四色問題の証明
Formal proof of the Four color theorem

加藤　一郎
Ichiro KATO
e-mail : ***kato@my.zaq.jp***

要約

四色問題とは、「平面上の如何なる地図も、隣接する領域を塗り分けるには四色あれば十分である」ということを示す問題である。この問題は、「放電法」と呼ばれる手法を発展的に利用し、人手による実行が事実上不可能なほど複雑なプログラムによって 1976 年に証明された。その後も断続的にアルゴリズムやプログラムの改良が行われ、現在では四色問題は解決しているとみなされ、「四色定理」と呼ばれるようになった。しかし、その証明は計算機による済し崩しの演算による解決には違いなく、現在でもコンピュータを利用せずに済ませられる証明は得られていない。本稿は論理的な着色プロセスを示すことで、あらゆる地図が四色で塗分け可能なことを、計算機を用いずに証明するものである。

キーワード： 四色定理、グラフ理論、色スワップ、排他的連鎖ルート、逐次追加プロセス、海岸線3色原則
Keywords **:** Four Color Theorem, Graph Theory, Color Swap, Exclusive Chain Route, Incremental Addition Process, Coastline three-color principle

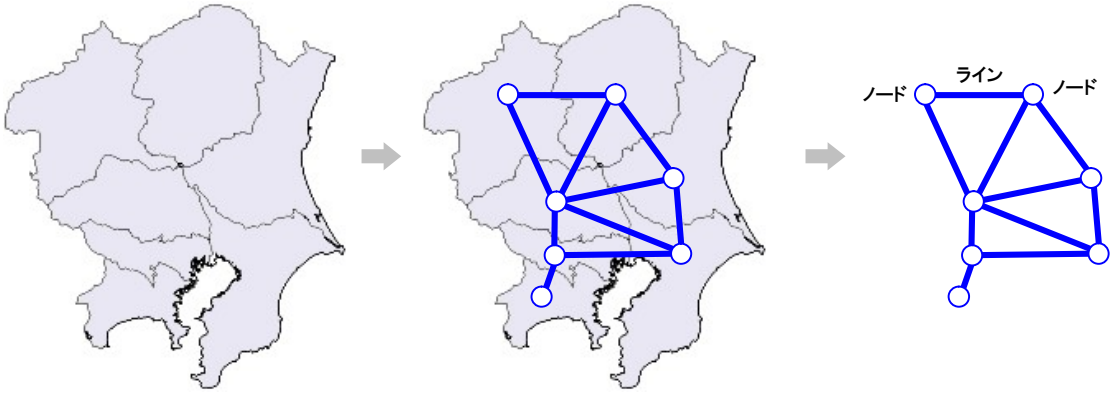
1. はじめに

本稿は、以下のアイディアを結び付けることで、あらゆる地図が四色で塗分けられることを証明する。

- 地図をグラフ化し、そこに補助線を加えた三角ネットのみの還元可能な構成
- 海岸線を3色で維持しながら、ノードを逐次追加していくという手続きを積上げて全体を着色するプロセス
- スワップ処理における排他的連鎖ルートの特性を利用し、接岸ラインから追加ノード色を除去する手法

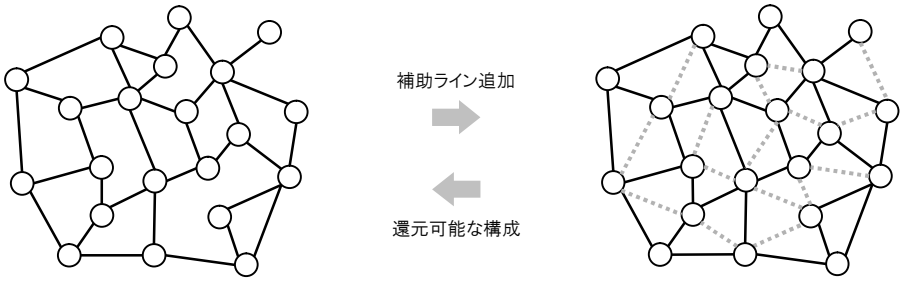
2. 平面グラフ化

極めて一般的な手法ではあるが、まずは地図の平面グラフ化を実施する。以下の例のように、地区ごとにノードを与え、地区間の境をラインで接続する。当然のことながら、これらのラインは交差することは無い。このノードのそれぞれに色を割り当てていくことになるが、ライン接続されたノード同士は異なる色を与えなければならない。この条件下において、どのような地図であろうとも4色で塗り分けられることを主張するのが、四色問題である。

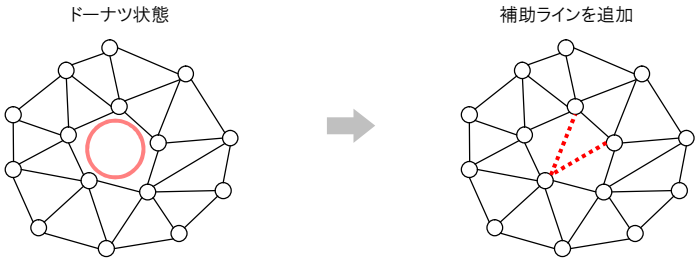


3. 補助ライン

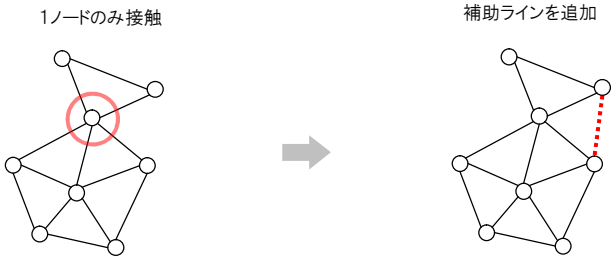
これも一般的な手法ではあるが、以降の分析のために一つの工夫を施す。得られたグラフの各ノードに補助ラインを追加し、すべて三角のネット網となるようにする。この処理はノードの隣接状態を複雑にし、4色の塗分け条件がオリジナルより厳しくなる。しかし、塗分けパターンは逆に単純化されるため、トポロジー分析において有利に働く。三角ネット網の状態でグラフ全体が4色に塗り分けられたら、追加した補助ラインを除去することで、元の地図のグラフが復元する。つまり、三角ネット網で塗分け可能ならば、元の地図も塗り分けられることが保証される。



すべてを三角ネット網にすることで、空洞を伴うマップにおいては、それを埋めるように補助ラインを追加することになり、結果としてどの様なマップにおいても穴が1つも無いトポロジー形状となる。

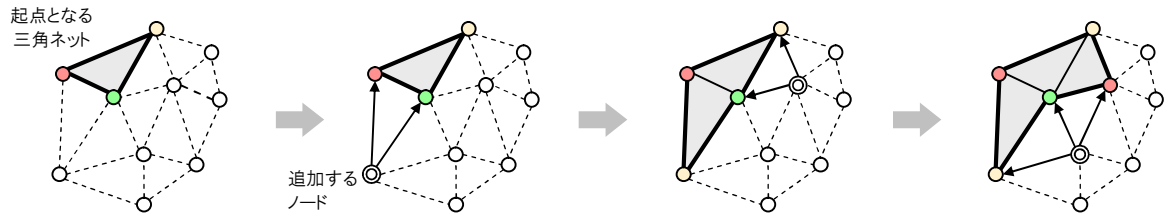


なお、1ノードのみ接触している三角ネットが存在する場合は必ず補助ラインを追加し、2 ノード以上の接触状態にしておく。これにより、着色プロセスの処理パターンがより簡素化する。



4. 着色プロセスの概略

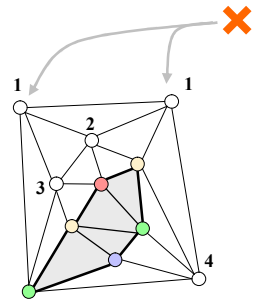
本稿が提案する着色プロセスは、まず起点となる任意の三角ネットの着色を行う。これには3色が必要となる。次に着色済みのノードに直接接続されている近傍のノードを追加し着色する。この際、色干渉しないように着色するには着色済みの色に変更を与えなければならない場合がある。この色変更処理の詳細に関しては後述するが、いずれにしてもその処理を施しながら順次着色ノードを追加していくことで、最終的にグラフ全体を着色する。



なお、ノード追加の際にいくつかの禁則事項が存在する。まず、追加する未着色ノードは 2 本以上のラインで着色ノードと接続されていなければならない。また、追加するノードから延びる接続ラインの先の着色ノードは連続して並んでいなければならず、別の未着色ノードが混入してはいけない。更に、着色済みの三角ネットに隣接してさえいれば着色順序は任意であるが、あくまでも稠密に実施し、空洞を伴う着色状態(ドーナツ状態)に陥らないように留意しなければならない。

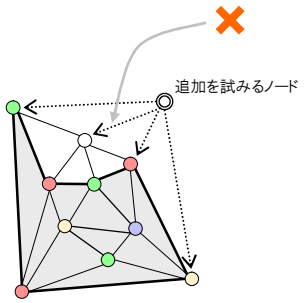
禁則1

着色ノードに接続しているラインが 1 本しかない未着色ノードはまだ追加できない。



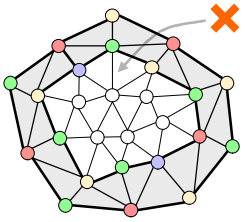
禁則2

接続先の着色ノードが連続して並んでいないノードはまだ追加できない。

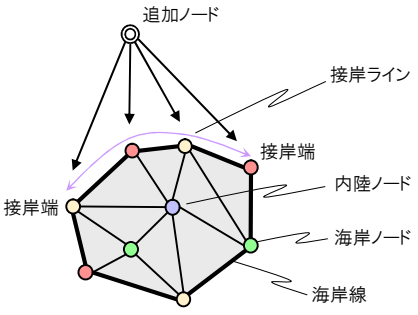


禁則3

着色プロセス途中の一時的な状態であっても、空洞を伴う着色状態にしてはならない。



ここで、以降の着色プロセスの詳細な検討を円滑に進めるために、以下のように用語の定義をする。

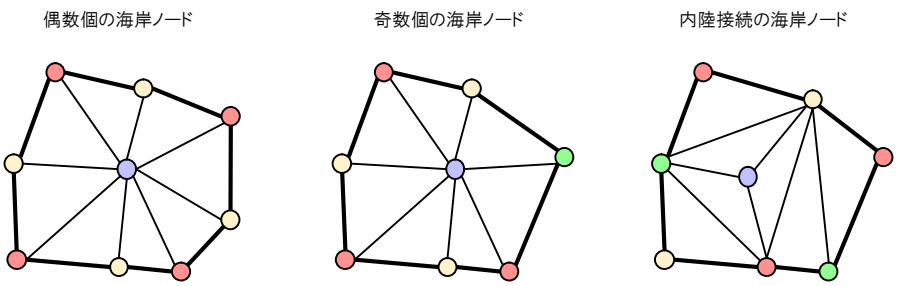


海岸線	着色済みの領域において、その輪郭を「**海岸線**」と呼び、そこに存在するノードを「**海岸ノード**」と呼ぶ。
内陸	海岸線の内部に位置する領域を「**内陸**」と呼び、そこに位置するノードを「**内陸ノード**」と呼ぶ。
追加ノード	着色プロセスにおいて、着色済みの領域に対して新たに追加するノードを「**追加ノード**」と呼ぶ。
接岸ライン	追加ノードを接続する海岸線上のノードを特に「**接岸ノード**」と呼ぶ。一般に接岸ノードは複数存在し、それらが連なる領域を「**接岸ライン**」と呼ぶ。また、接岸ラインの両端点を「**接岸端**」と呼ぶ。

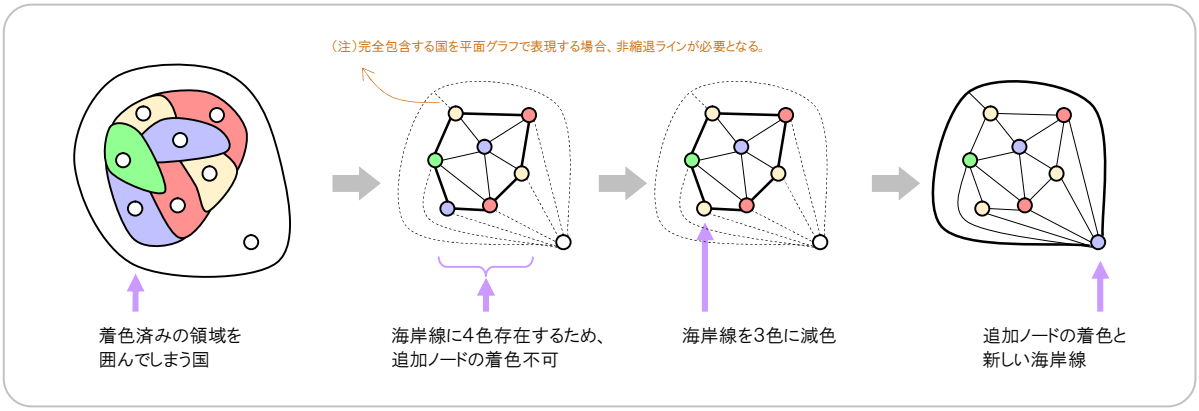
※ 追加ノードが接続された時点で、追加ノードは新たな海岸ノードとなり、接岸端を除く接岸ノードは海岸ノードから内陸ノードへと変更される。

5. 海岸線3色の原則

今、内陸ノードが1つのみ存在するグラフを考える。この時、海岸ノードの偶奇性によって、海岸ノードの着色に必要な色数が決まる。偶数個であれば2色で、奇数個であれば3色で塗分けが可能なことは自明である。奇数個の場合、1ノードのみ異なる色とし、他のノードは別の2色の対で交互に着色することで塗分けられる。しかし、これは一般的には成立しない。例えば、離れた海岸ノード同士が内陸で接続されている場合は、以下の例にみられるように、3色それぞれを複数使用しなければ塗分けすることができない。



しかしいずれにせよ、海岸線を3色以内で塗分ける状態を維持することには意義がある。もし海岸線に4色を使用した状態で、それらすべての海岸線に接続されているノードを追加する場合（つまり着色済みの領域をすべて囲んでしまうような国が存在する場合）、追加ノードには5色目が必要となってしまう。これを回避するには、着色済みの海岸線を3色に減色する必要がある。すなわち、「海岸線3色の原則」は四色問題をクリアするための必要条件であることがわかる。



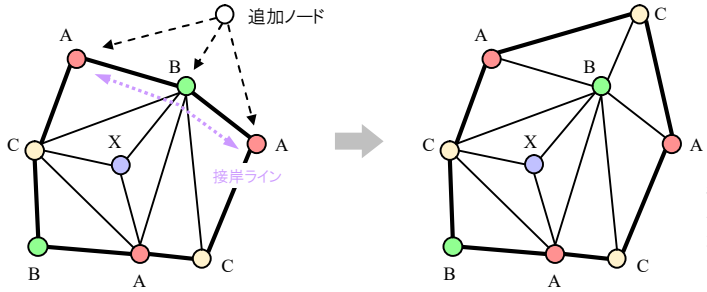
このことから、3色以内で塗分けられている海岸線に追加ノードを接続しても、新たな海岸線も3色以内で塗分けられている状態が維持できれば、その処理を逐次累積的に実施していくことで、最終的に地図全体の塗分けが可能であることがわかる。すなわち、「ノード追加時の海岸線3色維持」は、四色問題の十分条件であり、これにより四色問題を証明したことになる。

一般的な四色問題の証明アプローチは、新しい国を追加する際「4色」を維持できるか否かを議論している場合が多い。その場合は、5色になってしまう状況を4色に減色するという形で検討を進める。しかし本稿では、海岸線を「3色」のまま維持するという考え方を基本とする。この利点は2つあり、一つは検討対象の色数が少ないことで色の組み換えパターンが減り、検討が容易になるという点である。もう一つは、新しい国を追加する時点の手法さえ確立できれば、後は同じ手法を繰り返すことで幾らでも新しい国を追加できるという証明方法が使えるという点である。

6. ノード追加時の着色方針

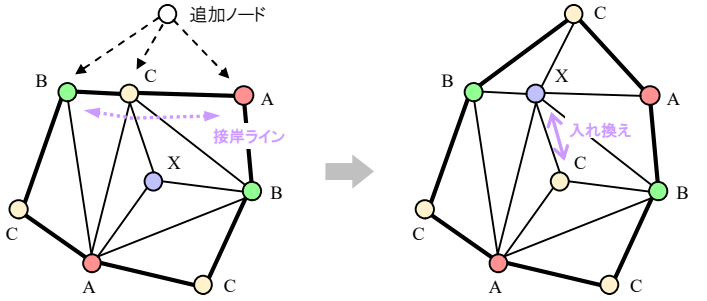
ノード追加時においても海岸線3色の原則を維持するための着色方針を示す。ここからは海岸線に使用できる3色を A,B,C とし、内陸でのみ使用できる1色をXと表現して検討を進める。(Xは湖の色と考えるとイメージしやすい。湖は内陸にのみ存在する)

まず、ノード追加時の接岸ノードが ABC のいずれか 2 色で構成されている場合、追加ノードは ABC の内の残りの1色を使用できるため、着色済みのノード色を変更することなく、新たな海岸線を3色(ABC)に維持できる。



接岸ノードがAとBの2色のみで構成されているため、残りの C を使って追加ノードを着色。結果的に新たな海岸線は ABC の3色が維持される。

一方、接岸ノードが3色(ABC)で構成されている場合、残された色はXのみとなるが、追加ノードは新たな海岸ノードとなるためXを使用できない。そこで、接岸ノードから ABC の内の任意の 1 色を取り除き、2 色に減色した後、取り除いた 1 色を追加ノードに割り当てることになる。ただし、この減色処理を実施するにあたり、「接岸端を除く接岸ノードには X が使用できる」という手段が与えられる。何故ならば、接岸端を除く接岸ノードは、ノード追加後に内陸ノードに変化するためである。この X の使用許可は、厳しい制約条件が課せられた減色処理の中で、極めて有効な手段となる。



この例では、C と X を入れ換えることで、追加ノードに C を割り当てることが可能となった。そして、新たな海岸線も ABC の3色が維持されている。

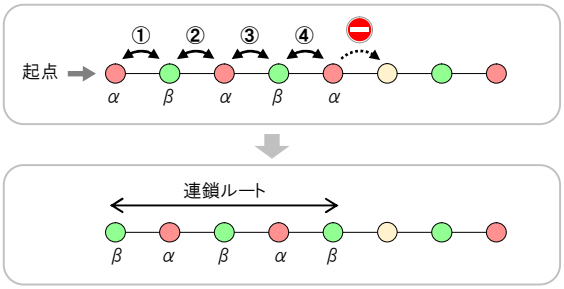
ノードを追加する実際の局面は、その接岸ノードの配列も様々であり、内陸や海岸線にある着色済みのノードとの関係も存在しているため、その場その場の状況を分析し、ヒューリスティックに減色処理を行うことは難しい。そもそも、規則性を伴わない着色プロセスでは、四色問題を証明したことにならない。本稿では、規則的かつ汎用的に減色を実現できる一意な手順を与えることを目指す。

いずれにせよ、色干渉していない着色済みのノードに対し、その状態を維持したまま接岸ラインの減色処理をシステマチックに実現するには、スワップ処理を利用する以外にない。しかるに、確実に減色を実現できるスワップの手続きを示すことができれば、四色問題を証明したことになる。

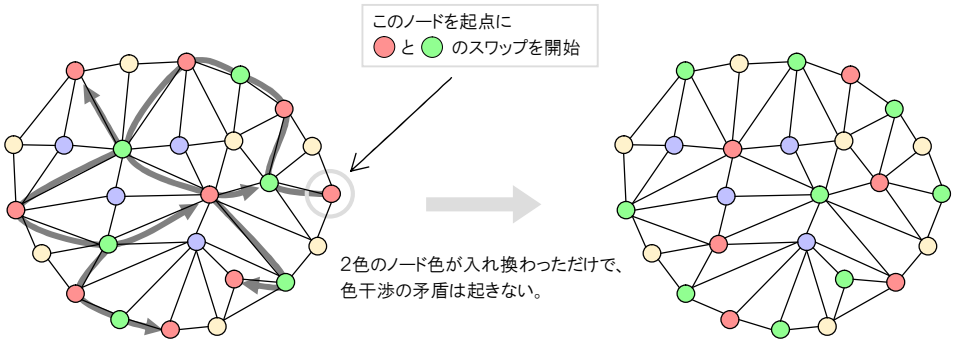
7. スワップ

スワップは、起点となるノードの色（α）を他の色（β）に変更することから始める。その際、起点ノードに（β）のノードが接続されていた場合（β）の干渉が起こる。そこで、干渉するノード（β）を（α）に色変更することで干渉を解決する。しかし、更にその先に（α）のノードがあった場合、そこで再び干渉が起こるため、そこでも色を変更する必要がある。このようにスワップは、起点ノードから（α）と（β）が交互に存在しているルート上の色を連鎖的に入れ換える処理となる。このルートを「連鎖ルート」と呼ぶ。また、（α）を起点に（β）とのスワップを行った際の連鎖ルートを「α β ルート」と呼称する。

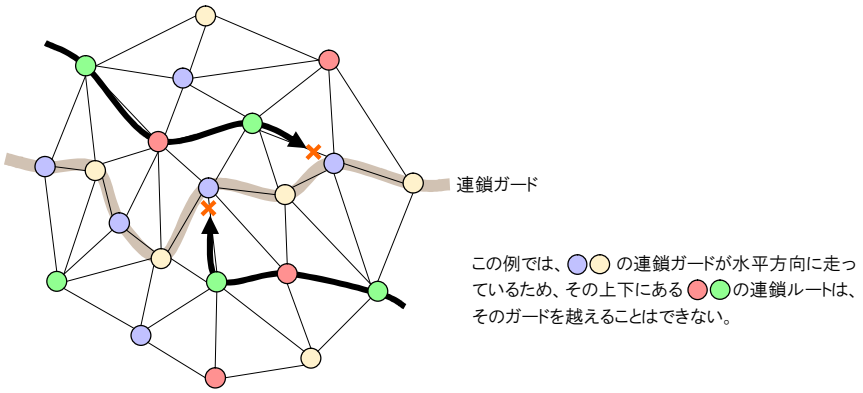
スワップは、接続されているノードのすべてが（α）（β）以外の色となる箇所に到達した時点で停止する。



スワップは、起点からの色の入れ換えが、分岐やループを伴いながら広範囲に及ぶ可能性がある。しかし、スワップ前のグラフが色干渉していなければ、たとえ分岐やループを発生したとしても、スワップ後も色干渉の破綻は起きない。この概念は、19世紀のアマチュア数学者アルフレッド・ブレ・ケンプが「ケンプ鎖」として提唱済みである。



また、ケンプ鎖においては、もう一つ重要な特性がある。それは、排他的な色の組合せで構成される2つの連鎖ルートは交差しないという特性である。一方の連鎖ルートが他方の連鎖ルートの連鎖範囲を限定するという意味で、ここでは特に「連鎖ガード」と呼ぶ。また、α β ルートによる連鎖ガードを「α β ガード」と呼称する。



連鎖ガードは接岸ラインの減色処理、特に X ノードの生成局面で活用する。

8. 接岸ラインの減色手順

海岸線を3色に維持したまま新しいノードを追加接続するための接岸ライン減色手順を示す。ここでは海岸色 ABC の内、減色対象を C として記述する。また、残存させる AB は処理上対称性を持っており、この AB を互いの「対色」と表記する。C の減色処理は片側の接岸端からもう片方の接岸端に向けて行う。以降の例示図では、接岸ラインに沿って右から左に向けて処理する方向で描いている。減色処理の大まかな流れは、接岸ラインの後方（右側）にある C を次々と前方（左側）にシフトさせ、最終的にすべての C を除去することである。接岸ラインから C が除去されれば、追加ノードに C を割り当てることができ、新しい海岸線も ABC3色での構成を保証できる。（なお、これは必須の処理ではないが、ABC の色割付はノード追加ごとに変更が可能なため、接岸ライン上の最後尾ノードがより前方にあるノード色を C に割り付けると処理効率が高まる。）

基本的に、対象とする C ノードを起点に CA スワップまたは CB スワップを実施する。スワップ成功とは、接岸ライン上からの起点 C の除去、あるいは後方へのシフトを指し、それを実現する状態は以下の場合がある。なお、これらの複合状態も成功となる。

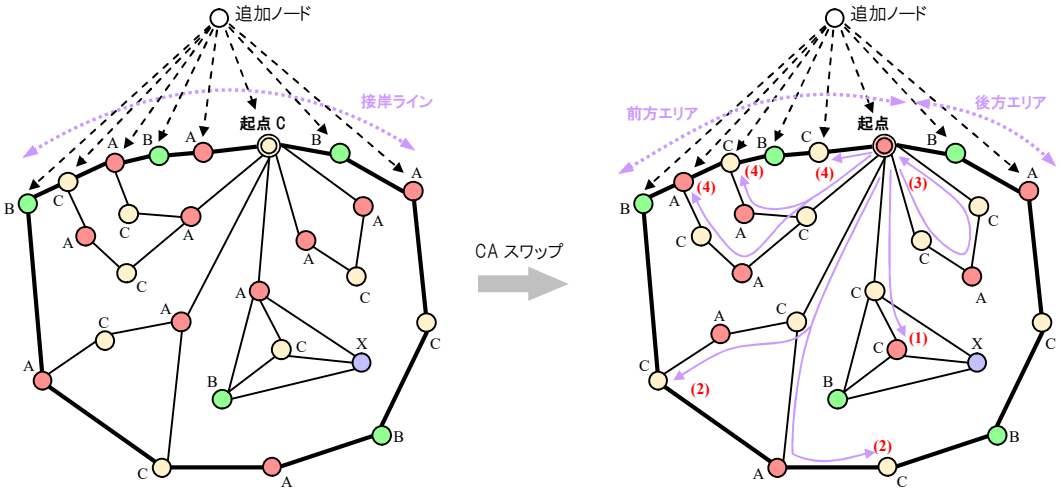
（1）内陸のノードで連鎖が停止した場合
（2）接岸ライン以外の海岸線に到達した場合
（3）起点の C ノード自身に戻った場合
（4）起点の C ノードよりも前方の接岸ノードに到達した場合

ここで、起点 C を含んだ前方の接岸ラインを「前方エリア」、後方を「後方エリア」と呼ぶことにすると、上記（1）～（4）の内容は、「スワップ成功条件」として以下のようにまとめて表現することができる。

C を起点としたすべての連鎖ルートが、後方エリアに1つも到達しない場合、スワップ成功とみなす

CA スワップの成功例

（1）（2）（3）の場合は、接岸ライン上から起点の C ノードが除去される。
（4）の場合は起点の C ノードが前方にシフトする。
なお、（4）における到達点が複数存在する場合、一時的に接岸ライン上の C が増加することになるが、最終的には除去されるため問題はない。
一方、到達点が C の場合は起点と到達点の両者の C が一気に除去される。



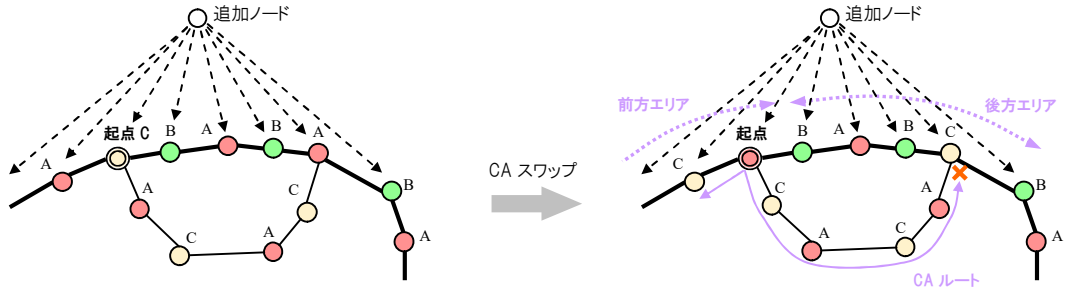
（注）図が煩雑になるのを避けるため、本件に直接関与しないラインやノードは描画を省略している。

一方、連鎖ルートが後方エリアに1つでも到達した場合はスワップ失敗と見なされ、そのスワップは実施せずに、追加の手続きを行う必要がある。この追加の手続きは、AX あるいは BX スワップであり、これによりノード X が発生する。
以下、詳細な減色手順の全工程を順を追って記載する。

（手順1）

接岸ライン上の最も後方にあるCを起点として選び、起点Cに隣接する後方ノードの対色でスワップを試みる。
例えば、後方ノードがBである場合は、CAスワップを試みる。
スワップが成功し、接岸ライン上からCが一掃されたら**処理は完了**する。Cが残存していたら、**（手順1）**を繰り返す。
なお、Cが後方の接岸端に位置している場合は、後方エリアが存在しないため、スワップは必ず成功する。
スワップが失敗した場合は、スワップを実施せず**（手順2）**に移行する。

（手順1）CA スワップの失敗例

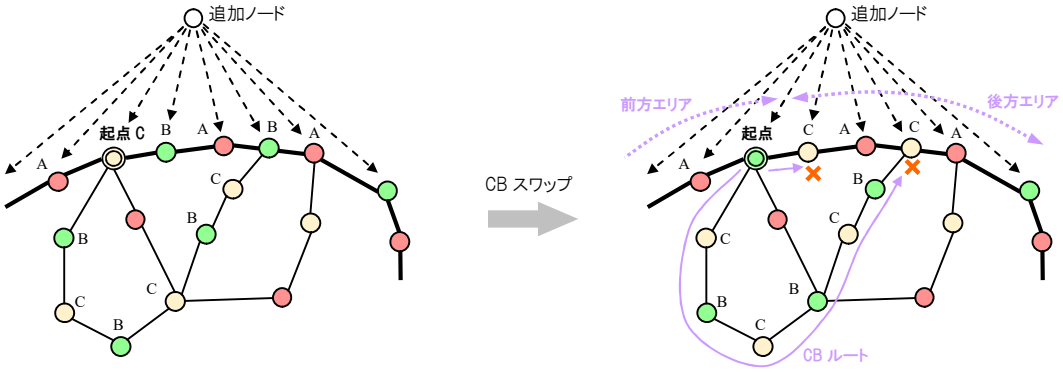


連鎖ルートが接岸後方エリアに到達してしまうため、このスワップは実施できない。

（手順2）

起点Cに隣接する後方ノードの色でスワップを試みる。
例えば、後方ノードがBである場合は、CBスワップを試みる。このスワップは、ノードBがCになるため、必ず失敗する。
それでも敢えて試みる意図は、起点Cから発生するすべてのCBルートの情報を収集するためである。

（手順2）CB スワップの失敗例



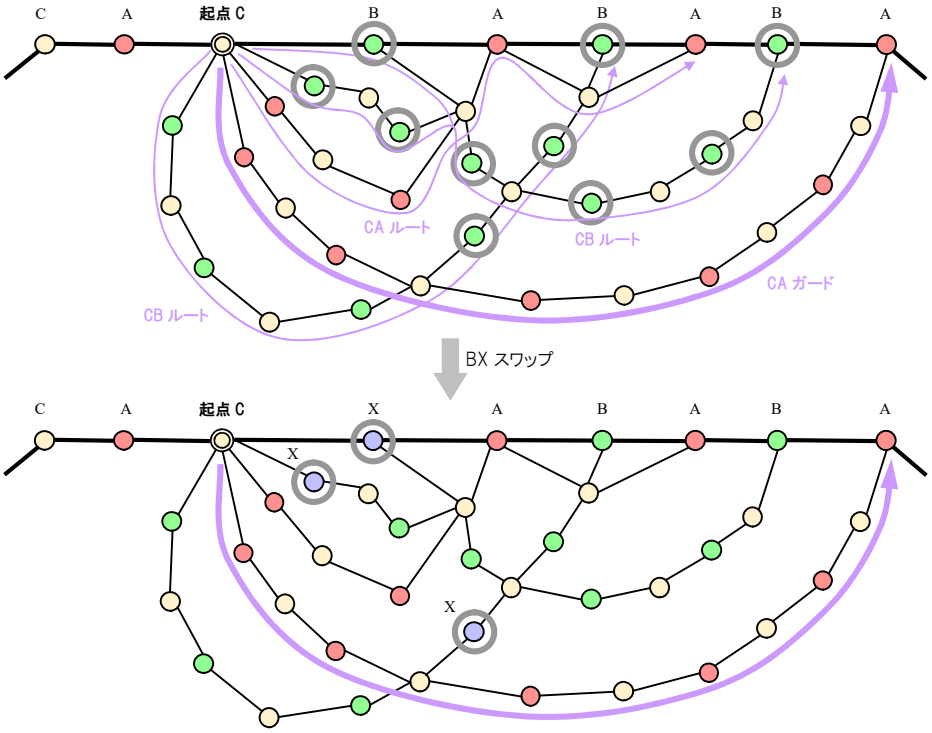
連鎖ルートは隣接する後方のノードに到達してしまうため、スワップは必ず失敗する。

この時点で、起点Cから延びる連鎖ルートにはCBルートとCAルートの2種類が存在している。そして、それらは1ルートずつとは限らず、共に複数存在している可能性がある。また、両者のルート上には C ノードが共通して含まれているため、そのノードで互いに交差している可能性もある。このように複雑に絡み合った連鎖ルートであるが、接岸ラインへの到達点に着目すると、その包含関係が明確になる。最も後方の到達点を持つ連鎖ルートは、それ以外のすべての連鎖ルートの到達点を包含している。どのような経路を辿ろうとも、そしてどのような交差状態であろうとも、最終的な到達点の位置関係で包含関係が決定するのである。よって、最も後方の到達点を持つ連鎖ルートは「連鎖ガード」として機能し、内包する対色のノードにXスワップを施した際、その連鎖でXが海岸線に到達することを阻止する。一方、Xに変色したノードを含む対色の連鎖ルートは、起点Cからのスワップを実行してもXに遮断されて後方エリアに到達できなくなる。

X スワップの起点は、連鎖ガード内に内包される対色ノードの内、起点 C に最も近いものを選択する。例えば、CA ガードが与えられていた時、X スワップはそこに内包されるすべての B ノードで実施する必要はない。接岸ラインに到達している B ノードから CB ルートを遡り、起点 C に最も近い B ノードを起点として BX スワップを実施すればよい。

（手順2）BX スワップの実施例

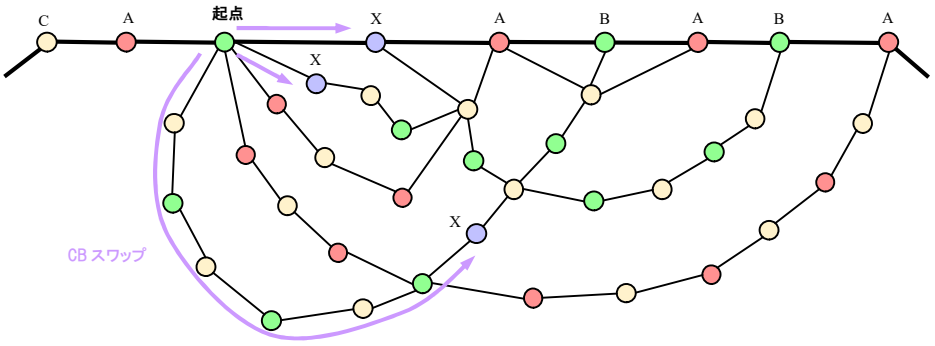
最も後方の到達点を持つ連鎖ルートを連鎖ガードとして採用する。（下記例では、CA ルートがそれに当たる）
CA ガードにより、それに内包される B ノードを起点にした BX スワップは、X が海岸線に到達しないことが保証される。



各 CB ルート上で、CA ガード内に入っている B ノードの内、起点 C に最も近いノードを起点に BX スワップを実行する。
これにより、起点 C から CB ルートを伝わって接岸後方エリアへ到達する経路は遮断される。

対象となるすべてのノードに X スワップを施した後、改めて起点 C からスワップを実施する。例えば、BX スワップを行った場合は、起点 C から CB スワップを実施する。既に連鎖ルートは X ノードで遮断されているため、このスワップは必ず成功する。
接岸ライン上から C が一掃されたら**処理は完了**する。C が残存していたら、**（手順1）**に戻り処理を継続する。

（手順2）CB スワップの実施例



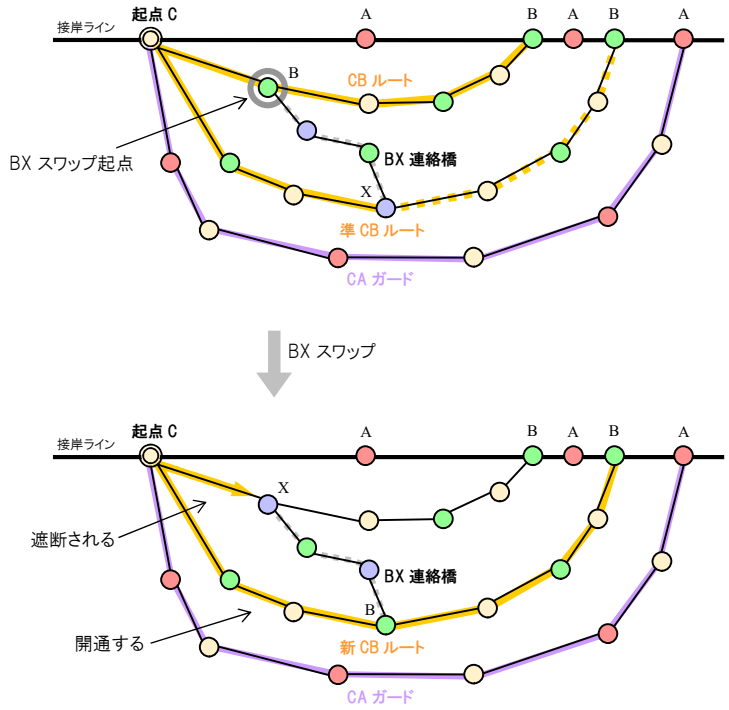
以上述べてきた接岸ラインの減色手順を、処理が完了するまで繰り返すことで、新たなノードを C として追加可能となる。すなわち、海岸線を3色に維持したままノードを次々と追加してマップを拡張することで、あらゆる地図を A,B,C,X の4色で塗分けることができる。ただし、**（手順2）**の X スワップにより新たな連鎖ルートが生成されてしまい、デッドロックが発生する事態も考えられるため、次節にその検証を加えておく。

9. 新たな連鎖ルート生成に関する検証

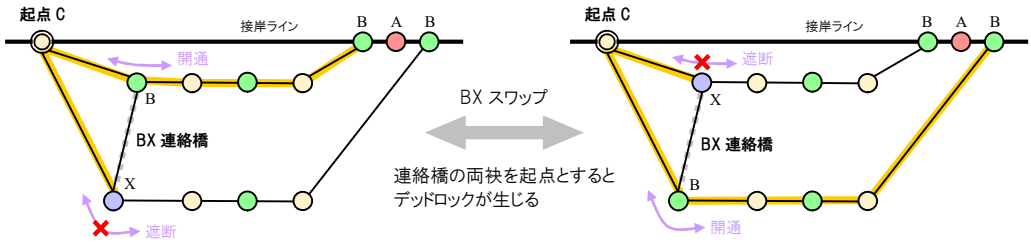
(手順2)で実施した X スワップによって新たな連鎖ルートが生成されてしまい、更にその生成が無限に繰返されるといった可能性に関して検証を行う。提示した着色手順が成立するためには、そのような事態が発生しないことを示しておく必要がある。

まず、「準連鎖ルート」を定義しておく。準連鎖ルートとは、通常の連鎖ルート上の1ノード(A あるいは B)が X ノードになっているルートである。例として、CA ガードに内包される CB ルートと準 CB ノードの場合を取り上げる。今、CB ルート上にある X スワップの起点 B から、準 CB ルート上の X ノードに向けて BX ルートが伸びているとしよう。この BX ルートを「連絡橋」と呼ぶ。この時、X スワップを実行すると、連絡橋の BX 連鎖を介して X ノードが B ノードとなり、準連鎖ルートが連鎖ルートに昇格する。

新たな連鎖ルートが生成される状況

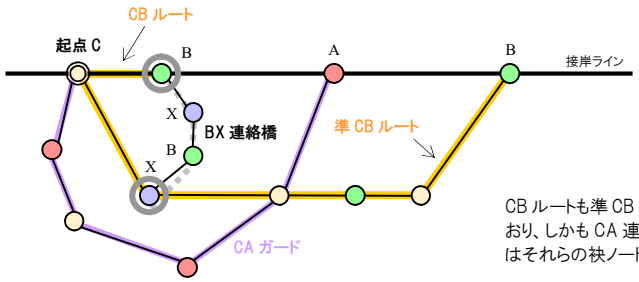


特に問題が生じるのは、X スワップの起点に連絡橋の袂を選択せざるを得なくなる場合である。その場合、より深刻なデッドロックの問題が生じる。デッドロックが生じる典型的な例は、両方の連絡橋の袂が起点 C にダイレクトに接続されている場合である。**(手順2)**の処理手順に従えば、起点 C に最も近いノードは橋の袂であるため、X スワップの起点として両ルートともそれぞれの橋の袂を選択することとなる。このパターンはフリップフロップ回路のように作用し、片方の連鎖ルートを遮断するともう片方の連鎖ルートが開通するという症状を無限に繰返すことになる。



(注) 連絡橋は間に1つもノードが無い状態、すなわち互いの袂が直接接続されている場合でも成立する。

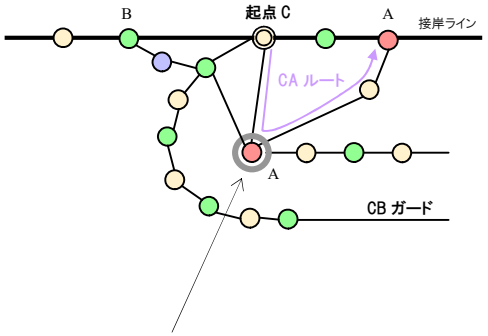
無論これを例外的処理として扱い、少なくとも片方の X スワップにおいて、連絡橋の袂以外のノードを起点にとって(**手順2**)を繰り返せば発症を抑えられる。しかし、それさえも不可能な場合が存在する。それは、片方の袂が接岸ライン上に存在し、もう片方の準連鎖ルートの到達点が連鎖ガードを越えて後方エリアに達している場合である。この場合、連鎖ガード内には袂ノードだけが存在している。よって X スワップ起点はどちらも連絡橋の袂を選択するしかない。すなわちデッドロックを回避できない。



CB ルートも準 CB ルートも、連絡橋の袂が直接起点 C と接続されており、しかも CA 連鎖ガードに内包されている X スワップの起点候補はそれらの袂ノードのみであるため、デッドロックを避けられない。

だが、本稿の着色手順に従うと、このデッドロックするパターン自体が起こり得ないことがわかる。それは、準連鎖ルートの X ノード(すなわち連絡橋の袂)が発生する局面に遡って考察すると明確になる。袂となる X ノードの発生は X スワップによるものであるから、それを包囲する連鎖ガードが存在していたはずである。そして、X ノード発生後の今回の起点 C は、前回の連鎖ガードの外に位置することは自明である。したがって、袂と今回の起点 C との間には、前回の連鎖ガードのラインが通っていることになる。結果として、袂と起点 C はそのライン上のノードを介してのみ接続可能となる。すると、今回の起点 C における連鎖ルートとして前回の連鎖ガードも候補に上ることになり、フリップフロップ回路自体が成立しなくなる。これは準連鎖ルートが今回の連鎖ガードの外側から入ってくる場合でも同様である。つまり、X スワップによる新しい連鎖ルートの生成は起こらない。

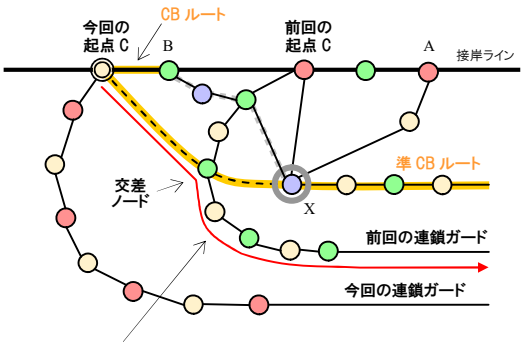
【ステップ1】



CB ガードに内包されている CA ルートを遮断するため、起点 C に最も近い A ノードを起点に AX スワップを実施する。それに伴い、A ノードが X に変色して連絡橋が生成される。

次の起点 C に移行

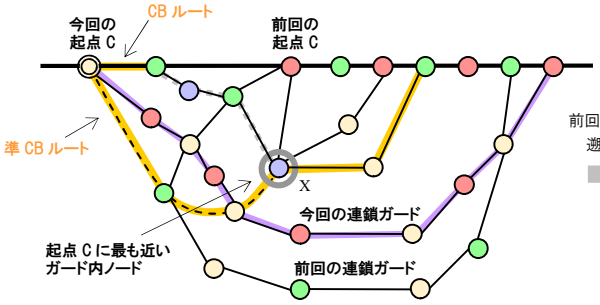
【ステップ2】



今回の起点 C と連絡橋の袂である X ノードを繋ぐラインは、前回の連鎖ガードと必ず交差する。したがって、準連鎖ルートではなく、前回の連鎖ガードを今回の連鎖ルートとして検出することになる。これは交差ノードが B であっても C であっても同様である。

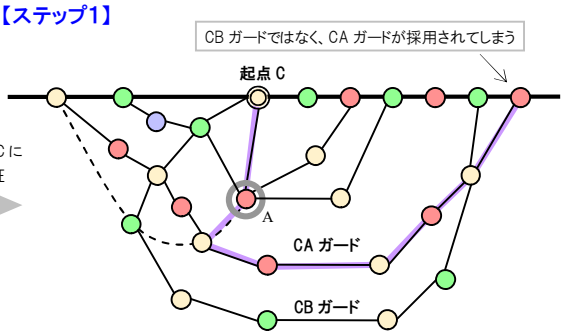
では、今回の連鎖ガードが前回の連鎖ガードの内側を通過している場合はどうであろうか。この場合、袂ノードが X スワップの起点として採用されることになってしまうが、前回の処理に遡って考察すると、連鎖ガードそのものが入れ替わってしまい、結果としてこの場合も新しい連鎖ルートの生成は起こらないことがわかる。

【ステップ2】

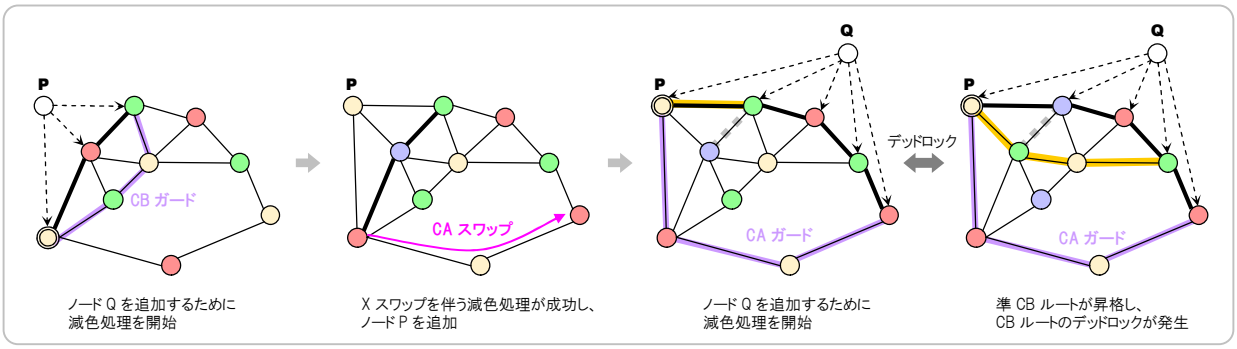


前回の起点 C に遡って検証

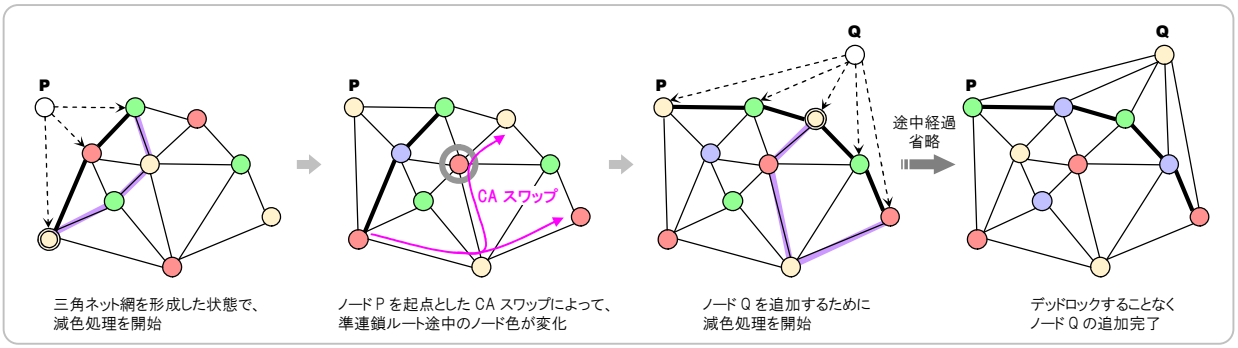
【ステップ1】



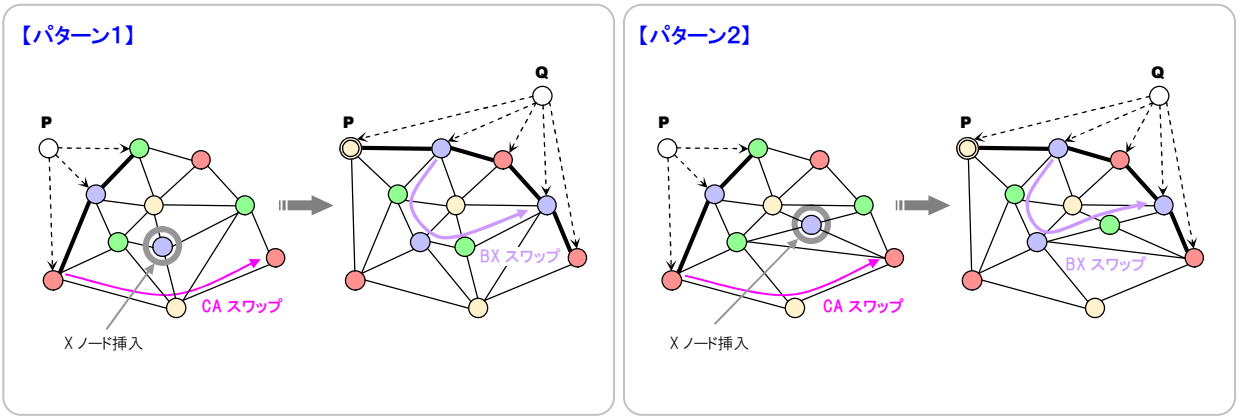
減色処理中の X スワップでは新たな連鎖ルートが発生しないことが確認できた。しかし、X スワップを伴うノードを追加した後、更に隣接するノードを追加する段階でデッドロックが生じる可能性が残っている。以下、その場合の検証を行う。今、1つ目の追加ノードを P とし、2つ目の追加ノードを Q として例を上げる。ノード P の追加時は CB ガードによる X スワップが実施され、続くノード Q の追加時はノード P を起点 C とした CA ガードによる X スワップが実施される。この時、準 CB ルートが形成されてしまい、X スワップによるデッドロックが発生する。



しかし、上記の検証例は重要な要素が欠落している。それは三角ネット網の形成である。実際は、以下の例のようにノード間を接続するラインが存在している。そのため、ノード P の最終処理である CA スワップがそのラインを通じて伝播し、準連鎖ルートのパターンが崩れることになる。結果としてデッドロックはもとより、新しい連鎖ルートの生成自体が起こらない。



ここで、CA ルート上に X ノードが存在し準連鎖ルートが乱されない場合を検証してみよう。ノード P 以前のノード追加によって、そのような X ノードが生成されている可能性は排除できない。しかし色干渉を避けて X ノードを挿入すると、それがどのようなパターンであろうとも、ノード Q の BX スワップはその X ノードを介して接岸ラインに到達する。何故あらゆるパターンでこのことが言えるかというと、CA 連鎖を断ち切るには BX 連鎖を通す以外になく、ここにも「連鎖ガードの原理」が関与しているためである。結果としてデッドロックは生じず、減色処理を進めることが可能となる。つまり、ここでも新しい連鎖ルートは生成しない。



以上の検証により、**(手順2)**における X スワップで新たな連鎖ルートは発生しないことが確認された。一般に、デッドロックが発生すると有限回のスワップ処理では解消することはできず、状況に応じた判断を要する広域的な色変更を施す必要がある。一方、本稿が提案する着色手順は、スワップのみで処理できる着色パターンを持続する機構が備わっていると予想される。

10.まとめ

海岸線3色の原則をベースにすることで、規則的かつ汎用的な着色手順が確立できることを示した。これにより、計算機を用いた済し崩しの演算に頼ることなく、あらゆる地図が四色で塗り分けられることが証明された。本稿が提案する手順は与えられた地図の場合分けを伴わず、コンピューターによる証明のように膨大なケースを準備する必要も無い。また、本稿が連鎖ガードとして活用した「排他的な色の組合せで構成される2つの連鎖ルートは交差しない」という特性は、何故地図の塗分けは3色では足りず、5色では多すぎるのかという根本的な問いへの答えも提供する。この特性は、独立な2色の連鎖ルート2本の関係を示すものであり、2色×2本、すなわち4色が必要十分な色数だからである。

Q. E. D

Appendix ： フローチャート

本稿が提案する海岸線3色の原理に基づいた着色プロセスの全体フローチャートを示しておく。ここでは、海岸線を A,B,C の3色とし、内陸での使用色を X として表現している。また、ノード追加の際の減色対象を C として記載している。