グラフとアルゴリズムとプログラムのやさしいおはなし

渡邉敏正

2021 年 7 月 22 日

第 4 回

1　ケーニヒスベルクの橋の問題を再び考えてみる

第 1 回で紹介した「ケーニヒスベルクの橋の問題」を再び取り上げて、どのようにしてイエス、ノーと判断するかを考えてみましょう。第 1 回の図 2 で示したグラフ G_0 を以下にもう一度図 19 として示しておきます。このグラフ G_0 にオイラーサイクルがあるかどうか、イエスかノーかを考えています。なお、特に断らずに使っているグラフの用語については第 3 回を参照してください。

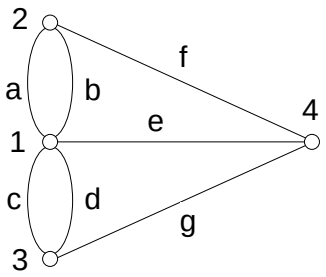


図 19　グラフ G_0

オイラーサイクルとは、グラフのすべての辺をちょうど一度だけ通って出発点に戻るパス (通過する辺の系列) のことです。通常は対象とするグラフを連結なものに限定して考えます。ここでも、問題の性質からグラフは連結であるとして説明します。

グラフが 1 点だけの場合は孤立点あるいは 1 本以上の自己ループを持ちます。点の次数は (0 も含めて) 偶数です。自己ループを 1 本以上持つときはオイラーサイクルの存在は明らかです。さらに、1 点のみで辺を持たないときも「オイラーサイクルはある」という言い方をします。また、2 点以上の連結グラフについては自己ループの有無はオイラーサイクルがあるか否かに影響を与えませんので、これ以後は

2 点以上を持つ自己ループのない連結グラフを前提として

説明します。このとき

辺は 1 本以上存在し、すべての点の次数は 1 以上である

ということに注意してください。

1.1 オイラーサイクルがあるとき

はじめに、連結グラフがオイラーサイクルを持っている場合にはどのようなことが成り立つだろうか考えてみましょう。以下のことはいいですね。

（1）どの点も次数が 1 以上だからオイラーサイクルはすべての点を通る。したがって、どの点もオイラーサイクルの出発点であり、終点である。

1.1.1 具体例で考えてみる

では、図 20 のグラフ G_2 を例として考えてみましょう。これは既に第 3 回の図 10 ででで示したグラフです。図 21 に G_2 の一つのオイラーサイクルを示します。通過する辺に向きを付けており、辺の横に記載の (1)〜(11) が通過の順番を表しています。このオイラーサイクルには全ての点が含まれますし、どの点についても接続するすべての辺はこのサイクルに含まれます。オイラーサイクルが点を通るたびに入る辺と出る辺がペアでありますので、この点に接続する辺の数は偶数です。ですから、どの点も次数は (2 以上の) 偶数ということになります。

いま G_2 のオイラーサイクルを例として説明しました。これでイメージはつかめたと思いますが、以下に一般的な場合を説明しておきます。でも読みたくない場合、あるいは抽象的な話は苦手という場合はスキップしてください。以後の話に影響はありません。

1.1.2 少し一般的な説明

図 22、図 23 を見ていただくとイメージはつかみやすいと思います。出発点 s では、オイラーサイクルが出るとき、入るときそれぞれで通る辺（別々の辺です）がペアで存在します。二度以上出入りしても常にペアで存在しますし、s に接続する辺は必ずオイラーサイクルに含まれます。ですから、s の次数は偶数です。s 以外の途中の点 v についても、図 23 で入るときに通る辺と出るときに通る辺（やはり別々の辺です）を見ていただけば、s と同じことが成り立つことはわかると思います。まとめると以下の（2）となります。

（2）オイラーサイクルがあるならば、すべての点の次数は (2 以上の) 偶数である。

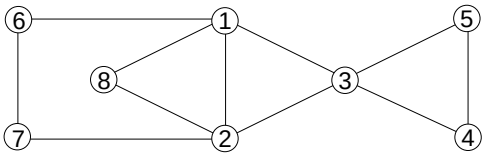


図 20　グラフ G_2（すべての点の次数は偶数）

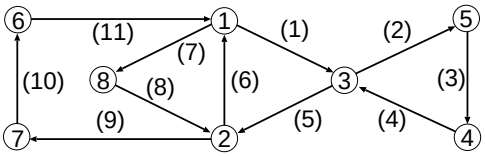


図 21　G_2 の一つのオイラーサイクル (辺に向きを入れて示しています)

1.2 すべての点の次数が偶数のとき

次に、全ての点の次数が偶数のときにオイラーサイクルがあるかどうかを考えてみましょう。ここでは、すべての点の次数は 2 以上です。

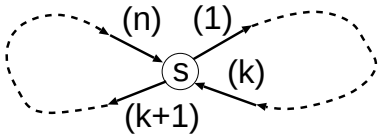


図 22　出発点（および終点）s の次数

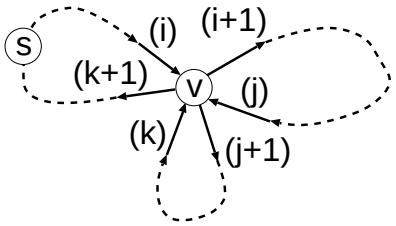


図 23　途中の点 v の次数（i の後に j, その後に k の順で出現）

1.2.1　具体例で考えてみる

まず、図 20 の G_2 を例として考えてみましょう。

(**ステップ 1**：サイクル C_0 の選択)

すべての点の次数は 2 以上の偶数です。点 1 を出発点 s として選ぶことにします。(実は、どの点を出発点としても以下の説明は同様です。) 図 24 に示すように s から s へのサイクル C_0：$1 \to 3 \to 2 \to 1$ を選びます。全ての点が含まれていれば終了ですが、今は違います。ここで、C_0 に含まれる辺をすべて G_2 から削除します（点は残し、線分のみ除去します)。残ったグラフを G_2' としましょう。図 25 を見てください。G_2' は必ずしも連結とは限りません。この例では G_2' は H_1, H_2 という 2 つの (2 点以上を含む) 連結グラフに分かれています。[*1]　ここで大切なことは、

- H_1, H_2 はどちらも (2 点以上で) すべての点の次数が 2 以上の偶数である

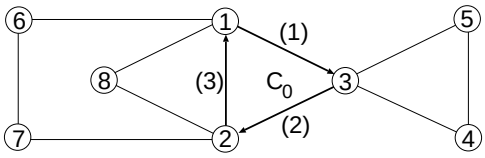


図 24　G_2 における 1 から 1 へのサイクル C_0

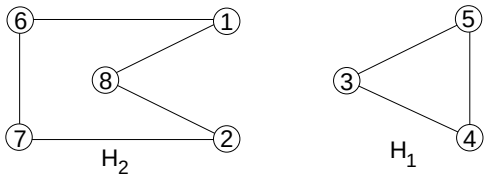
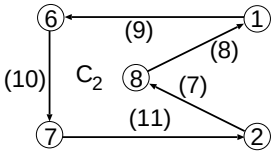


図 25　G_2 から C_0 に含まれる辺を除去したグラフ G_2'（2 つの連結グラフ H_1, H_2 に分かれる：すべての点の次数は偶数のまま）



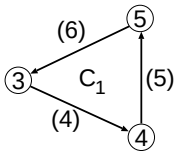


図 26　H_1 の 3 から 3 へのサイクル C_1、H_2 の 2 から 2 へのサイクル C_2

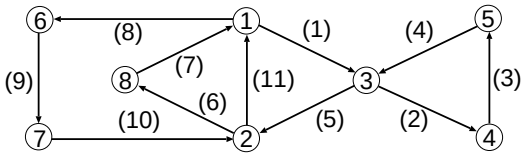


図 27　C_0, C_1, C_2 をつないでできる G_2 のオイラーサイクル：どの点を出発点としてもよい

ということです。G_2 ではすべての点の次数は 2 以上の偶数です。サイクル C_0 は通過する点に接続する辺を

[*1] G_2' において 1 点のみの連結グラフがあったとしても、それは C_0 に含まれる点であり、以後のサイクル拡大には影響しません。

2 本ずつ含みますので、削除によってこれらの通過する点の次数は 2 ずつ減少します。したがって、G_2' での点の次数も偶数ですが、H_1, H_2 どちらも 2 点以上含みますからどの点も次数は 2 以上の偶数です。

(**ステップ 2**：各連結グラフでのサイクル選択および各サイクルと C_0 との接続)

(**2.1**) 図 26 に示すように、H_1, H_2 それぞれで C_0 に含まれていた点を出発点とするサイクルを選びます。H_1 では 3 から 3 へのサイクル C_1、H_2 では 2 から 2 へのサイクル C_2 を選びました。この例ではどちらもそれぞれのグラフでオイラーサイクルになっています。(いつもそうなるとは限りません：同様なサイクル選びがさらに続く場合もあります。)

(**2.2**) 次に、図 27 に示しますようにこれらのサイクルを C_0 との共有点で接続します。C_1, C_2 はどちらも C_0 に含まれる点を出発点としましたので、C_0 と C_1、C_0 と C_2 それぞれのサイクルのペアには共有点がありますから、接続することができます。その結果サイクルが拡大されます。この例ではこの時点で G_2 のオイラーサイクルになっています。ただ、辺の通過順序は C_0 を回る途中で C_1 の周回動作を入れ、それから C_0 の周回動作に戻り、再び途中で C_2 の周回動作を入れ、C_0 の周回操作を再開して C_0 の周回動作を終える形に変更します。

(サイクル拡大における周回動作について) 具体的に言いますと、C_0 を $1 \to 3$ と進み、ここで $3 \to 4 \to 5 \to 3$ と C_1 を回り、再び C_0 を $3 \to 2$ と進んだところで今度は $2 \to 8 \to 1 \to 6 \to 7 \to 2$ と C_1 を回り、最後に C_0 を $2 \to 1$ と進むように周回動作を変更します。この結果サイクルが拡大され、今の場合は 1 から 1 へのオイラーサイクルとなっています。

ここまで、サイクルを見つけて拡大していき、最終的にオイラーサイクルを作る方法を図 20 を例として説明しましたが、実はこれはオイラーサイクルを作ることができる一般的な方法です。以下に少し一般的な説明をしておきますが、ここはスキップしても差し支えありません。スキップしても以後の話に影響はありません。

1.2.2 少し一般的な説明

注 4.1 いま対象とするグラフは連結で 2 点以上を持ち、すべての点の次数が 2 以上の偶数です。1 点 s を選んで、各点からまだ通過していない辺に沿って隣接点へ移動することを反復して s からパスをどんどん伸ばしていきます。もし s に戻る前にどこかの点 v でそれ以上パスを伸ばせなくなったとすると[*2]、v の次数は奇数です (理由を考えてみてください)。これは前提に矛盾します。したがって、s から s へのサイクルが存在することになります。このことについての詳しい説明は別の機会に譲ることとし、ここでは

- (2 点以上の連結グラフでは) すべての点の次数が偶数ならば、どの点 v についても v を含むサイクルを選ぶことができる

として話を進めます。

図 28 に一般的な状況を示してみました。

(**ステップ 1**) グラフ G で s を含むサイクル C を見つけ、C に含まれる辺を G から削除します。この結果、いくつかの連結グラフが生じます。(図 28 は、3 つの連結グラフ H_1, H_2, H_3 が残る状況のイメージで、網掛け部分が連結グラフを示しています。)

[*2] v へ到達した後で v を通るサイクルを何度も反復する形でパスが伸びるとしても、v の次数が偶数ならば v に接続する辺でこれらのサイクルに含まれないものがありますので、v からこの接続辺を通ってパスを伸ばすことができます。

どれも 2 点以上としていいですし、その規模（点や辺の数）は小さくなっていきます。また、削除によって、C に含まれる点の次数が 2 ずつ減少します。したがって、

- 削除後に残った (2 点以上の) 連結グラフの点の次数は (2 以上の) 偶数である

ということが維持されます。

(**ステップ 2**) 新しく生じた各連結グラフ H_i(ここでは $H_1,\ H_2,\ H_3$) について、以下を実行します。

2.1　C 上の点 v_i を含むような一つのサイクル C_i を見つける;
2.2　v_i などの共有点で C と C_i を接続し、C の周回動作の途中 v_i のところで、C_i に入って一周して v_i に戻る操作を挿入する;　/* s を含むサイクルの拡大 */
2.3　C_i に含まれる辺を H_i から削除する;

(**ステップ 3**) 拡大されたサイクルを新たに C とし、操作対象を新しく生じた連結グラフすべてとしてステップ 2 の実行に戻ります。　/* 以後、同様の操作を反復します */

注 4.2 ステップ 1 では、(C 除去後に) 連結グラフが生じていない状況であれば操作全体を終了しますが、そのことを明確には記述していません。なお、/*$\cdots$*/は $\cdots$ 部分が注釈で、操作には無関係な文章です。

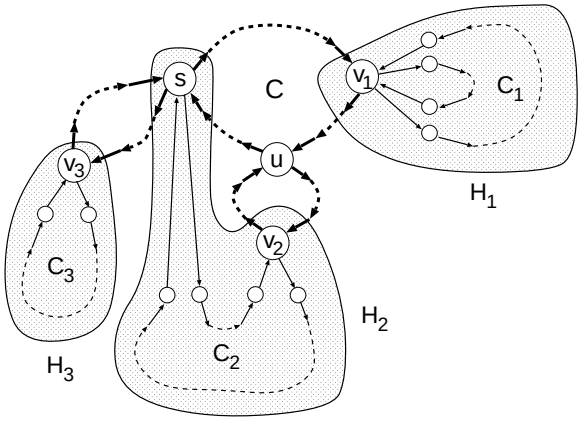


図 28　G からサイクル C に含まれる辺を削除して生じる連結グラフ $H_1,\ H_2,\ H_3$（網掛け部分で、2 点以上の場合を考えればよい）；C は $s \dashrightarrow v_1 \dashrightarrow u \dashrightarrow v_2 \dashrightarrow u \dashrightarrow s \dashrightarrow v_3 \dashrightarrow s$ と進む

このような拡大方法でオイラーサイクルが構成できることの証明は H_1 などの新しく生じた連結グラフの点数、辺数が G より少なくなっていることがポイントなのですが、ここでは深入りすることは避けて、「このような操作を続けていけば最終的にオイラーサイクルができる」ということだけ伝えておきます。その雰囲気は例題や上記の説明である程度感じていただけると思っています。

結局、(2 点以上の連結グラフについて) 以下の（3）が成り立つことがわかります。

（3）すべての点の次数が偶数であるならばオイラーサイクルが存在する。

1.3 まとめ

さて、ここまでのまとめをしておきましょう。2 点以上の連結グラフが前提です。どの点の次数も 1 以上です。はじめに

（2）グラフにオイラーサイクルがあるならば、すべての点の次数は偶数である。

が成り立つことを示しました。次に、

（3）すべての点の次数が偶数であるならば、オイラーサイクルが存在する。

が成り立つことを示しました。

（2）や（3）のような文章を**命題**（めいだい）といいます。そして、これらの命題が成り立つ（あるいは、命題が正しい）ことがわかりました。（2）のように命題が

『A:オイラーサイクルがある』ならば『B:すべての点の次数は偶数である』

という「A ならば B」の形であるとき、「『B の否定』ならば『A の否定』」の形の命題を (元の命題の) **対偶命題** (contrapositive) といいます。（対偶 (contraposition あるいは contrapositive) は「たいぐう」と読みます。）具体的には、（2）については

『B の否定：次数が奇数の点が一つ以上ある』ならば『A の否定：オイラーサイクルは存在しない』
(注：説明用に「B の否定：」、「A の否定：」を書いています)

が対偶命題ということになります。重要なことは、

- 元の命題が成り立つときには、必ず対偶命題も成り立つ

ということです。(理由の説明は少し準備が必要で長くなりますので、別の機会にします。) その結果、

（2）の対偶命題：次数が奇数の点が一つ以上あるならば、オイラーサイクルは存在しない

が成り立ちます。これらの命題を合わせると、オイラーサイクルがあるかどうかが、すべての点の次数が偶数か、一つでも次数が奇数の点があるか、で判定できることになります。

(オイラーサイクルが存在するか否かの判定)

- まず「すべての点の次数が偶数である」が成り立つときには、（3）によりオイラーサイクルがあります。
- 一方、「すべての点の次数が偶数である」が成り立たないとしましょう。これは「次数が奇数の点が一つ以上はある」という否定の形が成り立つことを意味しますので、（2）の対偶命題によって、「オイラーサイクルは存在しない」が成り立つことになります。

この判定方法を使うと、図 19 のグラフ G_0 ではすべての点の次数が奇数ですので、オイラーサイクルは存在しないことが分かります。すなわち、答はノーです。以上がオイラーの示した結果の筋道です。

これに関連して次のような表現を見かけることがあるかもしれません。意味は別の機会に説明しますので、見るだけで差し支えありません。

（オイラーサイクルの存在条件）連結グラフにおいて、オイラーサイクルが存在するための必要かつ十

分な条件はすべての点の次数が偶数であることである。

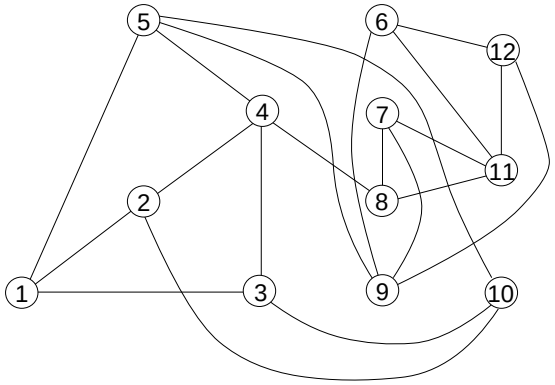


図 29　グラフ G_3（点数 12、辺数 20）

さて、ここで説明した点の次数に着目した判定法のありがたさは皆さんに伝わったでしょうか。図 19 のグラフ G_0 は点数 4、辺数 7 ですので、総当たり的にオイラーサイクル探しをしてみればあまり手数をかけずにノーの答を得ることができるかもしれません。では、図 29 のグラフ G_3 ではどうでしょうか。点数 12、辺数 20 ですが、総当たり的にオイラーサイクル探しをしてみるとその大変さが分かると思います。点の次数による判定法の方がはるかに楽だ、ということを実感することになるでしょう。

ちょっと一言　ところで、グラフの点数や辺数が何千、何万となった場合はどうでしょう。総当たり的なオイラーサイクル探しを諦めるのは当然かもしれませんが、点の次数をチェックすること自体にも手間がかかってくることになります。では、その場合にはどのように対処していくことが考えられるでしょうか。このような大規模なグラフの一筆書きをすることはないから考える必要はない、と思う人もいるかもしれません。一方で、このようなことに正面から向きあって、一筆書きではないかもしれないが今後大規模なグラフモデルを使って何か処理を行う場合に必要になるかもしれない、と考えてみましょう。すると、そこにはコンピュータの利活用が出てきます。そして、グラフをコンピュータで扱うにはどうすればいいか、という問題に行き着きます。

ここまで出てきたいくつかの話題の関係をまとめてみると、

- グラフにオイラーサイクルがあるかどうかの判定（問題）
- 点の次数に着目した判定法（問題解決のために行う処理の方法）
- この判定法をコンピュータで実行する

という流れになります。点の次数に着目した判定法は、問題を解決するために行う具体的な作業です。1 点ずつ次数を計算して偶数か奇数かを調べていく、という具体的な作業の記述（「処理の手順」ともいいます）であると考えることができます。このような物事の処理の手順を**アルゴリズム**（algorithm）といいます。何かを行う場合によく言う「段取り」なども同じ意味合いです。最終的には、アルゴリズムの意図する操作をプログラムとして記述してコンピュータに実行させることで実際の処理につながります。

結局、ここまでの流れは

グラフの問題 $\rightarrow$ アルゴリズムの設計 $\rightarrow$ プログラミング

という枠組みとなります。今回まではまだ最初の「グラフ」に関する説明が殆どですが、点の次数による判定法はアルゴリズムの設計に関係しています。今後は２番目、３番目の話題についても解説していきたいと思っています。