

# 四色問題 (V1.0)

## － 着色アルゴリズム検証プログラム －

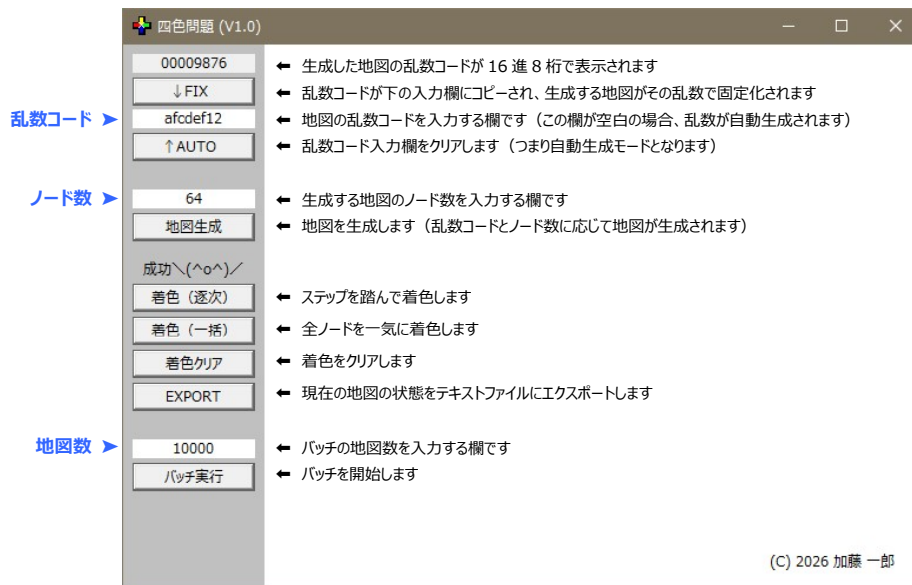
加藤一郎  
kato@my.zaq.jp  
<http://kato.chobi.net>

本プログラムは、四色問題におけるオリジナルの着色アルゴリズムを具現化したものです。  
詳細な処理内容や用語の意味については、作者ホームページに掲載予定の論文を参照してください。

### ■ インストールとアンインストール

インストールは、zip ファイルを任意のフォルダに展開するだけです。  
アンインストールは、インストールしたフォルダにあるファイルをすべて消去すれば完了です。レジストリへの書き込みは一切していません。  
パッチ実行やエクスポートを行うと、Log フォルダや Export フォルダが生成されますが、すべてインストールしたフォルダ配下です。

### ■ 画面構成



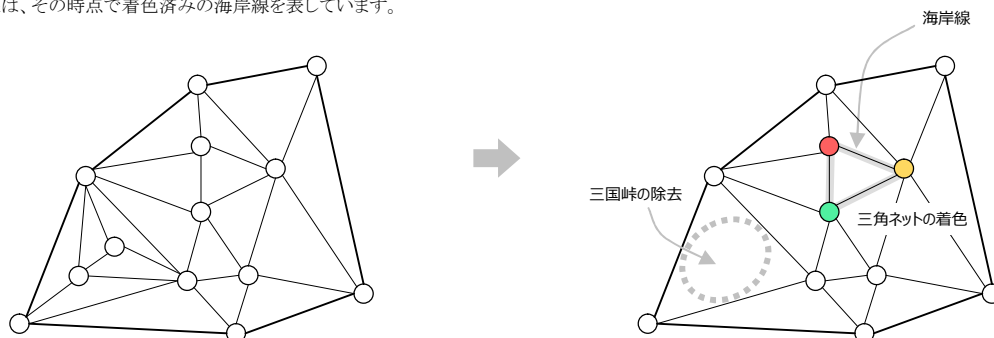
### ■ 基本的な操作

- ① **ノード数入力欄** に地図の規模(ノード数)を記入しておきます。
- ② [ **地図生成** ] ボタンを押下します。
- ③ ステップを踏んで地図を着色したい場合は [ **着色(逐次)** ] ボタンを押下します。
- ④ 一気に着色したい場合は [ **着色(一括)** ] ボタンを押下します。
- ⑤ 着色を一旦クリアしたい場合は [ **着色クリア** ] ボタンを押下します。

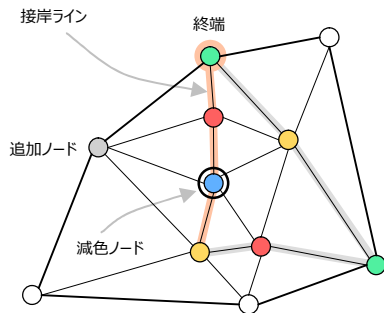
(注) 特定の地図を固定化して着色を繰り返したい場合は、その地図の乱数コードを **コード入力欄** に記入しておきます。  
コード入力欄が空白の場合は [ **地図生成** ] を行う毎に新しい乱数コードが自動生成され、それに基づいて地図が生成されます。

### ■ 描画内容

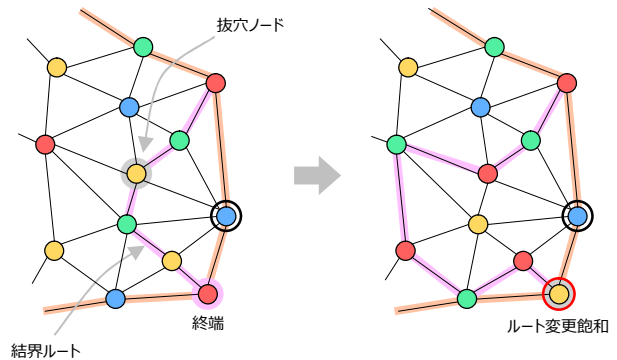
- 着色開始直後に、地図中央付近の三角ネット着色と、すべての三国峠の除去が起こります。
- 階層化している三国峠は、再帰的に除去されます。
- 灰色の太線は、その時点で着色済みの海岸線を表しています。



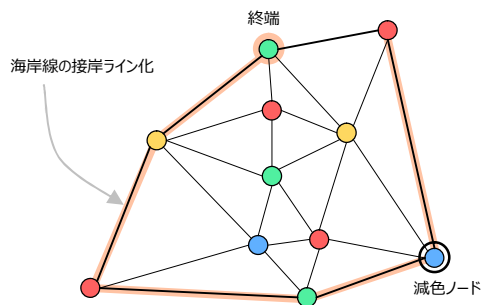
- 接岸ラインが発生するとオレンジ色の太線が表示されます
- 接岸ラインの終端にはオレンジ色の枠円が追加されます
- 減色ノードは黒い枠円が追加されます  
(減色ノードは基本的に青色です)
- 着色対象である追加ノードは灰色になります



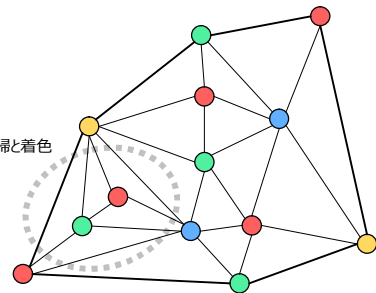
- 結界ルートが発生するとピンク色の太線が表示されます
- 結界ルートの終端にはピンク色の枠円が追加されます
- 抜穴ノードは灰色の枠円が追加されます  
(抜穴ノードは基本的に黄色です)
- ルート変更が飽和状態に達すると、抜穴ノードに赤色の枠円が追加されます



- 全ノード着色直後、海岸線を3色に減色する処理に入ります。(海岸線全体を接岸ラインとみなして処理されます)
- 着色が完了すると、三国峠の復帰とその着色が実施されます。



三国峠の復帰と着色



- 最後に以下のチェックを行い、着色の成否結果を表示します。
  - ✓ すべてのノードを着色できているか
  - ✓ 接続されているノード間の色を塗り分けできているか
  - ✓ 海岸線から青色を除去できているか

成功 \ (^o^ ) /

着色 (逐次)

着色 (一括)

(注) アルゴリズムに欠陥があり、着色が無限ループに陥ってしまう場合に備え、着色開始から約 10 秒でタイムアウトするようになっています。その際、結果表示は失敗となります。なお、逐次着色ではタイムアウトしません。

## ■ ショートカットキー

特定の操作を素早く行うため、いくつかのショートカットキーを提供しています。

キー押下はキーリピートが効くため、[ 着色(逐次) ] に利用すると、次々と着色が進む過程を確認できます。

また、乱数を自動生成状態にして [ 地図作成 ] に利用すると、様々な地図が生成される様子を確認することができます。

キー	等価な機能ボタン
[ A ]	AUTO
[ F ]	FIX
[ ↑ ]	地図生成
[ → ]	着色(逐次)
[ ↓ ]	着色(一括)
[ ← ]	着色クリア

以下のキーで、入力欄(乱数コードやノード数など)へのフォーカス制御ができます。

キー	機能
[ TAB ]	入力欄にフォーカスが移動します。 フォーカスが入力欄にある状態で押下すると、次の入力欄に移動します。 [ SHIFT ] の併用で逆向きに移動します。
[ ENTER ]	入力欄のフォーカスを解除します。

## ■ バッチ処理

自動生成される複数の地図をバッチ的に着色し、その結果をログに出力する機能です。

### <準備>

「 **地図のノード数** 」と「 **生成する地図の数** 」をそれぞれの入力欄に記入しておきます。

次に、必要に応じて「 **乱数コード** 」の欄を記入します。

乱数入力欄に乱数コードが記入されているか否かで初期値の乱数コードを制御できます。

- 空白の場合 : 初期値自体を乱数で決定します。
- 記入がある場合 : そのコードを初期値として採用します。

地図は、初期値として与えられる乱数コードを1つずつインクリメントして連続生成していきます。

### <実行>

- ① [ **バッチ実行** ] ボタンを押下します。
- ② バッチ処理が始まるとバッチモードに移行し、進捗に応じて成否のカウンタが表示されます。
- ③ バッチ処理が終了すると、総処理時間が表示されます。また、地図生成と着色に要した各処理時間がミリ秒単位で表示されます。

### <ログ>

以下の名称で、テキストファイルのログが生成されます。

フォルダ名	ファイル名
Log_YYYY	(nn)mmmmmm_@MMDDHHMMSS.txt
西暦	ノード数 地図数 月日時分秒のタイムスタンプ

ログの出力例を以下に示します。

乱数コード(ノード数)の形式で出力されます

着色に失敗した地図は、末尾に Error 表記が与えられ、エラー検出ノードの ID が(n)の形式で添えられます

着色に失敗した地図

バッチ実行の乱数範囲

バッチ結果の諸元

```
5a5f3d58 (32) ;Error (5)
5a63b69b (32) ;Error (15)
5a669203 (32) ;Error (31)
5a6875a7 (32) ;Error (11)
5a6d0384 (32) ;最終地図
5a5dc145 (32) ;初期地図

-----

ノード数 : 64
地図数 : 10000

-----

総処理時間 : 00 分 07 秒
地図 : 3090 (msec)
着色 : 775 (msec)
```

## ■ エクスポート書式

[EXPORT]ボタンを押下すると、現在の地図状態が「Export」フォルダにテキストファイルでエクスポートされます。

ファイル名は **xxxxxxxx(nnn).txt** の形式です。(乱数コードとノード数の組合せ)

エクスポートの出力例を以下に示します。

```
[G] 1 (71ff, 797f) 15, 17, 42, 43, 51, 65, 86
[B] 2 (051e, 6982) 50, 59, 87, 93, 95
[Y] 3 (2486, 315b) 16, 24, 28, 55, 90
[ ] 4 (2988, 6196) 50, 58, 74, 83, 84, 92, 97
[G] 5 (4a8c, 630a) 12, 13, 25, 27, 54, 66, 79
[ ] 6 (025c, 0877) 34, 40, 45, 93, 94
[*] 7 (634a, 0096) 29, 32, 33, 34, 57, 81, 96
[B] 8 (23a3, 2038) 11, 23, 63, 71, 90
[R] 9 (7334, 354a) 12, 19, 21, 30, 73, 76, 98
[B] 10 (69c4, 52d6) 12, 25, 27, 47, 61, 98
[ ] 11 (2ccd, 1b55) 8, 38, 55, 71, 88, 90
[Y] 12 (56ec, 49f6) 5, 9, 10, 19
```

接続先のノード ID

ノードの XY 座標 (16 進 4 桁)

ノード ID

着色状態 (R:赤 / G:緑 / Y:黄 / B:青 / \*:着色中 / 空白:未着色)