社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－AL－102 (6)
2005／9／16

クエリを高速にサポートする方形描画のコンパクトなコード化

山中 克久† 中野 眞一†

概要 本文では，方形描画をコンパクトにコード化する手法を提案する．我々は，方形描画に対する $\frac{5}{3}m + o(n)$ bit の文字列を構成し，隣接・次数クエリの解を定数時間で求める方法を与える．ここで，m とは方形描画の辺の本数である．はじめに，方形描画 R が与えられたとき，R に対する長さ $\frac{5}{3}m$ bit の文字列が存在することを示す．次に，長さ $o(n)$ bit の文字列をその文字列に加えることにより，隣接・次数クエリを定数時間でサポートする方法を与える．

A Compact Encoding of Floorplans with Efficient Query Support

Katsuhisa YAMANAKA†, Shin-ichi NAKANO†

abstract In this paper we give a simple coding scheme for floorplans. The coding scheme needs $\frac{5}{3}m + o(n)$ bits for each floorplan, and supports adjacency and degree queries in constant time. First, given a floorplan R, we show that R has a code with $\frac{5}{3}m$ bits, where m is the number of edges in R. Next, by appending $o(n)$ bits string, we give a method to support adjacency and degree queries in constant time.

1 はじめに

与えられたグラフ G から，次の 3 つの条件 (1)-(3) を満たすような 0 と 1 のみからなる文字列 S を求める問題を考えよう．

(1) S から G を再構成できる．

(2) S の長さはできるだけ短い．

(3) G に関する様々なクエリ (質問) の解を S から高速に求めることができる．

例えば，一般のグラフ G が与えられたとき，G の隣接行列を行ごとに分解したのちに結合することにより，文字列 S_G が得られる．グラフの点の個数を n とし，辺の本数を m としよう．このとき，

(1) S_G から G を再構成でき，

(2) S_G の長さは n^2 bit であり，

(3) G の指定された 2 点が隣接しているかどうかを，S_G から，$O(1)$ 時間で判定できる．

いくつかのクラスのグラフに対しては，より効率的な文字列 S が存在することが知られている．

例えば，順序木に対しては，次のような長さが $2m$ bit の文字列 S_c が古くから知られている．(図 1 参照.) 順序木 T が与えられたとき，T の深さ優先探索を行おう．このとき，辺を葉にむかって下るときに符号 0 を出力し，辺を根にむかって上るときに符号 1 を出力する．これらの出力から，図 1 に示すような $2m$ bit の文字列 S_c が得られる．このとき，(1) S_c から T を再構成でき，(2) S_c の長さは $2m$ bit である．ただし，T 中の指定された 2 点が隣接しているかどうかを，$O(1)$ 時間で S_c から求める方法は知られていない．しかし，S_c に $o(n)$ bit の文字列を追加すれば，T 中の指定された 2 点が隣接しているかどうかを $O(1)$ 時間で求めることができる [M97, M01]．

一方，n 点の順序木の個数は，カタラン数 C_{n-1} として知られている．カタラン数は以下のように定義される [R00, p.145]．

$$C_n = \frac{1}{(n+1)}\frac{(2n)!}{n!n!}$$

例えば，4 点の順序木の個数は，図 1 に示すように，$C_{4-1} = 5$ である．n 点の順序木のそれぞれを異なる文字列に対応させなくてはならないので，一般に n 点の順序木 T に対応する文字列 S_T の平均の長さは，少なくとも $\log C_{n-1}$ である．表記を工夫すること [G94, p495] により，カタラン数は次のようにも表現できる．

$$C_n = \frac{4^n}{(n+1)\sqrt{\pi n}}\left(1 - \frac{1}{8n} + \frac{1}{128n^2} + \frac{5}{1024n^3} - \frac{21}{32768n^4} + O(n^{-5})\right)$$

したがって，一般に n 点の順序木 T に対応する文字列 S_T の長さは，少なくとも $\log C_{n-1} = 2n - o(n) = 2m - o(n)$ である．先に紹介した S_c の長さは $2m$ であるので，S_c は漸近的に最適である．

また，平面グラフに対する文字列に関しても多くの研究がある [C01, C98, K95, M97, M01, P86, T84]．平面グラフに対しては，O(1) 時間で隣接・次数クエリの解を求めることができるような，$2m + 2n + o(n)$ bit の文字列が知られている [C01]．また，特に，極大平面グラフに対しては，隣接・次数クエリの解を $O(1)$ 時間で求めることができるような，$2m + n + o(n)$ bit の文字列が知られている [C98]．

† 群馬大学情報工学科 〒 376–8515 群馬県桐生市天神町 1–5–1.
Department of Computer Science, Gunma University, 1–5–1 Tenjin-Cho, Kiryu, 〒 376–8515
{yamanaka, nakano}@msc.cs.gunma-u.ac.jp

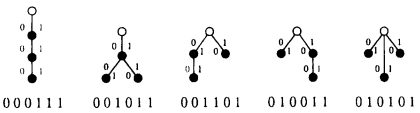


図 1: 順序木 T に対する文字列 S_c の例

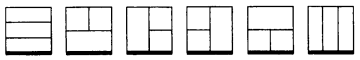


図 2: ちょうど 3 個の内面をもつ底辺つき方形描画

本文は方形描画を扱う. 各面が方形 (長方形) であるように, かつ, 辺が交差することなく, 平面に描画したグラフを方形描画という. 方形描画の, 外面の方形の 4 つの線分のうちひとつを底辺として選んだものを, 底辺つき方形描画という. 図 2 に, ちょうど 3 個の内面をもつ底辺つき方形描画のすべてを示す. 底辺は太線で描かれている. 方形描画は, 工学的にも応用が広く, 理論的にも多くの研究がある [H99, N02a, N02b]. 本文では内点の次数が全て 3 である方形描画のみを扱う. 次数が 5 の点をもつグラフには方形描画は存在しないことに注意しよう.

本文は, 方形描画 R が与えられたとき, 次の (1)-(3) を満たす文字列 S が構成できることを示す.

(1) S から R を再構成でき,

(2) S の長さは $\frac{5}{3}m + o(n)$ bit であり,

(3) R に関する様々なクエリの解を, S から $O(1)$ 時間で求めることができる.

(3) のクエリとは例えば次のようなものである. 詳細は 4–6 章で説明する.

(例 1) 面 x の北方向に, 面 y が隣接しているか ?

(例 2) 面 x の北方向に隣接する面の個数は ?

(例 3) 面 x の西方向に隣接する面のうち最も南にある面は ?

(例 4) 面 x の左下隅の点には南方向に接続する辺があるか ?

方形描画 R の各面を点に, 面と面との隣接関係を辺に, それぞれ置き換えて得られるグラフを, R の双対グラフ D という. 図 3(a) の方形描画の双対グラフを図 3(b) に示す. ただし, 外面については特別な扱いをするが, これについては後で説明する. 方形描画 R の内点の次数が全て 3 であるならば, R の双対グラフ D は (内部) 極大平面グラフである. したがって [C98] の手法を用いれば, D に対する $2m+n+o(n) = \frac{7}{3}m+o(n)$ bit の文字列が存在する. ただし, 各辺が, 縦横いずれかの隣接関係を表しているか, という情報は失なわれてしまう.

これに対し, 我々は, 方形描画 R に対する $\frac{5}{3}m+o(n)$ bit の文字列 S を構成する. さらに, 多様なクエリの解を S から $O(1)$ 時間で求めることができることを示す.

本文のアイデアは次のとおりである. 方形描画 R が与えられたときに (図 3(a) 参照), その双対グラフ D を作り (図 3(b) 参照), D の辺を次のように 2 種類に分割する (図 3(c) 参照).

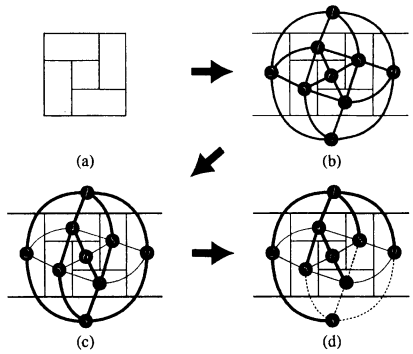


図 3: 方形描画の双対グラフ

(1) 面の縦方向の隣接関係に対応する辺 (図 3(c) では太線で示す.)

(2) 面の横方向の隣接関係に対応する辺 (図 3(c) では細線で示す.)

さらに (1) の辺を, (1a) 全域木に含まれる辺と, (1b) それ以外の辺, の 2 つに分割する.(図 3(d) 参照.) 我々は, (1a) と (2) の辺の情報のみを, $\frac{5}{3}m$ bit の文字列 S_R に格納する. (1b) については保存しなくてもかまわないことを示す. また, 様々なクエリの解を $O(1)$ 時間で求めるために, $o(n)$ bit の文字列 S_A を準備する. $S = S_R + S_A$ とする.

本文の構成は次のとおりである. 2 章は定義の説明である. 3 章は方形描画 R に対する文字列 S_R について説明する. 4 章は基本クエリを高速に求める方法について説明する. 5 章は, 隣接クエリを高速に求める方法を説明する. 6 章は次数クエリを扱う. 7 章は, $S = S_R + S_A$ から R を再構成できることを示す. 8 章はまとめと今後の課題である.

2 定義

本章では, いくつかの定義を与える.

G を連結なグラフとする. **木**はサイクルのない連結グラフである. **根つき木**とは, 1 点 r が根として指定された木である. **順序木**とは, 兄弟の間の順序が定められている根つき木である.

グラフの描画で, 互いに辺が交差しないものを, **平面描画**と呼ぶ. 平面描画は, 平面を面という連結な領域に分割する. 無限遠点を含む唯一の面を**外面**と呼び, 他の面を**内面**と呼ぶ. 面の**輪郭**とは面の境界の時計回りの線分列である. 平面描画できるグラフを**平面的グラフ**という. 平面への埋め込みを固定した平面的グラフを**平面グラフ**という. n はグラフの点の個数とし, m はグラフの辺の本数とし, f はグラフの面の個数とする. オイラーの公式より, 平面グラフでは

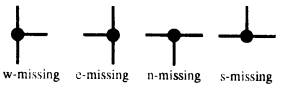


図 4: w-missing, e-missing, n-missing, s-missing

$n-m+f=2$ である. また, 各面は 3 本以上の辺で囲まれており, 各辺はちょうど 2 つの面の周上にあるので, 平面グラフでは $3f \leq 2m$ である. これら 2 つの式から平面グラフでは $m \leq 3n-6$ が成立する.

方形描画とは全ての面 (外面を含む) が方形であるような平面描画である. 最大次数が 5 以上のグラフは方形描画をもたないことに注意しよう. 方形描画で, 外面の方形の輪郭の 4 つの直線分のうち, ちょうど 1 つの直線分を指定したものを**底辺つき方形描画**と呼ぶ. 指定した直線分を**底辺**と呼ぶ. 本文では, 底辺を常に図中で最も低い水平線分として書く. 例として, ちょうど 3 個の内面をもつ底辺つき方形描画のすべてを図 2 に示す. 各底辺は太線で示されている. もし, 2 つの面 F_1 と F_2 が水平線分を輪郭上で共有していれば, F_1 と F_2 は **ns 隣接**しているという (ここで ns とは北と南を意味している). もし, 2 つの面 F_1 と F_2 が垂直線分を共有していれば, F_1 と F_2 は **ew 隣接**しているという (ここで ew とは東と西を意味している). 2 つの方形描画 P_1 と P_2 が与えられたとき, 必要ならば, 一方の描画を回転した後に, 各面に ns 隣接および ew 隣接関係を保存するような 1 対 1 対応があるならば, P_1 と P_2 は**同型**であるという. また, 底辺つき方形描画 P_1 と P_2 の各内面に, ns 隣接および ew 隣接関係を保存するような 1 対 1 対応があり, それぞれの底辺も互いに対応するとき, P_1 と P_2 は互いに同型であるという.

次数 3 の点 v を, 上下左右の 4 方向の, いずれの方向に辺が接続していないかによって, 次の 4 つのタイプのいずれかに分類する. もし, v の, 上下および右方向に辺が接続するが, 左方向に辺が接続していないならば, v は **w-missing** であるという (ここで w は west をあらわす). 同様に, **e-missing**, **n-missing**, **s-missing** を定める (図 4 参照).

3 文字列 S_R

本章では, 方形描画 R に対する文字列 S_R を定義する. R の点の個数を n とし, 辺の本数を m とする. (1) S_R から R は再構成でき, (2) S_R の長さは $\frac{5}{3}m$ bit である. ただし, (3) R に関するクエリの解を S_R から高速に求めることはできない. しかし, S_R に $o(n)$bit の長さの文字列 S_A を追加すれば, クエリの解を $O(1)$ 時間で求めることができる. S_A については, 次章で説明する.

S_R のおおまかな構成法は次の通りである. まず, 方形描画 R の双対グラフ D_R を求める. 次に, D_R の辺を 3 種類に分割し, このうち 2 種類の辺から, 文字列 S_1 と S_2 を作る. 最後に, S_1 と S_2 を連結して S_R とする. いくつかの準備からはじめよう.

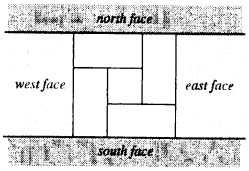


図 5: 外面の分割

まず, R をそれぞれ 0 度, 90 度, 180 度, 270 度 だけ時計回りに回転した 4 つの底辺つき方形描画を作る. このうち s-missing な点の個数が最も多いものを R と仮定してよい. もしそうでない場合は, どれだけ回転したかを 2 bit の符号で記憶しておき, クエリの解を求めた後に, この回転を考慮してから最終的な解を求めれば良いからである. 次の補題がいえる.

補題 3.1 *s-missing* な内点の個数を n_S とする. このとき, $n_S \geq (n-4)/4$ である.

底辺つき方形描画 R が与えられたとき, 最も高い水平線分と最も低い水平線分を水平方向に引き伸ばすことによって, 外面を 4 つの面 (北面, 南面, 西面, 東面) に分割する.(図 5 参照.) 方形描画 R の, 各面を点に, 任意の 2 面間の隣接関係を辺に, それぞれ置き換えて得られる平面グラフを R の双対グラフ D_R とする. 4 つの外面もそれぞれ点に置き換えることに注意しよう. (図 3(b) 参照.)

D_R の辺を次のように 2 種類に分割しよう. R 中の, 2 つの面の南北方向の隣接関係に対応する辺の集合を E_{NS} とし, 東西方向の隣接関係に対応する辺の集合を E_{EW} とする.(図 3(c) 参照.) 辺集合 E_{NS} が誘導する D_R の部分グラフを D_{NS} とし, E_{EW} が誘導する D_R の部分グラフを D_{EW} とする.

次に, E_{NS} の辺を, 次のようにしてさらに 2 種類に分割しよう. R の北面以外の各面 f について, f の**親面** $p(f)$ を次のように定義する. f の北方向に隣接する面のうち, 最も西にある面を $p(f)$ とする. E_{NS} 中の辺のうち, 任意の面 f と, その親面 $p(f)$ の隣接関係に対応する辺からなる集合を E^T_{NS} とする. E^T_{NS} は D_R の全域木を誘導する. この全域木を T_{NS} としよう. 例を図 3(d) に示す. T_{NS} は太線で示されている. $E_{NS}-E^T_{NS}$ 中の辺は破線で示されている.

我々は, T_{NS} から文字列 S_1 を作成し, T_{NS} と D_{EW} から文字列 S_2 を作成する. そして, $S_R=S_1+S_2$ を R に対する文字列とする.

まず S_1 について説明しよう. (これは, 1 章で説明した順序木に対する文字列と同様である.) 北面に対応する点を根として, T_{NS} の深さ優先探索を行おう. このとき, 辺を葉にむかって下るときに符号 "(" を出力し, 辺を根にむかって上るときに符号 ")" を出力する. このようにして得られた文字列を S'_1 とする. S'_1 の最初に "(" を付け加え, 最後に ")" を付け加えて得られる文字列を S_1 とする. (厳密にはさらに "("

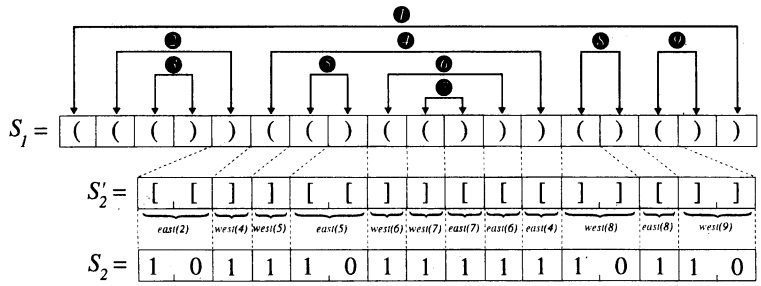


図 6: 図 3 の方形描画 R に対する文字列 $S_R = S_1 + S_2$

を "0" に, ")" を "1" に変換する.) 例を図 6 に示す. S_1 中の対応する括弧の組は, T_{NS} の点に対応することに注意しよう. T_{NS} の点に preorder で番号をつけよう.(図 3(d) 参照.) S_1 の i 番目の "(" と, これに対応する ")" とが, i 番の点に対応する. (図 3 および 6 参照.)

次に, S_2 について説明する. まず, S_1 から次のようにして S_2' をつくろう. S_1 中の各 i 番目の "(" を $|west(i)|$ 個の "]" に置き換え, i 番目の "(" に対応する ")" を $|east(i)|$ 個の "[" に置き換える. ここで, $west(i)$ とは, i 番の点に対応する R の面 f_i の, 西方向に隣接する面からなる集合であり, $east(i)$ とは, f_i の, 東方向に隣接する面からなる集合である. ここで, $west(1) = west(2) = west(3) = \phi$ であり, $east(1) = east(f+3) = east(3) = \phi$ であるので, これらに対応する文字列はないとしよう. ここで f は面の個数とする. このとき, R の任意の内面 f_i について, $|west(i)| \geq 1$ かつ, $|east(i)| \geq 1$ であることに気をつけよう. $|west(i)|$ 個の "]" のそれぞれは, i 番の点から西方向に接続する各辺に対応する. このとき k 番目の "]" は, 真北から反時計回りに k 番目の辺に対応するとしよう. 同様に, $|east(i)|$ 個の "[" のそれぞれは, i 番の点から東方向に接続する各辺に対応する. このとき, k 番目の "[" は, 真南から反時計回りに k 番目の辺に対応するとしよう. このとき, D_R は平面グラフであることより, S_2' は "[" と "]" の入れ子構造となることに注意しよう. (入れ子構造は 4 章で定義する.)

次に S_2' から S_2 を作ろう. i 番の点に対応する $|west(i)|$ 個の "]" を, 1 つの "1" と $(|west(i)| - 1)$ 個の "0" とに置き換える. 同様に, $|east(i)|$ 個の "[" を, 1 つの "1" と $(|east(i)| - 1)$ 個の "0" とに置き換える. 例として, 図 3 の方形描画 R に対する文字列 S_1 と S_2 を図 6 に示す. "1" は各 $west(i)$ や $east(i)$ の区切りとなっていることに注意しよう. また, $E_{NS} - E_{NS}^T$ 中の辺に関する情報は記憶しないことに注意しよう.

次に, $S_R = S_1 + S_2$ の長さを見積もろう. R の点の個数を n とし, 辺の本数を m とし, 面の個数を f とする. このとき, D_R の辺の本数も m である. R の外面の四隅の 4 点の次数は 2 であり, 他の点の次数は 3 である. したがって

$$2m = 3(n-4) + 2 \cdot 4 \tag{1}$$

である.

はじめに, S_1 の長さを見積もる. T_{NS} の各辺に対し, 1 組の "(" と ")" を記憶し, さいごに先頭の "(" と最後の ")" を加えるので, $|S_1| = 2|E_{NS}^T| + 2$ となる.

次に, S_2 の長さを見積もろう. 東西方向の隣接関係に対する各辺に対し, 1 組の "[" と "]" を記憶するので, $|S_2| = 2|E_{EW}| = 2(m - |E_{NS}|)$ である.

よって,

$$\begin{aligned} |S| &= |S_1| + |S_2| \\ &= (2|E_{NS}^T| + 2) + (2m - 2|E_{NS}|) \\ &= 2m + 2 - 2|E_{NS} - E_{NS}^T| \end{aligned}$$

である. 次の補題がいえる.

補題 3.2 $|E_{NS} - E_{NS}^T| = n_S$ である.

補題 3.2 より,

$|S| = 2m + 2 - 2|E_{NS} - E_{NS}^T| = 2m + 2 - 2n_S$

である. 補題 3.1 より,

$$|S| \leq 2m + 2 - 2\frac{n-4}{4} = 2m - \frac{n}{2} + 4$$

である. 式 (1) より, $|S| \leq \frac{5}{3}m + \frac{10}{3}$ である.

また, 次の補題がいえる.

補題 3.3 S_R から R は再構成できる.

4 基本クエリ

本章は文字列の基本クエリについて説明する.

まず, いくつかの定義を与える. 文字列 S の部分文字列で, i 文字目から j 文字目までからなるものを, $S[i,j]$ と書く. 特に $S[i,i] = S[i]$ とする.

S_1 は "(" と ")" からなる文字列とする. $S_1[1,i]$ 中の "(" の個数を $rank(S_1, i, ()$ と書く. S_1 中の k 番目の "(" が S_1 の j 文字目であるとき $select(S_1, k, () = j$ とする. $j = select(S_1, k, ()$ ならば, $k = rank(S_1, j, ()$ であることに注意しよう. 同様に, $rank(S_1, i,))$ と $select(S_1, k,))$ を定義する.

次に, **入れ子構造**を次のように定義する. 文字列 $S_a = ()$ は入れ子構造である. 文字列 S_b と S_c が入れ子構造のとき, 文字列 $S_d = S_bS_c$ と $S_e = (S_b)$ は入れ子構造である.

入れ子構造をもつ文字列において, "(" と ")" の間には自然な1対1対応がある. $S_1[i] =$ "(" と $S_1[j] =$ ")" が対応するとき, $S_1[i]$ と $S_1[j]$ は**対応する**という. このとき $match(S_1, i) = j$ と書く. $match(S_1, i) = j$ のとき $match(S_1, j) = i$ である.

次に, $enclose(S_1, i)$ を定義しよう. 2つの場合にわけて定義する. はじめに $S_1[i] =$ "(" の場合を扱う. $match(S_1, i) = j$ としよう. このとき, $S_1[i, j]$ を真に含む最小の長さの入れ子構造である部分文字列を $S_1[a, b]$ とする. このとき, $S_1[i]$ と $S_1[j]$ からなる括弧の組を, $S_1[a]$ と $S_1[b]$ からなる括弧の組が囲んでいることに注意しよう. 次に, $S_1[i] =$ ")" の場合を扱う. $match(S_1, i) = j$ とする. このとき, $S_1[j, i]$ を真に含む最小の長さの入れ子構造である部分文字列を $S_1[a, b]$ とする. 2つのいずれの場合も $enclose(S_1, i) = a$ と定義する. また, $S_1[a]$ と $S_1[b]$ は $S_1[i]$ と $S_1[j]$ を *enclose* するという. 例えば, 図6の S_1 について, $enclose(S_1, 9) = 6$ である.

次に, $wrapped(S_1, i)$ を定義しよう. 2つの場合にわけて定義する. はじめに $S_1[i] =$ "(" の場合を扱う. このとき, $enclose(S_1, k) = i$ を満たす k の個数を c としよう. 次に $S_1[i] =$ ")" の場合を扱う. $match(S_1, i) = j$ としよう. このとき, $enclose(S_1, k) = j$ を満たす k の個数を c としよう. 2つのいずれの場合も $wrapped(S_1, i) = c$ とする. S_1 は入れ子構造であるので, $wrapped(S_1, i)$ の値は常に偶数になることに気をつけよう.

次がいえる.

補題 4.1 ([C96, C01, J89, M97, M01])
S_1 に $o(n)$ *bit* の長さの文字列 S_1^A を追加することにより, 次の値を $O(1)$ 時間で求めることができる.
(1) $rank(S_1, i, ()$ および $rank(S_1, i,))$ [C96, J89].
(2) $select(S_1, k, ()$ および $select(S_1, k,))$ [C96].
(3) $match(S_1, i)$ [M97, M01].
(4) $enclose(S_1, i)$ [M97, M01].
(5) $wrapped(S_1, i)$ [C01].

S_2' は "[" と "]" からなる文字列とする. S_2 は, 3章で示した方法によって得られた文字列とする. このとき, S_2' の入れ子構造により S_2 の入れ子構造を定義できる. この入れ子構造にもとづいて S_2 についても同様に *rank*, *select*, *match*, *enclose*, *wrapped* を定める. S_2 についても補題4.1と同様のことがいえる.

次に, 文字列 $S_R = S_1 + S_2$ から基本的な情報を高速に取り出す基本クエリについて説明する.

S_1 中には D_R の各点に対応して1組の "(" と ")" が存在する. i 番目の点に対応する "(" と ")" の位置は, それぞれ $select(S_1, i, ()$ と $match(S_1, select(S_1, i, ()))$ である.

定義より, S_2 は, 各 $west(i)$ や $east(i)$ に対応した $(|S_1| - 6)$ 個の部分文字列に分割できる.(図6参照.) S_2 中の $west(i)$ に対応する部分文字列を L_i としよう. S_2 中から L_i を次のようにしてさがすことができる. $select(S_1, i, () = a$ としよう. このとき, $L_i = S_2[select(S_2, a-4, 1), select(S_2, a-3, 1)) - 1]$ である. S_2 中の "1" は必ず $west(i)$ や $east(i)$ の区切りとなっていることに注意しよう. L_i は, i 番目の点から西方向に接続する $|west(i)| = |L_i|$ 個の各辺に対応する. このとき, k 番目の "]" は, 真北から反時計回りに k 番目の辺に対応する. 特に, 1番目の "]", すなわち $select(S_2, a-4, 1)(= p_1$ とする) と, これに対応する "[", すなわち $match(S_2, p_1)(= p_2$ とする) の組は, i 番目の点に対応する面 f_i と, f_i の西方向に隣接する面のうち最も北にある面 f_{WN} との隣接関係に対応する. 面 f_{WN} に対応する点を v_j とする. このとき, $f_{WN}(i) = j$ と書こう. すなわち, 面 f_i の西 $(= W)$ 方向に隣接する面のうち最も北 $(= N)$ にある面が f_{WN} である.

$$f_{WN}(i) = rank(S_1, match(S_1, rank(S_2, p_2, 1)+4), ()$$

である. 同様に, 面 f_i の西方向に隣接する面のうち最も南にある面を f_{WS} とすると,

$$f_{WS}(i) = rank(S_1, match(S_1, rank(S_2, match(S_2, select(S_2, a-3, 1) - 1), 1) + 4), ()$$

である. すなわち, (例3) のクエリの解は $O(1)$ 時間で求めることができる.

同様にして, $east(i)$ に対応する部分文字列 R_i をさがすことができ, f_i の東方向に隣接する面のうち, 最も北 (南) にある面 $f_{EN}(f_{ES})$ をそれぞれ求めることができる.

補題 4.2 S_R に $o(n)$ *bit* の長さの文字列を追加することにより L_i, R_i, $f_{WN}(i)$, $f_{WS}(i)$, $f_{EN}(i)$, $f_{ES}(i)$ を $O(1)$ 時間で求めることができる.

i 番目の点に対応する面 f_i の左下隅の点を v としよう. このとき v は s-missing, もしくは, w-missing のいずれかである. もし, v が s-missing であるならば, $s\text{-}miss(i, WS) = True$ とし, そうでないとき $s\text{-}miss(i, WS) = False$ とする. 次のようにして $s\text{-}miss(i, WS)$ を計算できる. もし $f_{ES}(f_{WS}(i)) = i$ ならば $s\text{-}miss(i, WS) = True$ である. もし $f_{ES}(f_{WS}(i)) \neq i$ ならば $s\text{-}miss(i, WS) = False$ である. 次の補題がいえる.

補題 4.3 指定された面の4隅の各点について, どの方向が *missing* であるかを $O(1)$ 時間で判定できる.

すなわち, (例4) のクエリの解を $O(1)$ 時間で求めることができる.

5 隣接クエリ

本章では, 3章で示した方形描画 R に対する文字列 $S_R = S_1 + S_2$ に, $o(n)$ bit の長さの文字列 S_A を追加することにより, R の隣接クエリの解を $O(1)$ 時間で計算できることを示す. ただし, 文字列 S_A の詳細は文献 [C96, C01, J89, M97, M01] 参照.

方形描画 R の2つの面 f_i と f_j が与えられたとき, 次の4つのクエリを考えよう.

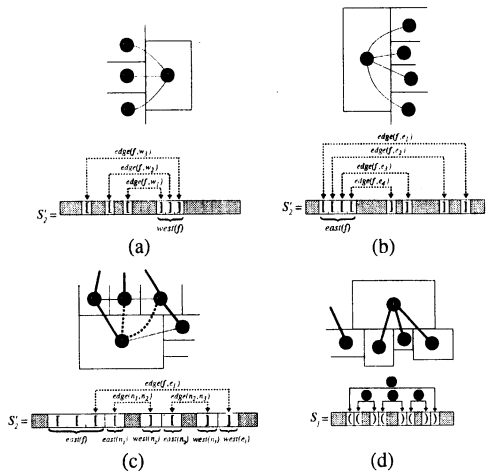


図 7: クエリの解の求め方の説明図

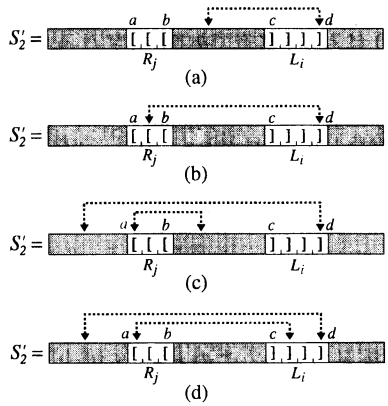


図 8: (wa) クエリの説明図

(wa) $f_j \in west(i)$: 面 f_i の西方向に面 f_j が隣接するかどうか? (図 7(a) 参照)
(ea) $f_j \in east(i)$: 面 f_i の東方向に面 f_j が隣接するかどうか? (図 7(b) 参照)
(na) $f_j \in north(i)$: 面 f_i の北方向に面 f_j が隣接するかどうか? (図 7(c) 参照)
(sa) $f_j \in south(i)$: 面 f_i の南方向に面 f_j が隣接するかどうか? (図 7(d) 参照)
$west(i)$ と $east(i)$ に対応する S_2 中の 2 つの部分文字列をそれぞれ L_i, R_i と書こう. R の双対グラフを D_R とする. 4 つのクエリを順に考えよう.

(wa) $f_j \in west(i)$: 面 f_i の西方向に面 f_j が隣接するかどうか ?

R_j 中のある "[" と L_i 中の "]" が対応するとき, またこのときに限り, $f_j \in west(i)$ である. (図 7(a) 参照.) そのような括弧の組があるかどうかを次のように調べよう.

$R_j = S_2[a,b]$, $L_i = S_2[c,d]$ とする. $a < b < c < d$ としてよい. 3 つの場合に分けて考えよう.

場合 1: $match(S_2,d) > b$ のとき. (図 8(a) 参照.)
L_i 中のどの "]" も R_j 中の "[" と対応しない. ゆえに, $f_j \notin west(i)$ である.

場合 2: $a \le match(S_2,d) \le b$ のとき. (図 8(b) 参照.)
$S_2[d]$ と $S_2[match(S_2,d)]$ が目的の括弧である. ゆえに, $f_j \in west(i)$ である.

場合 3: $match(S_2,d) < a$ のとき. (図 8(c) 参照.)
さらに 2 つの場合に分けて考えよう.

場合 3a: $match(S_2,a) < c$ のとき. (図 8(c) 参照.)
R_j 中のどの "[" も L_i 中の "]" と対応しない. ゆえに, $f_j \notin west(i)$ である.

場合 3b: $c \le \mathrm{match}(S_2,a) < d$ のとき. (図 8(d) 参照.)
$S_2[a]$ と $S_2[\mathrm{match}(S_2,a)]$ が目的の括弧である. ゆえに, $f_j \in west(i)$ である.

このようにして, 面 f_i の西方向に面 f_j が隣接するかどうかを調べることができる. 計算時間は $O(1)$ 時間である.

(ea) $f_j \in east(i)$: 面 f_i の東方向に面 f_j が隣接するかどうか ?
(wa) と同様である. 略.

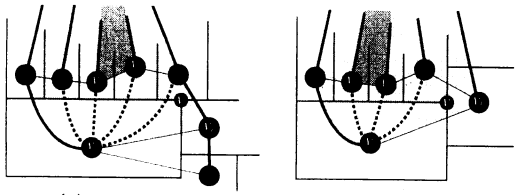
次に, 面 f_i の北方向についての隣接クエリの解を求める方法を説明する. この隣接関係に対応する辺が $E_{NS} - E_{NS}^T$ に含まれるときは, S_1 や S_2 中には直接格納されていないことに注意しよう. いくつかの準備からはじめよう. E_{NS} 等の定義については 3 章を参照されたい. D_R の点集合を $\{v_1, v_2, \ldots, v_{f+3}\}$ とする. v_1 は北面に対応し T_{NS} の根である.

はじめに, S_2 中の括弧の入れ子構造は, 双対グラフ D_R 中の領域の包含関係に対応することを示す.

各辺 $e = (v_i, v_k) \in E_{EW}$ に対して, D_R の領域 $R(v_i, v_k)$ を次のように割りあてよう. T_{NS} 上の, v_i から v_1 へのパスを P_i とし, v_k から v_1 へのパスを P_k とする. P_i と P_k と辺 e によって囲まれる面からなる領域を $R(v_i, v_k)$ としよう. D_R は平面グラフであるので, 任意の 2 つの領域 $R(v_i, v_k)$ と $R(v_j, v_l)$ が与えられたとき, どちらか一方がもう一方の一部分のみを真に含むことはない. 次の補題がいえる.

補題 5.1 $R(v_i, v_k)$ が $R(v_j, v_l)$ を真に含むための必要十分条件は, 辺 (v_i, v_k) に対応する S_2 中の "[" と "]" の組の内側の部分文字列が, 辺 (v_j, v_l) に対応する S_2 中の "[" と "]" の組を含むことである.

さて, f_i の北方向の親でない隣接面 f_j について考えよう. $k = f_{EN}(i)$ とし, これに対応する点を v_k とする. このとき, f_i の右上隅の点 v が n-missing か



(a) v: n-missing　(b) v: e-missing

図 9: f_i の北方向の隣接面の様子

e-missing であるかによって 2 つの場合がある. (図 9 参照.) 点 v が (i) n-missing ならば $l = f_{WS}(j)$ とし (図 9(a) 参照), (ii) e-missing ならば $l = f_{ES}(j)$ としよう. (図 9(b) 参照.) これに対応する点を v_l とする. 辺 (v_i, v_k) と辺 (v_j, v_l) はともに東西方向の隣接関係に対応している. このとき, また, このときのみ辺 (v_i, v_k) に対応する "[" と "]" が辺 (v_j, v_l) に対応する "[" と "]" を *enclose* するならば, 領域 $R(v_i, v_k)$ から辺 (v_i, v_k) を輪郭上にもつ面を削除して得られる領域は辺 (v_j, v_l) を外周上に含むことに注意しよう.

次がいえる.

補題 5.2 f_i の親面でない面 f_j が面 f_i の北方向に隣接するための必要十分条件は, 辺 (v_i, v_k) に対応する "[" と "]" が辺 (v_j, v_l) に対応する "[" と "]" を *enclose* することである. (図 9 参照.)

補題 5.2 より, 次のようにして面 f_i の北方向についての隣接クエリの解を求めることができる.

(na) $f_j \in north(i)$: 面 f_i の北方向に面 f_j が隣接するかどうか?

次の 2 つの場合に分けて考える.

場合 1: f_j が f_i の親面のとき.

$f_j \in north(i)$ である. *enclose* 命令により, f_j が f_i の親面になっているかどうかを $O(1)$ 時間で計算できることに注意しよう.

場合 2: 場合 1 でないとき.

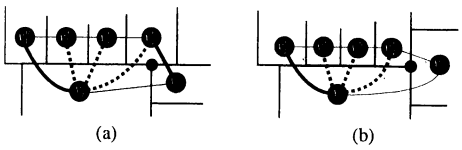
補題 5.2 より, 辺 (v_i, v_k) に対応する S_2' 中の "[" と "]" が, 辺 (f_j, f_l) に対応する S_2' 中の "[" と "]" を, *enclose* するときのみ $f_j \in north(i)$ である. f_i の右上隅の点を v とし, v によって次の 2 つの場合に分けて考える. $a = match(S_1, select(S_1, i, ()))$ とする. ここで, a は, v_i に対応する S_1 中の ")" を示している.

場合 2(a): v が n-missing であるとき (図 9(a) 参照.)

このとき, $l = f_{WS}(j)$ である. $b = select(S_1, j, ()$ とすると,

$$enclose(S_2, select(S_2, b-3, 1) - 1) = select(S_2, a-3, 1) - 1$$

であるときのみ $f_j \in north(i)$ である. ここで, $select(S_2, b-4, 1)$ は $|west(j)|$ 個の "]" のうち最初の "]" を示しているので, $select(S_2, b-3, 1)$ は $|west(j)|$ 個の "]" の文字列の "次の" 文字を示している. よって, $select(S_2, b-3, 1) - 1$ は $|west(j)|$ 個の "]" の文字列中の最後の "]" を示しており, これは辺 (v_j, v_l) に対応することに注意されたい.



(a)　(b)

図 10: (nd) クエリの説明図

場合 2(b): v が e-missing であるとき (図 9(b) 参照.)

このとき, $l = f_{ES}(j)$ である. 場合 2(a) と同様に, $c = match(S_1, select(S_1, j, ()))$ とすると,

$$enclose(S_2, select(S_2, c-4, 1)) = select(S_2, a-3, 1) - 1$$

であるときのみ $f_j \in north(i)$ である.

最後に, 面 f_i の南方向についての隣接クエリの解を求める方法を説明する.

(sa) $f_j \in south(i)$: 面 f_i の南方向に面 f_j が隣接するかどうか?

f_i が f_j の北側に隣接するとき, またこのときに限り, f_j は f_i の南側に隣接する. よって, (nd) により, f_j が f_i の南側に隣接するかどうかを調べることができる.

6 次数クエリ

本章は, 方形描画 R に対する文字列 S_R に, $o(n)$bit の長さの文字列 S_A を追加することにより, R の面 f_i が与えられたとき, f_i の次数を $O(1)$ 時間で求める方法を説明する. ここで, f_i の次数とは, f_i に隣接する面の個数である. 4 方向のそれぞれに対応する 4 つのクエリ (wd), (ed), (nd), (sd) にわけて考えよう.

(wd) $|west(i)|$:面 f_i の西方向に隣接する面の個数は?

$west(i)$ の個数, すなわち, $|L_i|$ を求めればよい. よって, $O(1)$ 時間で十分である. (図 7(a) 参照.)

(ed) $|east(i)|$:面 f_i の東方向に隣接する面の個数は?

(wd) と同様である. 略.

(nd) $|north(i)|$:面 f_i の北方向に隣接する面の個数は?

f_i に対応する点を v_i とする. f_i の東方向に隣接する面のうち, 一番北にある面を $f_{EN}(i) = k$ とし, これに対応する点を v_k とする. (図 10 参照.) f_i の北方向に隣接する面を, 西から東へ順に, $n_1, n_2, \ldots, n_{|north(i)|}$ とする. これらの面に対応する点を $v_1^n, v_2^n, \ldots, v_{|north(i)|}^n$ とする. 面 n_1 は面 f_i の親面であることに注意しよう. f_i の右上隅の点を v とする. 2 つの場合にわけて考えよう.

場合 1: 点 v が n-missing のとき. (図 10(a) 参照.)

辺 (v_i, v_k) に対応する S_2' 中の "[" と "]" は，それぞれ a, b 文字目であるとする．基本クエリにより，a と b の値を $O(1)$ 時間で計算できることに注意しよう．このとき，$S_2'[a]$ と $S_2'[b]$ は，辺 $(v_1^n, v_2^n), (v_2^n, v_3^n), \ldots, (v_{|north(i)|-1}^n, v_{|north(i)|}^n)$ に対応する "[" と "]" の各組をすべて *enclose* し，かつ，$S_2'[a]$ と $S_2'[b]$ が *enclose* する括弧はこれらに限られる．したがって，$|north(i)| = (wrapped(S_2, a)/2) + 1$ である．

場合 2: 点 v が e-missing のとき．(図 10(b) 参照.)
$|north(i)| = wrapped(S_2, a)/2$ である．(詳細はテクニカルレポート参照.)

(sd) $|south(i)|$:面 f_i の南方向に隣接する面の個数は?
f_i の左下隅の点を v とする．次の 2 つに場合にわけて考えよう．

場合 1: 点 v が w-missing のとき
このとき，R の面が f_i の南側に隣接するための必要十分条件は f_i を親面とすることである．よって，f_i を親面とするような面の個数を求めればよい．ゆえに，$|south(i)| = wrapped(S_1, select(S_1, i, ()))/2$ 個である．このようにして，$|south(i)|$ を $O(1)$ 時間で求めることができる．

場合 2: 点 v が s-missing のとき
このとき，場合 1 とは異なり，f_i の南方向に隣接する面は，f_i を親面とするような面の他にも，ちょうど 1 個ある．その面は $south(i)$ の中で最も西に位置する．(図 7(d) 参照.) 以上により，$|south(i)| = 1 + wrapped(S_1, select(S_1, i, ()))/2$ である．よって，$|south(i)|$ を $O(1)$ 時間で求めることができる．

以上により，4 方向のそれぞれに関して，面 f_i に隣接する面の個数を $O(1)$ 時間で求めることができる．よって，f_i の次数を $O(1)$ 時間で求めることができる．

7 再構成

方形描画 R の文字列 $S = S_R + S_A$ が与えられたとき，4 章と 5 章のクエリを用いて R を再構成できる．次の定理がいえる．

定理 7.1 方形描画 R に対する文字列 S が与えられたとき，S から R を $O(n)$ 時間で再構成できる．

8 まとめ

本文では，方形描画 R に対するコンパクトな文字列 S を与えた．また，方形描画の文字列 S が与えられたとき，(1) 隣接クエリ・次数クエリの解を $O(1)$ 時間で求める方法と，(2) S からもとの方形描画 R を $O(n)$ 時間で再構成する方法，を与えた．

底辺つき方形描画を高速に列挙するアルゴリズムが知られている [N02a, N02b]．k 個の面をもつ底辺つき方形描画の個数を N_k としよう．我々は，上記の列挙アルゴリズムを実装することにより，底辺つき方形描画の個数の数え上げを行い，$N_{11} = 10948768$ という結果を得た．よって，11 個の面をもつ底辺つき方形描画に対する文字列の平均の長さは，少なくとも $24 > \log N_{11} = 23.5$ bit であるとわかる．一方，本文で求めた文字列は，クエリを高速にサポートするための文字列 S_A を除いても，$5 \times 11 = 55$ bit である．したがって，文字列の長さを改善する可能性がまだあることがわかる．高速にクエリの解を求めることができ，かつ，より短い文字列を設計することが今後の課題である．

本文では，紙面の都合上，補題の証明等を省略した．よって，それらを記述した文書をテクニカルレポートとして *http://www.msc.cs.gunma-u.ac.jp/~yamanaka/Tech/CEFEQ.pdf* にて公開する．

参考文献

[C01] Y.-T. Chiang, C.-C. Lin and H.-I Lu, *Orderly Spanning Trees with Applications to Graph Encoding and Graph Drawing,* Proc. of 12th Annual ACM-SIAM Symposium on Discrete Algorithms, (2001), pp.506-515.

[C98] R. C.-N Chuang, A. Garg, X. He, M.-Y. Kao, H.-I Lu, *Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses,* Proc. of 25th International Colloquium on Automata, Languages, and Programming, LNCS 1443, (1998), pp.118-129.

[C96] D. Clark, *Compact Pat Trees,* Ph. D, thesis, Department of Computer Science, University of Waterloo, (1996).

[G94] R. L. Graham, D. E. Knuth and O. Patashnik, *Concrete Mathematics, 2nd ed.,* Addison-Wesley, (1994).

[H99] X. He, *On Floor-Plan of Plane Graph,* SIAM J. on Computing, 28, (1999), pp.2150-2167.

[J89] G. Jacobson, *Succinct Static Data Structures,* Ph. D, thesis CMU-CS-89-112, Carnegie Mellon University, (1989).

[K95] K. Keeler and J. Westbrook, *Short Encodings of Planar Graphs and Maps,* Discrete Applied Mathematics, 58, No 3, (1995), pp.239-252.

[M97] J. I. Munro and V. Raman, *Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs,* Proc. of 38th IEEE Symposium on Foundations of Computer Science, (1997), pp.118-126.

[M01] J. I. Munro and V. Raman, *Succinct Representation of Balanced Parentheses and Static Trees,* SIAM Journal on Computing, 31(3), (2001), pp.762-776.

[N02a] S. Nakano, *Enumerating Floorplans with n Rooms,* IEICE TRANS. FUNDAMENTALS, Vol. E85-A, No. 7, (2002), pp.1746-1750.

[N02b] S. Nakano, *Enumerating Floorplans with Some Properties,* Interdisciplinary Information Sciences, Vol. 8, No. 2, (2002), pp.199-206.

[P86] C. Papadimitriou and M. Yannakakis, *A Note on Succinct Representations of Graphs,* Information and Control, Vol. 71, (1986), pp.181-185.

[R00] K. H. Rosen (Eds.), *Handbook of Discrete and Combinatorial Mathematics,* CRC Press, Boca Raton, (2000).

[T84] G. Turan, *Succinct Representations of Graphs,* Discrete Applied Math, Vol. 8, (1984), pp.289-294.