DAGの簡潔表現について

高木　拓也†　　有村　博紀†

† 北海道大学大学院情報科学研究科　〒 060–0814 北海道札幌市北区北 14 条西 9 丁目
E-mail: †{tkg,arim}@ist.hokudai.ac.jp

あらまし　本稿では，Directed Acyclic Graph(DAG) 構造の簡潔表現について考える．近年，簡潔データ構造と呼ばれる，様々なデータ構造について情報理論的下界に近い領域計算量を達成するような表現が提案されている．DAG 構造の表現については，Maruyama らによる CFG 構造の DAG に対する表現や (Maruyama et al., 2013)，Tabei らによる SLP の DAG の簡潔表現（Tabei et al., 2013），Denzumi らによる ZDD の圧縮索引が存在する（Denzumi et al., 2014）．これらのデータ構造は，DAG の source ノード（入次数が 0）と sink ノード (出次数が 0) のノードがそれぞれ 1 つしか存在せず，DAG の各ノードの出次数が高々 2 であるという制限の上で，DAG を 2 つの木構造に分解し表現している．本稿では，source ノードと sink ノードがそれぞれ 1 つのみである DAG について，上記の手法が適応可能な構造へ変形させる手法を与える．

キーワード　DAWG，簡潔データ構造

1. はじめに

閉路のない有向グラフは**有向非巡回グラフ** (Directed Acyclic Graph, DAG) と呼ばれ，コンピュータサイエンスの様々な分野で利用されている．例えば，文脈自由文法 (Context-free Grammar, CFG) の構造や，CFG を拡張した Straight Line Program(SLP) の構造は DAG で表現できる [1, 2]．論理関数を表現するデータ構造である二分決定グラフ (Binary Decision Diagram, BDD) [3] やゼロサプレス型 BDD(Zero-Suppressed BDD, ZDD) [4], 全文索引の 1 つである Directed Acyclic Word Graph(DAWG) [5] なども DAG である．また，近年，**簡潔データ構造** (Succinct Data Structure) と呼ばれる，入力データの最悪時エントロピーよりも小さい追加領域しか用いないにもかかわらず，様々なクエリを高速に解くことができるデータ構造が知られている [6].

本稿では，source ノード（入次数が 0）と sink ノード (出次数が 0) のノードがそれぞれ 1 つしか存在しないという制約下の DAG に対する簡潔表現を考察する．これまで，DAG 構造の表現については，Maruyama らによる CFG 構造の DAG に対する表現や [1]，Tabei らによる SLP の DAG の簡潔表現 [2]，Denzumi らによる ZDD の圧縮索引が存在する [7]．しかしながら，これらのデータ構造は，DAG の source ノード（入次数が 0）と sink ノード (出次数が 0) のノードがそれぞれ 1 つしか存在しないという制約に加えて，DAG の各ノードの出次数が高々 2 であるという制限がある．提案手法では，この出次数の制約がない DAG に対して，出次数が高々 2 になるよう DAG を変形させる手法を与える．これにより，上記の既存研究に似たアイデアが適応可能になる．結果として，通常のポインタを用いた表現ではノード数 n，辺数 m に対して，$O(m \log m)$ bits 必要となる有向非巡回グラフを $m \log m + O(m)$ bits で表現可能である．

初めに，2. 節で必要な定義を準備する．次に 3. 節では，DAG を木構造に分解し表現する手法を示す．最後の節では，まとめと今後の展望を示す．

2. 準　　備

本節では，以降の節で必要な用語と定義を与える．また，本論文で扱う主要なデータ構造である DAG を導入する．

.

2.1 有向非巡回グラフ (DAG)

有向グラフ $G = (V, E)$ が閉路を持たない時，G は**有向非巡回グラフ** (Directed Acyclic Graph, DAG) と呼ばれる．有向非巡回グラフ G のあるノード v の入次数が 0 のとき，v を *Source* ノードと呼び，*Source*(G) と表す．同様に，G のあるノード v の出次数が 0 のとき，v を *Sink* ノードと呼び，*Sink*(G) と表す．Source ノードと Sink ノードの個数がそれぞれ定数個であるような有効非巡回グラフ G は，文脈自由文法の表現や二部決定グラフ，全文索引の DAWG など様々な分野で用いられている．

2.2 長男次弟表現

与えられた多分木 T を 2 分木で表現する方法として，**長男次弟表現** (Left-Child Right-Sibring Representation) がある [8]．多分木のあるノード v がもつ子ノードの集合を C とする．長男次弟表現上では C を連結リストの形で表現し，v は C の一番最初の子ノード c へのポインタのみをもつ．これにより，すべての内部ノードは高々 2 つのポインタのみを持つ．

2.3 簡 潔 構 造

N を入力が取り得る種類数とする．すなわち入力は $O(\log N)$ bits で表現されている．ここで $o(logN)$ bits の追加領域で各種操作を高速化するデータ構造を簡潔データ構造は呼ばれる．

順序木に対しては，LOUDS [9] など多くの簡潔データ構造が知られている．

定理 1 ([9]). ノード数 n で，内部ノードがちょうど *2* つの子

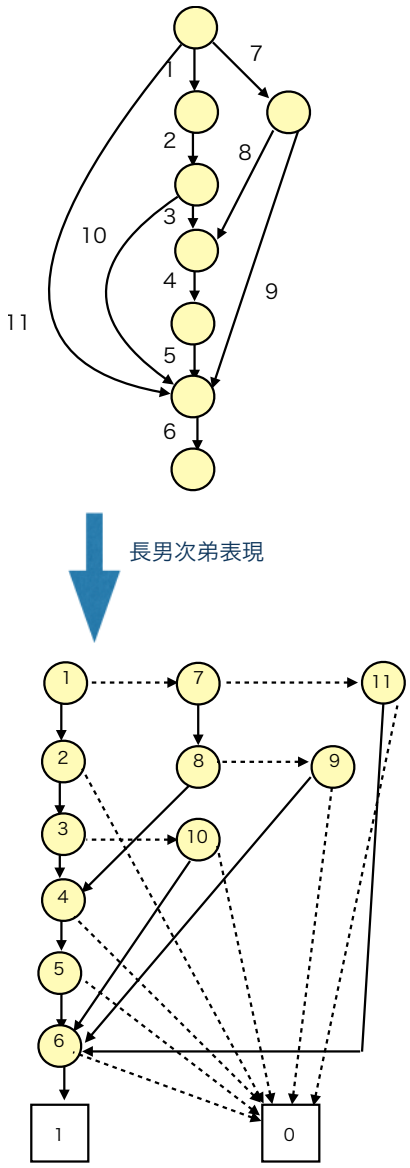

図 1　長男次弟表現による有効非巡回グラフの変形

ノードを持つような順序木 T を $2n + o(n)$ *bits* で表現し，あるノード v の子ノード c の計算，親ノード u の計算は定数時間で実行可能なデータ構造が存在する．

3.　提案アルゴリズム

この節では，*source* ノードと *sink* ノードのノードがそれぞれ *1* つしか存在しないという制約の元，有向非巡回グラフに対するよりコンパクトな表現を与える．

3.1　長男次弟表現の有向非巡回グラフへの拡張

まず，有向非巡回グラフ G に対する長男次弟表現を定義する．これにより，ノード数 n で辺数 m の有向非巡回グラフ G をノード数 m で辺数 $2m$ の有向非巡回グラフとして表現する．

以下に，*source* ノードと *sink* ノードのノードがそれぞれ *1* つしか存在しない有向非巡回グラフ G を *source* ノードと *sink* ノードのノードがそれぞれ *1* つしか存在せず各ノードの出次数は高々 2 つの有向非巡回グラフ G' に変形する方法を示す．まず，G の各辺に *ID* をつける．この各辺が G' のノードとなる．つぎに G のノード v から出ている辺の集合 $edge(v)$ を考える．辺集合 $edge(v)$ に対応する G' のノード集合を連結リストで表現する．そして G' の各ノードは各連結リストの先頭ノードへのポインタをもつ．このポインタを *1* 辺と呼ぶ．また連結リストにおける辺を *0* 辺と呼ぶ．これにより，各ノードの出次数は *1* 辺と *0* 辺の高々 2 つになる．

3.2　木への分解と簡潔表現

こうして得られたグラフ G' の各連結リストの末尾を共通の仮想ノードに連結する．これを 0 ノードと呼ぶ．また，G の *sink* ノードへ向かう辺に対応する G' のノードは共通の仮想ノードへ連結する．これを 1 ノードと呼ぶ．このとき，*1* ノードを根とし *1* 辺からなるグラフは木となる．これを 1 木と呼ぶ．また，*0* ノードを根とし *0* 辺からなるグラフも木となる．これを 0 木と呼ぶ．これらの木構造は順序木に対する簡潔データ構造で $O(n)$ *bits* で表現可能である．また，*1* 木と *0* 木の各ノードは 1 対 1 対応している．よって 1 木と 0 木の各ノードに *ID* をふり，ノード間の対応をサイズ m の順列として保持できる．これにより，有向非巡回グラフを表現可能である．

定理 2. *source* ノードと *sink* ノードのノードがそれぞれ *1* つしか存在しない有向非巡回グラフ G は，$m \log m + O(m)$ *bits* で表現可能である．

4.　まとめと今後の課題

本稿では *Directed Acyclic Graph(DAG)* 構造の簡潔表現について考察した．その結果，*source* ノード（入次数が *0*）と *sink* ノード（出次数が *0*）がそれぞれ 1 つのみである *DAG* について，木構造に対する長男次弟表現を *DAG* 構造に応用することで．既存の簡潔表現が可能な構造へ変形させる手法を提案した．今後の課題として，より一般的な *DAG* 構造の簡潔表現の提案や *DAWG* などの特別な構造をもった *DAG* 構造に対するより効率の良い表現の提案が必要である．

文　献

[1] *Shirou Maruyama, Masaya Nakahara, Naoya Kishiue, and Hiroshi Sakamoto. Esp-index: A compressed index based on edit-sensitive parsing.* J. Discrete Algorithms, *Vol. 18, pp. 100–112, 2013.*

[2] *Yasuo Tabei, Yoshimasa Takabatake, and Hiroshi Sakamoto. A succinct grammar compression. In* Combinatorial Pattern Matching, 24th Annual Symposium, CPM 2013, Bad Herrenalb, Germany, June 17-19, 2013. Proceedings, *pp. 235–246, 2013.*

[3] *S. B. Akers. Binary decision diagrams.* IEEE Trans. Comput., *Vol. 27, No. 6, pp. 509–516, June 1978.*

[4] *Shin-ichi Minato. Zero-suppressed bdds for set manipulation in combinatorial problems. In* DAC, *pp. 272–277, 1993.*

[5] *A. Blumer, J. Blumer, D. Haussler, R. Mcconnell, and A. Ehrenfeucht. Complete inverted files for efficient text retrieval and analysis.* J. ACM, *Vol. 34, No. 3, pp. 578–595, 1987.*

[6] *Gonzalo Navarro.* Compact Data Structures - A Practical Approach. *Cambridge University Press, 2016.*

[7] *Shuhei Denzumi, Jun Kawahara, Koji Tsuda, Hiroki Arimura, Shin-ichi Minato, and Kunihiko Sadakane. Densezdd: A compact and fast index for families of sets.*

In Experimental Algorithms - 13th International Symposium, SEA 2014, Copenhagen, Denmark, June 29 - July 1, 2014. Proceedings*, pp. 187–198, 2014.*

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. *The MIT Press, 2 edition, 2001.*

[9] Guy Joseph Jacobson. Succinct Static Data Structures. *PhD thesis, Pittsburgh, PA, USA, 1988. AAI8918056.*