Vol.2012-AL-139 No.2
2012/3/14

# リーフの最近分岐点を用いたグラフの2辺連結化アルゴリズム

間島利也[†1]　田岡智志[†2]　渡邉敏正[†2]

グラフの2辺連結化問題とは，無向グラフ $G=(V,E)$ が与えられたときに，$G$ に辺集合 $F$ を付加することにより得られるグラフが2辺連結となるような，辺数最小の付加すべき辺集合 $F$ を求める問題である．本稿では，グラフの2辺連結化問題を解く線形時間アルゴリズムとして，リーフの最近分岐点を用いたアルゴリズムを示す．

# Algorithms to 2-Edge-Connect a Graph by Using Nearest Branch-Vertices of Leaves

Toshiya Mashima,[†1] Satoshi Taoka[†2]
and Toshimasa Watanabe[†2]

The 2-edge-connectivity augmentation problem of a graph, 2ECA, is defined as follows: "Given an undirected graph $G=(V,E)$, find a smallest set $F$ of edges such that $(V,E\cup F)$ is 2-edge-connected." In this paper, we show two linear time algorithms for solving 2ECA. Each of the two algorithms is based on the method for searching nearest branch-vertices of leaves in a tree, where a branch-vertex of a tree is a vertex of degree at least three.

## 1. はじめに

与えられたグラフにコスト最小の辺集合を付加して所望の性質を満たすグラフを構成する問題を一般にグラフの辺付加問題という．グラフの連結度増大問題は，グラフの辺付加問題の一種であり，所望の連結度を持つグラフをコスト最小の辺集合を付加することにより構成する問題である．付加辺のコストが均一である場合には，辺数最小の付加すべき辺集合を求める問題となる．グラフの連結度増大問題はグラフに関する基本的な問題の一つであり，通信ネットワークの設計等への応用があることから，それを解くためのアルゴリズムの研究が多く為されている．

本稿では，グラフの連結度増大問題の中で最も基本的な問題の一つである，付加辺のコストが均一である場合のグラフの2辺連結化問題を扱う．以下，特に断りの無い限り，付加辺のコストは均一であるとする．グラフの2辺連結化問題 (2ECA と略記する) とは，無向グラフ $G=(V,E)$ が与えられたときに，$G$ に辺集合 $F$ を付加して得られるグラフ $G+F$ が2辺連結となる（すなわち，$V$ の全ての2点間に辺を共有しない道が2本以上存在するグラフとなる），辺数最小の辺集合 $F$ を求める問題である．より一般に，最小本数の辺付加でグラフを $k$ 辺連結にする問題をグラフの $k$ 辺連結化問題と呼び，$k$ECA と表す．グラフが $k$ 辺連結であるとは，点の除去によりグラフを非連結にするために少なくとも $k$ 本以上の辺を除去する必要があることを意味する．

2ECA に関連する既知結果について述べる．2ECA の最適解を求める線形時間アルゴリズム[1] が Eswaran と Tarjan によって提案されている．この Eswaran と Tarjan によるアルゴリズムは，深さ優先探索を用いて，辺を付加すべき点集合に番号を付け，その番号を基に，最適解となる付加辺集合を一度に求めるものである．付加する辺にコストがついており，付加辺のコスト総和を最小にするような2辺連結化問題については，与えられたグラフが木で付加辺のコストが1または2でも NP-困難であることが知られている[2]．2辺連結化問題をより一般化した問題であるグラフの $k$ 辺連結化問題 $k$ECA については，付加辺の本数を最小にする場合の多項式時間アルゴリズム[3] が知られている．更に，Frank[4] により，点対ごとに局所辺連結度の要求がある場合に最小本数の辺付加で要求を満たすグラフを構成する問題が多項式時間で解けることが示されている．

最近，著者らは文献 5) で，リーフの最近分岐点を用いたグラフの2点連結化アルゴリズムを提案した．本稿では，リーフの最近分岐点を用いた手法を2辺連結化問題に適用した場合のアルゴリズムを示す．提案アルゴリズムは，2辺連結成分木におけるリーフの最近分岐点の探索を基礎としており，最適解に含まれる辺を1本ずつ求めるものである．

†1 広島国際大学
Hiroshima International University
†2 広島大学
Hiroshima University

## 2. 準　　備

### 2.1 諸 定 義

無向グラフ $G = (V, E)$ は，有限な空でない点集合 $V$ と無向辺の集合 $E$ からなる．$V$，$E$ をそれぞれ $V(G)$，$E(G)$ と表すこともある．2 つの点 $u, v$ $(u, v \in V)$ を結ぶ辺を $(u, v)$ と表し，$u, v$ をその辺の端点という．本稿では，グラフといえば無向グラフを意味する．また，無向辺のことを単に辺と呼ぶ．$G$ において，点 $v \in V$ に接続している辺の数を点 $v$ の次数と呼び．$d_G(v)$ で表す．グラフ $G' = (V', E')$ が，$V' \subseteq V$ かつ $E' \subseteq E$ であるならば，$G'$ はグラフ $G$ の部分グラフであるという．グラフ $G$ から辺集合 $E' \subseteq E$ を除去したグラフを $G - E'$ と表す．

グラフ $G$ のどの 2 点 $u, v$ に対しても $u, v$ をつなぐ経路が存在するとき，$G$ は連結であるという．または $G$ は連結グラフであるという．連結でないグラフを非連結である，または非連結グラフであるという．連結成分とは極大な連結部分グラフのことである．閉路を含まないグラフを森，閉路を含まない連結グラフを木という．森において，次数が 0 である点を孤立点，次数が 1 である点をリーフ，次数が 3 以上である点を分岐点と呼ぶ．

切断辺とはその辺を除去するとグラフの連結成分が増える辺のことである．切断辺を含まない連結グラフを 2 辺連結グラフという．グラフ $G$ の 2 辺連結成分（以降，単に 2 成分と呼ぶ）とは $G$ の極大な 2 辺連結部分グラフのことである．2 成分内の任意の 2 点間には 2 本以上の辺を共有しない道が存在する．ちょうど 1 本の切断辺とだけ接続している 2 成分をリーフ 2 成分，どの切断辺とも接続していない 2 成分を孤立 2 成分という．

### 2.2 下 界 値

本節では，グラフ $G$ が与えられたとき，$G$ を 2 辺連結化するために $G$ に付加すべき辺の本数の下界値[1] について説明する．

$G$ のリーフ 2 成分の数を $t$，$G$ の孤立 2 成分の数を $s$ とする．但し，$G$ 全体が孤立 2 成分であるならば $s = 0$ とする．$G$ の各リーフ 2 成分には少なくとも 1 辺を，$G$ の各孤立 2 成分には少なくとも 2 辺を付加する必要がある．従って，$G$ を 2 辺連結にするには，$t + 2s$ 以上の点次数の増加を必要とする．1 本の辺付加で点次数は 2 だけ増加させることができるので，少なくとも $\lceil (t + 2s)/2 \rceil = \lceil t/2 \rceil + s$ 本の辺を付加する必要がある．よって，付加辺数の下界値は $\lceil t/2 \rceil + s$ である．$G$ が連結ならば単に $\lceil t/2 \rceil$ となる．

本稿で示すアルゴリズムを含め，2ECA を解くアルゴリズムはここで示した下界値と等しい本数の辺付加で $G$ を 2 辺連結化するアルゴリズムである．

### 2.3 2 成 分 森

$G$ の各 2 成分を 1 点に縮約して得られるグラフを $G_S = (V_S, E_S)$ と表す．$V_S$ の各点は $G$ の 2 成分に，$E_S$ の各辺は $G$ の切断辺に対応している．$G_S$ は $G$ から線形時間で構成できる．$G_S$ は閉路を含まないので $G_S$ を 2 成分森と呼ぶ．特に，$G$ が連結ならば，$G_S$ は木であり，$G_S$ を 2 成分木と呼ぶ．なお，前節で示した付加辺数の下界値は，2 成分森 $G_S$ の情報を使って簡単に計算することができる．これは，$G$ のリーフ 2 成分の数 $t$ は $G_S$ のリーフの数，$G$ の孤立 2 成分の数 $s$ は $G_S$ の孤立点の数，として計算できるからである．

$G_S$ を 2 辺連結化することが $G$ を 2 辺連結化することと同等であることが，次のように示されている[1]．$G$ の各点 $v \in V$ に対し，$\alpha(v)$ を $v$ を含む 2 成分が縮約された $G_S$ の点を表すとし，$u \in V_S$ に対し，$\alpha^{-1}(u)$ を $u$ に縮約された $G$ の 2 成分中の任意の点を表すとする．$G$ に付加する辺集合 $F$ に対して，$\alpha(F) = \{(\alpha(u), \alpha(v)) \mid (u, v) \in F, \alpha(u) \neq \alpha(v)\}$，$G_S$ に付加する辺集合 $F_S$ に対して，$\alpha^{-1}(F_S) = \{(\alpha^{-1}(u), \alpha^{-1}(v)) \mid (u, v) \in F_S\}$ とする．このとき，$G + F$ が 2 辺連結になるような任意の付加辺集合 $F$ に対して $G_S + \alpha(F)$ は 2 辺連結であり，かつ $|\alpha(F)| \leq |F|$ が成り立つ，また逆に，$G_S + F_S$ が 2 辺連結であるような任意の付加辺集合 $F_S$ に対して，$G + \alpha^{-1}(F_S)$ は 2 辺連結であり，$|\alpha^{-1}(F_S)| \leq |F_S|$ が成り立つ，以上のことから，$G$ に対する問題は，$G_S$ を 2 辺連結化する問題に帰着することができる．

### 2.4 連結化アルゴリズム

グラフが非連結の場合，連結成分数を $h$ とすると，$h - 1$ 本の辺の付加で連結グラフを構成できる[1]．2 つの連結成分を 1 辺の付加で連結にする場合には，付加する辺の端点をリーフ 2 成分あるいは孤立 2 成分の点から選び辺を付加する．その操作を連結成分が 1 つになるまで（すなわち，$h - 1$ 回）繰り返す．この連結化アルゴリズムは線形時間で実行可能であり，また，前述の下界値が $h - 1$ だけ減少していることは容易に確認できる．

以降，本論文では 2ECA に対する入力グラフは連結グラフであると仮定する．

### 2.5 リーフの最近分岐点

連結グラフ $G$ の 2 成分木 $G_S$ を $T$ をおく．$T$ のリーフ（次数 1 の頂点）$u$ に対して，$u$ から $T$ を探索したときに，初めて訪れる分岐点（次数 3 以上の点）を $u$ の最近分岐点と呼び，$\mathrm{b}_T(u)$ と表す．また，少なくとも 1 つ以上のリーフに対して最近分岐点となっている分岐点を，リーフ付き分岐点と呼ぶ．

### 2.6 2 成分木の変形

$T$ を連結グラフ $G$ の 2 成分木とし，リーフが 4 つ以上存在すると仮定する．$T$ の 1 つの

リーフを $u$ とする．$u$ から $u$ の最近分岐点 $\mathrm{b}_T(u)$ までの経路 $P(u, \mathrm{b}_T(u))$ を $T$ から削除する (但し，$\mathrm{b}_T(u)$ は削除しない) ことによって得られる木を $\mathrm{H}(T, u)$ と表す．$T' = \mathrm{H}(T, u)$ とする．$T'$ のリーフ数は $T$ のリーフ数より 1 だけ小さく，また，$d_{T'}(\mathrm{b}_T(u)) = d_T(\mathrm{b}_T(u)) - 1$ である．$T'$ における各リーフの最近分岐点は，$d_T(b_T(u)) = 3$ でない限り，$T$ での最近分岐点と一致する．$d_T(b_T(u)) = 3$ の場合には，$d_{T'}(b_T(u)) = 2$ となり，$b_T(u)$ が $T'$ の分岐点でなくなり，$T'$ の高々1 つのリーフについて，最近分岐点の更新が必要となる．新しい最近分岐点は，$b_T(u)$ からの $T'$ の未探索な方向への探索を実行することにより見つけられる．

本稿で提案するアルゴリズムでは，付加辺を 1 つ見つける度に，木の変形を行い，必要ならば最近分岐点の更新を行う．

$G$ の 2 成分木から，上記の操作，すなわち，リーフからその最近分岐点までの経路を削除する操作を何回か繰り返して得られる木を $T''$ とする．$T''$ の点集合によって表されている 2 成分から誘導される $G$ の部分グラフを $G[T'']$ と表す．

### 2.7 リーフ接続条件

本節では，付加する辺はリーフ同士を結ぶものと仮定して，下界値を 1 だけ減らせるような辺付加の条件を記述する．なお，$G$ は連結であり，リーフ数は 3 以上であると仮定する．

$T$ を $G$ の 2 成分木とする．$T$ のリーフ数を $t$ とする．$u, v$ を $T$ の 2 つのリーフとし，辺 $(u, v)$ を付加するものとする．$u$ の $T$ における最近分岐点 $\mathrm{b}_T(u)$ を $u'$，$v$ の $T$ における最近分岐点 $\mathrm{b}_T(v)$ を $v'$ とおく．

$t$ が偶数のとき，下界値を 1 だけ減らせるための条件，すなわち，リーフ数を 2 だけ減らすための条件は，(i) $u'$ と $v'$ が異なる，または (ii) $u'$ と $v'$ が等しくかつ $d_T(u') \geq 4$，のいずれかが成り立つことである．ここで，$T' = \mathrm{H}(T, v)$ とする．上記の条件が成り立つならば，$u$ の最近分岐点は $T'$ においても $u'$ である．更に，$T'' = \mathrm{H}(T', u)$ とすると，$T''$ のリーフ数は $T$ のリーフ数より 2 だけ小さく，すなわち，$G[T'']$ の下界値は $G$ の下界値より 1 だけ小さくなることが確認できる．このように，$t$ が偶数の場合には，1 辺の付加に伴い，下界値が 1 だけ小さい $G$ の連結部分グラフを構成できる．

$t$ が奇数の場合には，$u$ と $v$ は任意の異なるリーフとしておいて，下界値が 1 だけ小さい $G$ の連結部分グラフを構成できる．すなわち，$T' = \mathrm{H}(T, v)$ とすれば，$G[T']$ の下界値は $G$ の下界値より 1 だけ小さくなることが確認できる．

提案アルゴリズムでは，1 辺の付加で下界値を 1 だけ減らすための上記の条件を満たすように辺付加を行い，1 辺を付加するごとに，下界値が 1 だけ小さい $G$ の連結部分グラフの 2 成分木を構成する．

## 3. リーフの最近分岐点を用いたアルゴリズム

本章では，リーフの最近分岐点を用いたアルゴリズムを示す．始めに，リーフの最近分岐点を用いた基本となるアルゴリズムを示し，次に，その詳細版アルゴリズムとして，(分岐点数+1) 個のスタックを用いるアルゴリズム，更に，使用するスタック数を 1 個にしたアルゴリズムを示す．なお，本章では入力グラフ $G$ を連結グラフと仮定する．

### 3.1 基本となるアルゴリズム

リーフの最近分岐点を用いた基本のアルゴリズムは次のようである．$G_S$ のリーフ $v$ の最近分岐点を $v'$ と表す．

**アルゴリズム** *Basic-Algorithm*

**入力**：連結な無向グラフ $G = (V, E)$.

**出力**：$G + F$ が 2 辺連結となるような辺数最小の付加辺集合 $F$

(1) $G$ の 2 成分を求め，各 2 成分を 1 点に縮約したグラフ $G_S = (V_S, E_S)$ を構成し，$G_S$ のリーフ数 $t$ を計算する．$F_S \leftarrow \emptyset$ とする．

(2) リーフ数 $t$ が奇数のとき，$G_S$ の 2 つのリーフ $a, b$ を任意に選び，$F_S \leftarrow F_S \cup \{(a, b)\}$ とする．$a$ から $a'$ への経路（$a'$ は含まない）を削除したグラフを改めて $G_S$ として，$t \leftarrow t - 1$ とする．

(3) リーフ数 $t$ が 4 以上である間，次の (a) と (b) の操作を繰り返す．

(a) $G_S$ の 2 つのリーフ $a, b$ で，$a'$ と $b'$ が異なるか，または同じでその次数が 4 以上であるような $a, b$ を求め，$F_S \leftarrow F_S \cup \{(a, b)\}$ とする．

(b) $a$ から $a'$ への経路（$a'$ は含まない），$b$ から $b'$ への経路（$b'$ は含まない）を削除したグラフを改めて $G_S$ とし，$t \leftarrow t - 2$ とする．

(4) $t = 2$ ならば，2 つのリーフを $a, b$ とし $F_S \leftarrow F_S \cup \{(a, b)\}$ とする．

(5) 各辺 $(a, b) \in F_S$ に対して，$a, b$ に対応する $G$ の 2 成分間を結ぶ 1 辺を作り，そのような辺からなる集合を $F$ として出力する．

アルゴリズムの正当性について説明する．アルゴリズム *Basic-Algorithm* では，手順 (2) と (3) でリーフの接続条件を満たす 2 つのリーフを選んでいる．よって，$G_S$ の更新操作により，下界値が 1 だけ小さい $G$ の連結部分グラフ $G[G_S]$ が間接的に構成されている．アルゴリズムは，いずれは $t = 2$ の状態に達し，最後の付加辺を求める．

$G$ に関する付加辺の下界値を $p$ とする．アルゴリズムによって，構成される $G_S$ の連結部分グラフの列を $G_p = G_S$，$G_{p-1}$，$G_{p-2}$，…，$G_2$，$G_1$ とする．但し，添え字はそのグ

ラフに関する下界値を表している．また，アルゴリズムによって手順（2）〜（4）で逐次求められる付加辺を求められた順に，$e_p$，$e_{p-1}$，$e_{p-2}$，...，$e_2$，$e_1$ とする．$e_p$ が最初に求められた付加辺，$e_1$ が最後に手順（4）で求められた付加辺を表す．

アルゴリズムの正しさを示すために，$G_k$ に対して辺集合 $\{e_i \mid 1 \leq i \leq k\}$ の付加で 2 辺連結化できることを示す．まず，$G_1$ については辺 $e_1$ の付加で 2 辺連結化できることは明らかである．次に，$G_{k-1}$ に対して辺集合 $\{e_i \mid 1 \leq i \leq k-1\}$ の付加で 2 辺連結化できると仮定して，$G_k$ に対して辺集合 $\{e_i \mid 1 \leq i \leq k\}$ の付加で 2 辺連結化できることを示す．$G_{k-1}$ は $G_k$ の部分グラフであり，仮定より $\{e_i \mid 1 \leq i \leq k-1\}$ の付加で 2 辺連結化できる．従って，2 辺連結グラフ $G_{k-1} + \{e_i \mid 1 \leq i \leq k-1\}$ に，$G_k$ から $G_{k-1}$ を構成した時に削除した 2 本もしくは 1 本の経路および $e_k$ を付加してできるグラフに切断辺が生じないことは容易に確認できる．よって，アルゴリズム *Basic-Algorithm* は 2ECA の最適解となる最小本数の付加辺集合を求める．

### 3.2 スタックを複数用いるアルゴリズム

本節では，前節で示した基本アルゴリズムの詳細化版アルゴリズムを示す．集合を保持するためのデータ構造としてスタックを用いることとし，スタックを複数用いるアルゴリズムを記述する．文献 5) での 2 点連結化アルゴリズムと同様の考えで，各分岐点 $v$ に対して，$v$ を最近分岐点とするリーフの集合を保持するためのスタック $S(v)$ を用意する．また，リーフ付き分岐点の集合を保持するためのスタック $B$ を準備しておく．スタック $B$ に入っている要素数を $|B|$ などと表す．アルゴリズム記述中のおいては，上記のスタックを単に集合として記述している．また，$G_S$ のリーフ $v$ の最近分岐点を $v'$ と表す．

**アルゴリズム** *2-Edge-Connect_A*

**入力**：連結な無向グラフ $G = (V, E)$

**出力**：$G + F$ が 2 辺連結となるような辺数最小の付加辺集合 $F$

（1） $G$ の 2 成分を求め，各 2 成分を 1 点に縮約したグラフ $G_S = (V_S, E_S)$ を作る．$G_S$ のリーフ数 $t$，$G_S$ のリーフ $v_1, v_2, \ldots, v_t$ を求める．$B \leftarrow \emptyset$，各分岐点 $v$ に対して $S(v) \leftarrow \emptyset$ とする．$F_S \leftarrow \emptyset$ とする．

（2） $t = 2$ ならば，$F_S \leftarrow F_S \cup \{(v_1, v_2)\}$ として手順 8 に進む．

（3） $t \geq 3$ のとき，各リーフ $v_i$（$1 \leq i \leq t$）に対して以下の (a)–(c) を実行する．
(a) $v_i$ の最近分岐点 $v_i'$ を求める．
(b) $v_i'$ が新たなリーフ付き分岐点ならば（すなわち，$|S(v_i')| = 0$ ならば），$B \leftarrow B \cup \{v_i'\}$ とする．
(c) $S(v_i') \leftarrow S(v_i') \cup \{v_i\}$ とする．

（4） $G_S$ のリーフ付き分岐点が 2 つ以上存在し，かつ $t$ が奇数ならば，次の (a)–(f) を実行する．/* 実行後 $t$ は偶数となる．*/
（a） $B$ から分岐点 $b_1$ を取り出す．/* $B \leftarrow B - \{b_1\}$ */
（b） $S(b_1)$ からリーフ $x$ を取り出す．/* $S(b_1) \leftarrow S(b_1) - \{x\}$ */
（c） $B$ に属す任意の分岐点を $b_2$ とし，$S(b_2)$ に属す任意のリーフを $y$ とする．/* $b1 \neq b2$ */
（d） $F_S \leftarrow F_S \cup \{(x, y)\}$ とする．
（e） $x$ から $b_1$ への経路 ($b_1$ は含まない) を削除したグラフを改めて $G_S$ とする．$b_1$ が分岐点であり $|S(b_1)| > 0$ ならば，$B \leftarrow B \cup \{b_1\}$ とする．$b_1$ が分岐点でなくなり $|B(b_1)| = 1$ ならば，$B(b_1)$ からリーフ $z$ を取り出し，$z$ の新しい最近分岐点 $z'$ を $b_1$ からの探索により求め，$S(z') \leftarrow S(z') \cup \{z\}$ とし，更に，$z'$ が新たなリーフ付き分岐点ならば $B \leftarrow B \cup \{z'\}$ とする．
（f） $t \leftarrow t - 1$ とする．

（5） $|B| \geq 2$ かつ $t \geq 6$ である間，次の (a)–(f) の操作を繰り返す．
（a） $B$ から分岐点 $b_1$，$b_2$ を取り出す．/* $B \leftarrow B - \{b_1, b_2\}$ */
（b） $S(b_1)$ からリーフ $x$ を取り出し，$S(b_2)$ からリーフ $y$ を取り出す．/* $S(b_1) \leftarrow S(b_1) - \{x\}$，$S(b_2) \leftarrow S(b_2) - \{y\}$ */
（c） $F_S \leftarrow F_S \cup \{(x, y)\}$ とする．
（d） $x$ から $b_1$ への経路 ($b_1$ は含まない) を削除したグラフを改めて $G_S$ とする．$b_1$ が分岐点であり $|S(b_1)| > 0$ ならば，$B \leftarrow B \cup \{b_1\}$ とする．$b_1$ が分岐点でなくなり $|B(b_1)| = 1$ ならば，$B(b_1)$ からリーフ $z$ を取り出し，$z$ の新しい最近分岐点 $z'$ を探索し，$S(z') \leftarrow S(z') \cup \{z\}$ とし，更に，$z'$ が新たなリーフ付き分岐点ならば $B \leftarrow B \cup \{z'\}$ とする．
（e） $y$ から $b_2$ への経路 ($b_2$ は含まない) を削除したグラフを改めて $G_S$ とする．$b_2$ が分岐点であり $|S(b_2)| > 0$ ならば，$B \leftarrow B \cup \{b_2\}$ とする．$b_2$ が分岐点でなくなり $|B(b_2)| = 1$ ならば，$B(b_2)$ からリーフ $z$ を取り出し，$z$ の新しい最近分岐点 $z'$ を探索し，$S(z') \leftarrow S(z') \cup \{z\}$ とし，更に，$z'$ が新たなリーフ付き分岐点ならば $B \leftarrow B \cup \{z'\}$ とする．
（f） $t \leftarrow t - 2$ とする．

（6） $|B| \geq 2$ かつ $t = 4$（このとき，$|B| = 2$ である）ならば，2 つの分岐点を $b_1$，$b_2$，$b_1$

を最近分岐点とする 2 つのリーフを $v_1, v_2$，$b_2$ を最近分岐点とする 2 つのリーフを $v_3, v_4$ とおき，$F_T \leftarrow F_T \cup \{(v_1, v_3), (v_2, v_4)\}$ とする．

(7) リーフ付き分岐点が 1 つであるとき，その唯一の分岐点を $b$ とし，$S(b)$ に属す $t$ 個のリーフを $v_1, v_2, \ldots, v_t$ として，$F_S \leftarrow F_S \cup \{(v_i, v_{i+\lfloor t/2 \rfloor}) \mid 1 \leq i \leq \lceil t/2 \rceil\}$ とする．

(8) 各辺 $(a, b) \in F_S$ に対して，$a, b$ に対応する $G$ の 2 成分間を結ぶ 1 辺を作り，そのような辺からなる集合を $F$ として出力する．

アルゴリズム *2-Edge-Connect_A* はリーフ付き分岐点とそれを最近分岐点とするリーフ集合を格納するために，スタックを (分岐点数 + 1) 個使用している．次節で示すアルゴリズムは，使用するスタック数を少なくし，スタックを 1 個だけ使用し，しかも，スタックに入る要素数は高々3 であるというものである．

### 3.3 スタックを 1 つだけ用いるアルゴリズム

本節では，アルゴリズム *2-Edge-Connect_A* を改良し，使用メモリ量がより少ないアルゴリズム *2-Edge-Connect_B* を示す．アルゴリズム *2-Edge-Connect_A* では，分岐点ごとに，その分岐点を最近分岐点とするリーフの集合を保持するために分岐点数のスタックを使用している．アルゴリズム *2-Edge-Connect_B* では，使用するスタック数を 1 個にして，なおかつ，スタックに入る要素数が最大 3 個であるようにできた．

アルゴリズム *2-Edge-Connect_A* では，リーフの最近分岐点を求めることをすべてのリーフに対して実行してから，付加辺集合を求める手順に入るが，その際には，分岐点とそれを最近分岐点とするリーフ集合の関係を分岐点数個のスタックを用いて保持していた．アルゴリズム *2-Edge-Connect_A* では，1 つのリーフの最近分岐点を求めることを実行するごとに，可能ならば付加辺を求めることも行うことにより，1 個の分岐点についてのスタックを使うだけで適切な付加辺集合を求められるというものである．しかも，その唯一のスタックには高々3 個の要素しか蓄えられない．

次に提案アルゴリズムを示す．分岐点 $b$ を最近分岐点とする（探索済みの）リーフを格納するためのスタック $S$ を用意しておく．スタックに入っている要素数を $|S|$ などと表す．$G_S$ のリーフ $v$ の最近分岐点を $v'$ と表す．

**アルゴリズム** *2-Edge-Connect_B*

**入力**：連結な無向グラフ $G = (V, E)$

**出力**：$G + F$ が 2 辺連結となるような辺数最小の付加辺集合 $F$

(1) $G$ の 2 成分を求め，各 2 成分を 1 点に縮約したグラフ $G_S = (V_S, E_S)$ を作る．$G_S$ のリーフ数 $t$，$G_S$ のリーフ $v_1, v_2, \ldots, v_t$ を求める．$S \leftarrow \emptyset$，$F_S \leftarrow \emptyset$ とする．$t$ が奇数ならば $p \leftarrow true$，そうでなければ $p \leftarrow false$ とする．

(2) $t = 2$ ならば，$F_S \leftarrow F_S \cup \{(v_1, v_2)\}$ として手順 7 に進む．

(3) $t = 3$ ならば，$F_S \leftarrow F_S \cup \{(v_1, v_2), (v_2, v_3)\}$ として手順 7 に進む．

(4) $t \geq 4$ ならば，$i \leftarrow 1$ として，$i \leq t - 3$ である限り次の (a)–(b) の操作を繰り返す．

(a) $v \leftarrow v_i$ とし，$v$ の最近分岐点 $v'$ を求め，次の i から v までの操作のいずれか 1 つを実行する．

(i) $|S| = 0$ ならば，$b \leftarrow v'$ とし，$S \leftarrow S \cup \{v\}$ とする．

(ii) 上記の操作 i が実行されずかつ $p = true$ ならば，$x \leftarrow v$，$S$ の先頭要素を $y$ とし，$F_S \leftarrow F_S \cup \{(x, y)\}$ とする．$v$ から $v'$ への経路 ($v'$ は含まない) を削除したグラフを改めて $G_S$ とし，$|S| = 1$ で $z \in S$ の最近分岐点の更新が必要ならば，新しい最近分岐点を $z$ からの探索で求め，それを改めて $b$ とする．$p \leftarrow false$ とする．

(iii) 上記の操作 i と ii のどちらも実行されずかつ $v'$ と $b$ が異なるならば，$x \leftarrow v$，$S$ から $y$ を取り出し，$F_S \leftarrow F_S \cup \{(x, y)\}$ とする．$x$ から $x'$ への経路 ($x'$ は含まない) と $y$ から $y'$ への経路 ($y'$ は含まない) を削除したグラフを改めて $G_S$ とし，$|S| = 1$ で $z \in S$ の最近分岐点の更新が必要ならば，新しい最近分岐点を $z$ からの探索で求め，それを改めて $b$ とする．

(iv) 上記の操作 i と ii と iii のどれもが実行されずかつ $|S| \leq 2$ ならば，$S \leftarrow S \cup \{v\}$ とする．

(v) 上記の操作 i から iv のどれもが実行されない（すなわち $b = v'$ かつ $|S| = 3$）ならば，$x \leftarrow v$，$S$ から $y$ を取り出し，$F_S \leftarrow F_S \cup \{(x, y)\}$ とする．$x$ から $x'$ への経路 ($x'$ は含まない) と $y$ から $y'$ への経路 ($y'$ は含まない) を削除したグラフを改めて $G_S$ とする．

(b) $i \leftarrow i + 1$ とする．

(5) $|S| = 3$ ならば，$S$ のリーフを $x, y, z$ とし，$F_S \leftarrow F_S \cup \{(x, v_{t-2}), (y, v_{t-1}), (z, v_t)\}$ として，手順 7 に進む．

(6) $|S| = 1$ ならば，$S$ のリーフを $x$ とおき，次の操作 (a)–(b) を実行する．

(a) $v \leftarrow v_{t-2}$ とし，$v$ の最近分岐点 $v'$ を求める．

(b) $v'$ と $b$ が異なるならば，$F_S \leftarrow F_S \cup \{(x, v), (v_{t-1}, v_t)\}$ とする．そうでなければ，$F_S \leftarrow F_S \cup \{(x, v_{t-1}), (v, v_t)\}$ とする．

(7) 各辺 $(a,b) \in F_S$ に対して，$a,b$ に対応する $G$ の 2 成分間を結ぶ 1 辺を作り，そのような辺からなる集合を $F$ として出力する．

### 3.4 アルゴリズムの実装について

ここでは，アルゴリズム 2-Edge-Connect_A および 2-Edge-Connect_B を線形時間で実行するための実装について，重要と思われる事柄について述べる．特に，アルゴリズムの記述においては木の変形を用いているが，実装の際に工夫すれば，木の変形を実際には実行せずに実装可能であることを記す．

2 成分木のデータは，各点の隣接点を線形リストで保持しているものとする．データ構造として，他に，各点 i の次数を保持する配列 deg[i] を用いる．更に，アルゴリズム 2-Edge-Connect_A では，点 i を最近分岐点とするリーフを保持するためのスタック S[i] とリーフ付き最近分岐点の集合を保持するためのスタック B を用いる．アルゴリズム 2-Edge-Connect_B では，分岐点を格納するための変数 b と b を最近分岐点とするリーフを保持するためのスタック S を用いる．スタックについては，PUSH，POP 操作以外にもスタック内の要素数が 1 であるかを判定する操作，スタック内の要素数を戻す操作，スタック内の先頭要素を参照する操作も実行可能であるとする．

点 i の次数 deg[i] の計算は，点 i の隣接点のリストを辿ることで可能である．リーフの最近分岐点を求めるには，リーフからの探索をして，次数 3 以上の点が出現するまで直線的に探索をする．点 i を探索している時には，deg[i] が 2 以下であれば i は通過点であり，3 以上ならば i が最近分岐点である．i が通過点である場合には，次の探索点を見つける作業が必要であるが，その作業に入る前に，deg[i] を 0 に設定（削除扱い）しておけば，次の探索点は点 i の隣接リストを辿って deg[j] が 0 でない隣接点 j として見つけられる．

次に，最近分岐点の更新する場面について記す．$v$ をリーフとし，$v$ の最近分岐点が $u$ であるとする．$v$ から $u$ への経路を $T$ から削除する操作は，その時点で，$v$ から $u$ までの経路上のすべての点 $x$（$u$ は除く）の次数 deg[$x$] は既に 0 になっているので削除は実行済みと考えられるので，単に deg[$u$] の値を 1 減らすだけである．但し，次数を 1 減らした結果，deg[$u$] の値が 2 になり，かつスタック S[$u$] に要素が 1 つだけ入っているならば，S[$u$] 中のリーフについて最近分岐点の更新が必要である．その際も，deg[$u$] の値を 0 にして，上記と同様に，点 $u$ の隣接リストを辿り，deg[j] が 0 でない隣接点 j を探せばそのような j が次の探索点である．引き続き探索を続ければ新たな最近分岐点を見つけられる．

このように実装することで，結局は，グラフのデータを変更することなく，すなわち，実際はグラフ変形をせずに実装可能である．

次数を 0 にして（削除扱いにして），隣接リストを辿る操作は，各点について各アルゴリズム全体で 1 回だけであり，また，スタック B，S[i]，S の維持管理についても，$O(|V|)$ の手間で実行できるので，提案アルゴリズム 2-Edge-Connect_A および 2-Edge-Connect_B はいずれも $O(|V|+|E|)$，すなわち，線形時間で実行可能である．

最後に，使用メモリ量について考察する．グラフを保持するための線形リストの他には，アルゴリズム *2-Edge-Connect_A*，*2-Edge-Connect_B* ともに $G_S$ の各点の次数を保持するための配列 1 個を使用している．スタックについては，アルゴリズム *2-Edge-Connect_A* では，(分岐点数 + 1) 個のスタックを使用し，全体で $|V|$ 個の要素（分岐点またはリーフ）を保持している．一方，アルゴリズム *2-Edge-Connect_B* では，スタック 1 個だけを使用し，そのスタックに入る最大要素数は 3 である．従って，アルゴリズム *2-Edge-Connect_B* は，大きさ $|V|$ の配列 1 個と定数個の変数で実装できる．

## 4. まとめと今後の課題

本稿では，リーフの最近分岐点を用いたグラフの 2 辺連結化アルゴリズムを示した．スタックを複数用いる単純なアルゴリズムと，およびそれを改良した，スタックを 1 個だけ用いる，しかもそのスタックに入る要素数が高々3 であるアルゴリズムを示した．

今後の課題としては，更なる使用メモリ量の削減が可能かどうか検討すること，本稿での使用メモリ量削減手法を文献 5) の 2 点連結化アルゴリズムへ適用することなどが挙げられる．

## 参 考 文 献

1) Eswaran, K. P. and Tarjan, R. E.: Augmentation problems, *SIAM J. Comput.*, Vol.5, No.4, pp.653–665 (1976).
2) Frederickson, G. N. and Ja'ja', J.: Approximation algorithms for several graph augmentation problems, *SIAM J. Comput.*, Vol.10, pp.270–283 (1981).
3) Watanabe, T. and Nakamura, A.: Edge-connectivity augmentation problems, *J. Computer and System Sciences*, Vol.35, No.1, pp.96–144 (1987).
4) Frank, A.: Augmenting graphs to meet edge connectivity requirements, *SIAM J. Discrete Math.*, Vol.5, No.1, pp.25–53 (1992).
5) 花中雄太，間島利也，田岡智志，渡邉敏正：グラフの 2 点連結化問題に対する線形時間アルゴリズムについて，IPSJ SIG Technical Report AL138-1, 情報処理学会 (2012).