グラフの連結成分を数えるカードプロトコル

縫田 光司(NUIDA, Koji)

九州大学マス・フォア・インダストリ研究所
／産業技術総合研究所

CSS2024@神戸　3H5-1　 2024年10月24日

本研究の概要

- **グラフの連結成分の個数**を得るカードプロトコル
  - グラフの頂点集合の部分集合Sを入力として、Sの情報を秘匿しつつ
    Sの連結成分の個数だけを得る
- （秘密計算でない）アルゴリズム構成 → カードプロトコルへ変換
- 既存手法よりもシャッフル回数のオーダーが改善

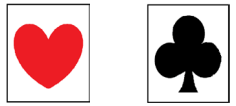
目次

- 研究の背景と問題設定
- 成果：秘密計算でないアルゴリズム
- 成果：秘密計算のカードプロトコル

目次

「物理暗号」、カードベース暗号

- 「物理的」(非電子的)な道具で**暗号的操作**を実現する分野
  - 秘密計算
  - ゼロ知識証明
  - ランダムネス生成
  - …
- カードベース暗号:カードを伏せる/めくる、シャッフルする、…
  - 2色カード、上下カード、トランプ、UNO、…

表側: 

裏側: 

パズルの解の知識のゼロ知識証明

- 「このパズルの解を知っている」ことを立証する
- ただし、解そのものは検証者にまったく明かさない

- カードベース暗号の代表的研究テーマの一つ
  - 数独の研究が最も有名
  - 多様なパズルが研究対象：スリザーリンク([LMM+21]他)、ぬりかべ([RML+22]他)、数コロ([SS23])、…

[LMM+21] P. Lafourcade, D. Miyahara, T. Mizuki, L. Robert, T. Sasaki, H. Sone: Theoretical Computer Science, vol.888, pp.41-55 (2021)

[RML+22] L. Robert, D. Miyahara, P. Lafourcade, T. Mizuki: New Generation Computing, vol.40, no.1, pp.149-171 (2022)

[SS23] 佐々木 駿、品川 和雅：SCIS 2023, 3D2-4 (2023)

パズルの解の知識のゼロ知識証明

- 基本方針：
  1. 証明者はパズルの解を（伏せた）カードで表示
  2. 検証者は、パズルの解の条件を一つずつ確認
     例（数独の場合）：「各行の数字が異なる」＆「各列の数字が異なる」
     ＆「各ブロックの数字が異なる」
- よくある「条件」の一つ：「解が連結である」
  - 例（ぬりかべ）：黒マスたちが**一つながり**である
  - 例（スリザーリンク）：線たちが**一つの**輪っかになっている

解の連結性の確認：既存手法の特徴

- 各パズルの特性に応じた**特注品**である
  - 例（数コロ [SS23]）：実際には「解領域の既知の点たちが同一の連結成分に属する」ことを確認
- **証明者がオンライン**である必要がある（[LMM+21]、[RML+22]の場合）
  - 例（スリザーリンク [LMM+21]）：既知の輪っかから出発して証明者と検証者が**対話的に**解の輪っかへと変形させる

- 課題：**汎用的**で**非対話的**な連結性確認手法を得たい

本研究の成果

- （有限単純無向）**グラフの連結成分の個数**を得る非対話プロトコル
  - グラフの頂点集合の部分集合Sを入力として、Sの情報を秘匿しつつ
    Sの連結成分の個数だけを得る
  - 「連結成分が1個かどうか」の計算にも転用可能
- （秘密計算でない）アルゴリズム構成 → カードプロトコルへ変換
- [SS23]の手法よりもシャッフル回数のオーダーが改善
  - $M \times M$ 格子グラフ: $O(M^4) \rightarrow O(M^3 \log M)$

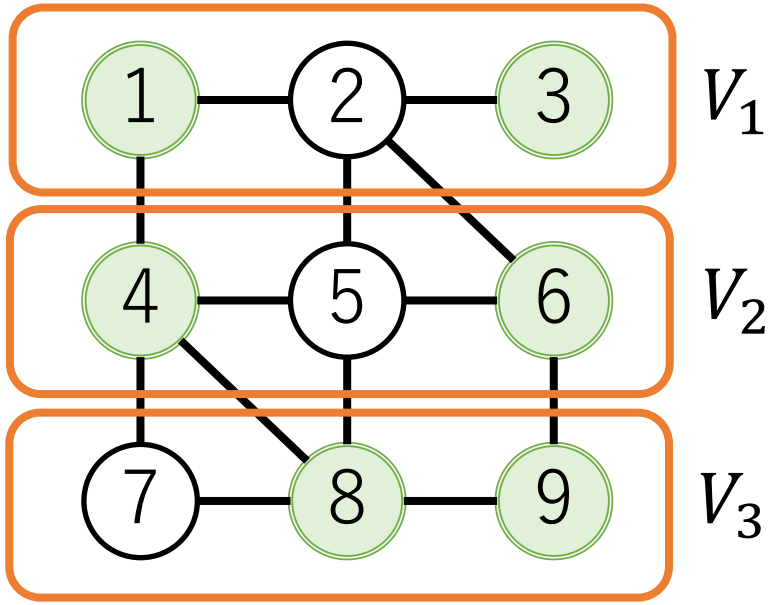
目次

- 研究の背景と問題設定
- 成果：秘密計算でないアルゴリズム
- 成果：秘密計算のカードプロトコル

準備:グラフの分割

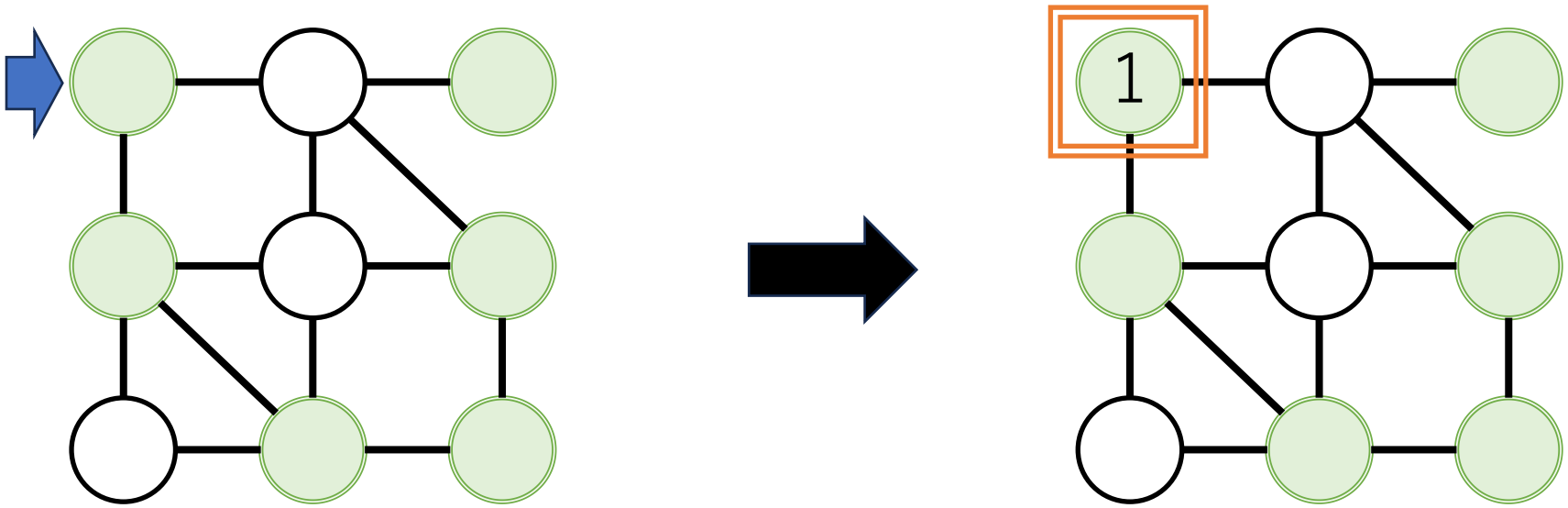
- グラフの頂点集合を区画 $V_1, \ldots, V_L$ に分割(公開情報)
- 区画 V_i の頂点と隣接する頂点は、V_{i-1}, V_i, V_{i+1} のどれかに入る
  - 二つ以上離れた区画の頂点とは隣接しない

- 例:



(塗られた点がSの点)

提案アルゴリズム(例)

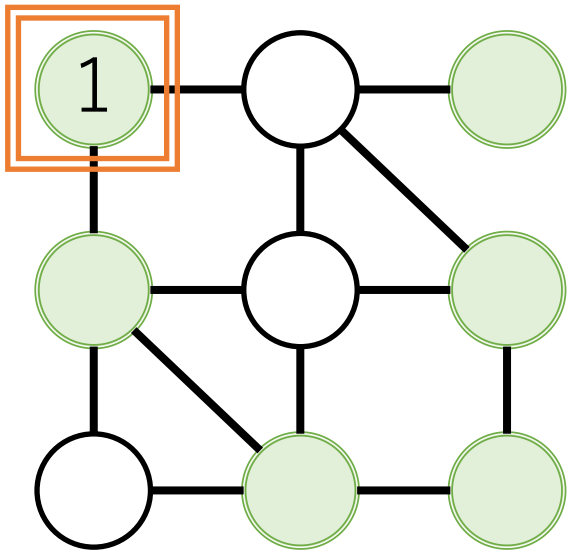


先頭の区画から順に、各頂点 v に着目:

Sの点なら、自身の番号を入れて「印」を付ける

連結成分の代表点

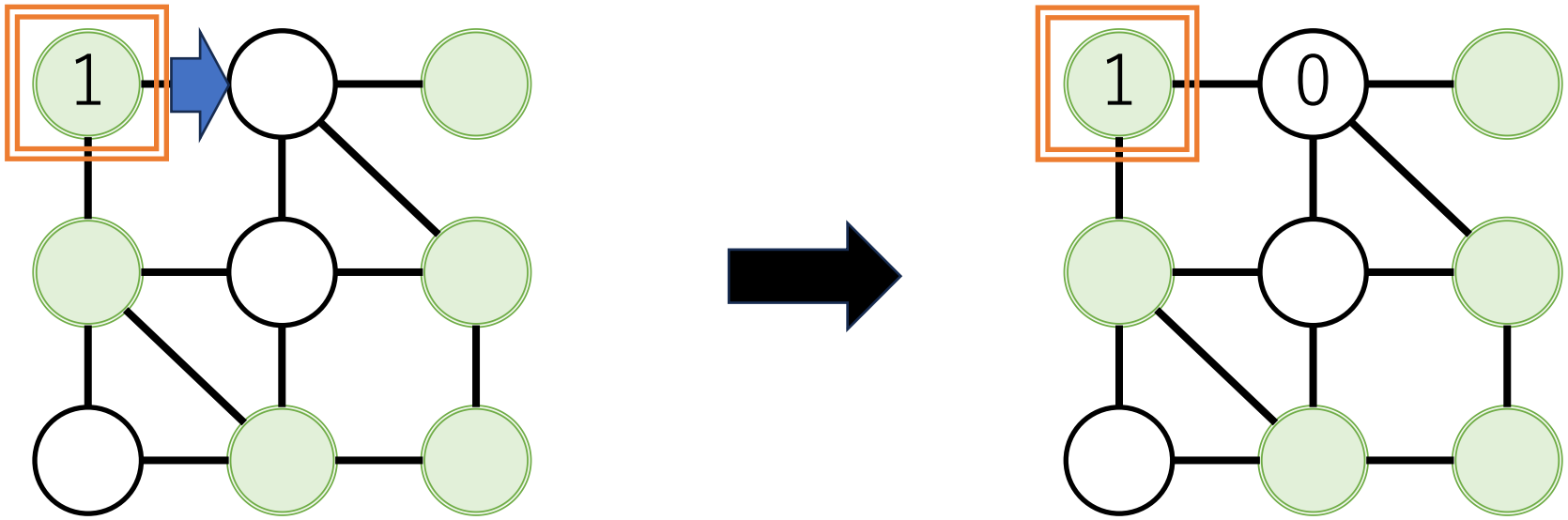
提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

(今は該当なし → 次の点に進む)

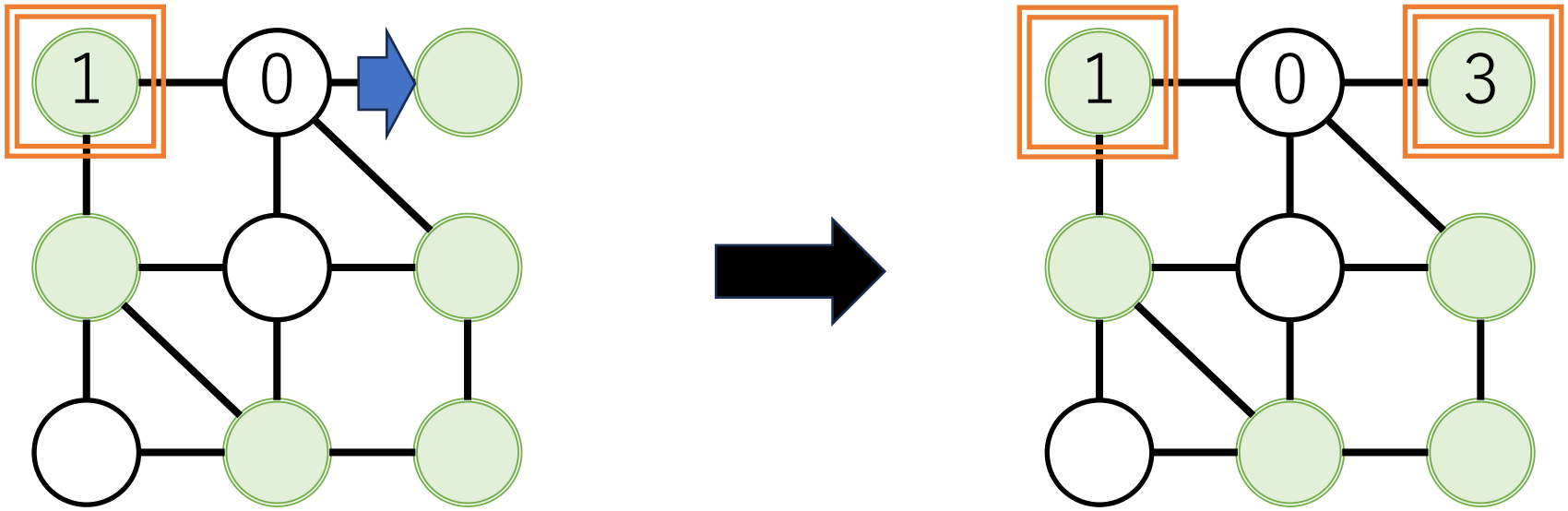
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

(Sの点でない → 「0」を入れて次の点に進む)

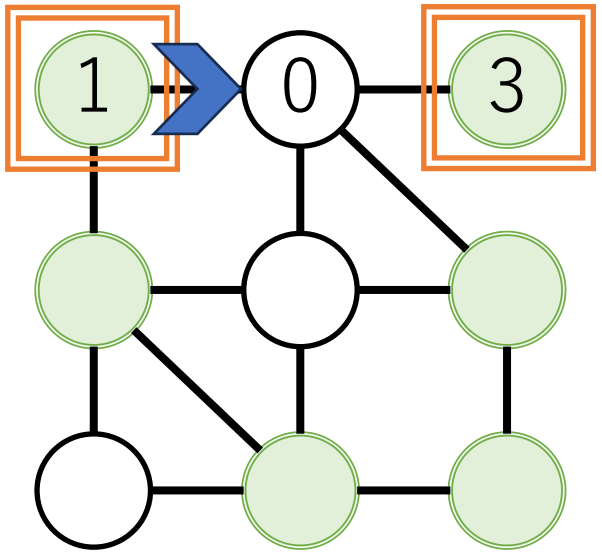
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

Sの点なら、自身の番号を入れて「印」を付ける

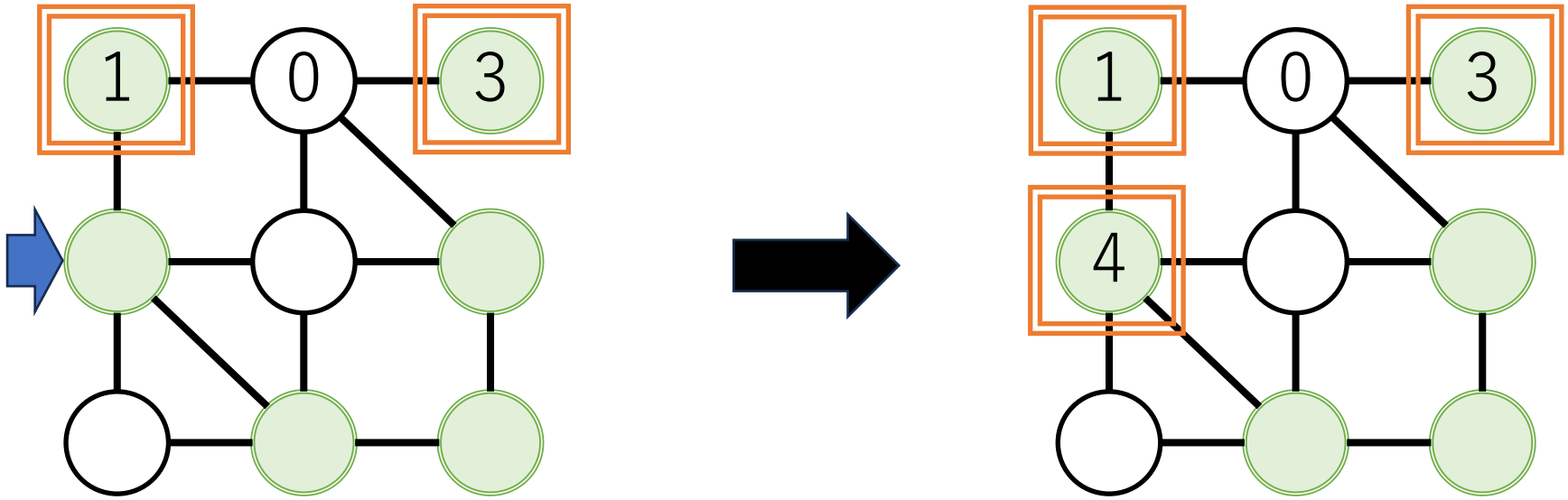
提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

(「0」の点しかないので、次に進む)

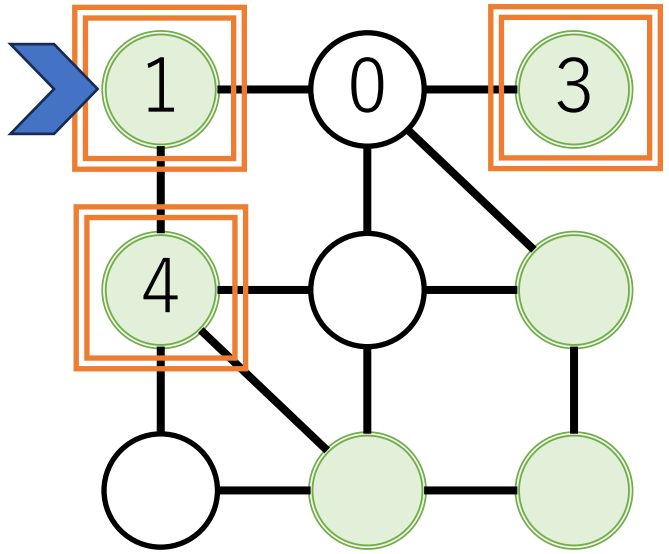
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

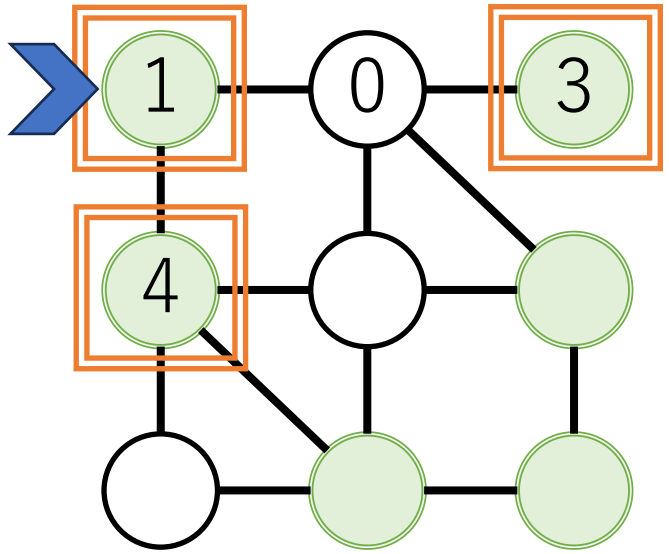
Sの点なら、自身の番号を入れて「印」を付ける

提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

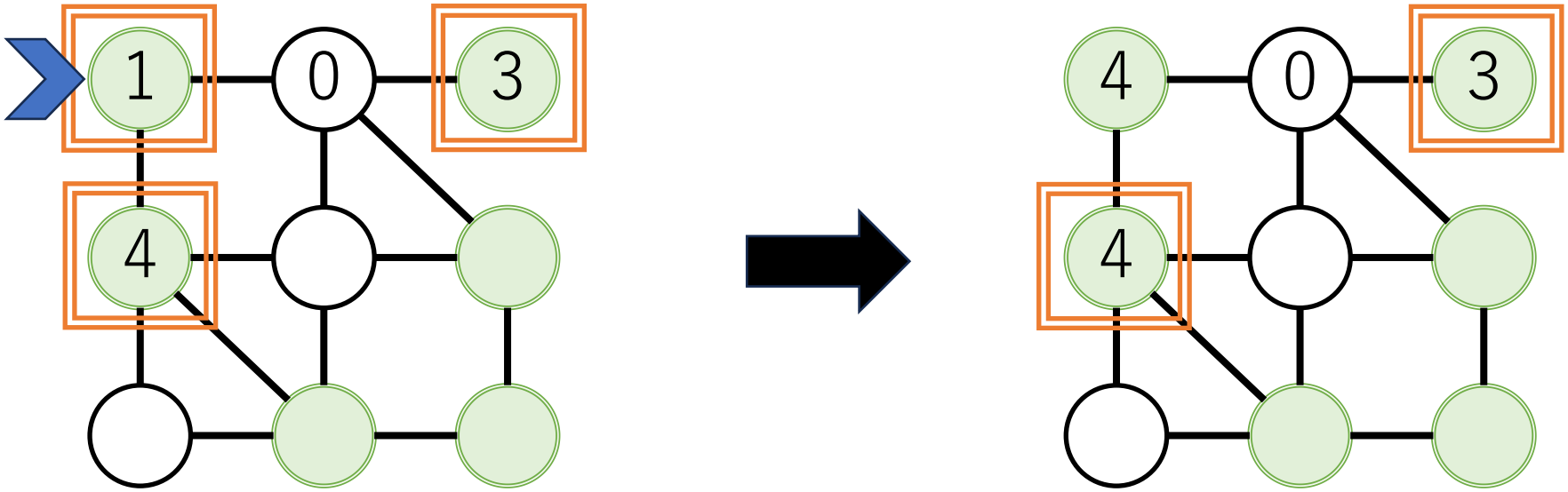
提案アルゴリズム(例)



v と同じか一つ前の区画で、v' と同じ値の点 v'' に着目:

(今は該当なし)

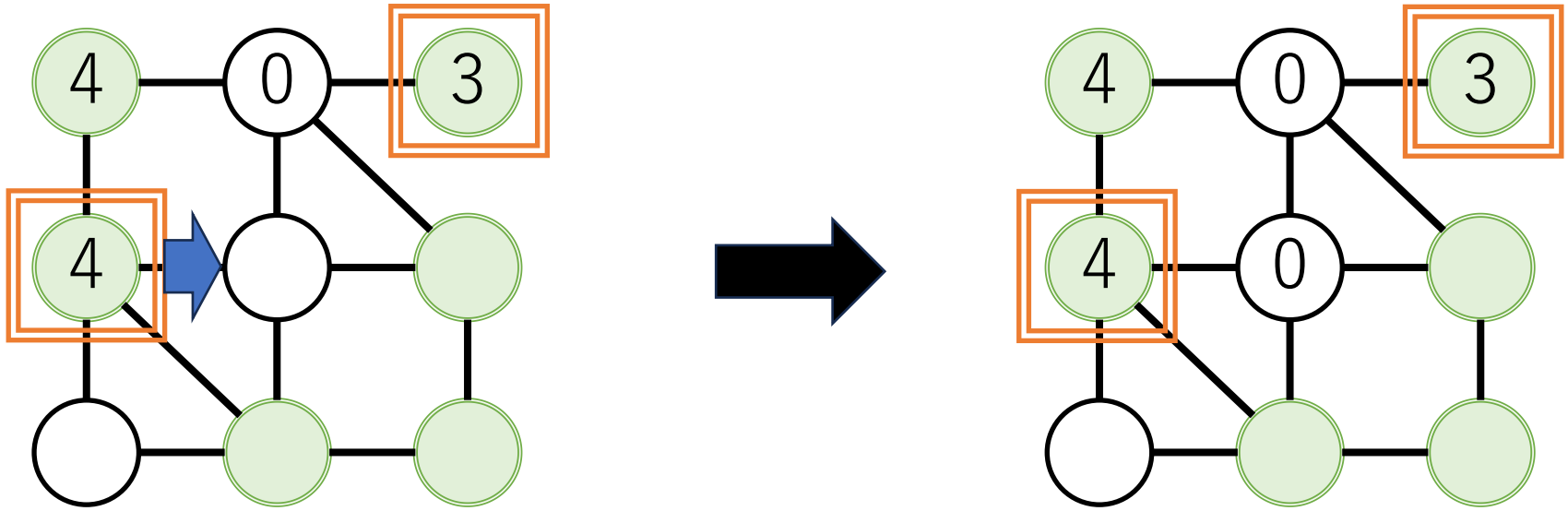
提案アルゴリズム(例)



v' を v と同じ値にして、「印」を削除

次の点に進む

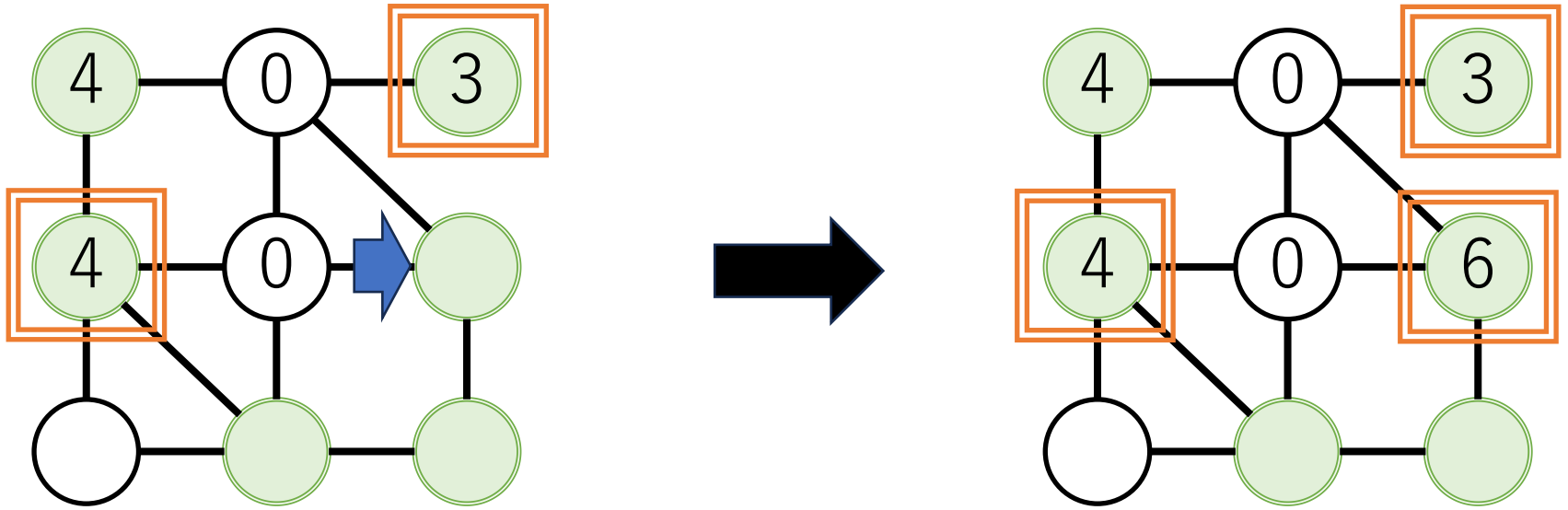
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

(Sの点でない → 「0」を入れて次の点に進む)

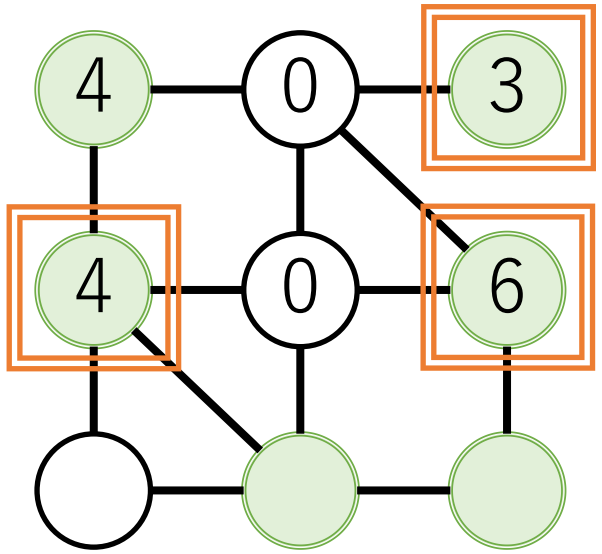
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

Sの点なら、自身の番号を入れて「印」を付ける

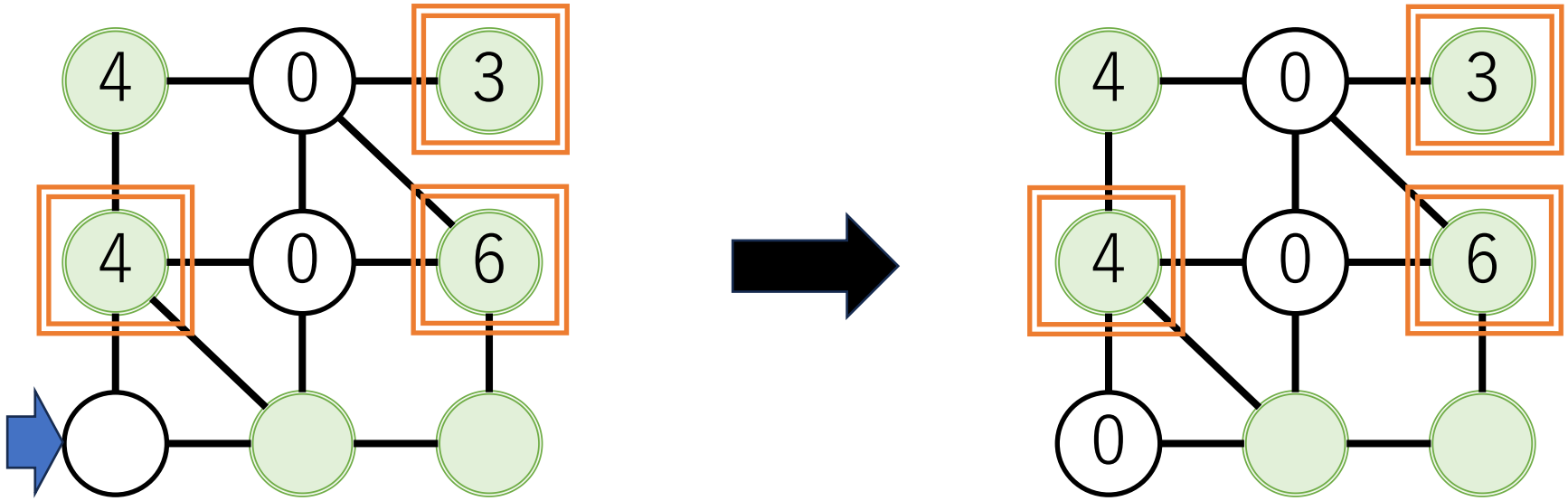
提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

(「0」の点しかないので、次に進む)

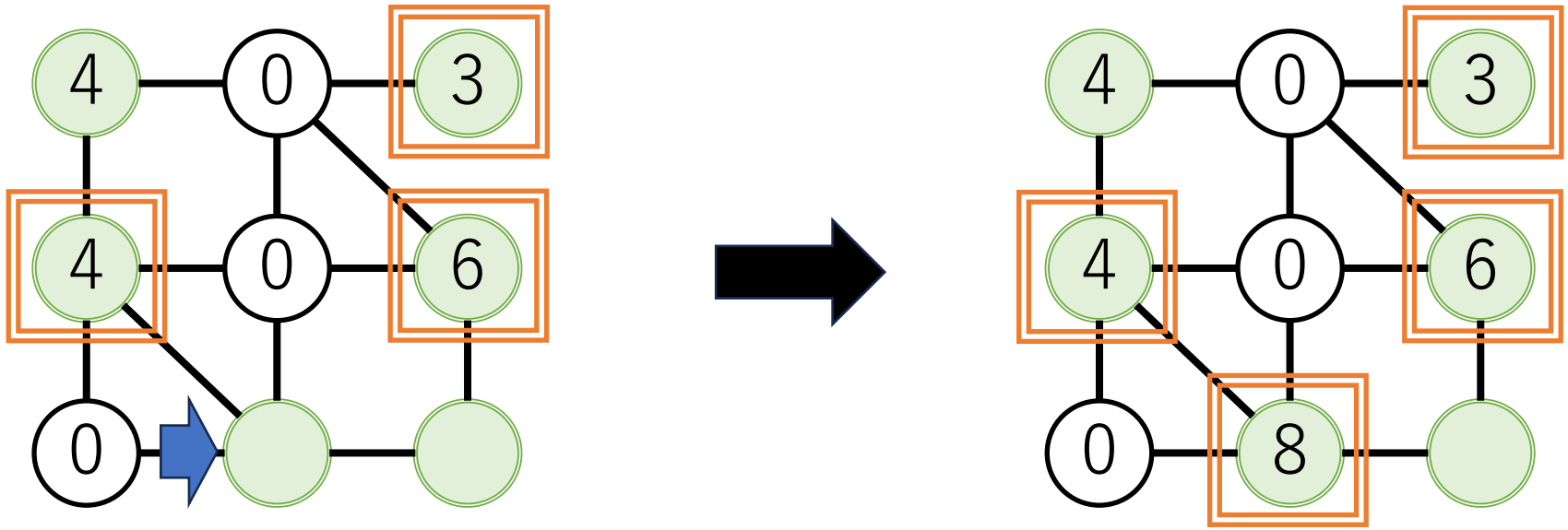
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

(Sの点でない → 「0」を入れて次の点に進む)

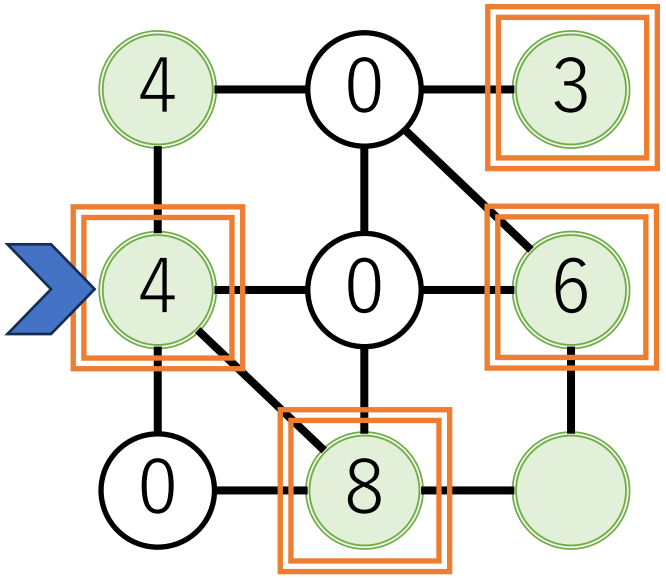
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

Sの点なら、自身の番号を入れて「印」を付ける

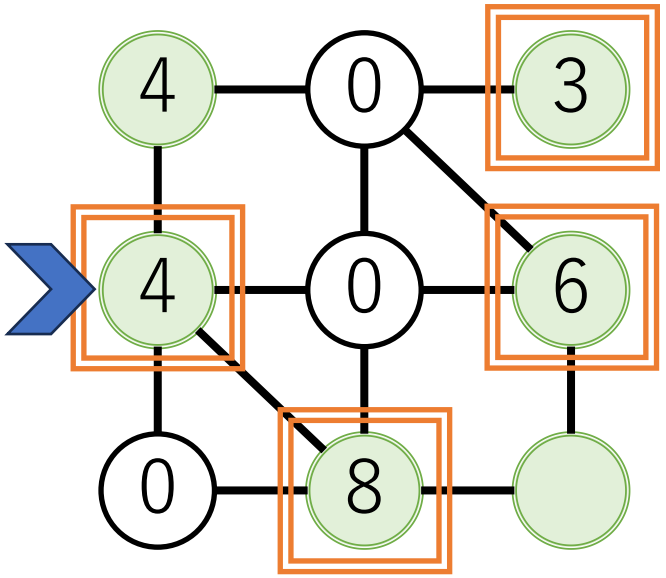
提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

「0」でない点にのみ着目する

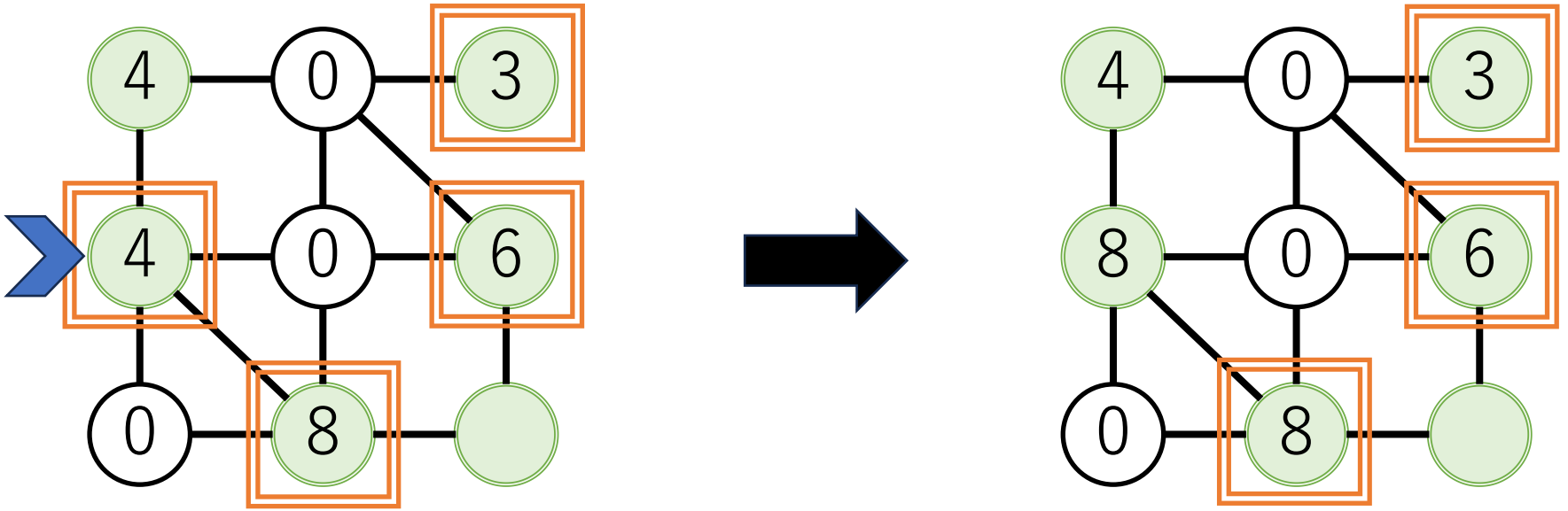
提案アルゴリズム(例)



v と同じか一つ前の区画で、v' と同じ値の点 v'' に着目:

(今は該当なし)

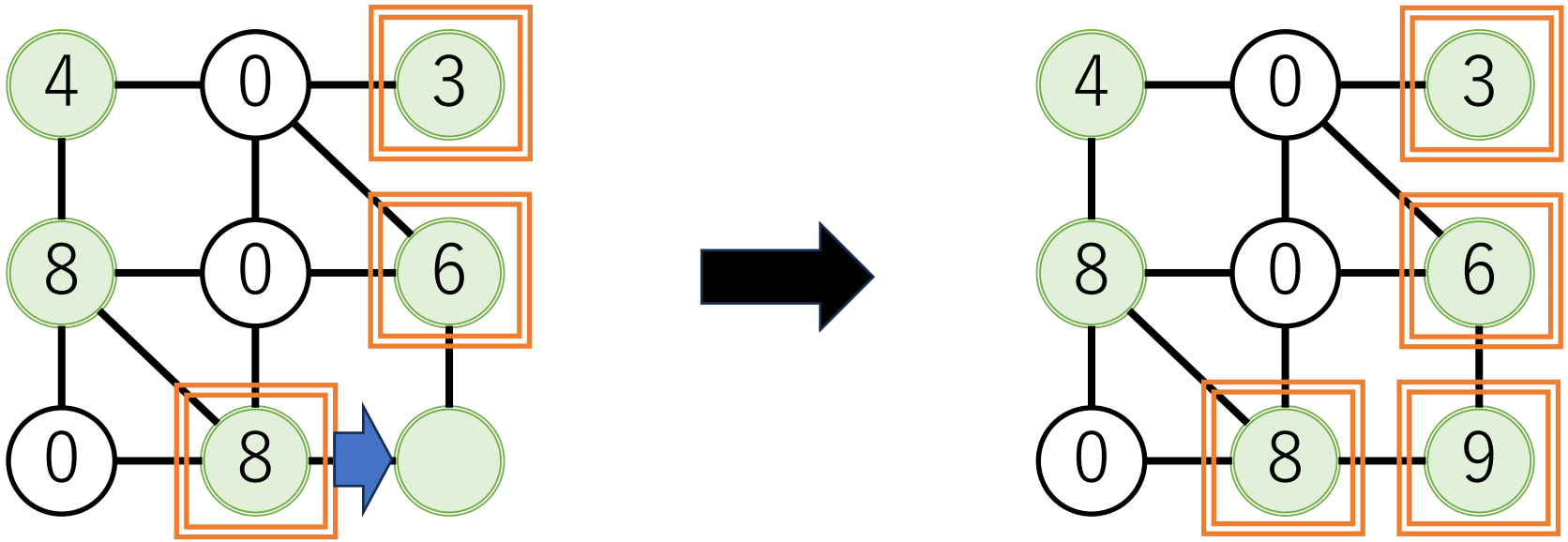
提案アルゴリズム(例)



v' を v と同じ値にして、「印」を削除

次の点に進む

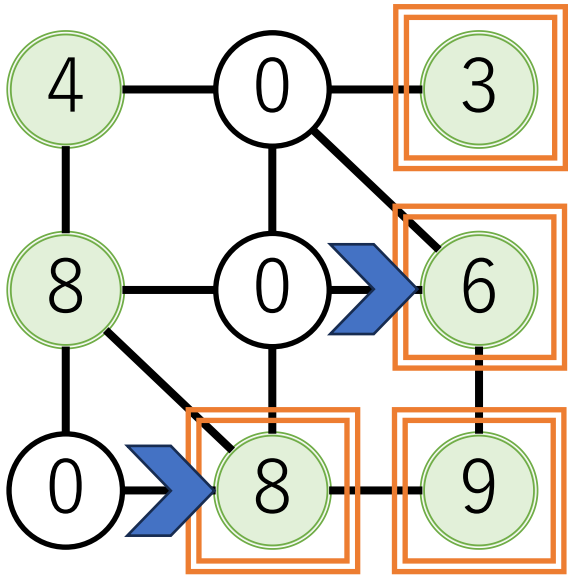
提案アルゴリズム(例)



先頭の区画から順に、各頂点 v に着目:

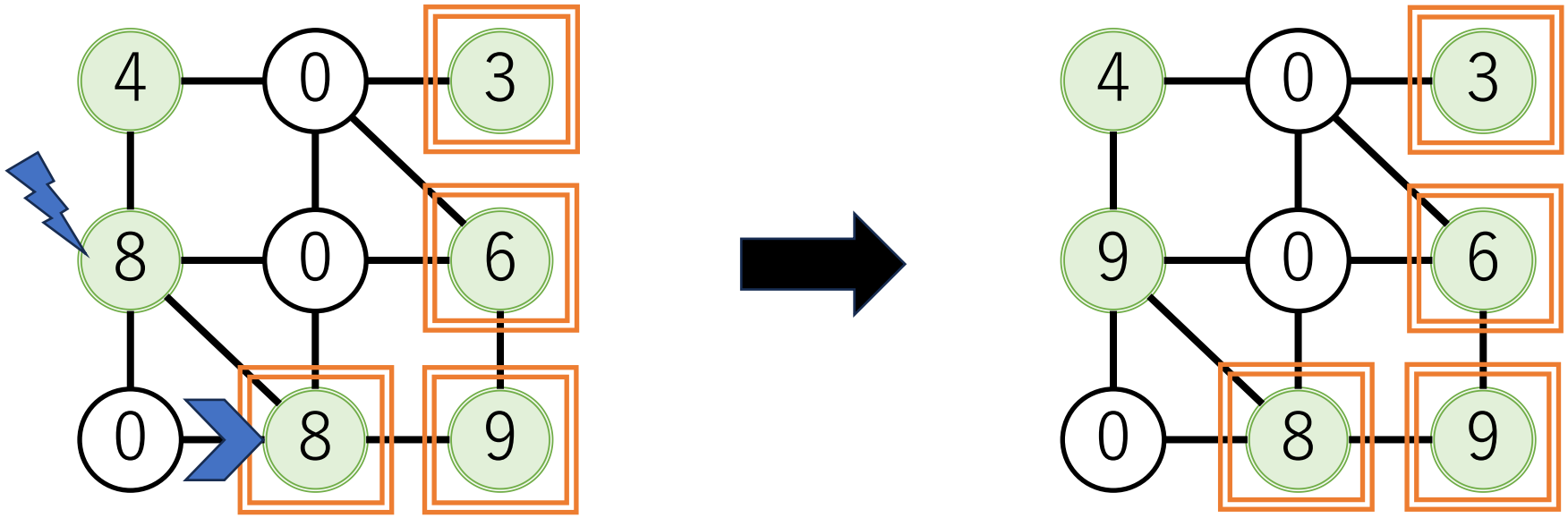
Sの点なら、自身の番号を入れて「印」を付ける

提案アルゴリズム(例)



v よりも前にある隣接点 v' に着目:

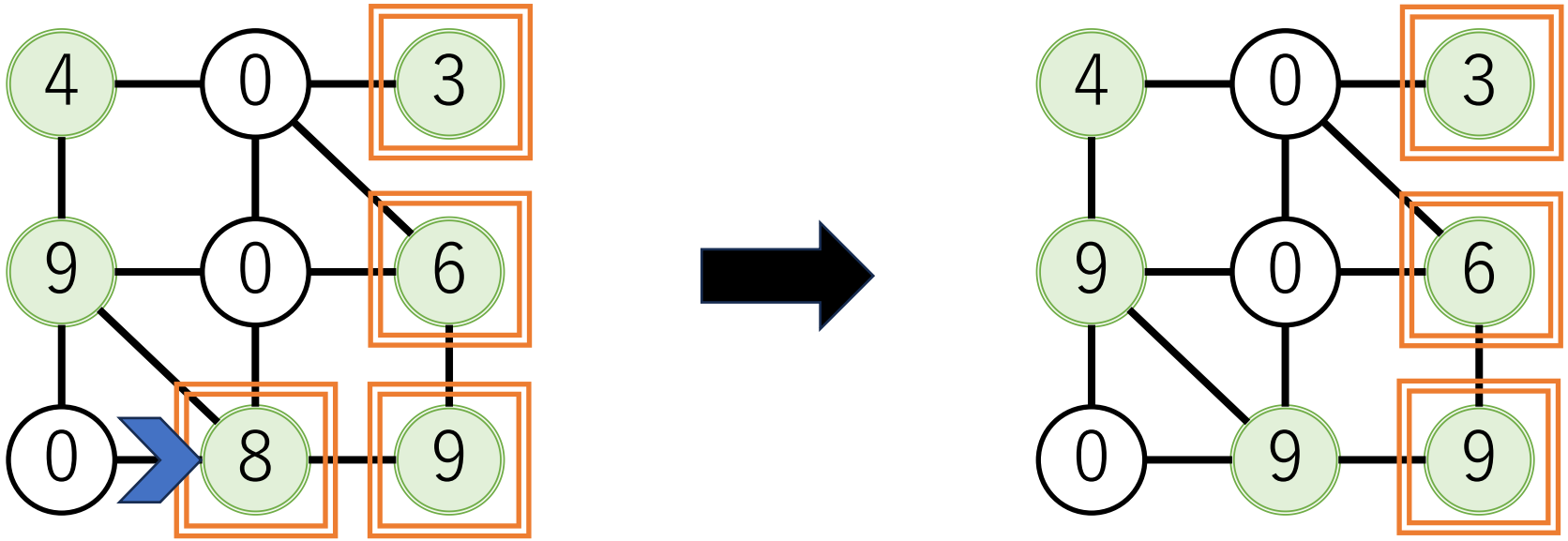
提案アルゴリズム(例)



v と同じか一つ前の区画で、v' と同じ値の点 v'' に着目：

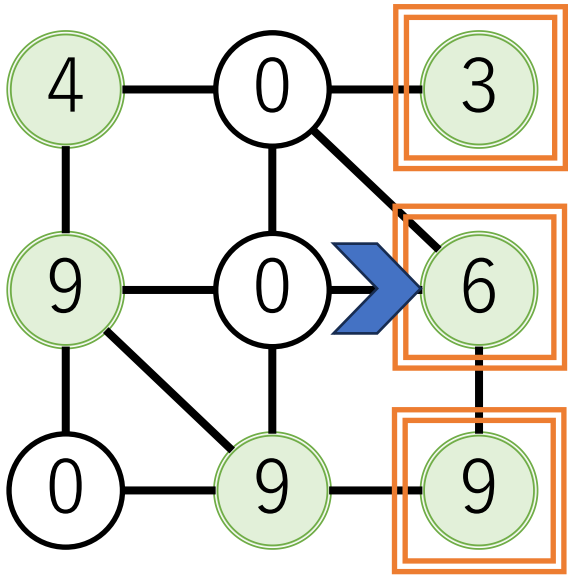
v'' の値を v と同じ値にする

提案アルゴリズム(例)



v' を v と同じ値にして、「印」を削除

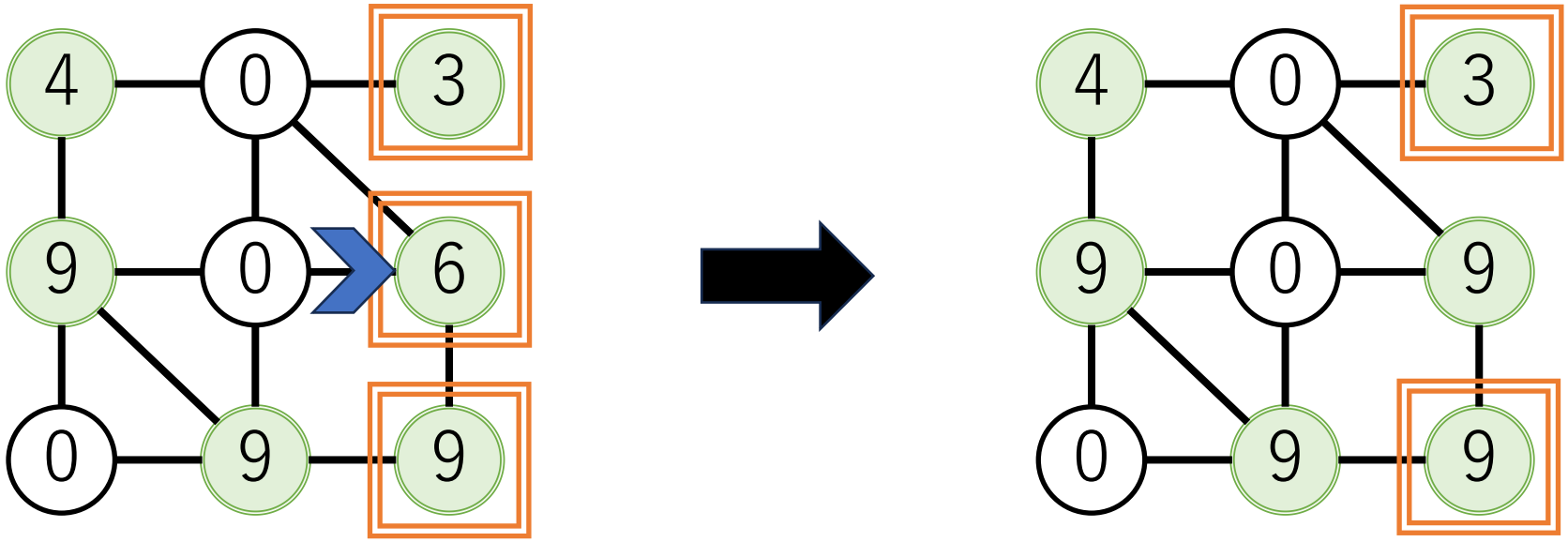
提案アルゴリズム(例)



v と同じか一つ前の区画で、v' と同じ値の点 v'' に着目:

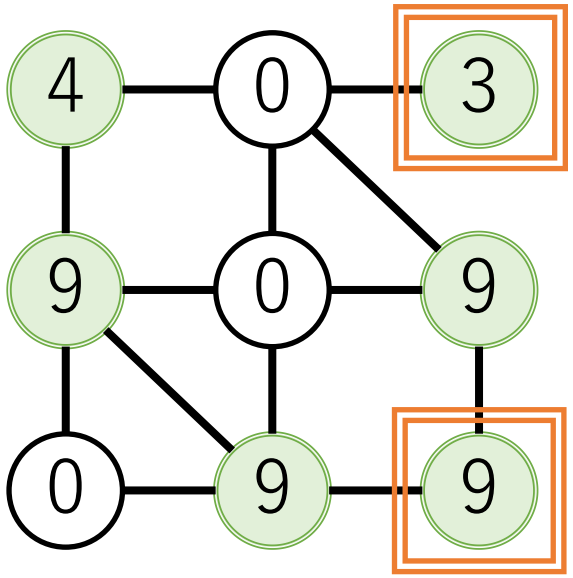
(今は該当なし)

提案アルゴリズム(例)



v' を v と同じ値にして、「印」を削除

提案アルゴリズム(例)



「印」は各連結成分の代表点を与える

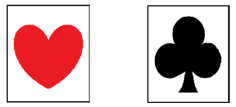
→「印」を数えれば連結成分の個数がわかる

目次

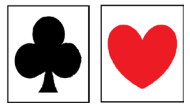
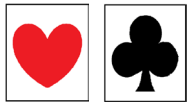
- 研究の背景と問題設定
- 成果：秘密計算でないアルゴリズム
- 成果：秘密計算のカードプロトコル

値の表示(コミットメント)

- カードベース暗号:カードを伏せる／めくる、シャッフルする、…
  - 2色カード、上下カード、トランプ、UNO、…

表側: 

裏側: 

- ビットのコミットメント(実際は裏向きに置く)$Com(b)$:
  0 →  、1 → 

- 整数の表示:二進法表記のビットごとのコミットメント

提案プロトコルの入出力

- 入力：各点がSの点か否かを表すビットのコミットメントたち
- 出力：各連結成分の代表点の集合
  - 「印」が1、それ以外が0というビットのコミットメントたち

- 出力の総和で連結成分の個数を計算できる
- 「総和が1か否か」で連結性を判定できる

使用するサブプロトコル

ランダム二等分割カット(2束のランダムな交換)

$1+\ell$ 個に複製

Mizuki-Sone

プロトコル	シャッフル回数	カード枚数 入力	追加	解放	廃棄
Dup^{ℓ} (*)	k RBC	$2k$	$2\ell k+2$	2	0
$\neg$	0	2	0	0	0
$\oplus$	1 RBC	4	0	2	0
$\wedge$	1 RBC	4	2	2	2
$\oplus_{\mathsf{Dup}}$	2 RBC	4	4	2	0
OExc, OSel, OSub	1 RBC	2 (§)	0	2	0
$\mathsf{OExc}_{\mathsf{Dup}}$, $\mathsf{OSel}_{\mathsf{Dup}}$, $\mathsf{OSub}_{\mathsf{Dup}}$	1 RBC	2 (§)	2	2	0
$\wedge^{k}_{\mathsf{Dup}}$	k RBC	$2k$	$2k+2$	2	$2k-2$
OApp (†)	$2k$ RBC	$2k$ (§)	2	$2k+2$	0
$\mathsf{Com}(x)^{(k)} -_{\mathsf{Dup}} y$ $(k \geq 2)$	$6k-8$ RBC $+1$ CS	$2k$	$4k$	$2k$	0

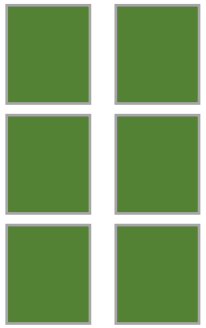
完全シャッフル(廃棄カードを再利用する場合)

入力を残すXOR $\oplus_{DUP}$

- 入力: $Com(b_1), Com(b_2)$
- 出力: $Com(b_1), Com(b_2), Com(b_1 \oplus b_2)$

- 素朴な方法:COPYを2回とXORを1回
  - カード計10枚、RBCを3回
- 次頁の方法:カード計8枚、RBCを2回

入力を残すXOR $\oplus_{DUP}$

1. 複製 $Com(b_1) \rightarrow Com(b_1), Com(b'_1)$
2.  $\leftarrow Com(b_2)$
   $\leftarrow Com(b'_1)$
   $\leftarrow Com(0)$
3. 列をRBC → 一番上の行をめくる
4. 左側が❤️なら列を入れ替える
5. 2行目が $Com(b_1 \oplus b_2)$ 、3行目が $Com(b_2)$

秘匿交換 OExc

- 入力: $Com(b)$、カード束 α_0, α_1
- 出力: $b = 1$ なら α_0, α_1 を入れ替える
- Mizuki-Sone ANDの応用で可能

- 「入力を残す」版: $Com(b)$ の複製と上記を同時に行えばよい
  - どちらも「ビット b の値による場合分け」なので

秘匿選択 OSel

- 入力: $Com(b)$、カード束 α_0, α_1
- 出力: α_b
- 秘匿交換 OExc と同様に実現可能
- 「入力を残す」版も先ほどと同様

秘匿代入 OSub

- 入力: $Com(b)$、カード束 α_0, α_1
- 出力: $b = 1$ なら α_0 を α_1 と交換
- 秘匿交換 OExc と同様に実現可能
- 「入力を残す」版も先ほどと同様

入力を残す多入力AND $\wedge_{DUP}^{k}$

- 2入力($k = 2$)の場合：入力を残す秘匿選択で実現可能
  - [NHM+15]の手法よりもカード枚数が多いがシャッフル回数は少ない
- 一般の入力数：再帰的に実行

[NHM+15] T. Nishida, Y. Hayashi, T. Mizuki, H. Sone: TAMC 2015, pp.110-121 (2015)

適用先を秘匿した操作 OApp

- 入力: $Com(x)$(kビット)、カード束 $\alpha_0, \ldots, \alpha_{2^k-1}$
- 出力: α_x に何らかの操作を施す
- 既存手法のCard Choosing Protocol [DLM+19] や Chosen Pile Protocol [RML+22] と同様

[DLM+19] J.-G. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, T. Sasaki, H. Sone: COCOON 2019, pp.166-177 (2019)

公開値による減算 $Com(x) -_{DUP} y$

- 入力: $Com(x)$(kビット)、公開の整数 y
- 出力: $Com(x), Com(x-y)$
- 具体的な手順は予稿を参照
- 解放カードと廃棄カードの枚数が y によって変わる

- 使う場面:OAppの入力 $Com(x)$ の値を0起点に調整する

提案プロトコルのシャッフル回数

- $N \times M$ の格子状のグラフの場合、RBCの回数は最悪時で
$$(3M^2 + 9M)(N+1)\log_2(N+1)$$
$$+(3M^2 + 13M)N\log_2 M + 15M^2N + 12MN$$
$$\sim 3M^2N(\log_2 M + \log_2 N)$$
  - 出力の事後処理は含まない（頂点数の線型オーダー）
- $M = N$ のとき、$\sim 6M^3 \log_2 M$
- [SS23] の手法のシャッフル回数：$M^3(11M - 8) \sim 11M^4$
  - シャッフルの種類は同じくRBC

まとめ

- **グラフの連結成分の個数**を得る非対話カードプロトコル
  - 「連結成分が1個かどうか」の計算にも転用可能
- (秘密計算でない)アルゴリズム構成 → カードプロトコルへ変換
- [SS23]の手法よりもシャッフル回数のオーダーが改善
  - $M \times M$ 格子グラフ: $O(M^4) \rightarrow O(M^3 \log M)$

- 今後の課題:効率の改善、具体的なパズルへの適用