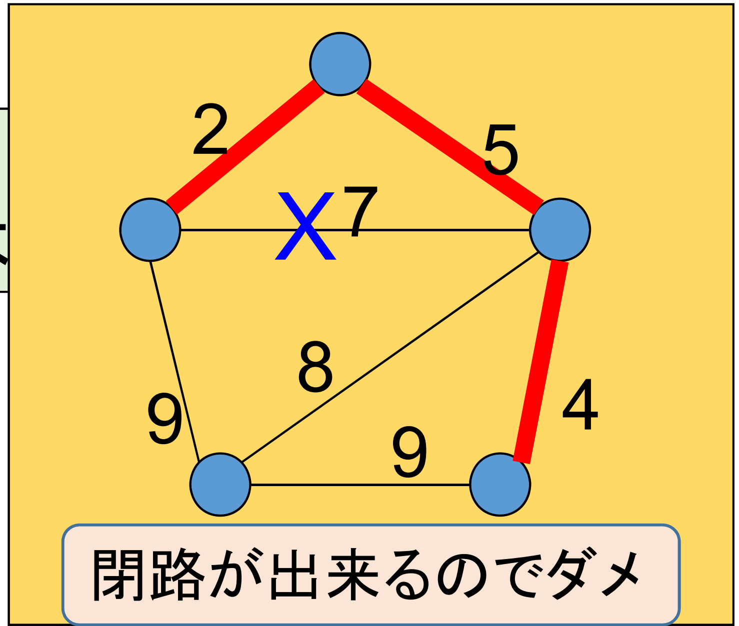
(s,t) より費用が小さい（または等しい）枝のみ含む

➔枝 (s,t) の費用 ≧ 枝 (u,v) の費用

■

クラスカルのアルゴリズムの正当性１：全域木の最適性

補題2：
クラスカルのアルゴリズムが出力する全域木



証明　最適性条件を利用

「T に含まれない任意の枝 (s,t) と,

　T と(s,t)に関する基本閉路任意の枝 (u, v)

　枝 (s,t) の費用 ≧ 枝 (u,v) の費用」　を示せば良い

- (s,t) は T に含まれない➔ (s,t)を加えようとしたら閉路ができたから

　（これが基本閉路）

- T と(s,t)に関する基本閉路:

　(s,t) より費用が小さい（または等しい）枝のみ含む

➔枝 (s,t) の費用 ≧ 枝 (u,v) の費用

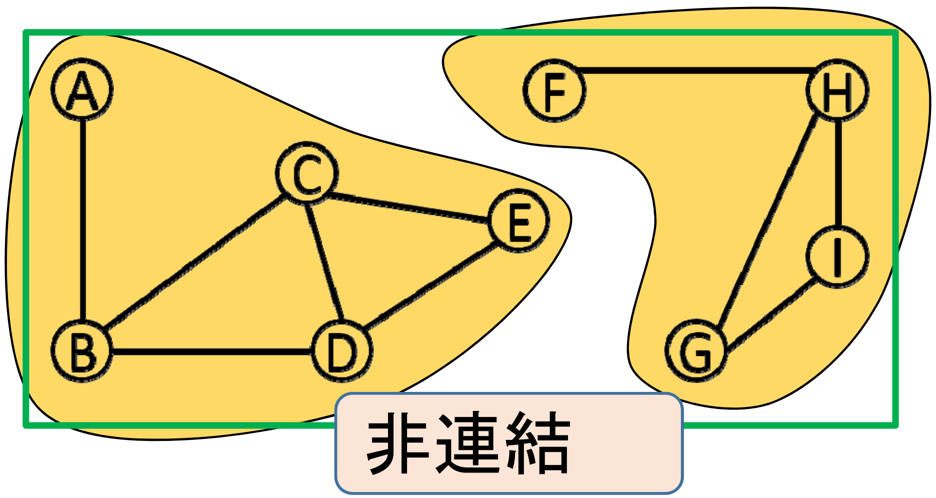
■

カットセットと連結性

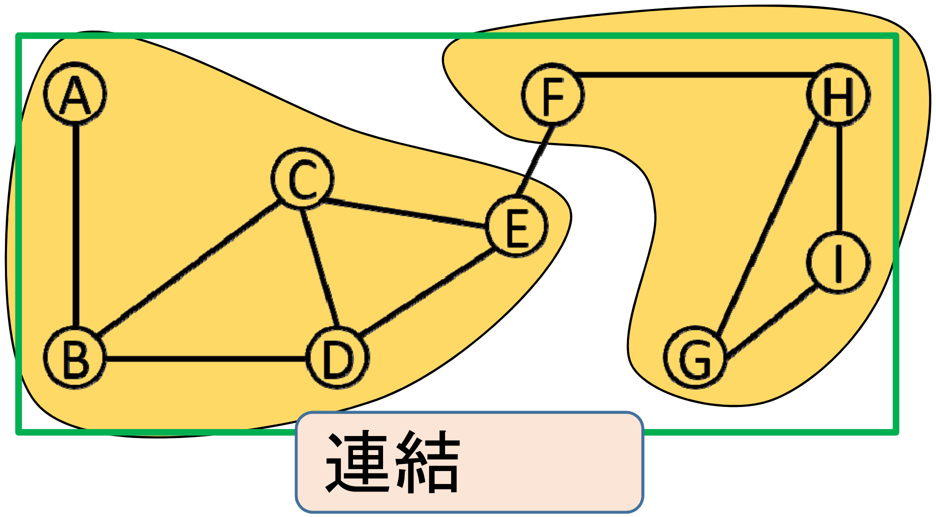
補題3：枝集合Tは連結
⟷任意のカットセットに対し，T はカットセットの枝を1つ以上含む

（証明の概略）
- 「あるカットセットの枝を含まない➔非連結」は自明
- 非連結➔ある頂点 u から到達できない頂点が存在
  V'= u から到達可能な頂点，V''=残りの頂点

➔ V', V'' に関するカットセットの枝を含まない ■



あるカットセットの枝を含まない



任意のカットセットの枝を含む

クラスカルのアルゴリズムの正当性2:出力は全域木

補題4:クラスカルのアルゴリズムの出力Tは全域木

証明

- T＝アルゴリズムの出力の枝集合
- 定義より，「Tは閉路を含まない，連結」を示せばOK
- 「Tは閉路を含まない」は枝の追加ルールより成立
- 「Tは連結」の証明:補題3より，

  任意のカットセットの枝を含めばOK

  - 任意のカットセット$E(V',V'')$を考える
  - その中で費用最小の枝(u,v)に注目
  - アルゴリズムで(u,v)を追加しようとするとき，閉路は生じない

    （∵閉路には，カットセット$E(V',V'')$の枝がもう1本以上必要）

    ➔枝(u,v)は必ず追加される

    ➔カットセットの枝を1つ以上含む ■

定理1の証明の準備

補題5：2つの異なる全域木T, T’ および枝 $(u, v) \in T - T'$ に対し，
ある枝$(s, t) \in T' - T$ が存在して，
T – (u,v)+(s,t) および T’ + (u,v) – (s,t) はともに全域木

証明は省略
（基本閉路と基本カットの性質を利用）

補題6：2つの異なる全域木T, T’ の枝の数は等しい

補題5より導かれる

定理1の証明

- 対偶を証明

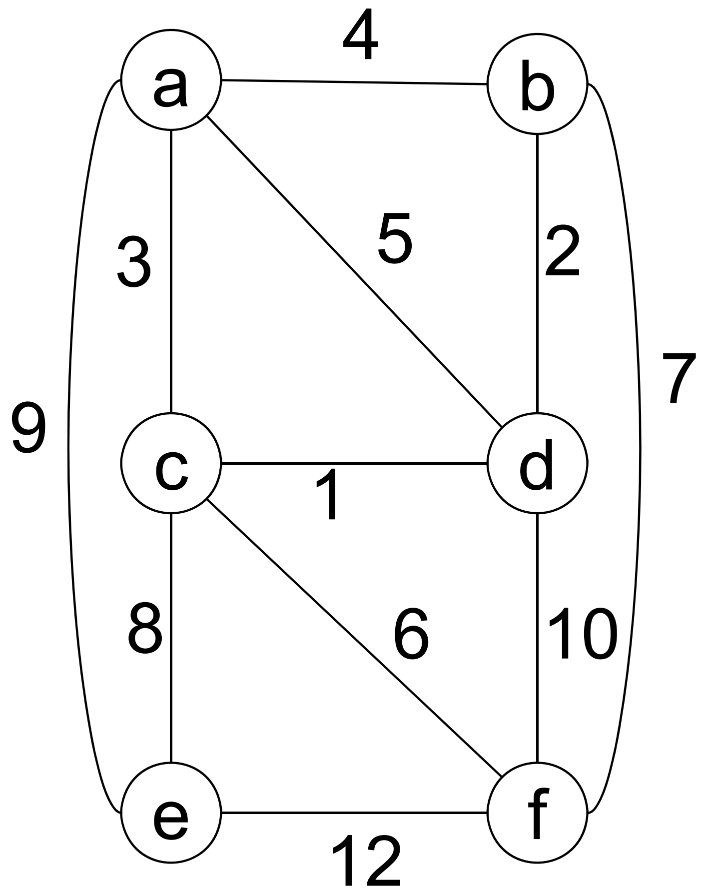
定理1の対偶:
全域木　T　は費用が最小ではない
⇔ (＊) T の枝を一つだけ交換して得られる，ある全域木 T′ の費用 ＜ T の費用

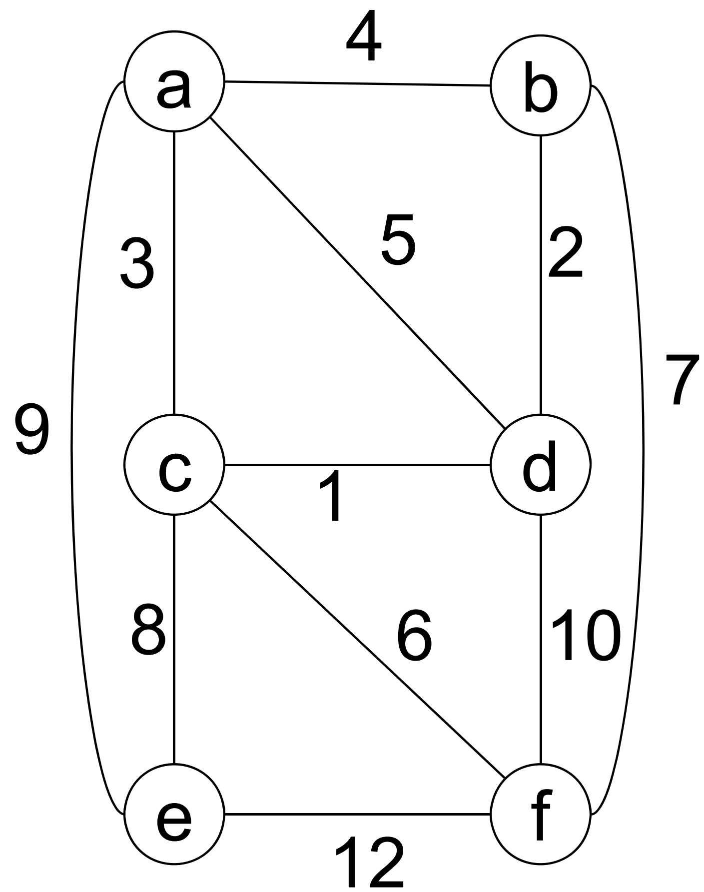
- 「⇐」は明らか．以下では「⇒」を帰納法で示す．
- T*: 最小全域木 ⇒ T* の費用＜T の費用 ⇒ T*≠ T
- T* - T に含まれる枝数に関する帰納法で証明
- T* - T に含まれる枝数＝1のとき:
  T* は T の枝を一つだけ交換して得られる全域木なので，
  条件(＊)が成り立つ

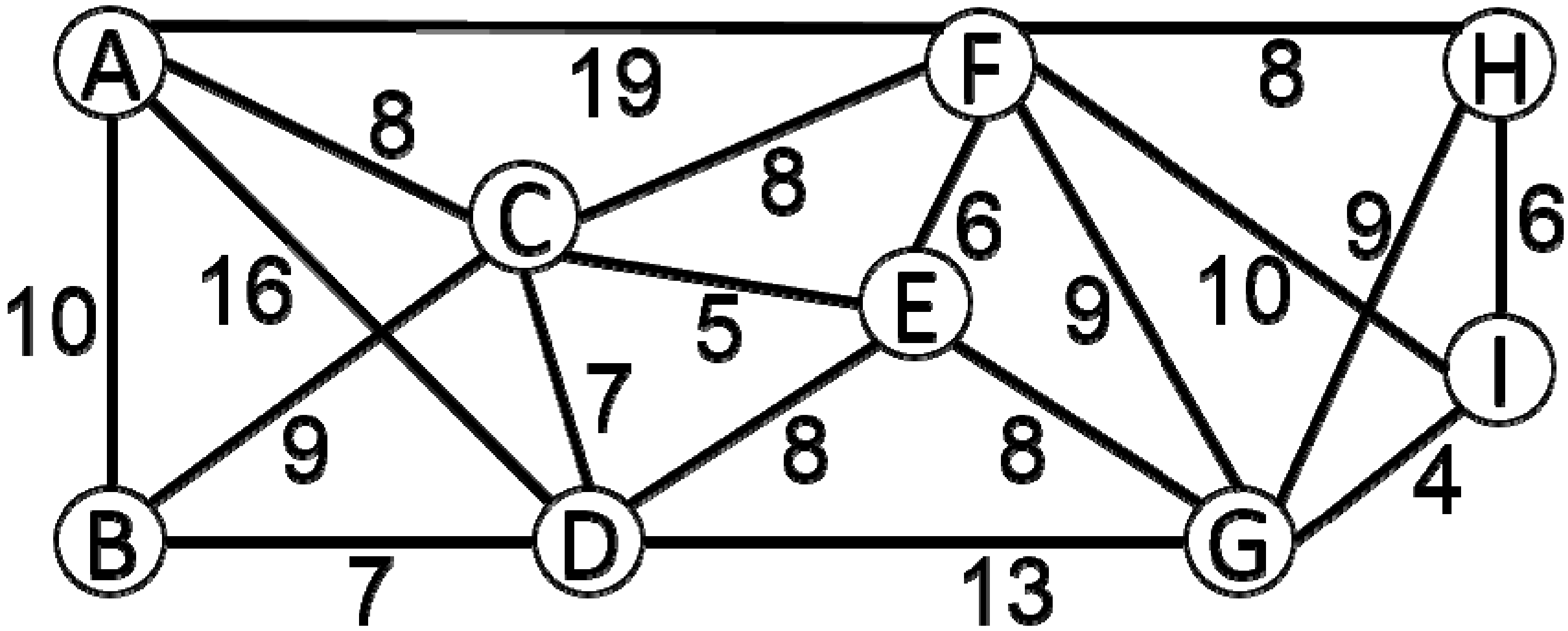
定理1の証明（続き）

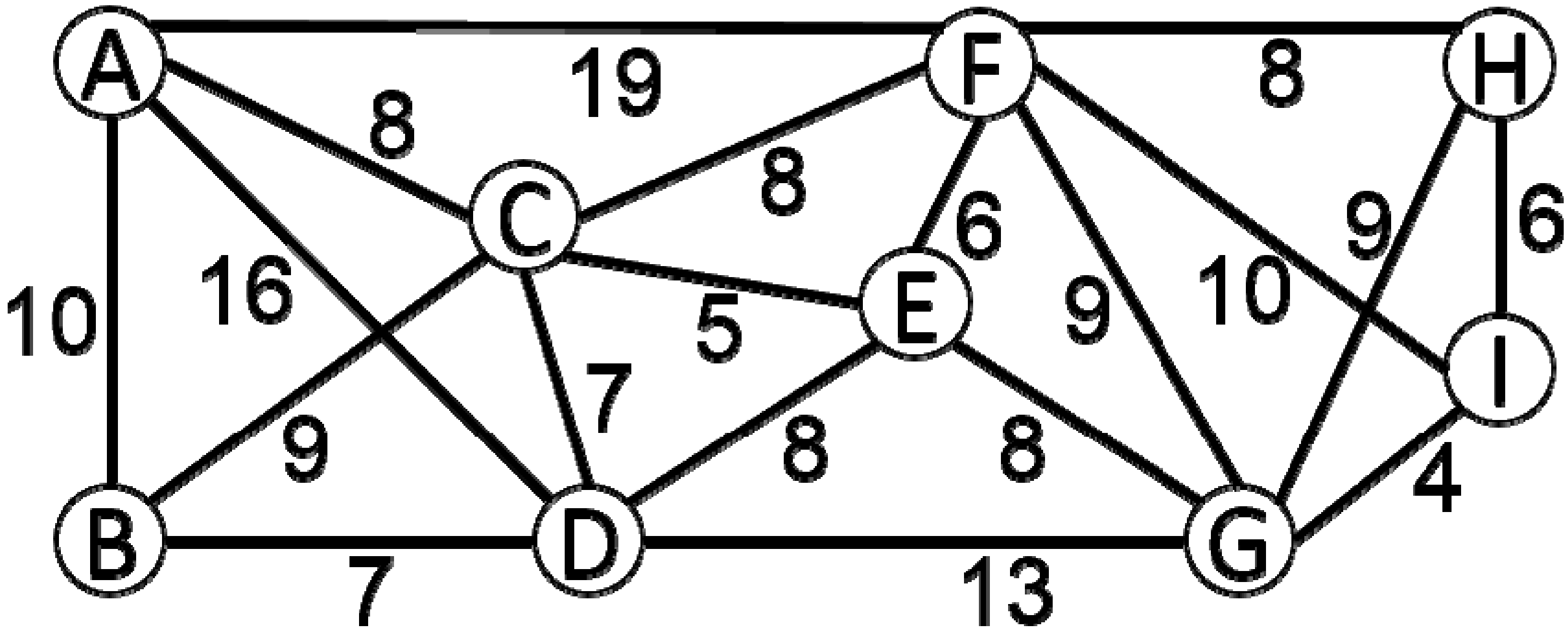
- T* - T に含まれる枝数＝k-1 で成り立つと仮定，= k のとき：
  補題5より，
  任意に選んだ枝 $(u, v) \in T - T^*$ に対し，
  ある枝$(s, t) \in T^* - T$ が存在して，
  T – (u,v)+(s,t) および T* + (u,v) – (s,t) はともに全域木
- ここで，次の等号が成立
  「T – (u,v)+(s,t) の費用」＋「T* + (u,v) – (s,t)の費用」
  ＝「T の費用」＋「T* の費用」 …（式A）
- 「T – (u,v)+(s,t) の費用」<「T の費用」ならば，ただちに条件(*)成立
- 「T – (u,v)+(s,t) の費用」=「T の費用」ならば，(式A)より，
  「T* + (u,v) – (s,t)の費用」＝「T* の費用」
  ∴ T** = T* + (u,v) – (s,t) もまた最小全域木
- T と T** に帰納法の仮定を適用➔条件（＊）が成立 ■

クラスカルのアルゴリズム，プリムのアルゴリズムを実行してみなさい

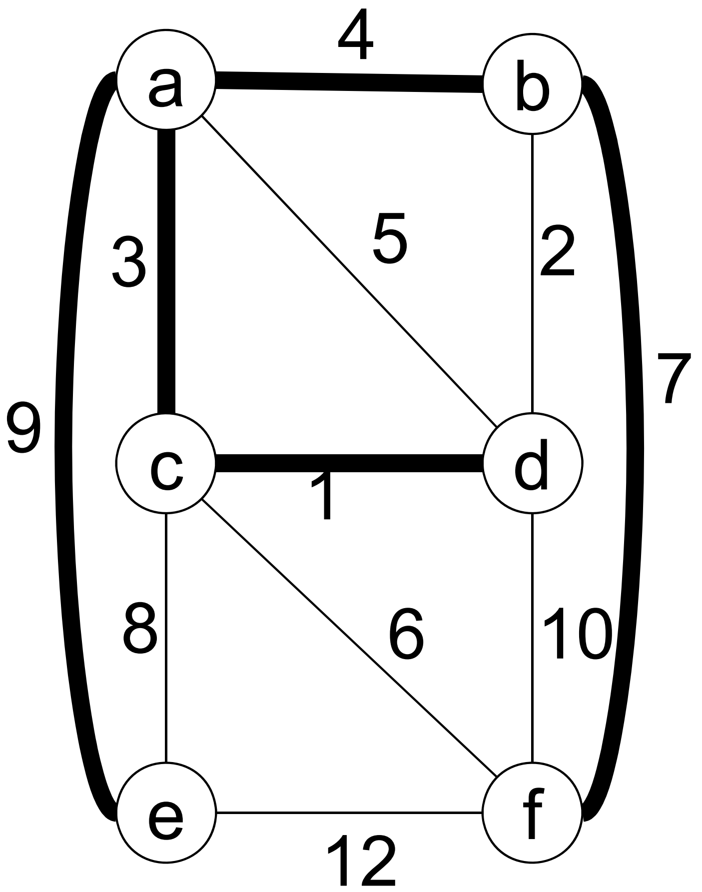


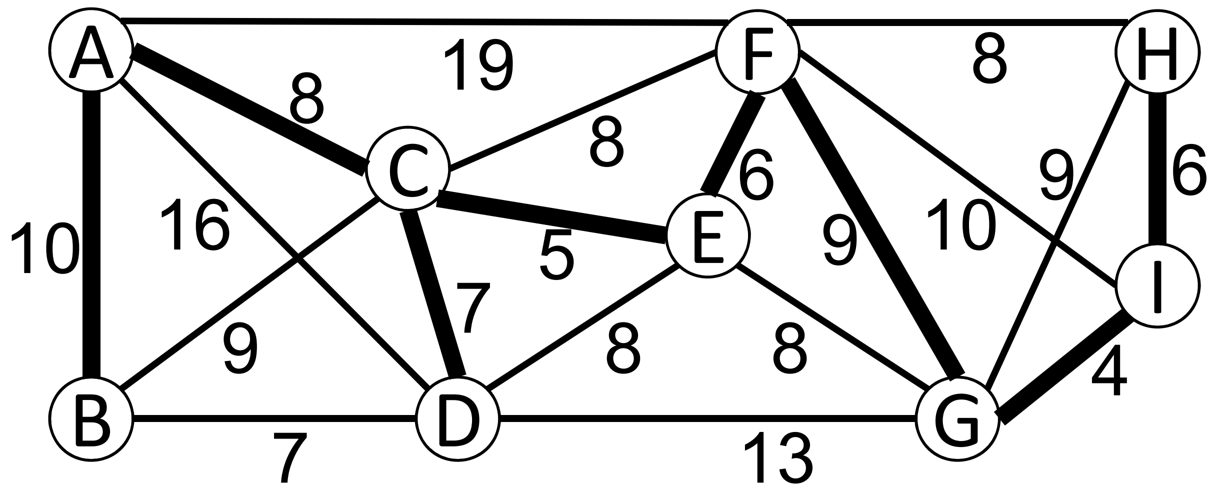



A
F
H
19
8
8
C
8
6
6
9
10
E
16
10
9
5
7
I
9
8
8
4
B
D
G
7
13

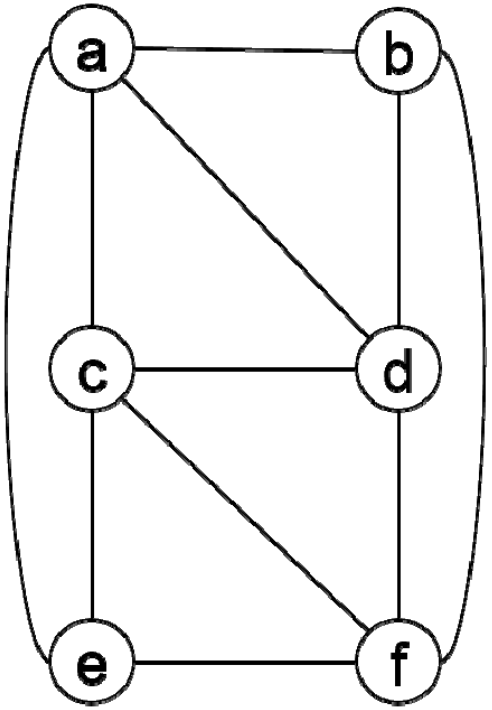

A
F
H
19
8
8
C
8
6
6
9
10
E
16
10
9
5
7
I
9
8
8
4
B
D
G
7
13

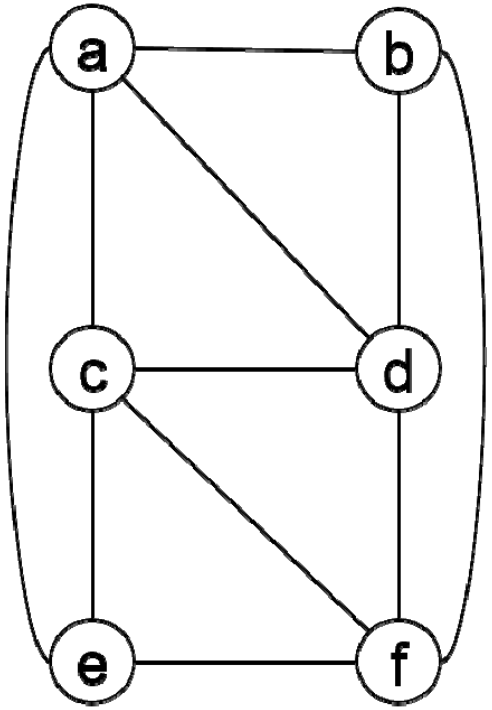
- 下記の全域木が費用最小でないことを示せ
  （ヒント：枝を1つ交換すればわかる）

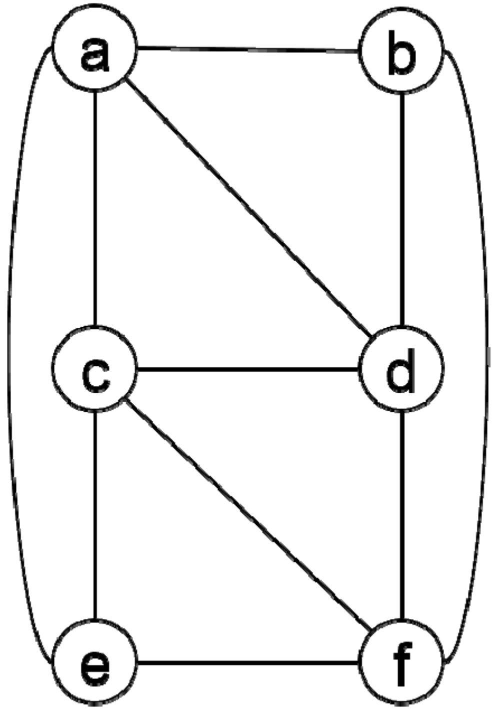


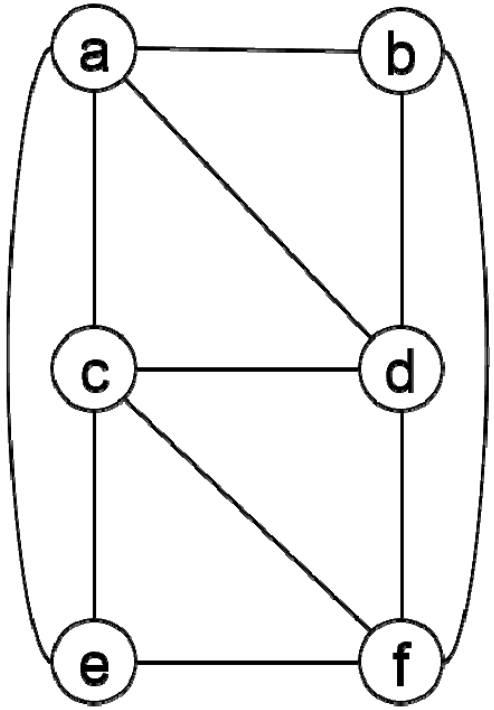


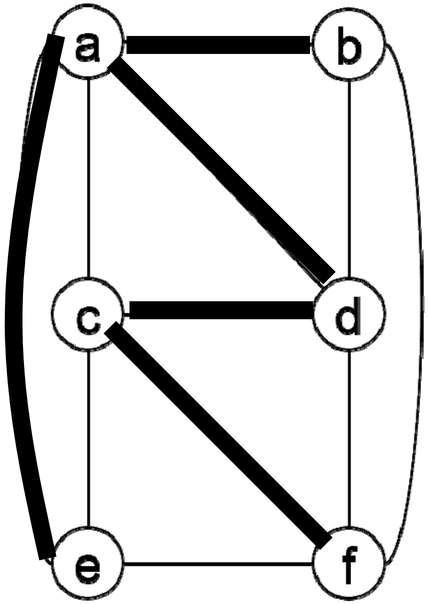
ある全域木から別の全域木を，枝の入れ替えのみによって得ることが出来ることを確認しよう











スタート

ゴール