レシピフローグラフへの簡潔データ構造の適用

Representation of recipe flow graphs in succinct data structures

並木 拓哉　尾崎 知伸 *
Takuya Namiki　Tomonobu Ozaki

日本大学 文理学部
College of Humanities and Sciences, Nihon University

Abstract: The recipe flow graphs are directed acyclic graphs for representing the procedures in cooking recipes. It is expected for recipe flow graphs to contribute the research for understanding the entire recipe texts. However, the recent rapid increase of recipe data and flow graphs posted in community sites on cooking activities causes the problems of the decrease in processing speed and pressure on storage capacity. In this research, in order to cope with these problems, we apply a simplified representation of a graph called GLOUDS to recipe flow graph data. In addition, initial experiments using real world dataset are conducted to assess the effectiveness of the proposed data structures.

1　はじめに

レシピフローグラフ [6, 7] とは，森・山肩らによって提案された料理レシピの抽象表現であり，各ノードをレシピ用語，各エッジをノード間の関係で表現する根付き有向無閉路グラフである．レシピフローグラフは，グラフ表現を利用した料理手順の理解に加え，固有表現認識や述語項構造解析，共参照解析といったレシピテキスト全体の理解に必要とされる諸技術への貢献が期待されている．クックパッドでは，2018 年 11 月末に全世界合計で投稿レシピ数が 500 万件を突破し [13]，レシピ文書やレシピフローグラフを用いた料理理解に関する研究がますます重要となるが，その一方で，データの増加に伴う処理速度の低下とデータ記憶領域の圧迫が一つの問題となると予想される．

簡潔データ構造 [11] とは，情報理論的下界に近い領域量だけを使いつつ，検索等の処理を行う際にはあたかも非圧縮のデータに対してアクセスしているように扱えるデータ構造である．近年の情報サービスやビッグデータの増加に伴い，簡潔データ構造の持つ高い空間効率は注目を集めている．その一種である順序木を表現する LOUDS[5] は，非常に少ないメモリで木構造を表現しながら，高速な検索を実現している．これまでに，そのパフォーマンスの向上を図る LOUDS++[10] や，LOUDS を応用したグラフの簡潔表現 GLOUDS[3]，ファイルシステムの簡潔表現 FLOUDS[9]，LOUDS におけるビットベクトルに付与する索引の最適化 [12] など，LOUDS に関連した様々な応用が提案されてきた．

本研究では，LOUDS と GLOUDS に基づくレシピフローグラフデータの簡潔データ表現を提案・実装する．また，それらを処理速度と空間効率の観点から検証し，その実用性を評価する．

本論文の構成は以下の通りである．2 章でレシピフローグラフについて述べる．3 章で LOUDS と GLOUDS に関連する簡潔データ構造の基礎事項を示す．4 章では簡潔データ構造を用いたレシピフローグラフデータの実装について述べる．5 章で実験と考察を行い，最後に 6 章でまとめと今後の課題を述べる．

2　レシピフローグラフ

料理レシピのコミュニティサイトに投稿されるレシピは主に，タイトル・食材リスト・調理手順の形式に従い記述される．レシピフローグラフ [6, 7] は，このうちの調理手順に対する意味内容の表現形式であり，各ノードが食材や道具，継続時間などのレシピ用語，各辺がレシピ用語間の関係を表す．図 1 に「鶏肉の唐揚げ」料理に対する（簡略化した）レシピフローグラフの例を示す．グラフで示される通り，食材と道具に対して調理動作が繰り返され，最後の調理動作（「揚げ」）が完成品（唐揚げ）に相当する．

なおレシピフローグラフは本来辺ラベルを有するが，本研究ではグラフ構造とノードラベルのみを扱うこととし，辺ラベルはモデル化の対象外とする．

*連絡先：日本大学文理学部情報科学科
〒 156-8550 東京都世田谷区桜上水 3-25-40
E-mail: tozaki@chs.nihon-u.ac.jp

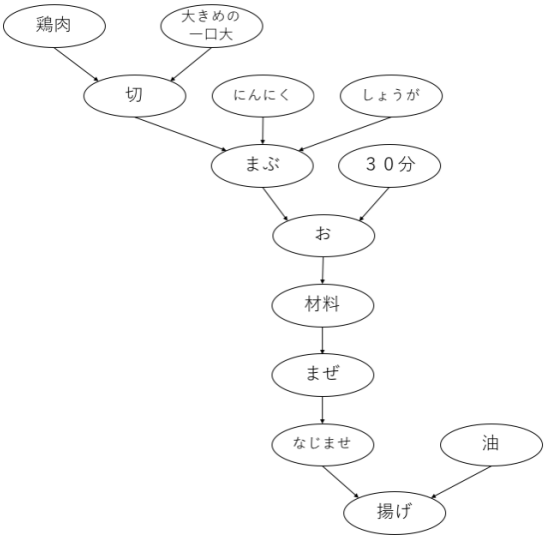


図 1: 鶏肉の唐揚げに関するレシピフローグラフの例

3　木構造・グラフの簡潔表現

本章では，木およびグラフ構造に対する簡潔表現技術の概要を示す．なお，計算モデルとして語長 w の word-RAM モデルを用い，CPU は w ビットの 2 つの整数の論理演算や四則演算を定数時間で実行できるとする．

3.1　rank と select

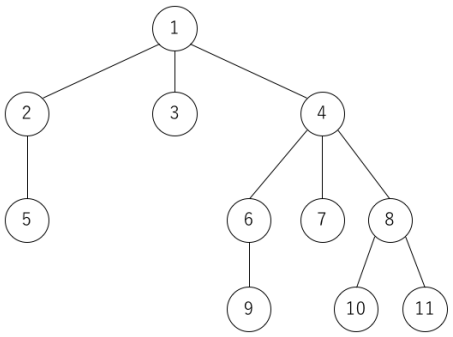
$B \in \{0,1\}^n$ を長さ n のビットベクトルとする．$i \in \{0,1,\ldots,n-1\}$ に対し，$B[i]$ を B の位置 i の値とする．また $i,j \in \{0,1,\ldots,n-1\}$ に対し，$B[i..j]$ を $B[i], B[i+1], \ldots, B[j]$ からなるビットベクトルとする．このときビットベクトル B に対し，次の 2 つの演算を定義する．

- $rank_x(B,i)$: $B[0..i-1]$ 中の x の数を返す
- $select_x(B,i)$: B 中の先頭から i 番目の x の位置を返す

これらの演算は，付加情報（索引）を用いない場合，線形時間を要する．これに対し Jacobson[5] は，$O(n \log\log n/\log n)$ ビットの領域を使用し，rank を $O(1)$ 時間，select を $O(\log n)$ 時間で求める索引を提案した．またその後，データ構造が改良され，$O(n \log\log n/\log n)$ ビットの索引で select を定数時間で求めることが可能になっている．

3.2　LOUDS

これまでに，ノード数 n の根付き木を $2n$ ビットで表現する方法がいくつか提案されており [5, 8, 1]，LOUDS[5] はその代表的な方法の一つである．LOUDS は，まず空のビットベクトルとして B を初期化する．その後，対象となる木に対して幅優先探索を行いながら，訪問した各ノードに対して子の数だけのビット 1 と一つのビット 0 を B に付加する操作を繰り返す．この操作を通じて得られるビットベクトル B を LOUDS と呼ぶ．LOUDS では，各ノードが親として見做されたときは 0，子として見做されたときは 1 で表現される．したがって，LOUDS のビット長は $2n+1$ となる．また 1 と 0 の両方のビットが順序を保ったまま表現されるため，各ノードを識別することが可能である．LOUDS の例を図 2 に示す．



B = 1 0 1 1 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0 0 0 0

図 2: 順序木およびその LOUDS

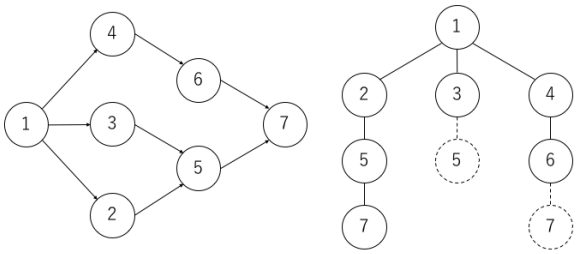
LOUDS は，rank, select 演算のための索引を持つ．これにより，定数時間で実行される rank, select 演算を通じ，任意のノード $i \in \{1,2,\ldots,n\}$ の親ノードおよび子ノードの番号，および子の数を効率的に求めることができる．

3.3　GLOUDS

GLOUDS[3] は，LOUDS を基にした有向グラフ G に対する簡潔データ構造である．

簡略化のため GLOUDS では，根ノード，すなわち全てのノードへの経路が存在するノード root を仮定する．root から幅優先探索を行い，その結果を T_G^{BFT} とする．ノード v の幅優先探索中に検査されたが訪問されていない各ノード w に対し，w のコピー（シャドウノード）を作成し，それを幅優先探索木の v の子として追加する．こうして得られる木を T_G と表記する．

GLOUDS では，通常のノードとシャドウノードを区別するために，$\{0,1,2\}$ からの系列としてトリットベクトル B を構築する．まず B を，2 トリット 1,0 と初期化する．次に T_G の各ノードをレベルオーダで訪問し，その子が通常ノードならばトリット 1 を，シャ



$B = 1011101020101020 0$

$H = 57$

図 3: グラフおよびその GLOUDS

ドウノードならばトリット 2 を B に付加する．子の検査が終了すると，B にトリット 0 を付加する．ここで，シャドウノードは訪問されないので，B においては 2 でのみ表現される．また B に加え，T_G に対する対するナビゲーション操作のために，B 中に現れるシャドウノードを順にリストするベクトル H を構築する．B と同様 H に対しても，効率的な rank，select 演算のための索引が必要となる．索引には，ウェーブレット木 [4] やウェーブレット行列 [2] が用いられる．GLOUDS の例を図 3 に示す．図中においてシャドウノードは点線で表される．

3.4 ウェーブレット木

ウェーブレット木は，整数列（あるいは文字列）に対するデータ構造である．図 4 に示すように，上位のビットから順にその値に注目しながら，整数を再帰的に左右にフィルタリングすることで，木を構築する．各ノードは保持するビット列に対して索引を持ち，rank，select を効率的に求めることができる．整数列に対する rank，select は，対象の整数 n の各ビットの値に沿って木を辿り求める．ウェーブレット木における rank，select では，木の高さ（すなわち，整数の表現に使用される最大ビット長）が実行時間にそのまま影響する．分岐数を増やせば木の高さを抑えることができるが，保持するポインタの増加により，実行時間と空間使用量のトレードオフが発生する．

図 4: ウェーブレット木

4 レシピフローグラフの簡潔表現

本章では，レシピフローグラフデータベースに対する LOUDS および GLOUDS の適用について検討する．

4.1 簡潔表現構築の単位

簡潔表現の構築単位については，(1) 各フローグラフに対して一つの簡潔表現を構築する，または (2) 複数のフローグラフ全体に対して一つの簡潔表現を構築するの 2 つが考えられる．LOUDS および GLOUDS では，ビット（トリット）ベクトルに対する索引を必要とするが，ビット（トリット）ベクトルのサイズが大きくなるほど，それに対する索引サイズの割合が小さくなるという特徴を有する．この特徴を踏まえ，索引の割合を抑えるために，本研究では，複数のデータを一つにまとめ，それに対して一つの簡潔データ表現を構築することとする．

4.2 GLOUDS の適用

本節では，レシピフローグラフに対する簡潔表現の一つとして，GLOUDS を適用することを考える．便宜上，これを手法 1 と呼ぶ．

レシピフローグラフは一般に，(材料に相当する）親を持たないノードを複数持つ．GLOUDS では，全てのノードに連結されている root を必要とするため，これらの親なしノードへエッジを持つ仮想的な「開始ノード」をレシピフローグラフに付与し，これを根とする．また各レシピフローグラフの根を繋ぐノードを準備する．二つののレシピフローグラフデータから生成した GLOUDS の例を図 5 に示す．

4.3 LOUDS の適用

本節では，レシピフローグラフに対する簡潔表現の一つとして，LOUDS を適用することを考える．便宜上，これを手法 2 と呼ぶ．

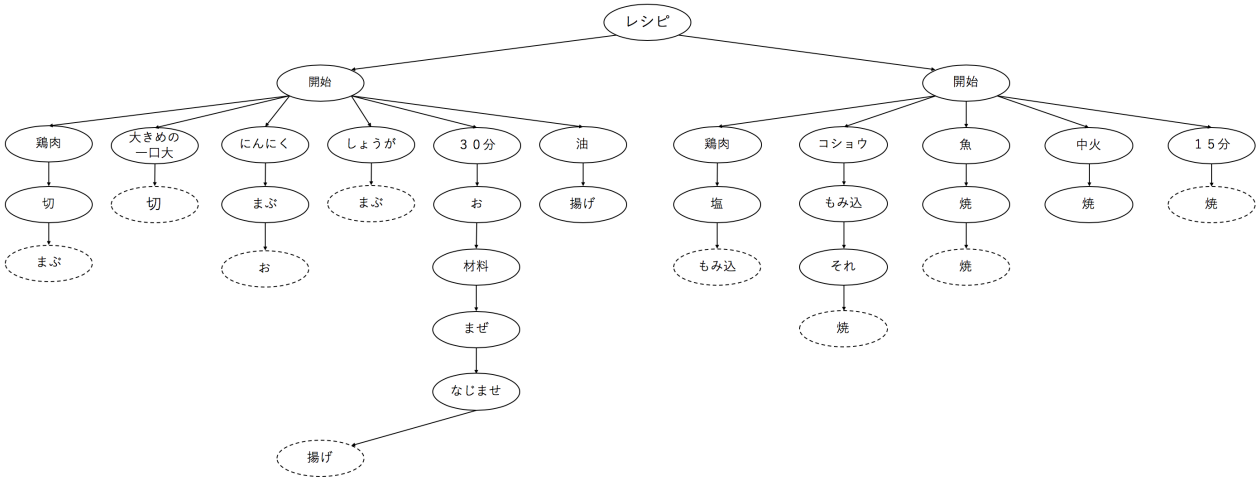


図 5: 手法 1 : GLOUDS のためのレシピフローグラフの木構造表現

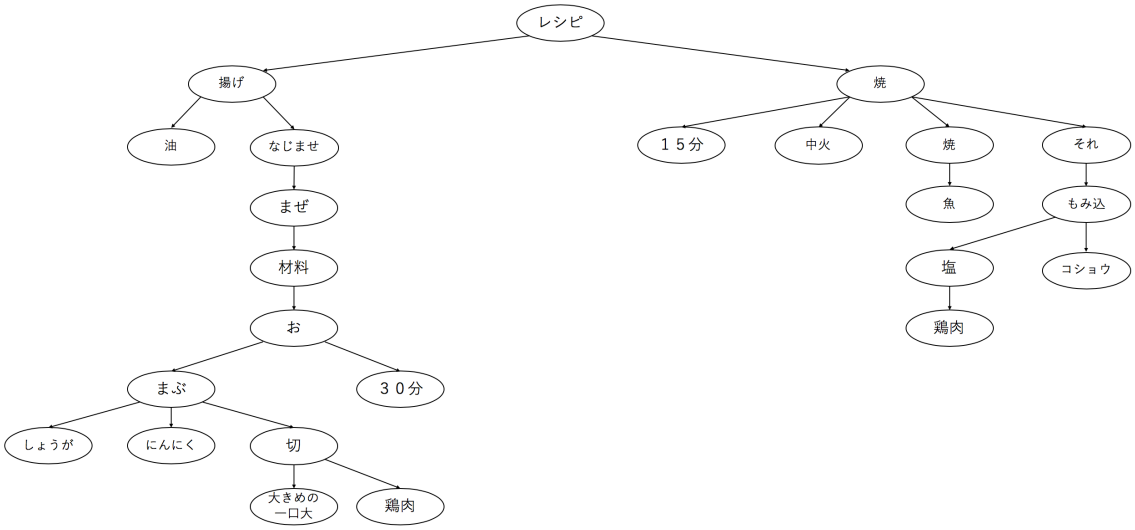


図 6: 手法 2 : LOUDS のためのレシピフローグラフの木構造表現

手法 2 では，レシピフローグラフの完成品ノードを除く全てのノードがただ一つの子を持つという特徴を利用する．ここで，グラフの各ノードの親子関係を反転させると，完成品ノードを根とし，完成品ノード以外がただ一つの親を持つ木構造と見做すことができる．手法 2 では，この木構造を LOUDS によって表現する．二つのレシピフローグラフデータから手法 2 に依り生成した LOUDS を図 6 に例を示す．

ここで手法 1 と手法 2 を簡単に比較する．手法 1 では，各レシピの根の配下に元々親なしであったノードが存在する．これらのノードはレシピ中における所謂材料を表していることが多いため，例えばある材料を持つレシピ数，という問い合わせがあれば，それらのノードが葉として木の深い部分に存在している手法 2 に比べて高速に答えられることが見込まれる．一方手法 2 では．手法 1 とは異なりシャドウノードが存在せず，その構造の表現が簡潔になっている．

4.4 実装

今回の実装では，LOUDS のビットベクトル B は 8 ビットを 1byte で表現した．また GLOUDS で利用するトリットベクトル B に関しては，5 トリットを 1byte で表現し，索引付け H の実装にはウェーブレット木 [4] を用いた．ノードラベルに関しては，LOUDS によるトライ木を構築，利用した．なおその際，GLOUDS・LOUDS におけるノード番号に対して，そのノードが

表 1: メモリ使用量

手法	与式	値
配列（完成品が子なし）	エッジ数 $*$ $4byte$	231.31 MB
配列の配列（完成品が根）	(ノード数 $*$ 2 + エッジ数) $*$ $4byte$	706.00 MB
提案手法 1	(トリット列の長さ/5)$byte$ + シャドウノード数 $*$ $4byte$	63.95 MB
提案手法 2	(ノード数 $*$ 2 + 1)bit	14.83 MB

持つラベルの LOUDS トライ木におけるノード番号をマッピングした．

ところで，LOUDS や GLOUDS の構築には幅優先探索による木の巡回が必要とされ，一般に大量のメモリを必要とする．実際の実装ではメモリ不足を避けるために，各レシピデータに対する簡潔表現を一時ファイルに出力し，後処理としてそれらを合成するという処理を行っている．

5 評価実験

5.1 実験データ

実験には，クックパッド株式会社と国立情報学研究所が提供しているクックパッドデータセット[1] を基に，森・山肩らが構築したレシピフローグラフデータセット[2][6, 7] を用いた．なお，データセットに含まれるレシピフローグラフの総数は 1,582,541 であり，各グラフに含まれるノード数の平均は 39.31，標準偏差は 24.07 である．

対象全データを，4 つの方法で表現した場合のメモリ使用量（理論値）を表 1 に示す．ここでは，ノード間の関係のみを表現するとし，ラベルの保存については考慮しない．配列による表現は，手法 1 と同じグラフ構造を対象とし，各ノードが持つ子のノード番号を配列で保持する．配列の配列による表現は，手法 2 と同じ木構造を対象とし，子の配列へのポインタ，子のノード番号，子の配列の長さを保持する．手法 1 による表現は，トリット列とシャドウノードの登場順のリストを保持する．手法 2 による表現は，ビット列を保持する．

共通の構造を表現する手法同士のメモリ使用量を比較すると，配列と手法 1 では，手法 1 が配列の約 27% に圧縮され，配列の配列と手法 2 では，手法 2 が配列の配列の約 2% に圧縮されていることがわかる．

[1] https://www.nii.ac.jp/dsc/idr/cookpad/cookpad.html
[2] http://www.ar.media.kyoto-u.ac.jp/data/recipe/

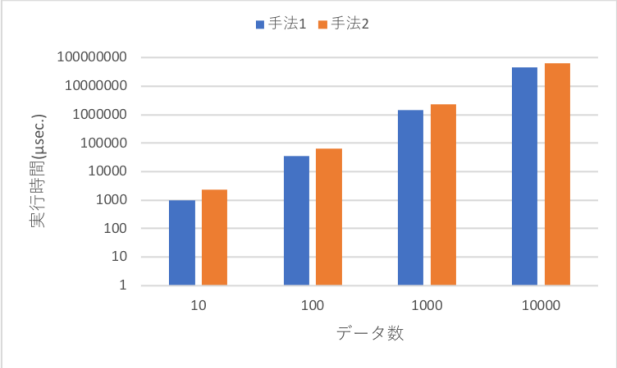


図 7: クエリ 1 の実行時間

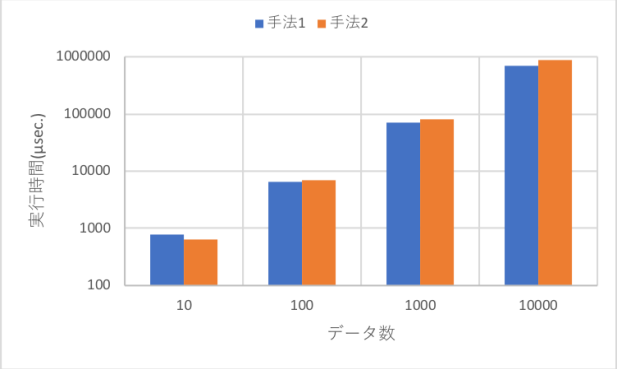


図 8: クエリ 2 の実行時間

5.2 評価実験と考察

複数のフローグラフデータから簡潔表現を生成し，以下に示す実用的な問い合わせを行い，その実行時間の比較を行う．

- クエリ 1：材料に豚肉を含むレシピ数を返す
- クエリ 2：手順数が 20 ステップ以内のレシピ数を返す

対象レシピ数を，$10^1, 10^2, 10^3, 10^4$ へと変化させた場合の各クエリの実行時間を図 7 と図 8 にそれぞれ示す．結果より，いずれのクエリにおいても，データ数 $= 10^1$ の場合を除いて手法 1 の実行時間が優れていることが分かる．

クエリ 1 では，手法 1 がレシピの材料ノードを根の直下に持っているため，所望のラベルをいち早く見つけることができるのに対し，手法 2 では材料ノードが木の深い部分に集中していることが多く，探索に時間を要してしまっていることが結果の差を広げてしまっていると考えられる．一方クエリ 2 では，手法 1 の完成品を除く全てのノードが子を必ず持っているため新たなノードを走査する可能性が高いのに対し，手法 2 では材料ノードが子を持たないために新たなノードを走査する可能性が低く，その差が結果の差を広げてしまっていると考えられる．

空間効率では手法 2 に軍配があがるが，ラベルを使用した実用的な検索では，根の直下に材料ノードを持つ手法 1 の方が高速に答えることが明らかになった．これらの結果から，実用的な検索に速度を要する処理では手法 1，記憶領域を優先する処理では手法 2 が望ましいと考えられる．空間使用量と処理速度にトレードオフが生じることは，簡潔データ構造においては一般的である．

6 まとめと今後の課題

本研究では，巨大化するレシピフローグラフデータベースに対する簡潔表現を実装し，初期的な実験結果を以てその有用性を示した．今後は，各研究に具体化した実装や実験，さらなる効率化の検討を行う必要があると考えられる．

謝辞： レシピフローグラフをご提供頂きました東京大学の山肩洋子氏，京都大学の森信介教授に感謝いたします．また本研究では，クックパッド株式会社と国立情報学研究所が提供する「クックパッドデータ」を利用しました．

参考文献

[1] D. Benoit, E. D. Demaine, J. I. Munro, R . Raman, V. Raman, and S. S. Rao: Representing trees of higher degree, *Algorithmica 43 (4)*, pp.275–292 (2005)

[2] F. Claude and G. Navarro: The Wavelet Matrix, *SPIRE*, pp.167–179 (2012)

[3] J. Fischer and D. Peters: GLOUDS: Representing tree-like graphs, *Jounal of Discrete Algorithms*, Vol.36, pp.39-49 (2016)

[4] R. Grossi and A. Gupta: High-order entropy-compressed text indexes, *SODA '03*, pp. 841–850 (2003)

[5] G. Jacobson: Space-efficient static trees and graphs, In *Proc. 30th SFCS*, pp.549–545 (1989)

[6] 前田 浩邦, 山肩 洋子, 森 信介: 手順文書からの意味構造抽出, 人工知能学会論文誌, Vol.32, No.1, pp.1–8 (2017)

[7] 森 信介, 山肩 洋子, 田中 克己: レシピテキストのためのフローグラフの定義, 情報処理学会研究報告, 2013-NL-214(13), pp.1–7 (2013)

[8] J.cI. Munro and V. Raman: Succinct representation of balanced parentheses and static trees, *SIAM J. Comput. 31 (3)*,pp.762–776 (2001)

[9] D. Peters, J. Fischer, F. Thiel, and J.-P. Seifert: FLOUDS: A Succinct File System Structure, *ACSIS*, Vol.12, pp.51–57 (2017)

[10] N. Rahman and R. Raman: Engineering the louds succinct tree representation, In *Proc. of WEA*, p.145 (2006)

[11] 定兼邦彦 (2018). 簡潔データ構造 共立出版

[12] D. Zhou, D. G. Andersen and M. Kaminsky: Space-Efficient, High-Performance Rank & Select Structures on Uncompressed Bit Sequences, *SEA*, pp.151–163 (2013)

[13] "クックパッド、投稿レシピ数が世界 500 万品を突破！～ 海外進出を加速マレーシアが新たに加わり 70 カ国でサービスを展開 ～". cookpad. https://info.cookpad.com/pr/news/press_2018_1220, (参照: 2019/01/14)