Vol.2012-AL-138 No.1
2012/1/28

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

花中雄太[†1]　間島利也[†1]
田岡智志[†2]　渡邉敏正[†2]

グラフの 2 点連結化問題とは，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 点連結となるような，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 点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の探索を基礎とするアルゴリズムを示す．

On a Linear Time Algorithm for 2-Vertex-Connectivity Augmentation Problem

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

The 2-vertex-connectivity augmentation problem of a graph, 2VCA, is defined as follows: "Given an undirected graph $G = (V, E)$, find a smallest edge set F of edges such that $(V, E \cup F)$ is 2-vertex-connected." This paper shows a linear time algorithm for solving 2VCA, which 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 広島国際大学
Hiroshima International University
†2 広島大学
Hiroshima University

1. はじめに

本稿では，グラフの連結度増大問題の中でも最も基本的な問題の一つであるグラフの 2 点連結化問題を扱う．グラフの 2 点連結問題 (2VCA と略記する) は次のように定義される：無向グラフ $G = (V, E)$ が与えられたときに，G に辺集合 F で，G に F を付加することにより得られるグラフが 2 点連結となるような，辺数最小の F を求めよ．より一般に，最小辺付加でグラフを k 点連結にする問題をグラフの k 点連結化問題と呼び，kVCA と表す．グラフが k 点連結であるとは，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 個以上の点を除去する必要があることを意味する．kVCA（$k \geq 3$）に関する結果については文献 1)–8) を参照されたい．$k \leq 4$ までは多項式時間アルゴリズムが知られているが，一般に kVCA を解く多項式時間アルゴリズムが存在するかどうかは未解決である．

2VCA に対する既存結果としては，Eswaran と Tarjan[9] により，付加辺の本数の下界値が証明され，多項式時間アルゴリズムが示されている．2VCA を解く線形時間アルゴリズムとしては，Rosenthal-Goldner アルゴリズム[10]，Hsu-Ramachandran アルゴリズム[11]，Hsu アルゴリズム[12] が提案されている．文献 10) のアルゴリズムは 2VCA の最適解を線形時間で求める初めてのアルゴリズムであり，最適解に含まれる辺を 1 辺ずつ求めるものである．文献 11) は，2VCA を解く線形時間アルゴリズムを示すとともにその並列化を行っている．文献 12) は，従来のアルゴリズムを改良し，より簡単にかつ高速にしたアルゴリズムであり，最適解に含まれる複数の付加辺を同時に求めるようにしたものである．これら 3 つのアルゴリズムはいずれも，時間計算量は線形時間である．入力グラフから得られるブロックカット木と呼ばれるデータ構造を用いて，ブロックカット木を 2 点連結化することで，元の入力グラフを 2 点連結化するものである．最近，文献 13) で新しいデータ構造を用いて 2VCA または 3VCA を解く線形時間アルゴリズムが提案されている．そのアルゴリズムは深さ優先探索や基数ソートを用いている．

本論文では 2VCA を解くアルゴリズムについて，リーフの最近分岐点を求める探索を基礎とする線形時間アルゴリズムを提案する．提案アルゴリズムは，ブロックカット木を使って最適解に含まれる辺を 1 本ずつ求める．この点は，従来解法[10] と同じであるが，複雑な経路探索を用いず，リーフの最近分岐点を求める探索を基礎としている．

本論文の構成は次の通りである．第 2 節で準備として，諸定義，ブロックカット木，付加辺数の下界値など記述する．第 3 節で提案アルゴリズム，第 4 節でまとめと今後の課題を述べる．

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 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．

グラフ G から除去するとグラフが非連結になる点のことを，カット点という．カット点を含まない連結グラフを 2 点連結グラフという．グラフ G の 2 点連結成分とは G の 2 点連結な極大な部分グラフであり，ブロックとも呼ばれる．カット点を 1 点だけ含むブロックをリーフブロック，カット点を含まないブロックを孤立ブロックという．カット点の分離度とは，そのカット点を含むブロックの個数のことである．

2.2 ブロックカット森

グラフ G のカット点とブロックの包含関係により定まる森をブロックカット森という．ブロックカット森の点集合は，c 点と呼ばれる G のカット点に対応する点と b 点と呼ばれる G のブロックに対応する点からなる．ブロックカット森の 2 点が辺で結ばれるのは，2 点が c 点と b 点の対であり，かつ c 点が表す G のカット点が b 点が表す G のブロックに含まれるときまたそのときに限られる．G が連結であるときには，ブロックカット森は木であり，ブロックカット木と呼ばれる．ブロックカット森，あるいはブロックカット木は G から線形時間で構成できる．

T を連結グラフ G のブロックカット森とする．T のリーフ (次数 1 の頂点)u 対して，u から T を探索したときに，初めて訪れる次数 3 以上の点を u の最近分岐点といい，$\mathrm{b}_T(u)$ 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．

2.3 ブロックカット木の変形

T を連結グラフ G のブロックカット木とし，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 (ただし，$\mathrm{b}_T(u)$ 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．提案アルゴリズムでは，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．

2.4 下 界 値

ここでは，グラフ G が与えられたとき，G を 2 点連結化するために G に付加すべき辺の本数の下界値[9] について説明する．

G のリーフブロックの数を t とする．G の孤立ブロックの数を s とする．ただし，G 全体が孤立ブロックになっているならば $s = 0$ とする．G の各リーフブロックには少なくとも 1 辺を付加する必要があり，G の各孤立ブロックには少なくとも 2 辺を付加する必要がある．したがって，G を 2 点連結にするには，$t + 2s$ 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．

G の連結成分数を h とし，G の分離度最大のカット点の分離度を d とする．G の分離度最大のカット点を v_m とすると，G から v_m を除去した後の連結成分数は $d + h - 1$ となる．G を 2 点連結にするためには，$d + h - 1$ 個の連結成分を連結にする必要があるので，$d + h - 2$ 本の辺を付加する必要がある．

以上の議論より，次の付加辺数の下界値 LB を得る．

$$LB = \max\{\lceil t/2 \rceil + s, d + h - 2\}$$

これは，G が連結の場合（$s = 0, h = 1$ の場合）には，単に $LB = \max\{\lceil t/2 \rceil, d - 1\}$ である．提案アルゴリズムでは，LB 本の辺の付加で，G を 2 点連結化する付加辺集合を求める．

なお，下界値は，G のブロックカット森 F の情報を使って簡単に計算することができる．これは，G のリーフブロックの数 t は F のリーフの数，G の孤立ブロックの数 s は F の孤立点の数，h は F の連結成分数，d は F の c 点の最大次数として計算できるからである．

2.5 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[9]．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ブロックのカット点でない点，あるいは孤立ブロック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．このアルゴリズムはブロックカット森から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．

2.6 リーフ接続条件

最小本数の辺の付加で G を 2 点連結にするには，1 本辺を付加するごとに，前節で述べた下界値を 1 だけ減らさなければならない．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[11] を記述する．なお，G は連結であると仮定する．

T を G のブロックカット木とする．u, v を 2 つのリーフとし，辺 (u, v) を付加するものとする．$G' = G + \{(u, v)\}$ とし，G' のブロックカット木を T' とする．

- リーフ数を 2 減らすための条件
  T' のリーフ数が T のリーフ数より 2 小さくなるための十分条件は，T 上の u から v への経路上に，(i) 少なくとも 2 つ以上の次数 3 以上の点が存在する，または，(ii) 少なくとも 1 つ以上の次数 4 以上の b 点が存在する，のいずれかが成り立つことである．
- 最大分離度を 1 減らすための条件
  T' の最大分離度が T の最大分離度より 1 小さくなるための十分条件は，T 上の u から v への経路上に，最大分離度と等しい次数を持つ T のすべての c 点が存在することである．

提案アルゴリズムでは，上記のリーフ数を 2 減らすための条件，最大分離度を 1 減らすための条件のいずれかまたは両方を満たすように，辺付加を行い，1 辺を付加するごとに下界値を 1 だけ減少させる．

3. 2 点連結化アルゴリズム

本節では，グラフの 2 点連結化問題に対する提案解法を示す．なお，本稿では，入力グラフは連結であると仮定する．連結でない場合は，2 点連結化の最適解に含まれることになる (連結成分数) − 1 本の辺を付加して連結グラフを構成し，その連結グラフに対して本アルゴリズムを適用すれば最適解が得られる．

3.1 アルゴリズムの概要

T を G のブロックカット木とする．従来解法は T に対して深さ優先探索などを実施し，最適解を得るための情報を得ているが，提案解法では深さ優先探索を採用せず，リーフの最近分岐点を求めるためのリーフからの探索を基本とする．

各リーフからの探索で最近分岐点を求め，リーフ付き分岐点ごとに，その分岐点を最近分岐点とするリーフの集合を線形リストによるスタックなどで維持管理する．

付加辺を構成する際に，基準となるのは，リーフ付き分岐点の最大次数 d' と，リーフ数 t である．最大次数 d' をもつリーフ付き分岐点の一つを v_m とする．まず，T の分岐点が 1 つだけならば，容易である．t 個のリーフを $v_1, \ldots, v_t$ とする．なお，$d' = t$ である．このとき，付加辺集合 F は，v_m が c 点ならば $F = \{(v_i, v_{i+1}) \mid 1 \le i < t\}$，$v_m$ が b 点ならば $F = \{(v_i, v_{i+\lfloor t/2 \rfloor}) \mid 1 \le i \le \lceil t/2 \rceil\}$ として得られる．

以下，T の分岐点が 2 つ上存在するときの，アルゴリズムの概要を説明する．

$d' - 1 \ge \lceil t/2 \rceil$ の場合には，v_m を最近分岐点とするリーフと v_m と異なるリーフ付き分岐点を最近分岐点とするリーフを見つけ，それらのリーフ同士を結ぶ辺を付加すべき辺とする．$d' - 1 < \lceil t/2 \rceil$ の場合には，v_m と異なる 2 つのリーフ付き分岐点を見つけ，それらのそれぞれを最近分岐点とするリーフ同士をつなぐ辺を付加すべき辺とする．

付加すべき辺を見つけると，次に，ブロックカット木の変形を行う．実際に G に辺を付加して，辺付加後のブロックカット木の構成をするのではないことに注意されたい．ここでいう，ブロックカット木の変形は単なる縮小変形であり，具体的には，リーフ u, v 間に辺を付加するとすると，T から，u と $\mathrm{b}_T(u)$ を結ぶ経路，および v と $\mathrm{b}_T(v)$ を結ぶ経路を削除するだけである．新たに得られた木を T' とする．このとき，付加辺の下界値について比較すると，T' についての下界値が，T についての下界値より 1 だけ小さくなることが証明できる．また，一つの経路を削除するごとに，高々1 つのリーフについて最近分岐点を更新する必要があるがその場合には（元の）分岐点からの探索により最近分岐点を求める．その後，T' を新たに T とおいて，上記のことを分岐点が 2 つ以上存在する限り繰り返す．

3.2 アルゴリズム

入力: 無向連結グラフ $G = (V, E)$
出力: $G + F$ が 2 点連結となる辺数最小の付加辺集合 F

(1) G の 2 点連結成分とカット点を求め，G のブロック・カット木 $T = (V_T, E_T)$ を構成する．$F_T \leftarrow \emptyset$，T のリーフ数を t とする．

(2) $t = 2$ ならば，2 つのリーフ u, v をペアにして，$F_T \leftarrow \{(u, v)\}$ として，手順 8 へ進

む．そうでなければ，すなわち，$t \geq 3$ ならば，次の手順へ進む．

(3) T の各リーフの最近分岐点をリーフからの探索により求め，リーフ付き分岐点のうち次数が最大である点の一つを v_m，さらに，その最大次数を d' とする．/* v_m は c 点とは限らない．*/

(4) T の分岐点が 2 つ以上存在し，かつ t が奇数ならば，次の (a)–(d) を実行する．/* 実行後 t は偶数となる．*/

(a) $d' - 1 \geq \lceil t/2 \rceil$ ならば，$b_1 = v_m$ とし，b_1 を最近分岐点とするリーフを 1 つ選び v_1 とする．そうでなければ，すなわち $d' - 1 < \lceil t/2 \rceil$ ならば，v_m と異なるリーフ付き分岐点を任意に 1 つ選び，b_1 とし，b_1 を最近分岐点とするリーフを任意に 1 つ選び v_1 とする．

(b) リーフ付き分岐点のうち b_1 とも v_m とも異なる分岐点を任意に 1 つ選び，b_2 とし，b_2 を最近分岐点とするリーフを任意に 1 つ選び v_2 とする．

(c) v_1 から b_1 までの経路を T から削除して得られる木を改めて T とする．このグラフ変形によって b_1 が分岐点でなくなり，かつ b_1 を最近分岐点としていたリーフがあるならばそれに対する新たな最近分岐点を b_1 からの探索により求める．加えて，リーフ付き分岐点の最大次数 d'，次数が d' であるリーフ付き分岐点 v_m を更新する．

(d) $t \leftarrow t - 1$，$F_T \leftarrow F_T \cup \{(v_1, v_2)\}$ とする．

(5) T の分岐点が 2 つ以上存在し，かつ $t \geq 6$ である間，次の (a)–(e) を繰り返す．

(a) $d' - 1 \geq \lceil t/2 \rceil$ ならば，$b_1 = v_m$ とし，b_1 を最近分岐点とするリーフを任意に 1 つ選び v_1 とする．そうでなければ，すなわち $d' - 1 < \lceil t/2 \rceil$ ならば，v_m と異なるリーフ付き分岐点を任意に 1 つ選び，b_1 とし，b_1 を最近分岐点とするリーフを任意に 1 つ選び v_1 とする．

(b) b_1 とも v_m とも異なるリーフ付き分岐点を任意に 1 つ選び，b_2 とし，b_2 を最近分岐点とするリーフを任意に 1 つ選び v_2 とする．

(c) v_2 から b_2 までの経路を T から削除して得られる木を改めて T とする．このグラフ変形によって b_2 が分岐点でなくなり，かつ b_2 を最近分岐点としていたリーフがあるならばそれに対する新たな最近分岐点を b_2 からの探索により求める．加えて，リーフ付き分岐点の最大次数 d'，次数が d' であるリーフ付き分岐点 v_m を更新する．

(d) v_1 から b_1 までの経路を T から削除して得られる木を改めて T とする．このグラフ変形によって b_1 が分岐点でなくなり，かつ b_1 を最近分岐点としていたリーフがあるならばそれに対する新たな最近分岐点を b_1 からの探索により求める．加えて，リーフ付き分岐点の最大次数 d'，次数が d' であるリーフ付き分岐点 v_m を更新する．

(e) $t \leftarrow t - 2$，$F_T \leftarrow F_T \cup \{(v_1, v_2)\}$ とする．

(6) T の分岐点が 2 つ以上存在し，かつ $t = 4$ 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) T の分岐点が 1 つだけならば，v_m を最近分岐点とするリーフを $v_1, v_2, \ldots, v_t$ とし，v_m が c 点ならば，$F_T \leftarrow F_T \cup \{(v_i, v_{i+1}) \mid 1 \leq i \leq t - 1\}$，$v_m$ が b 点ならば，$F_T \leftarrow F_T \cup \{(v_i, v_{i+\lfloor t/2 \rfloor}) \mid 1 \leq i \leq \lceil t/2 \rceil\}$ とする．

(8) これまでに求まった，リーフの各ペア $(u, v) \in F_T$ に対して，ペアになったリーフに対応する，G のブロック間を結ぶ 1 辺（カット点でない点同士を結ぶ辺）を作り，そのような辺からなる集合 F を解として出力する．

3.3 アルゴリズムの正当性

本節では，前述のアルゴリズムの正当性の証明について，その概略を示す．

まず，T の分岐点が 1 つだけの場合について説明する．T の分岐点を u とする．u が c 点の場合は，u に対応する G のカット点を G から除去すると $d_T(u)$ 個の連結成分が生じる．これらの連結成分を連結させるような $d_T(u) - 1$ 本の辺を付加する必要がある．手順 7 ではそのような辺集合を求めており，手順 8 で得られる F を付加すると 2 点連結になる．u が b 点の場合は，u に対応する G のブロックが $d_T(u)$ 個のカット点を含んでおり，T のカット点の分離度はすべて 2 である．したがって，効率良くリーフ数を減らしていくことだけを考えればよく，そのためにはリーフのペアを順次作成していけばよい．手順 7 ではそのような $\lceil d_T(u)/2 \rceil$ 本の辺集合を求めており，手順 8 で得られる F を付加すると 2 点連結になる．

手順 6 における，分岐点が 2 つ以上で，かつ $t = 4$ の場合は，分岐点が 2 つあり，それらの次数がどちらも 3 の場合である．よって，リーフ数を 2 減らすための条件と最大分離度を 1 減らす条件を同時に満たすように辺を付加する必要があるが，手順 6 ではそのようにリーフのペアを求めており，手順 8 で得られる F を付加すると 2 点連結になる．

次に，手順 4 について，1 辺の付加で下界値が 1 だけ減少していることを示す．手順 4 では，片方のリーフからの経路だけを除去するので，リーフ数が 1 減るのは明らかであり，t が奇数よりリーフ数の半分の切り上げは 1 だけ減少する．したがって，$d - 1 \geq \lceil t/2 \rceil$ のと

きに最大分離度が 1 減少することを示す．$d-1 \geq \lceil t/2 \rceil$ を仮定する．このとき，t は奇数なので，d を次数とする c 点は T に一つだけ存在し，その点の次数が T の最大次数であること示せる．次数が $\lceil t/2 \rceil + 1$ 以上の点は，リーフ付き分岐点であるので，すなわち，v_m がその唯一の c 点である．手順 4 では，2 つのリーフを選び，その間の経路が v_m を通るようにしており，したがって，1 辺の付加後に最大分離度は 1 減少する．

最後に，T の分岐点が 2 以上でかつ $t \geq 6$ の場合について説明する．この場合も手順 5 で，異なる最近分岐点をもつ 2 つのリーフを選んでいるので 1 辺の付加後にリーフ数が 2 だけ減るのは明らかである．したがって，$d-1 \geq \lceil t/2 \rceil$ のときに最大分離度が 1 だけ減少することを示す．$d-1 \geq \lceil t/2 \rceil$ を仮定する．もし $d-1 > \lceil t/2 \rceil$ ならば，t が奇数のときと同様に，d を次数とする c 点は T に一つだけ存在し，その点の次数が T の最大次数であること示せる．また，$d-1 = \lceil t/2 \rceil$ ならば，最大次数をもつ c 点は多くても 2 つ存在し，もし 2 つ存在するならそれらが T のすべての分岐点である．これらのことから，いずれの場合においても，手順 7 で選ばれるリーフ間の経路上に分離度最大の c 点が含まれており，したがって，1 辺の付加後に最大分離度は 1 減少する．

また，手順 5 の 1 回の繰り返しで，リーフ数が 2 ずつ減っていくので，繰り返しによりいずれは，分岐点が 1 つになるか，リーフ数が 4 になる．手順 6 または手順 7 へ進むことになる．手順 6 または手順 7 の正当性は前述の議論から得られるので，結局，T の分岐点が 2 以上の場合についてもアルゴリズムは正しく最小本数の付加辺集合を求める．

3.4 アルゴリズムの実装

ここでは，提案アルゴリズム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．

ブロックカット木のデータは，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]`，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]`，v_m 以外の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．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である考えられ，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，各点についてアルゴリズム全体で 1 回だけであり，また，v_m，スタック `B`，および `S[i]` の維持管理についても，$O(|V|)$ の手間で実行できるので，提案アルゴリズムは $O(|V|+|E|)$，すなわち，線形時間で実行可能である．

4. まとめと今後の課題

本稿では，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として，リーフの最近分岐点の探索を基礎とするアルゴリズムを示した．従来解法と異なり，深さ優先探索やソーティングアルゴリズムを用いないという特徴がある．また，実装の際の工夫により，グラフ変形を行わずに実行可能である．

本アルゴリズムの基礎であるリーフの最近分岐点探索は，2 点連結化問題だけでなく，2 辺連結化や 3 点連結化など，基本的な連結度増大問題にも適用可能である．特に，2 点連結グラフの 3 点連結化は本アルゴリズムと同様に解くことができる．

実装の節で触れたが，本アルゴリズムの実装では，$|V|+1$ 個のスタックを用いた．今後の課題として，アルゴリズムを工夫し，定数個のスタックで実装できるよう改良することなどがある．

Vol.2012-AL-138 No.1
2012/1/28

参 考 文 献

1) Watanabe, T. and Nakamura, A.: A minimum 3-connectivity augmentation of a graph, *J. Computer and System Sciences*, Vol.46, No.1, pp.91–128 (1993).
2) Hsu, T.-S. and Ramachandran, V.: A linear time algorithm for triconnectivity augmentation, *Proc. 32th Ann. IEEE Symp. on Found. of Comp. Sci.*, pp.548–559 (1991).
3) Hsu, T.-S.: On four-connecting a triconnected graph, *Proc. 33rd Ann. IEEE Symp. on Found. of Comp. Sci.*, pp.70–79 (1992).
4) Hsu, T.-S.: Undirected vertex-connectivity structure and smallest four-vertex-connectivity augmentation, *Proc. 6th Intl. Symp. on Algorithms and Computation*, Lecture Notes in Computer Science 1004, Springer-Verlag, pp.274–283 (1995).
5) Jackson, B. and Jordán, T.: Independence free graphs and vertex connectivity augmentation, *Proc. 8th Intl. Integer Programming and Combinatorial Optimization Conference*, Lecture Notes in Computer Science 2081, Springer-Verlag, pp. 264–279 (2001).
6) Jordán, T.: On the optimal vertex-connectivity augmentation, *J. Combinatorial Theory, Series B*, Vol.63, pp.8–20 (1995).
7) Jordán, T.: A note on the vertex-connectivity augmentation problem, *J. Combinatorial Theory, Series B*, Vol.71, pp.294–301 (1997).
8) Vegh, L.A.: Augmenting undirected node connectivity by one, *Proc. of the 42nd ACM Symposium on Theroy of Computing*, pp.563–572 (2010).
9) Eswaran, K. P. and Tarjan, R. E.: Augmentation problems, *SIAM J. Comput.*, Vol.5, No.4, pp.653–665 (1976).
10) Rosenthal, A. and Goldner, A.: Smallest augmentations to biconnect a graph, *SIAM J. Comput.*, Vol.6, pp.55–66 (1977).
11) Hsu, T.-S. and Ramachandran, V.: Finding a smallest augmentation to biconnect a graph, *SIAM J. Comput.*, Vol.22, pp.889–912 (1993).
12) Hsu, T.-S.: Simpler and faster biconnectivity augmentation, *J. Algorithms*, Vol.45, pp.55–71 (2002).
13) Narayanaswamy, N.S. and Sadagopan, N.: A novel data structure for biconnectivity, triconnectivity, and k-tree augmentation, *Proc. 17th Computing: The Australasian Theory Symposium*, pp.45–54 (2011).