D-009

重み付き有向非巡回グラフに対する効率良いテキスト索引の構築アルゴリズム
An Efficient Index Construction Algorithm for Weighted Directed Acyclic Graphs

山岸　大騎 *,‡ Daiki Yamagishi　高木　拓也 * Takuya Takagi　渋谷　哲朗 † Tetsuo Shibuya　有村　博紀 * Hiroki Arimura

1. はじめに

ソーシャルネットワーク（SNS）や，地理情報システム（GIS），生命情報ネットワークなどの大規模グラフデータと，その上の行動・機能情報のモデルとして，ネットワークの辺に遷移確率を表す重みと情報をラベルとしてもつ重み付き DAG（weighted DAG, WDAG）が広く用いられている．確率モデルの一種である隠れマルコフモデルも WDAG で表現できる．

本稿では，あらかじめ固定されたパラメータ $z \geq 1$ に対して，入力パターンに文字列ラベルが一致し，パス上の重みの積が $1/z$ 以上であるようなパスを検索する問題として，WDAG における重み付き文字列照合問題を導入する．主結果として，入力として頂点数 n と辺数 m の WDAG $G = (V(G), E(G), \bot, lab, w)$ と重みしきい値 $z \geq 1$ を受け取り，任意のパターン P に対する重み付き文字列照合クエリを $O(|P| \log \sigma + occ)$ 時間で実行可能な，$O(zn)$ 語のテキスト索引を与える．さらに，この $O(zn \log(n+m))$ 時間で構築可能なことを示す．

本結果は，Barton ら [2] による重み系列（position weight matrix, PWM）に対する重み付き文字列照合のためのテキスト索引を，任意の WDAG に一般化したものである．

1.1 WDAG

$\Sigma = \{a, b, \ldots\}$ を文字の有限集合とする．WDAG は，多重辺を許した有向非巡回グラフ $\underline{G} = (V(G), E(G))$ と，**シンク**と呼ばれる特別なノード $\bot \in V(G)$，**ラベル関数** $lab : E(G) \to \Sigma$，**重み関数** $w : E(G) \to [0, \infty)$ からなる組 $G = (V(G), E(G), \bot, lab, w)$ である．ここに，任意のノード v から $\bot$ への有向パスが存在すると仮定する．また，各ノード $x \in V(G)$ に対して，x の出辺の重み和は 1 以下とする．図 1 に，WDAG の例を示す．

重みしきい値は，任意の正整数 $z \geq 1$ である．有向パス $\pi = e_1 \ldots e_m \in E(G)^*$ に対して，そのパス**重み**を $w(\pi) := w(e_1) \times \cdots \times w(e_m) \in [0, 1]$ と，そのパス**文字列**を $lab(\pi) := lab(e_1) \cdots lab(e_m) \in \Sigma^*$ と定義する．

与えられた長さ m の文字列 $P \in \Sigma^*$ が，グラフ G 中の有向パス $\pi = e_1 \ldots e_m \in E(G)^*$ に対して，重みしきい値 z に関して**重み付き照合する**とは，次の条件 (i)–(ii) が成立することをいう：(i) π の文字列ラベルが P に一致する，すなわち，$lab(\pi) = P$ が成立する．(ii) π の累積重みが $1/z$ 以上である，すなわち，$w(\pi) \geq 1/z$ が成立する．このとき，上の有向パス π の末端ノード $dest(\pi) := dest(e_m)$ を**照合位置**とよぶ．上の条件 (ii) を満たす有向パス π を，z-**ソリッドパス**（z-solid path）とよぶ．

パターン P を受け取り，その照合位置全てを返す演算を，**重み付き照合クエリ**と呼ぶ．以上の準備のもとで，

*北海道大学大学院情報科学研究科， IST, Hokkaido University
†東京大学医科学研究所ヒトゲノム解析センター，Human Genome Center, The University of Tokyo
‡山岸大騎, e-mail:yamagishi@ist.hokudai.ac.jp

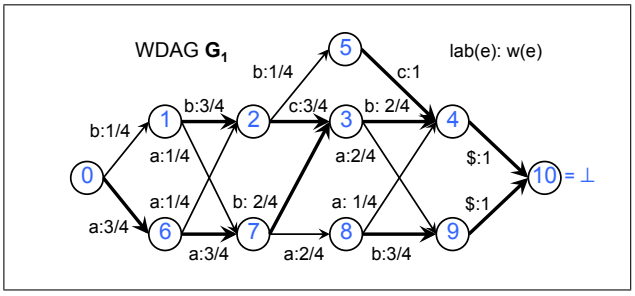

図 1: 文字集合 $\Sigma = \{a, b, c, \$\}$ 上の WDAG の例 $G_1 = (V_1, E_1, lab_1, w_1, sink_1)$．辺のラベル $c : w$ は文字 c と重み w を表す．

本稿の問題を述べる．

定義 1 重み付きグラフに対するテキスト索引構築問題は，入力として *WDAG* $G = (V(G), E(G), \bot, lab, w)$ と重みしきい値 $z \geq 1$ を受け取り，あとで与えられるパターン照合クエリを効率よく実行できるように，G を前処理して索引構造 $\mathcal{I}$ を構築する問題である．

一般に入力サイズは非常に大きいので，索引は，入力の線形程度の領域計算量をもつものが望ましい．

1.2 主結果

はじめに，極大 z-ソリッドパスの基本的性質を示す．次に，これらの性質に基づいて，頂点数 n と辺数 m を持つ WDAG と最小重みしきい値パラメータ $z > 0$ を入力としたとき，任意の文字列パターン P に対する重み付き照合計算を $O(|P| + occ)$ 時間で実現するような $O(zn)$ 語サイズの索引を与える．ここに，occ は P の出現回数である．さらに，この索引の $O(zn \log n)$ 時間構築アルゴリズムを与える．

2. 極大ソリッドパスの基本的性質

以降では，入力の重み付きグラフ $G = (V(G), E(G), \bot, lab, w)$ と重みしきい値 $z \geq 1$ を固定する．有向パス π が**極大** z-**ソリッドパス**（maximal z-solid path）であるとは，$w(\pi) \geq 1/z$ かつ任意の辺 $e \in E(G)$ に対して $w(\pi e) < 1/z$ が成立することをいう．

Amir 他 [1] の重み付き系列に関する結果を一般化して，次の補題を得る．

補題 1 任意の *WDAG* G と，その任意のノード $x \in V(G)$ に対して，x から始まる互いに異なる z に関する極大 z-ソリッドパスは高々 $\lfloor z \rfloor$ 個しか存在しない．

ここで，$\mathcal{M}(G, z)$ で，G 中の z に関する極大 z-ソリッドパス全ての集合を表す．すると，補題 1 から直ちに次が導かれる．

補題 2 任意の *WDAG* G としきい値 $z \geq 1$ に対して，$|\mathcal{M}(G,z)| = O(nz)$.

以後，$N := |\mathcal{M}(G,z)| = O(zn)$ で，極大 z-ソリッドパスの総数を表す．上の補題から，テキスト索引構造の設計の基本戦略として，与えられた重みしきい値 z のもとで，$\mathcal{M}(G,z)$ の要素，すなわち G に含まれる全ての極大 z-ソリッドパスを線形領域で格納し，効率よく検索を行えるようなデータ構造を設計すれば良いことがわかる．

しかし，単純に接尾辞木 $Stree(\mathcal{M}(G,z))$ へ格納するだけでは，z-ソリッドパスは $O(n)$ 文字長であるため，$O(zn^2)$ 語のサイズを必要とする．これは目標サイズに程遠い．

3. 索引構築アルゴリズム

3.1 アウトライン

そこで，提案手法では，次のようにテキスト索引 $\mathcal{I}_G = (\mathcal{CS}_G, \mathcal{ST}_G)$ を構成する．

1. はじめに，極大 z-ソリッドパス集合 $\mathcal{M}(G,z)$ 中の全ての文字列をうまく重ね合わせて，それらを枝のパスとする $O(zn)$ ノードの**非決定性逆トライ** $\mathcal{CS}(G,z)$ を構築する．本来，集合 $\mathcal{M}(G,z)$ の総文字数は $O(zn^2)$ だが，文字列同士をうまく重ね合わせ，小さくする．

2. 次に，いくつかの前処理をする．非決定性逆トライ $\mathcal{CS}(G,z)$ では，もとの極大 z-ソリッドパスの境界が失われているので，それらを表す情報として，*end*-リンクを逆トライの各ノードからその先祖への両方向ポインタとして付加する．

3. 逆トライ $\mathcal{CS}(G,z)$ の根から開始して深さ優先探索を行い，各ノードに入る辺に同じ分岐文字があればマージして，これを決定化する．同時に，*end*-リンクも同時にマージする．次に，逆トライを入力とする任意の接尾辞木構築アルゴリズム（例えば [3, 5]）を，$\mathcal{CS}(G,z)$ に適用し，接尾辞木 $\mathcal{ST}(G,z)$ を構築する．さらに，$\mathcal{CS}(G,z)$ 上の *end*-リンクを移植して，$\mathcal{ST}(G,z)$ をプロパティ接尾辞木 [1] に変換し，索引 $\mathcal{I}(G,z)$ を構築する．

以下の小節では，必要なデータ構造と各ステップを説明する．

3.2 最重みパス木

準備として，最重みパス木の概念を導入する．これは，Barton ら [2] による重み系列に対する最重みパスの概念を，DAG に一般化したものである．G の任意のノード x に対して，その出辺の中で最大重みを持つ辺（**最大重み辺**）$\mathtt{hv}(x) \in E(G)$ をちょうど一つ定める．同じ重みの出辺が複数あるときは，適宜，タイブレークする．

G の任意のノード y に対して，開始ノード y をもつ**最重み接尾辞パス**（heavy suffix path）とは，ノード y から G のシンク $\perp$ へ到達する最重み出辺だけからなる有向パス β である．最重み接頭辞パスは，開始点 y に対して一意に定まるので $\mathtt{hvpath}(y) := \beta$ と書く．最重みパス木を G の最重み接頭辞パス全体 $\mathcal{H}(G) := \{\, \mathtt{hvpath}(y) \mid y \in V(G) \,\}$ の誘導する部分グラフとする．以後，これも $\mathcal{H}(G)$ と書く．各頂点の最重み出辺はただ一つなので，次を得る．

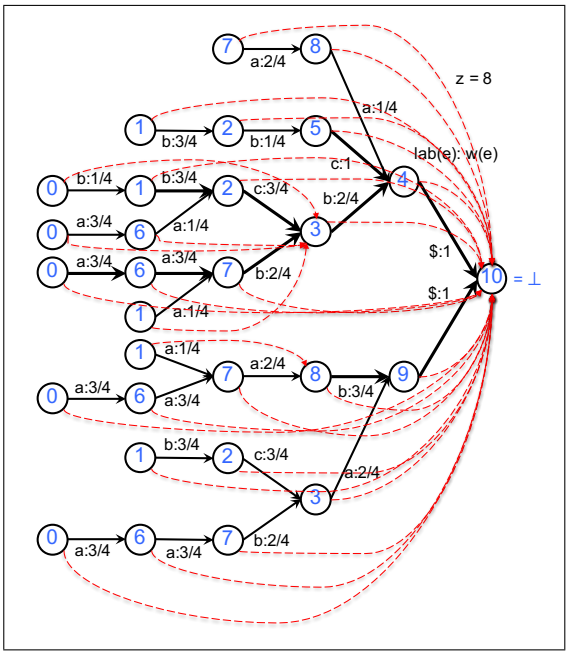

図 2: 図 1 の WDAG G_1 と重みしきい値 $z = 8$ に対する逆トライ $\mathcal{CS}_1$ の例．それぞれの円は，逆トライのノードを表わし，その中の数字は，対応する入力 WDAG の頂点番号を表す．矢印は辺を示し，とくに太い矢印は最重み辺を示す．辺のラベル $c : w$ は文字 c と重み w を表す．赤い点線矢印は *end*-リンクを表す．

補題 3 $\mathcal{H}(G)$ は，$\perp$ を根とする木である．

3.3 逆トライ $\mathcal{CS}(G,z)$

正規 z-ソリッド接尾辞パス．G の接尾辞パスとは，G のあるノード x から開始してシンク $\perp$ へ到達する有向パスである．指数個ある接尾辞パスの中から，**正規 z-ソリッド接尾辞パス**と呼ぶ，次のような形をした特別な有向パス γ だけを考える：(i) $\gamma = \alpha\beta$，かつ，(ii) α は開始ノード x から開始し，ある中間ノード y に到達するある極大 z-ソリッドパス，(iii) β はその中間ノード y から開始してシンク $\perp$ へ到達する最重み出辺だけからなる一意な有効パス（すなわち y の最重み接尾辞パス）．以後，$\mathcal{CSUF}(G,z)$ で，正規 z-ソリッド接尾辞パスのなす集合を表す．

逆トライ．逆トライ $\mathcal{CS} = \mathcal{CS}(G,z) = (\mathcal{V}, \mathcal{E}, root, lab, w)$ は，G の各正規 z-ソリッド接尾辞パス β に対して，一意なノード $[\beta]$（区別のため**トライノード**と呼ぶ）をもつような逆トライ（reverse-trie）である．パス $[e\beta]$ から先頭の辺 e を除去したパス $[\beta]$ へ，ラベル $lab_{\mathcal{CS}}(f) := lab(e)$ と重み $w_{\mathcal{CS}}(f) := w(e)$ をもつ逆辺 $f := ([e\beta], [\beta])$ を張る．図 2 に，逆トライの例を示す．

G の各正規 z-ソリッド接尾辞パスの集合 $\mathcal{CSUF}(G,z)$ は，接尾辞をとる演算について閉じているので，この構成法は，想定したサイズの木を正しく形成する．

補題 4 $\mathcal{CS}(G,z)$ は，高々 $O(zn)$ 個の頂点と辺をもつ Σ 上のトライである．

逆トライの正当性. G の任意の極大 z-ソリッドパス $\alpha \in \mathcal{M}(G,z)$ に対して，逆トライ $\mathcal{CS}(G,z)$ における α の開始点と終了点に対応するトライノード $\mathtt{head}(\alpha)$ と $\mathtt{tail}(\alpha)$ を次のように定めよう：$\mathtt{head}(\alpha) := [\mathtt{csuf}(\alpha)] = \gamma = \alpha\beta$ および $\mathtt{tail}(\alpha) := \beta$．ここに，$\beta := [\mathtt{hvpath}(tail(\alpha))]$ は，α の端点から $\perp$ までの一意な最重み接頭辞パスである．

反対に，$\mathcal{CS}(G,z)$ の任意のトライノード v に対して，$\mathtt{head}(\alpha) = v$ となるような G の極大 z-ソリッドパス α が一意に存在することが言える．すなわち，G の極大 z-ソリッドパス全体と $\mathcal{CS}(G,z)$ のトライノード全体は一対一に対応する．

このとき，γ_α は β_α を接尾辞として含むことから，次の補題を示せる．

補題 5 任意の極大 z-ソリッドパス α に対して，*(a)* 逆トライは $[\gamma_\alpha]$ から $[\beta_\alpha]$ までの逆パス $Q_\alpha = \mathtt{path}([\gamma_\alpha],[\beta_\alpha])$ をもつ．さらに，*(b)* $lab_{\mathcal{CS}}(Q_\alpha) = lab_G(\alpha)$，かつ，*(c)* $w_{\mathcal{CS}}(Q_\alpha) = w_G(\alpha)$ が成立する．

これは，構築した逆トライ $\mathcal{CS}(G,z)$ が，G の極大 z-ソリッドパス全てを，正しく含むことを主張する．

3.4 ステップ 1: 逆トライの構築

逆トライの構築には注意が必要である．G の各正規 z-ソリッド接尾辞パスは $O(n)$ 長さをもつ．したがって，定義通りに $\mathcal{M}(G,z)$ の各正規 z-ソリッド接尾辞パスを逆トライに挿入する方法だと，G からの構築に $O(Nn) = O(zn^2)$ 時間を要する．

そこで，次のステップからなる手続き BuildCSTree で逆トライを構築する．最初は，根 $root := [\varepsilon]$ を生成し，シンク $\perp$ から出発して，有向辺を逆向きに辿りながら G を深さ優先探索する．その際に，通過した逆パスに相当する新しい頂点を $\mathcal{CS}(G,z)$ に追加しながら，$\mathcal{CS}(G,z)$ を構築する．

探索で，現在までに，根から現在地までに通ってきた辺からなるパスを γ とおく．この際，手続きは，任意時点で次の不変条件を満たすように，二つのポインタ head と tail，実数変数 backw，ブール変数 onheavy を管理する．

- head は，現在訪問しているノード（現在地）を指す．
- tail は，根から現在地までに通ってきたパス上のノード（逆先祖）で，これまで最重み辺だけを通過し，最も現在地に近い（根から遠い）ものを指す．
- backw（back weight）は，現在までの経路を表すパス γ 上の tail から head までの部分パス α の累積重み $w(\alpha) \in [0,1]$ を保持する．
- onheavy は，γ が最重み辺のみからなるとき $true$，そうでないとき $false$ を保持する．

手続き BuildCSTree は，次のルールを用いて，上記の変数を更新しながら，深さ優先探索を実行する．

- 初期値として，根 $[\varepsilon]$ では，$\mathtt{head} := \mathtt{tail} := [\varepsilon]$ かつ，$\mathtt{backw} := 1$，$\mathtt{onheavy} := true$ とおく．
- もし $\mathtt{onheavy} = 1$ で，かつ，現在地から最重み辺 e を横断した際は，次を実行する：$\mathtt{head} := \mathtt{head}\cdot e$ かつ，$\mathtt{tail} := \mathtt{head}$，$\mathtt{backw} := 1$，$\mathtt{onheavy} := true$.
- それ以外の場合，すなわち，(i) もし一度でも最重み辺でない辺 e を横断した後（すなわち，$\mathtt{onheavy} = false$），または，(ii) もし現在地から最重み辺でない辺 e を横断した後は，次を実行する：$\mathtt{head} := \mathtt{head}\cdot e$ かつ，$\mathtt{tail} := \mathtt{tail}$，$\mathtt{backw} := \mathtt{backw} \times w(e)$，$\mathtt{onheavy} := false$.
- 親から辺 e を横断したら，現在のトライノード $v := [\gamma]$ に新しいトライノード $u := [\gamma \cdot e]$ を入辺 $f = (v,u)$ をもつ新しい子として挿入する．辺ラベル $lab(f) := lab(e)$ と辺重み $w(f) := w(e)$ を追加する．
- もし辺の横断後に，$\mathtt{backw} < 1/z$ となったら，直ちにその枝の探索を中止して，その前の親にバックトラックして，親の未訪問の子に対して探索を継続する．

上記の深さ優先探索は，ノード一つあたりならし定数時間で実行できるので，次を得る．

補題 6 （end-リンクなしの）逆トライ $\mathcal{CS}(G,z)$ は，G と z から $O(N) = O(zn)$ 時間で計算可能である．

3.5 ステップ 2：end-リンクの計算

任意の G 中の極大 z-ソリッドパス α に対して，対応する $\mathcal{CS}(G,z)$ 中の逆パス $Q_\alpha = \mathtt{path}([\gamma_\alpha],[\beta_\alpha])$ の開始点から終了点へリンク $end([\gamma_\alpha]) := [\beta_\alpha]$ を張る．さらに，後のマージのために，目的頂点 $[\beta_\alpha]$ に，その頂点に入る end-リンクの逆リンク $end^{-1}([\beta_\alpha]) := [\gamma_\alpha]$ のリスト $L([\beta_\alpha])$ を関連づける．各 end-リンクとその逆リンクは，各 α に一対一で対応しているので，以上の処理の後でも，逆トライのノード数は変わらず $O(zn)$ である．図 2 に，end-リンクの例を示す．

$\mathcal{CS}(G,z)$ には極大 z-ソリッドパスの情報は保持されていないので，このステップの計算は次のように行う．各トライノード $[\gamma] \in \mathcal{V}(\mathcal{CS}(G,z))$ に対して，その根方向へ遡った先祖 $[\beta]$ ($\gamma \sqsupset_{suf} \beta$) で，$w(\mathtt{path}([\gamma],[\beta])) \geq 1/z$ となる最遠の（すなわち，最も根に近い）先祖 γ を見つけると，パス $\mathtt{path}([\gamma],[\beta])$ は，極大 z-ソリッドパス α に正確に対応することが言える．

この計算は，$\mathcal{CS}(G,z)$ を根から深さ優先探索または幅優先探索しながら，$[\gamma]$ と $[\beta]$ を指す二つのポインタを更新しながら漸増的に行える．しかし，木の接頭辞での計算が重複するため，残念ながら，このやり方では $O(Nh) = O(zn^2)$ 時間を要する．ここに，$N = O(zn)$ と $h = O(n)$ は木の頂点数と高さである．

効率良い解法. この問題を解決するために，次のように重み付き先祖クエリ（weighted ancestor, WA）構造 [4] を用いる．まず重みの各辺 e の重み $w(e) \in [0,1]$ をその**負の対数重み** $\ell(e) := -\log w(e) \geq 0$ に変換して辺

に付加する．すると，パス $\pi = e_1 \cdots e_h$ について，累積積を累積和に変換できる．

$$w(\pi) := w(e_1) \times \cdots \times w(e_h) \geq 1/z$$
$$\iff \quad \ell(\pi) := \ell(e_1) + \cdots + \ell(e_h) \leq \log z$$

したがって，$\mathcal{CS}(G, z)$ のコピーを WA 構造に格納し，根からの深さ $h \geq 0$ の各トライノード v に対して，その根からのパス $\pi = (e_h, \ldots, e_1)$ 上の辺重みの累積和 $L(v) := \ell(\pi) = \ell(e_h) + \cdots + \ell(e_1)$ を WA に重みとして登録する．すると，与えられたトライノード $[\gamma]$ に対して，$L = L([\gamma]) - \log z$ を超えない重みをもつ最遠先祖を見つけるクエリを WA に対して発行することで，1 回あたり $O(\log n)$ 時間で望む先祖 $[\beta] = end([\gamma])$ を得られる．以上により，ステップ 2 を $O(N \log n) = O(zn \log n)$ 時間で実行可能である．

3.6 ステップ 4：接尾辞木の構築

逆トライの決定化．得られた逆トライ $\mathcal{CS}(G, z)$ は，根を初期状態とする有限状態機械（NFA）として見たときに非決定性であるので，根から開始して深さ優先探索を行い，各頂点から同じ文字をラベルにもつ異なる入辺があれば，それらの辺と端点を一つにマージして，$\mathcal{CS}(G, z)$ を決定化する．この際，両方向ポインタを用いて，end-リンクも同時にマージする．このステップは，$O(N \log \sigma) = O(zn \log \sigma)$ 時間で実行可能である．

木の接尾辞木の構築．決定化された $\mathcal{CS}(G, z)$ は通常の決定性の逆トライなので，逆トライを入力とする任意の接尾辞木構築アルゴリズム（例えば [3, 5]）をこれに適用し，全ての正規 z-ソリッド接尾辞パスの文字列ラベルからなる接尾辞木 $\mathcal{ST}(G, z)$ を構築する．ノード数 N の逆トライに対して，Breslauer [3] のアルゴリズムを用いた場合，$O(N \log \sigma)$ 時間で接尾辞木を構築可能である．よって，$\mathcal{ST}(G, z)$ は $O(zn \log \sigma)$ 時間で計算可能である．

プロパティ接尾辞木への変換．さらに，$\mathcal{CS}(G, z)$ 上の end-リンクを $\mathcal{ST}(G, z)$ 上に移植して，プロパティ接尾辞木 [1] に変換する．具体的には，$\mathcal{CS}(G, z)$ のある頂点 v から開始する接尾辞パス S_v のラベル文字列を挿入する場合には，その接尾辞 S_v を表す $\mathcal{ST}$ の葉 $leaf_v$ に対して，根からの距離が $len := v - depth(\mathtt{end}(v)) \geq 0$ であるような $leaf_v$ の先祖 u を同定し，そのノードをマークし，頂点 v へのポインタ情報を付加する．

先祖 u が実ノードならそのままマーク可能であり，辺上の仮想ノードなら Amir ほか [1] と同じ方法で，辺に保持したリストに根からの深さとマークの組を格納することで，マーク 1 個あたり $O(\log \log n)$ 時間を用いて仮想的に実現する．このステップは，Amir ほか [1] に従い，$O(N \log \log \log n) = O(zn \log \log \log n)$ 時間で実行可能である．

重み付き文字列照合クエリ処理．構築されたテキスト索引におけるクエリ処理は，次のように行う．任意の文字列 $P \in \Sigma^*$ がパターンとして与えられたとする．このとき，はじめに接尾辞木 $\mathcal{ST}$ 上で，通常の検索を行い，根から P で辺ラベルを読んで到達可能なノード位置 ϕ を見つける．上で述べた頂点と辺に付けられたリストを用いて，ϕ を根とする部分木を深さ優先探索して，ϕ より下の位置に対応づけられたマークを全て列挙し，解として出力する．以上の計算は，$O(|P| \log \sigma)$ 時間で実行可能である．

4. 解析

以上をまとめると，重み付きグラフに対するテキスト索引の構築について，本稿の主結果を得る．

定理 1 (重み付きグラフに対するテキスト索引構築) サイズ σ のアルファベットに対して，入力として頂点数 n と変数 m の $WDAG(G, lab, w)$ と，重みしきい値 $z \geq 1$ から，任意のパターン P に対する重み付き文字列照合クエリを $O(|P| \log \sigma + occ)$ 時間で実行可能な，$O(zn)$ 語のテキスト索引を $O(zn \log(n + m))$ 時間で構築可能である．

5. おわりに

本稿では，WDAG において，それが含む全パスに対して，重み付き文字列照合問題を効率良く解くためのテキスト索引構造を与えた．入力の二乗サイズを必要とする索引の領域量を改善し，$O(zn)$ 語のサイズをもち，入力 WDAG G と重みしきい値 z から $O(zn \log n)$ 時間構築可能で，重み付き文字列照合クエリを $O(|P| \log \sigma)$ 時間で解くような効率良い索引を与えた．

今後の課題として，重み付きの無向グラフに対する拡張があげられる．また，隠れマルコフモデル（HMM）のような確率生成モデルでの予測の高速化への応用は興味深い課題である．

参考文献

[1] Amihood Amir, Eran Chencinski, Costas Iliopoulos, Tsvi Kopelowitz, and Hui Zhang. Property matching and weighted matching. *Theoretical Computer Science*, 395(2-3):298–310, 2008.

[2] Carl Barton, Tomasz Kociumaka, Solon P. Pissis, and Jakub Radoszewski. Efficient index for weighted sequences. In *27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, June 27-29, 2016, Tel Aviv, Israel*, pages 4:1–4:13, 2016.

[3] Dany Breslauer. The suffix tree of a tree and minimizing sequential transducers. *Theoretical Computer Science*, 191(1-2):131–144, 1998.

[4] Tsvi Kopelowitz and Moshe Lewenstein. Dynamic weighted ancestors. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 565–574. Society for Industrial and Applied Mathematics, 2007.

[5] Tetsuo Shibuya. Constructing the suffix tree of a tree with a large alphabet. In *Algorithms and Computation, 10th International Symposium, ISAAC '99, Chennai, India, December 16-18, 1999, Proceedings*, volume 1741 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 1999.