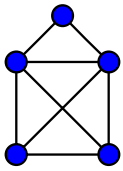
28

Eulerian Circuits

Michael Behrisch, Amin Coja-Oghlan, and Peter Liske

Humboldt-Universität zu Berlin, Berlin, Germany
University of Warwick, Coventry, UK
Humboldt-Universität zu Berlin, Berlin, Germany



Teasing your mates with riddles can be quite an amusing pastime – provided that you know the answer already! The "House of Santa Claus" provides a nice little teaser:

This figure consists of five *nodes* (the blue dots) and eight *edges* (the lines that connect the nodes). *Can you draw the House of Santa Claus in one sweep, without lifting the pen and without drawing any edge twice?*

Of course, it won't be long until all your friends know how to solve this one (as there are actually 44 different ways of drawing the House of Santa Claus). But fortunately there are plenty of other figures that can be drawn in one sweep as well, provided that you know how. In some other cases you may end up trying for quite a while, just to realize that drawing the figure in one go seems all but impossible.

In this chapter we present an *algorithm* that will *always* produce a way to draw a given figure in one go if this is possible. To devise such an algorithm, let us first try to deal with figures that can be drawn in one sweep such that

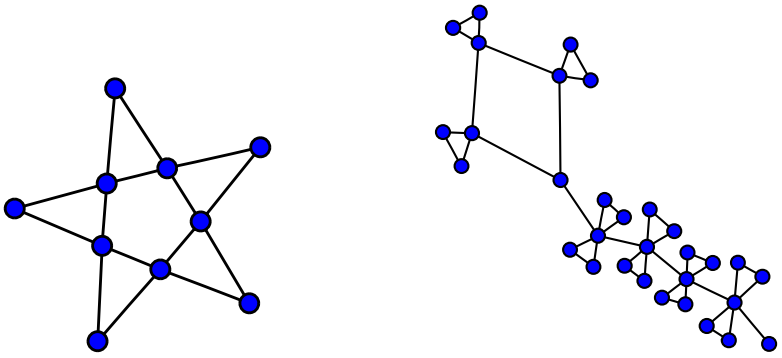


Fig. 28.1. Try the star and the dragon!

B. Vöcking et al. (eds.), *Algorithms Unplugged*,
DOI 10.1007/978-3-642-15328-0_28,

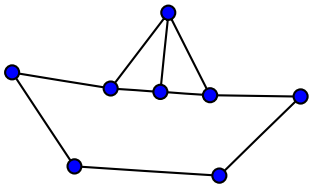


Fig. 28.2. The sailing boat provides a counterexample

the pen ends up at the same node where it started. The route that the pen traverses is then called an *Eulerian circuit.* This is because the mathematician *Leonhard Euler* was the first to find out what figures can be sketched in this way. (Actually Leonhard Euler was mostly interested in one particular figure, namely the roadmap of his home town of Königsberg.) Following Euler, we first deal with the question of whether an Eulerian circuit exists. Later on we sort out how to find an Eulerian circuit quickly if there is one.

In what follows we are going to figure out why, for example, the star has an Eulerian circuit, and why neither the House of Santa Claus nor the ship admit one. Bizarrely, the House of Santa Claus can be drawn in one go if we allow that the pen ends up at a different node than where it started, whereas even this is impossible in the case of the ship. To find out why, we need to take a closer look at the *nodes*, i.e., the places where the pen can change direction. (Recall that the nodes are just the blue dots in our figures.)

When Does an Eulerian Circuit Exist?

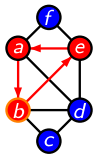
The *degree* of a node is the number of edges that pass through that node. For instance, the degree of the mast top of the ship is three. If we draw a figure in one sweep so that in the end the pen returns to the starting point, then the pen leaves every vertex exactly as many times as it enters that vertex. In effect, *the degree of each node is even.*

Observe that the ship has four nodes of odd degree (namely, all the nodes that belong to the sail). This shows that it is impossible to draw the ship in one go such that the starting point and the endpoint coincide. The same is true of the House of Santa Claus, because it has two nodes of degree three. However, there is a little twist (to be revealed later) that enables us to draw the figure in one sweep, but with different starting point and endpoint. By contrast, all nodes of the star have even degree. *But does any figure with this property feature an Eulerian circuit? And, if so, how do we actually find one?*

Finding Eulerian Circuits

Suppose that all nodes have even degrees. In the absence of a better idea, we could just *start somewhere and go ahead drawing*. That is, we start at an arbitrary node and follow an arbitrary edge to another node. Once we get there, we pick another edge that we haven't used before arbitrarily, and so on.

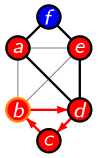
Each time we enter or leave a node, we use up two of its edges (because we are not allowed to use an edge more than once). Hence, if the node had an even degree initially, the number of available edges at that node will remain even. As a consequence, we won't get stuck in a dead end. In other words, our "just go ahead" strategy will eventually lead us back to the node where we started.



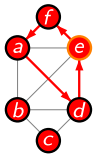
In the left figure a *circuit* (i.e., a route through several edges that leads back to the origin) consisting of three edges has emerged. The circuit passes through the vertices `b`, `e`, and `a`.

Yet following the above strategy ("start somewhere and keep on going until you get back to the origin") does not necessarily yield a circuit that covers *all* edges. Namely, we could have taken a "shortcut" at some node, thereby skipping a part of the figure. If this happens, we will need to *extend the circuit that we have constructed so far*. Before we do this, we remove the edges that we have visited already (because we are not allowed to pass through the same edges again anyway).

For instance, in the above figure upon returning to the origin `b`, we can repeat the same procedure to construct another circuit. In the above example the second circuit is a triangle through the nodes `b`, `d`, and `c` (right figure).



Linking the two circuits that we have obtained so far, we obtain a longer circuit `(b,e,a,b,d,c,b)`. Alas, even this circuit does not comprise all the edges. Hence, we are in for another extension. Of course, since all the edges that pass through the start vertex `b` are already used up, we need to pick another vertex to construct the next circuit.



Observe that removing all edges that our current circuit `(b,e,a,b,d,c,b)` passes through leaves us with a figure in which all nodes have even degrees. Hence, we can easily *find yet another circuit* as follows: Pick a node on the "old" circuit that has an edge that we haven't visited yet. Declare this node the new starting vertex and proceed to find another circuit by following the "just draw ahead" strategy. In the above example we get the quadrangle with the corners `a`, `d`, `e`, and `f` as shown in the left figure.

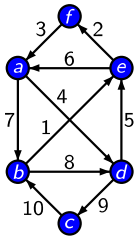
Thus, in addition to our "old" circuit we have got a new one that starts and ends at some node of the old circuit. Now, the plan is to hook the new

circuit into the old one. To achieve this, we first follow the old circuit until we reach the node where the new circuit starts. In the above example this is node `e`. We then proceed through the new circuit until we get back to its starting point (i.e., `e`). Finally, having completed the new circuit, we continue following the old circuit until the end. Hence, the complete route is `(b,e,f,a,d,e,a,b,d,c,b)`.

In summary, we have created a "big" circuit by combining two "small" ones. In our example the big circuit comprises the entire figure, as desired. But what do we do in other cases, where even the big circuit does not pass through all the edges?

Well, it's easy: Why not just repeat the entire procedure? That is, we search for a node on the big circuit that has an edge that we have not passed through yet. Since the original figure was just one connected object, such a vertex exists so long as we have not passed through all the edges. After removing all edges of the big circuit from the figure, we start at the chosen node and construct a new circuit just as before. Then, we link the two circuits to obtain a bigger one.

We keep doing this until all the edges of the figure are used up. Once all edges are finished, we have an Eulerian circuit.



In the above example we first combined the two circuits `(b,e,a,b)` and `(b,d,c,b)` to obtain the circuit `(b,e,a,b,d,c,b)`. Then, starting anew from node `e`, we obtained `(e,f,a,d,e)`. Linking this to the previously obtained circuit `(b,e,a,b,d,c,b)`, we finally constructed the Eulerian circuit `(b,e,a,d,e,f,a,b,d,c,b)`, which is depicted in the right figure. The numbers indicate the order in which the tour traverses the edges.

The Algorithm

The algorithm below works similarly to the example above except for the linking step, which is performed in place. Hence, after finding a subcircuit (e.g. `(b,e,a,b,d,c,b)`), the next circuit will be inserted right away. For instance, assume the current circuit is `(b,e,a,b,d,c,b)` and the node chosen in line 4 is $u =$ `a`. The edge selected could be `(a,d)`, which would extend the circuit preliminary to `(b,e,a,d,b,d,c,b)`. Obviously, this is not a valid circuit yet but the algorithm will continue until it reaches `a` again.

The algorithm EulerianCircuit calculates for a figure with even degree nodes the way to draw it in one sweep, and prints the order in which the edges have to be followed

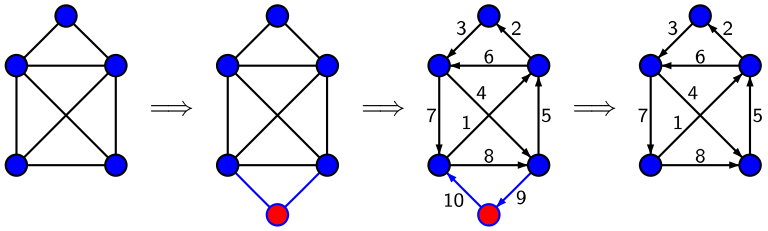
```
1   function EulerianCircuit(Figure F)
2   begin
3       Circuit := (s), for an arbitrary (start) node s in F
4       while there is a node u with an outgoing edge in the Circuit
5           v := u
6           repeat
7               take an edge v − w, starting in v
8               insert the other end node w into the Circuit after v
9               v := w
10              remove the edge from F
11          until v = u      // the circuit is closed
12      endwhile
13      return Circuit
14  end
```

The House of Santa Claus

Up to now, we have only cared about figures featuring an Eulerian circuit, which could thus be drawn in a single sweep with matching start and end point. We already know that this works only if all nodes of the figure have even degree. But this is not true for the House of Santa Claus: Both bottom nodes have degree 3. Nevertheless, it is possible to draw the figure in one sweep, if we do not insist on starting point and endpoint being identical.

How can we adapt the algorithm so that it works for the House of Santa Claus as well?

The simple trick is to insert a new node and connect it to both nodes of degree 3. The resulting figure has only nodes of even degree, and thus the algorithm will work and produce an Eulerian Circuit.



In the end we simply omit our "artificial" node from the Eulerian Circuit: We start drawing at the left "neighbor node" and finish at the right neighbor, thus getting a solution for the House of Santa Claus! This works particularly well in our example because the edges added are the last ones in the circuit.

If this is not the case we have to "shift" the circuit (this means, rotate the order of the edges such that 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 becomes 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, for example), to move those edges to the end.

If we have exactly two nodes of odd degree in our figure, this trick will always work. Figures with more than two nodes of odd degree cannot be drawn in one sweep at all, even if you allow the start and end point to differ.

Of Postmen and Garbage Collectors

In addition to the fact that you can impress your mates with the algorithm described (especially by deciding at "first sight", i.e., by testing the degrees of the nodes, whether the figure can be drawn in one sweep), there are also some serious applications. Assume the edges are streets and the nodes junctions. Then a path visiting each street exactly once is an Eulerian Circuit. Thus, if you have Eulerian Circuits in your road network they provide fast and fuel-saving routes for postal delivery and garbage collection. Our algorithm can calculate these routes, but we would, of course, implement it on a computer to process road networks with hundreds of streets.

Unfortunately, in reality there are often more than two junctions with an odd number of streets. That is why the calculation of the garbage collection routes has to take into account that some streets have to be used multiple times. When calculating the shortest route in this scenario the length of the streets comes into play as well. This leads to the so-called *Chinese Postman Problem*.

Thus, dealing with Eulerian Circuits does not only allow for a round trip through edges and nodes but also leads from one pleasant carrier (Santa Claus) to another (the postman).

Further Reading

1. http://en.wikipedia.org/wiki/Seven_Bridges_of_Koenigsberg
   This Wikipedia article explains everything about the starting point of this problem as a "touristical" question addressed by Euler more than 270 years ago.
2. http://en.wikipedia.org/wiki/Leonhard_Euler
   Life and work of the eponym not only of the Eulerian Circuit but also Eulerian Number and many other important mathematical achievements.
3. In contrast to an Eulerian Circuit, which visits all the edges of a given figure, Chap. 9 is about testing whether a figure contains a given vertex. The algorithm for this problem is quite similar to the one for finding Eulerian Circuits.

4. Chapter 40 (the Travelling Salesman Problem)
   In this chapter we saw an algorithm for finding an Eulerian Circuit, i.e., a circuit that traverses every *edge* of a given figure exactly once. Surprisingly, the problem of finding a cycle that merely visits every *vertex* precisely once seems much more difficult. If in fact the goal is to find a *shortest* such cycle, we end up with a notoriously difficult problem known as the *Travelling Salesman Problem*.
5. Think beyond: We have seen that the algorithm works for figures with an even number of outgoing lines at each point, we know that we can easily repair two "odd nodes" and that it does not work with four. But what about one or three "odd nodes"?