数理経済学
第６回
最小全域木問題

塩浦昭義
東京工業大学 経営工学系 准教授
shioura.a.aa@m.titech.ac.jp

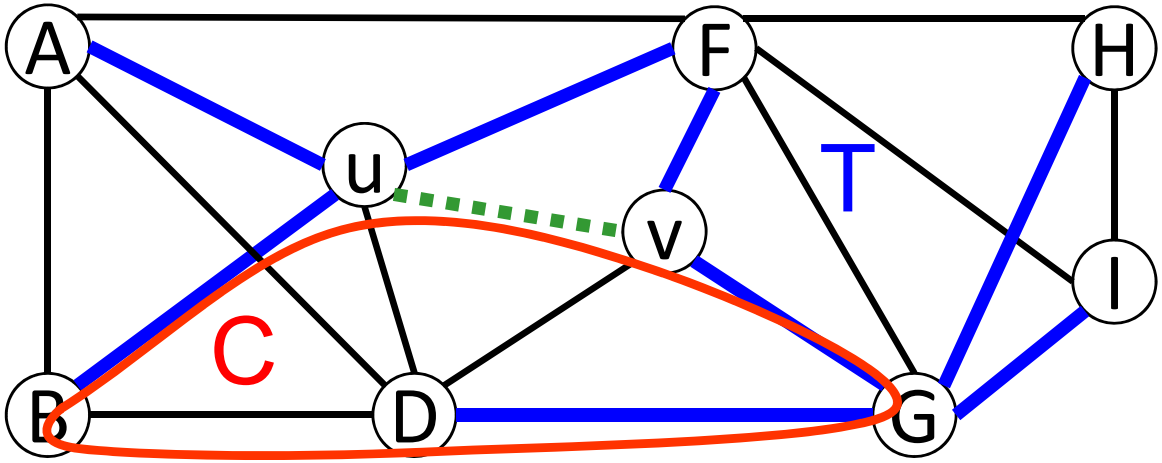
中間試験について

- 日時：7月12日（金）15:10～16:35 (85分)
- 手書きのA4用紙一枚のみ持ち込み可
  - 印刷やコピーは不可
  - これも採点の対象，試験終了後に回収します
- 本，ノート等の持ち込みは不可
- 座席はこちらで指定
- 試験内容：
  - 最短路，最大（重み）マッチング，最小全域木
  - 証明，アルゴリズムの実行，用語の定義など
  - 過去問は昨年までのOCWにおいてあります
- 50点満点，25点以下は不合格
- 成績は登録済みのメールアドレスに報告（7月末迄に）

閉路と最小全域木

定理: C: 任意の閉路, (u,v): 閉路Cの中で費用最大の枝
➔(u,v)を含まない最小全域木T*が存在

証明: 最小全域木 T* を任意に選ぶ.
（1）T* が (u,v) を含まない: ただちに証明終了



閉路と最小全域木（続き）

証明:　（2）T* が (u,v) を含む場合:

① T* - (u,v) は 2つの連結成分 (V', T'), (V'', T'') をもつ（演習問題）

P = C - (u,v) とおくと，これは u と v を結ぶ路.

一般性を失うことなく $u \in V', v \in V''$ と仮定する.

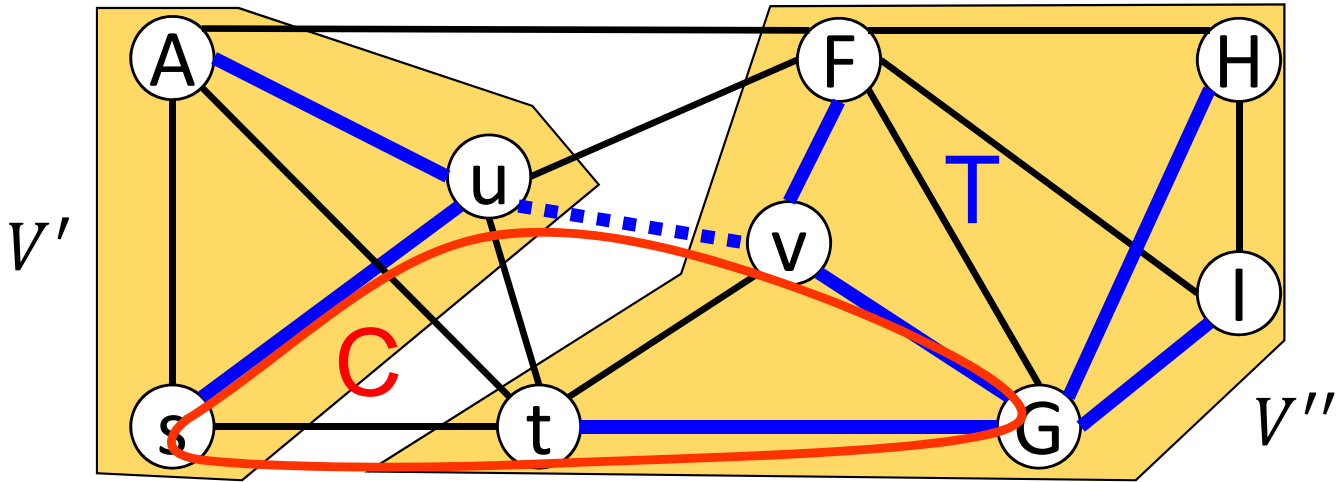
② P は(s,t) ∈ E(V', V'') なる枝 (s,t) を含む.

（証明）$P = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$（$v_0 = u,\ v_k = v$）とする.

i* : $v_i \in V'$ を満たす i の中の最大値

$v_k = v \in V''$ より i < k　∴ $v_{i^*} \in V', v_{i^*+1} \in V''$

∴　$(v_{i^*}, v_{i^*+1}) \in E(V', V'')$



閉路と最小全域木（続き）

証明:

$T^{**} = T^{*} - (u,v) + (s,t)$ とする

③ (V, T^{**}) は連結

（証明）$V = V' \cup V''$ の任意の2頂点 x, y の間に路があることを示す.

(case 1) $x,y \in V'$ または $x,y \in V''$

前者の場合を考えればよい.

(V', T') は連結なので, x と y の間の路 $P \subseteq T'$ が存在.

$T' \subseteq T^{*} - (u,v) \subseteq T^{**}$ なので, P は T^{**} に含まれる.

(case 2) $x \in V'$ かつ $y \in V''$,　　または　　$y \in V'$ かつ $x \in V''$

前者の場合を考えればよい.

$s,x \in V'$ であり, (V', T') は連結なので,

s と x の間の路 $P' \subseteq T'$ が存在.

同様に, t と y の間の路 $P'' \subseteq T''$ が存在.

$\therefore P = P' \cup \{(s,t)\} \cup P'' \subseteq T^{**}$ は x と y の間の路

閉路と最小全域木（続き）

証明:

T** = T* - (u,v) + (s,t) とする

④ T** は閉路を含まない（演習問題）

∴ T** は全域木

⑤ T** は最小全域木

（証明）T*は最小全域木なので，T** の費用 - T* の費用≧0.

一方，T** の費用 - T* の費用 = (s,t)の費用 - (u,v)の費用≦0

（∵(u,v)は閉路Cの中で費用最大の枝, (s,t)∈C）

よって，T** の費用 = T* の費用　　∴T** は最小全域木

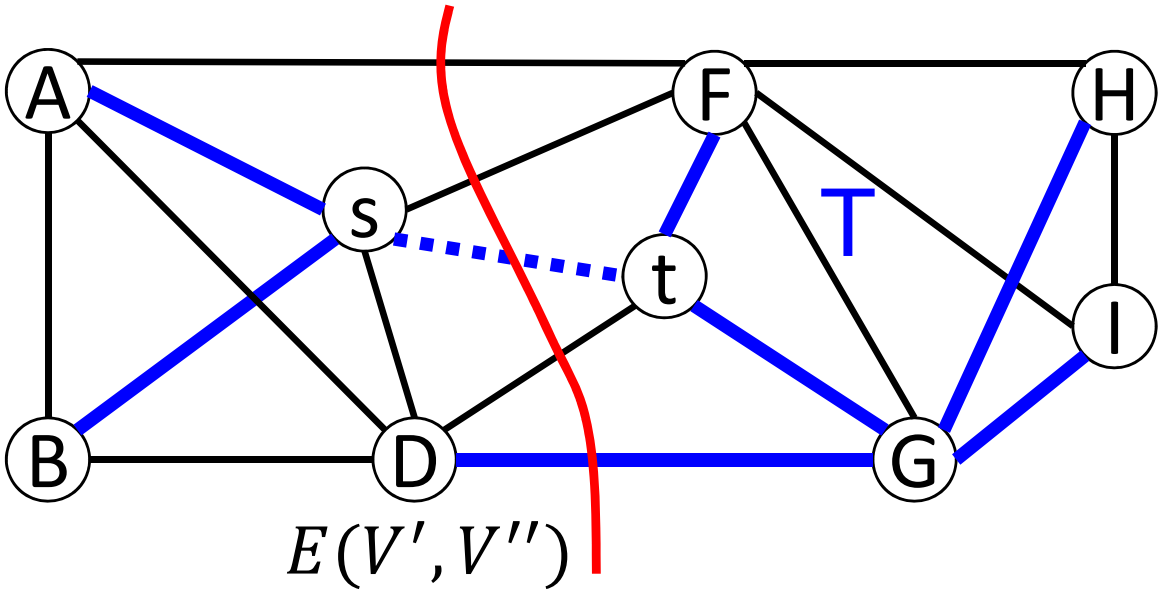
⑤より，T** は(u,v)を含まない最小全域木　■

カットセットと最小全域木：証明

- 任意のカットセットの枝のうち，少なくとも1つは全域木に含まれる
- カットセットの中で費用最小の枝は必要

定理: $E(V',V'')$: 任意のカットセット
(s,t): カットセット$E(V',V'')$ の中で費用最小の枝
➔(s,t)を含む最小全域木T*が存在

証明: 最小全域木 T* を任意に選ぶ.
（1）T* が (s,t) を含む: ただちに証明終了



カットセットと最小全域木：証明

証明：（2）T* が (s,t) を含まない場合：

T* は連結なので，s と t の間に路が存在 ← P とおく

① P の枝 (u,v) でカットセット $E(V',V'')$ に含まれるものが存在

（「閉路と最小全域木」の証明②参照）

② T – (u,v) は 2つの連結成分 (V’, T’), (V’’, T’’) になる

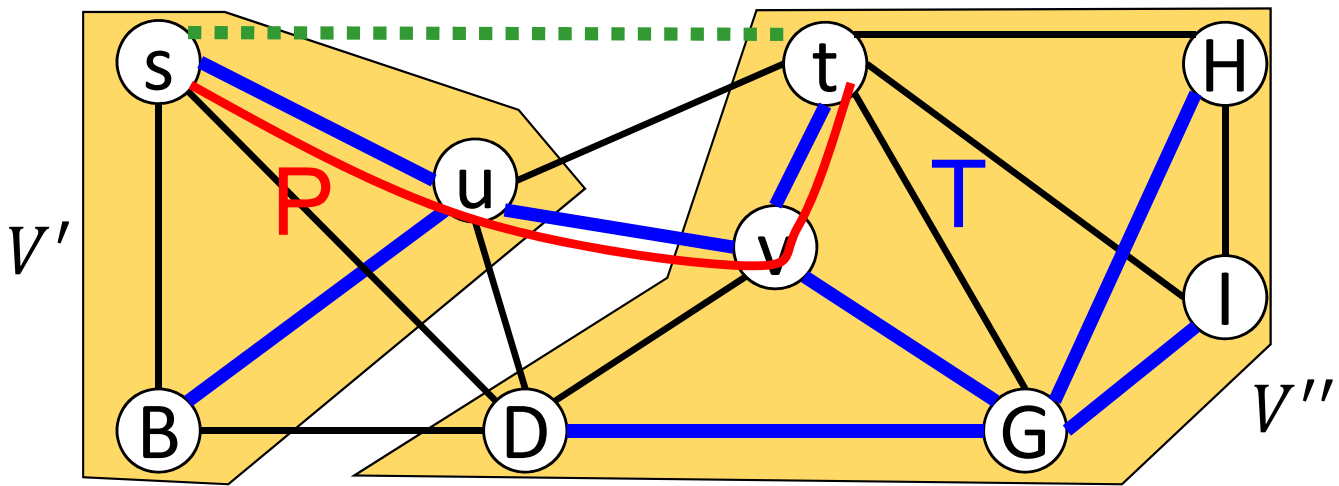
（前回の演習問題）

T** = T* - (u,v) + (s,t) とおく

③ T** は連結（∵仮定 $(s,t) \in E(V',V'')$ を使う.

「閉路と最小全域木」の証明③参照）

④ T** は閉路なし（前回の演習問題）



カットセットと最小全域木：証明

証明：

∴ T** は全域木

⑤ T** は最小全域木

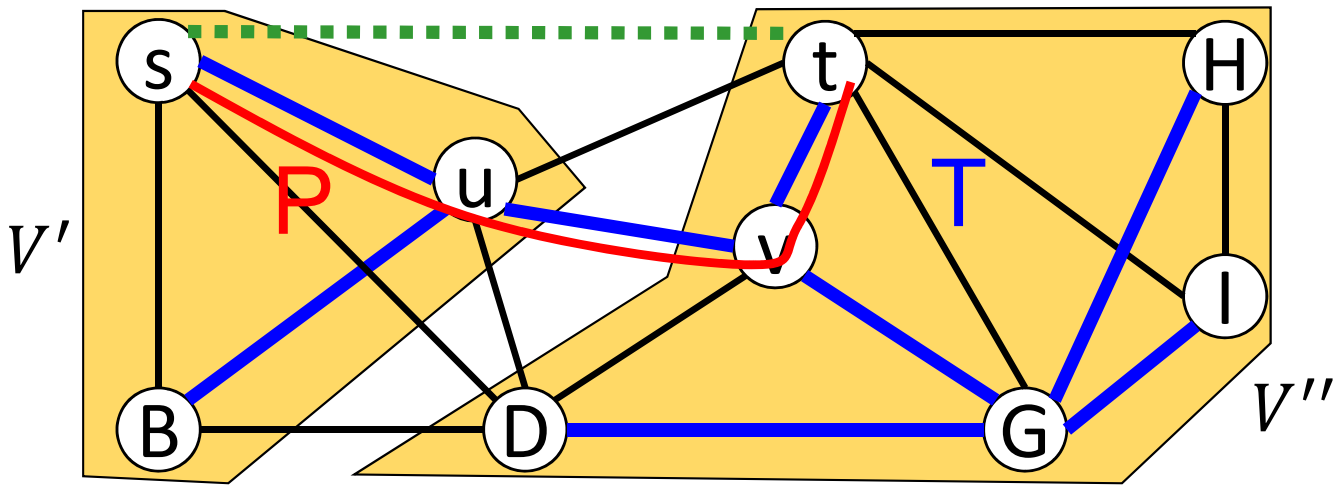
（証明）T*は最小全域木なので，T** の費用 - T* の費用≧0.

一方，T** の費用 - T* の費用 = (s,t)の費用 - (u,v)の費用≦0

（∵(s,t)はカットセットの中で費用最小の枝, (u,v)∈E(V’,V’)）

よって，T** の費用 = T* の費用　　∴T** は最小全域木

⑤より，T** は(s,t)を含む最小全域木　■



貪欲アルゴリズムとマトロイド

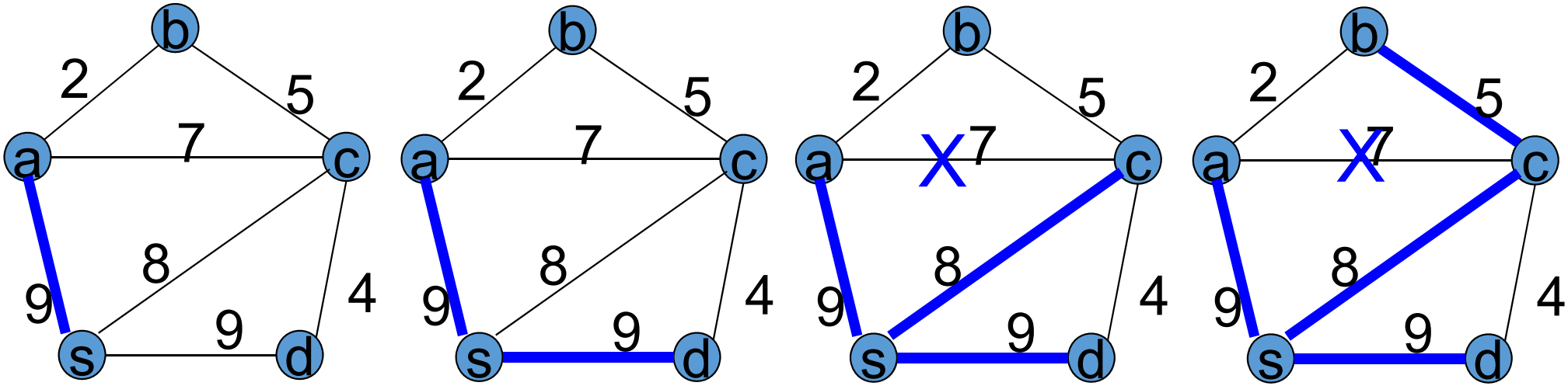
最大全域木と貪欲アルゴリズム

最大全域木問題：最小全域木問題の最大化版

各枝に非負重みが与えられる → 枝重みの和を最大化

最大全域木はクラスカルのアルゴリズム（の最大化版）で求められる

- 「閉路を含まない」という条件の下，重みの大きい順に枝を追加



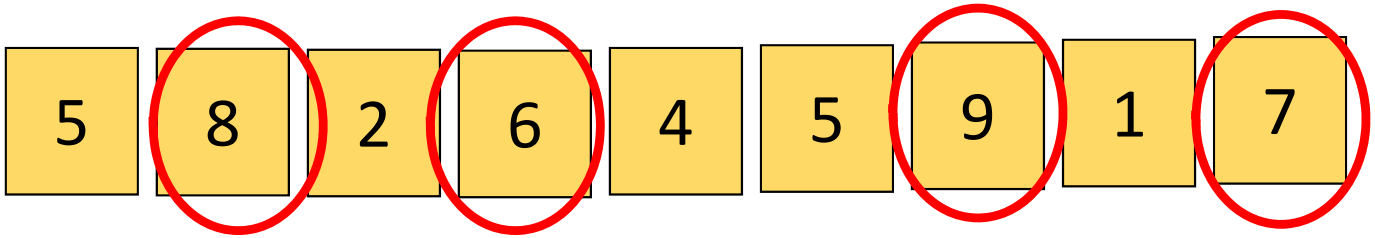
※枝集合 T は全域木 ←→ T は閉路を含まない極大な枝集合

（T は閉路を含まないが，これ以上枝を追加すると必ず閉路が生じる）

貪欲アルゴリズムで解ける簡単な問題

下記の問題も貪欲アルゴリズムで解くことができる

- 非負の数字の書かれた n 枚のカードが与えられる
- k 枚選んで，選んだカードの数字の合計を最大化したい



k=4のとき

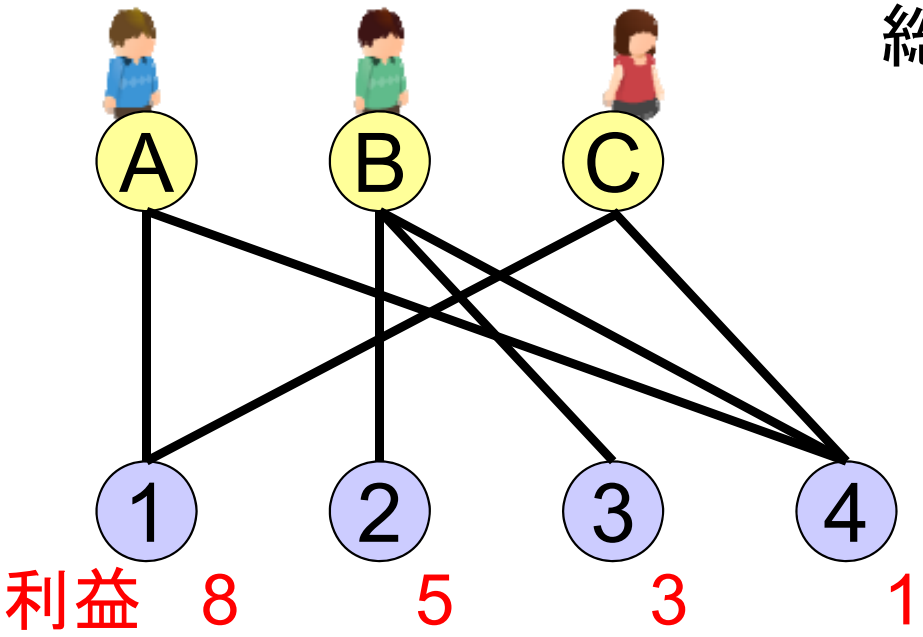
アルゴリズム：
「カードの枚数が5枚以下」という条件の下，
数字の大きい順にカードを追加

※ k 枚のカードの集合
⟷ k 枚以下のカードの極大な集合

仕事の割当問題と貪欲アルゴリズム

- 労働者集合と仕事集合が与えられている
- 各労働者は処理可能な仕事が限られている → 二部グラフ
- 各労働者には高々ひとつの仕事を割当可
- 各仕事は高々一人の労働者に割当可 → マッチング
- 仕事 j を処理 → 利益 p(j) を得る
- 処理可能な仕事の（極大な）集合のうち，

総利益最大のもの選びたい



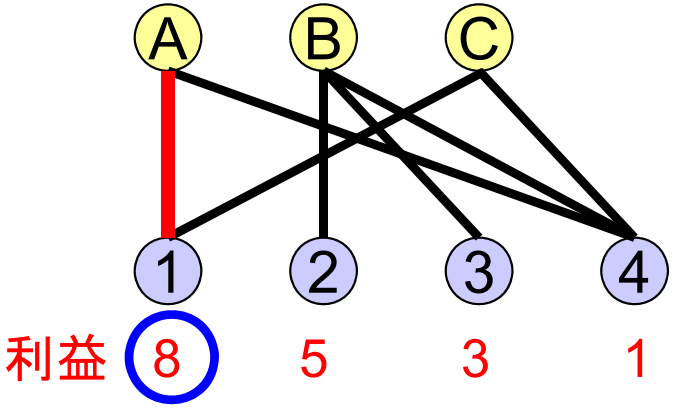
仕事①，③は
労働者A, B によって
処理可能
➔ 利益 8 + 3

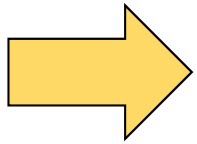
仕事の割当問題と貪欲アルゴリズム

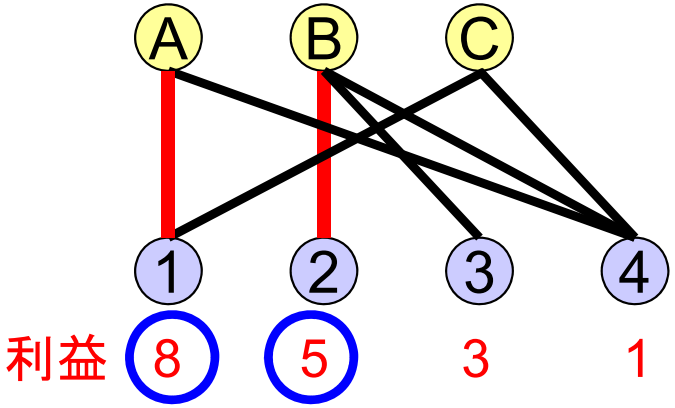
利益最大の仕事集合を求めるアルゴリズム:

「選んだ仕事すべてと労働者の間のマッチングが存在」

という条件の下，利益の大きい順に仕事を追加







仕事①はAとの間に
マッチングをもつ
➔①追加

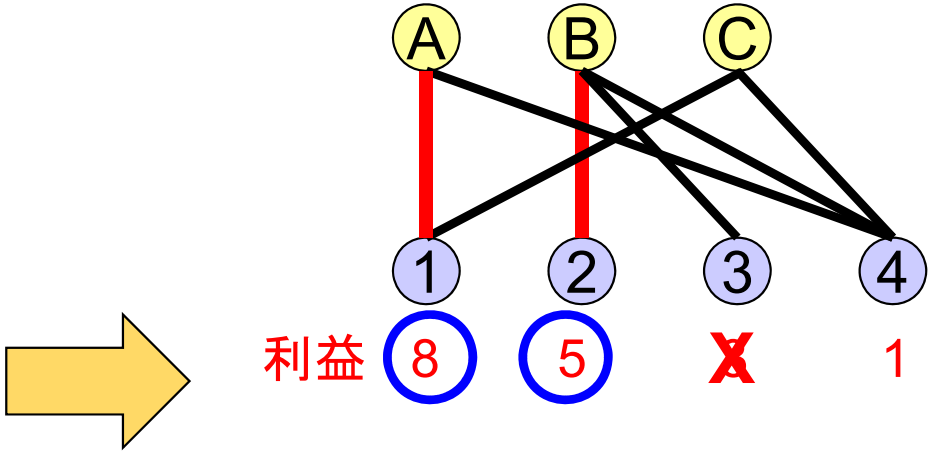
仕事①，②は
A, Bとの間に
マッチングをもつ
➔②追加

仕事の割当問題と貪欲アルゴリズム

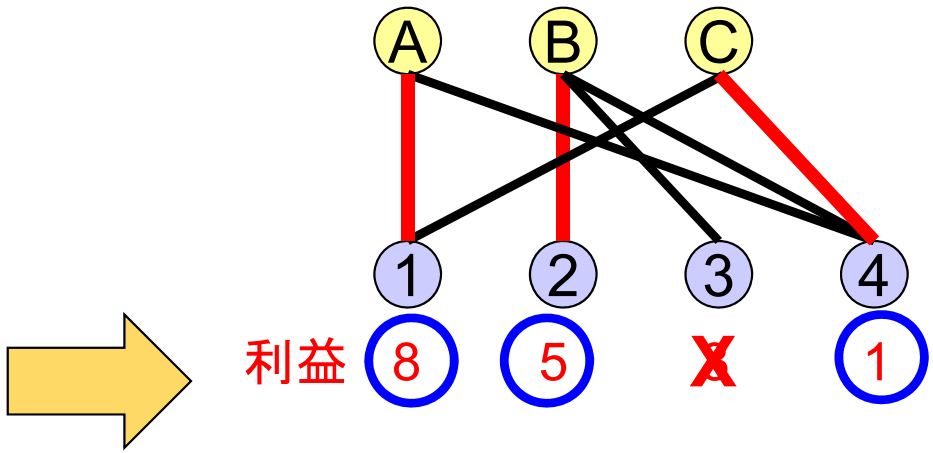
利益最大の仕事集合を求めるアルゴリズム:

「選んだ仕事すべてと労働者の間のマッチングが存在」

という条件の下，利益の大きい順に仕事を追加



仕事①，②，③は
A, B, Cとの間に
マッチングをもたない
➔③は除外



仕事①，②，④は
A, B, Cとの間に
マッチングをもつ
➔④追加

貪欲アルゴリズムで解けない問題

ナップサック問題

- n個の品物の中から持って行くものを選択
- ナップサックには b kg まで入れられる
- 品物 $i = 1,2,\ldots,n$ の重さは a_i kg, 価値は c_i
- 利用価値の合計を最大にする（極大な）品物集合を求めたい

i	1	2	3	4	5	6
c_i	10	7	25	24	28	10
a_i	2	1	6	5	7	3

b=7

品物{1,4}のとき価値の合計が最大 34

貪欲アルゴリズムで解けない問題

ナップサック問題の貪欲アルゴリズム

- 「総重量が b kg 以下」という条件の下，
  価値の大きい順に品物を追加

i	1	2	3	4	5	6
c_i	10	7	25	24	28	10
a_i	2	1	6	5	7	3

b=7

品物{1,4}のとき価値の合計が最大 34

貪欲アルゴリズム：最初に品物 5 を選んで終了
価値は 28 < 34

貪欲アルゴリズムの一般形

問題の一般形:

- $E = \{e_1, e_2, \ldots, e_n\}$ --- 選択可能な要素の集合
- $w(e)$ $(e \in E)$: 各要素の非負重み
- 要素集合に対する条件〇〇〇

目的: 「〇〇〇」という条件の下で,
重み和が最大の極大な要素集合を求める

一般的な貪欲アルゴリズム:

「〇〇〇」という条件の下, 重みの大きい順に要素を追加

- これまで紹介した3つの問題は上記の形の問題
- 上記の形の貪欲アルゴリズムにより, 最適解が得られる
- どのような問題ならば,
  貪欲アルゴリズムにより最適解が得られるのか？
  ➔ 解の集合族が「マトロイド」

貪欲アルゴリズムとマトロイド

- $\mathcal{F}$: 与えられた条件を満たす解集合をすべて集めたもの
  （例：最大全域木問題の場合，閉路を含まない枝集合すべて）

➔ 問題は下記のように記述される：

最大化　$\sum_{e \in X} w(e)$　　条件　$X \in \mathcal{F}$

定理:　$\mathcal{F}$ はマトロイド
⬅➔ 任意の非負重み $w(e)$ $(e \in E)$ に対し，
上記の問題の最適解は貪欲アルゴリズムで求められる

マトロイドの定義

定義: $\mathcal{F}$ はマトロイド ←→ 以下の3条件を満たす:

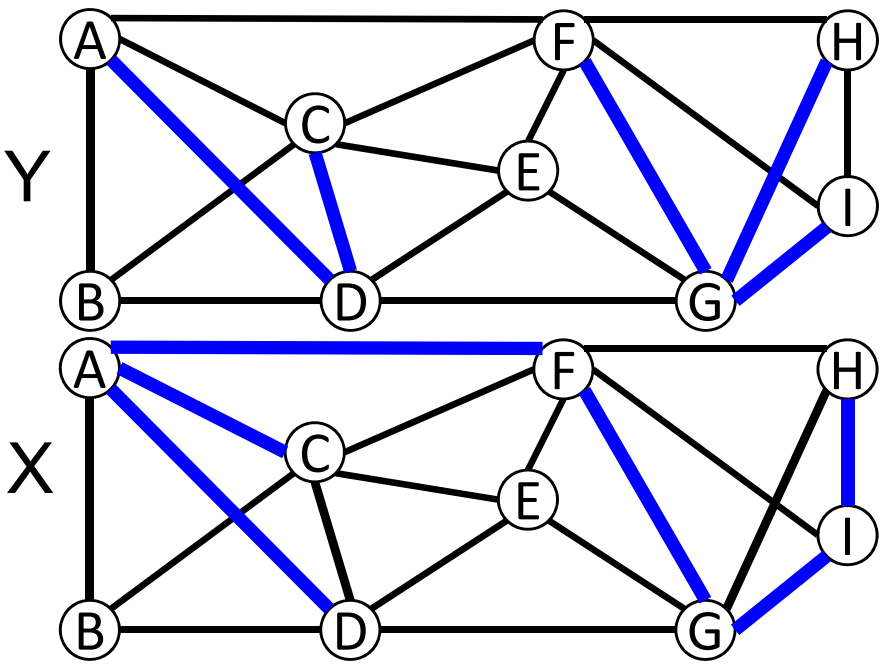
(P1) $\emptyset \in \mathcal{F}$ (空集合は実行可能解)

(P2) $X \in \mathcal{F},\ Y \subseteq X \rightarrow Y \in \mathcal{F}$

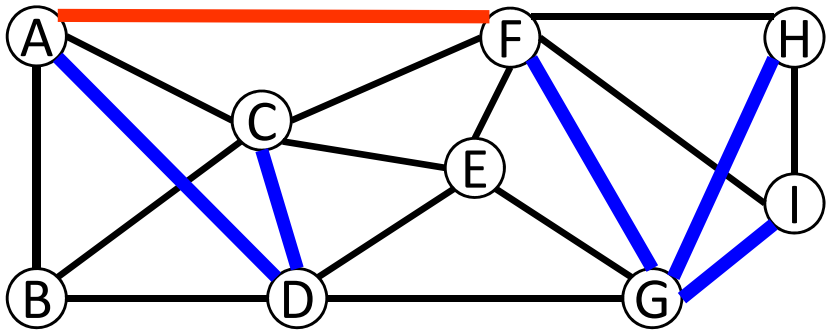
(集合 X が実行可能解のとき, その部分集合 Y は実行可能解)

(P3) $X, Y \in \mathcal{F},\ |Y| < |X| \rightarrow \exists j \in X - Y\colon Y \cup \{j\} \in \mathcal{F}$

ある要素 j∈ X - Y が存在して, Y + j も実行可能解



Y+(A,F) も閉路なし



マトロイドの他の例：一次独立なベクトル集合

行列の列ベクトルの集合で，一次独立なものの全体

matroid（マトロイド）= matrix（行列）＋ -oid（のようなもの）

$$\begin{array}{c} \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 1 & -1 \end{pmatrix} \\ \begin{array}{cccccc} a & b & c & d & e & f \end{array} \end{array}$$

$\mathcal{F}$ に含まれるベクトル集合：

空集合$\emptyset$

要素数1のベクトル集合すべて

{a,f} 以外の要素数2のベクトル集合すべて

{a,b,d}, {a,b,e}, {a,c,d}, {a,c,e}, {b,c,d}, {b,c,e}, {b,c,f}, {b,d,e}, {b,d,f}, {b,e,f}, {c,d,e}, {c,d,f}, {c,e,f}

$$\begin{array}{c} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \\ \begin{array}{cc} a & b \end{array} \end{array} \qquad \begin{array}{c} \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ 2 & 1 & -1 \end{pmatrix} \\ \begin{array}{ccc} c & e & f \end{array} \end{array} \Rightarrow \begin{array}{c} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \\ \begin{array}{ccc} a & b & e \end{array} \end{array}$$

一次独立ベクトルと貪欲アルゴリズム

行列の列ベクトルの集合の中から，一次独立なものを選ぶ.
そのベクトルの数を最大にしたい.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 1 & 1 \end{pmatrix}$$

○ ○ × ○ × ○

最大サイズの一次独立ベクトル集合を求めるアルゴリズム
「選んだベクトル集合が一次独立」
という条件の下，任意の順に列ベクトルを追加
これも貪欲アルゴリズム

マトロイドの利点

マトロイドのように抽象的な概念を扱う利点

- 個々の問題のもつ離散構造の本質（重要なポイント）が見えてくる
- マトロイドに対して新たな結果を得る

  ➔個々の問題に対して適用可能

  - 似たような定理・アルゴリズムを個々の問題に対して示す必要がなくなる

演習問題

定理: グラフGの任意の枝集合 $T \subseteq E$ に対し，以下の条件は等価
(1) T は全域木（(V, T) は連結，かつ枝集合 T は閉路を含まない）
(2) (V, T) の任意の2頂点の間に路はただ1つ存在
(3) (V, T) は連結，ただし T から任意の枝を削除すると非連結
(4) |T| = |V| - 1, かつグラフ (V,T) は連結
(5) |T| = |V| - 1, かつ T は閉路を含まない

問題1: (1)➔(2) を証明せよ.

問題2: (2)➔(3) を証明せよ.

問題3: (3)➔(1) を証明せよ.

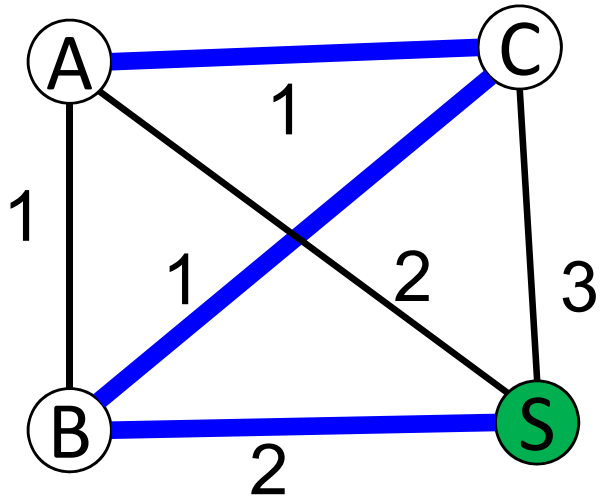
問題4（やや難）: 全域木 T に対し，「Tの枝数 ＝ Tの頂点数－1」が成り立つことを示せ.

（ヒント: 「(1)➔(3)」と頂点数に関する帰納法を使う）

最小全域木問題と協力ゲーム

応用：最小全域木構築の費用分担

- ネットワークを構築したい➔最小全域木を作る➔総費用4
  - Sは基地局（ソース）
  - その他の施設（A, B, C）の利用者（プレイヤー）で費用を負担
  - 費用をどう分担するか？



条件1：全プレイヤーの支払額の合計
＝最小全域木の総費用

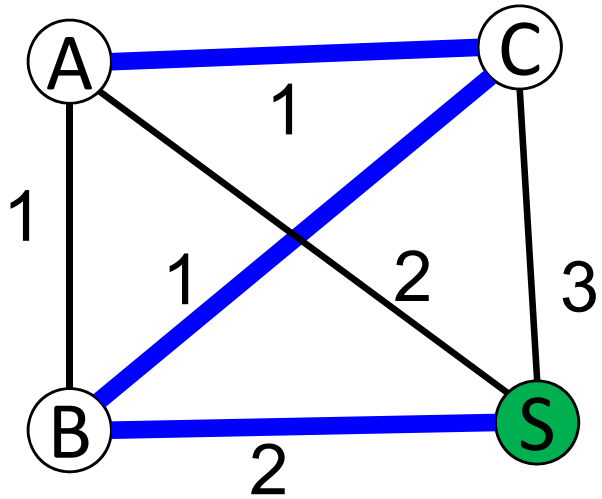
例：$(p_A, p_B, p_C) = (3,1,0)$

条件2：「公平な分担にする」

「公平な」費用分担を求める問題：最小全域木ゲーム
（協力ゲームの中の，特性関数形ゲームの特殊ケース）

応用：最小全域木構築の費用分担

- ネットワークを構築したい➔最小全域木を作る➔総費用4
  - Sは基地局（ソース）
  - その他の施設（A, B, C）の利用者（プレイヤー）で費用を負担
  - 費用をどう分担するか？



条件1：全プレイヤーの支払額の合計
＝最小全域木の総費用

例：$(p_A, p_B, p_C) = (3,1,0)$

コア：条件1，2を満たす費用分担

条件2：「公平な分担にする」← いろいろな定式化が可能

よく使われる条件：グループの支払額の合計
≦グループとソースに対する最小全域木の総費用

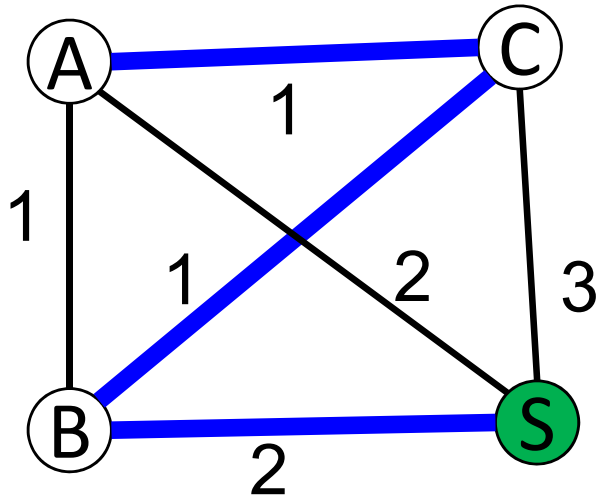
例：A の支払い = 3 ➔ A は単独でネットワークを作った方が費用小

最小全域木ゲームとコア

条件1: 全プレイヤーの支払額の合計＝最小全域木の総費用

条件2: グループの支払額の合計

≦グループとソースに対する最小全域木の総費用



$$p_A + p_B + p_C = 4,$$
$$p_A \leq 2,\ p_B \leq 2,\ p_C \leq 3,$$
$$p_A + p_B \leq 3,\ p_A + p_C \leq 3,\ p_B + p_C \leq 3$$

コア: 条件1，2を満たす費用分担

コアの例: $(p_A, p_B, p_C) = (2,1,1), (1,2,1)$ など

定理 (Bird 1976): 最小全域木ゲームにおいて，コアは常に存在

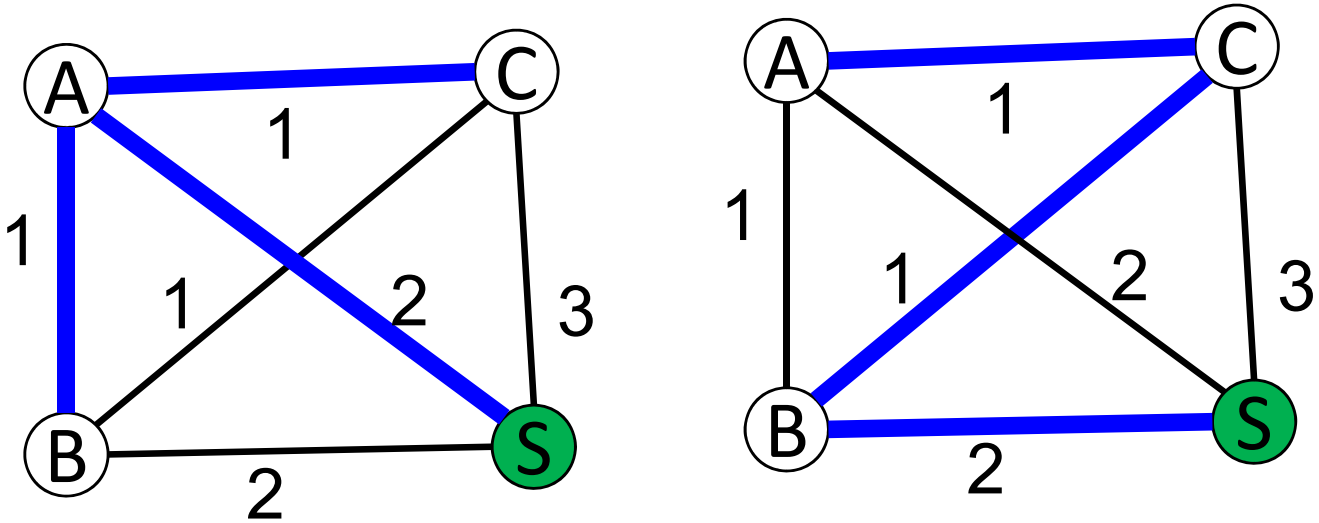
証明: 「Bird 分担」とよばれる費用分担がコアであることを示す

Bird 分担

Bird 分担の定義

- ソース s を根として，プリムのアルゴリズムにより最小全域木を求める
- $V = \{v_1, v_2, \dots, v_n\}$ の順に頂点が追加されたと仮定
- $(v_i, f(v_i))$: v_i が追加されたときの枝（ここで $f(v_i) \in \{v_1, \dots, v_{i-1}\}$）
- v_i の支払額 p_{v_i} =枝$(v_i, f(v_i))$の費用

Bird分担の例: $(p_A, p_B, p_C) = (2,1,1), (1,2,1)$



※Bird分担以外のコアも存在しうる

Bird分担がコアであることの証明

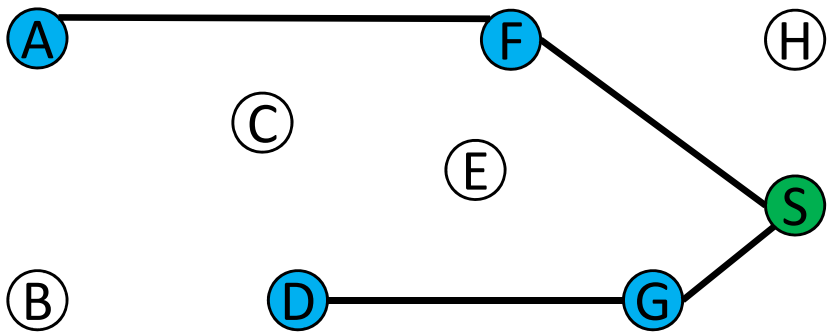
「条件2:グループの支払額の合計
≦グループとソースに対する最小全域木の総費用」
が成り立つことをチェックすれば良い

プレイヤーのグループ W に注目

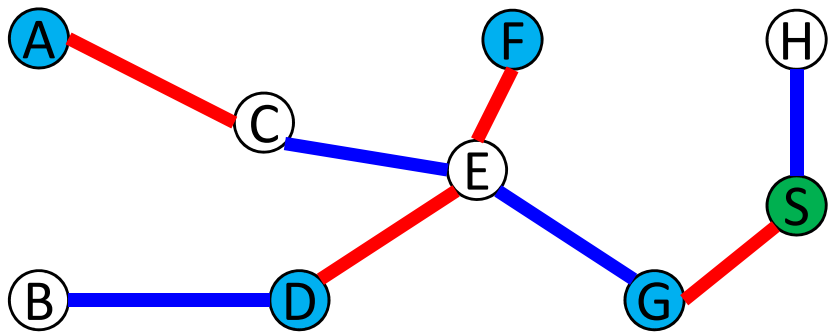
$\alpha = W \cup \{s\}$ に対する最小全域木T'の総費用

β =グループ W の支払額の合計= $\sum_{v \in W} c(v, f(v))$

$\gamma = \sum_{v \in V \setminus W} c(v, f(v))$ ➔プリムによる最小全域木の総費用= $\beta + \gamma$



T'の総費用α



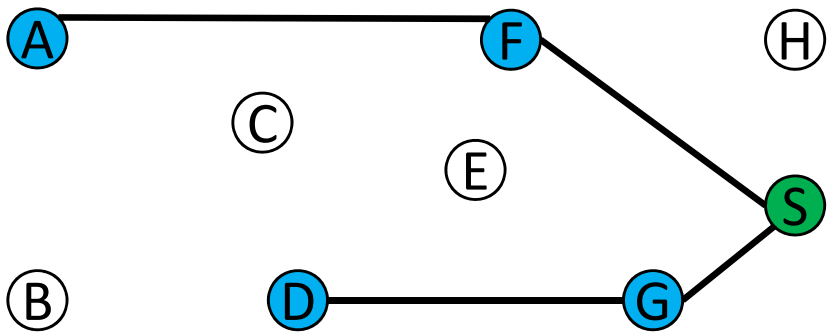
最小全域木の総費用= $\beta + \gamma$

参考資料:岡本吉央さんのスライド
http://jpn.nec.com/rd/datamining/rd/datamining/images/nec_datamining_seminar7.pdf

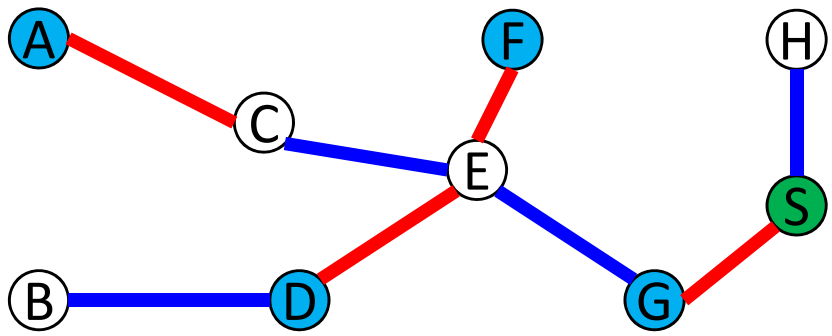
Bird分担がコアであることの証明（続き）

T' に枝集合$\{(v, f(v)|v \in N \setminus W\}$を追加
➔全域木になる（なぜ？）

この全域木の総費用$\alpha + \gamma$≧最小全域木の総費用＝$\beta + \gamma$
$\therefore \alpha \geq \beta$ が成立



T'の総費用α



最小全域木の総費用＝ $\beta + \gamma$

参考資料：岡本吉央先生のスライド
http://jpn.nec.com/rd/datamining/rd/datamining/images/nec_datamining_seminar7.pdf