MAU22C00 Lecture 21

John Stalker

Trinity College Dublin

Finding an Eulerian circuit

A trail is a path without repeated edges. A circuit is a closed trail, i.e. one where the initial and final vertices are the same. An Eulerian trail is a trail which traverses each edge.

If a graph has an Eulerian circuit then at most one connected component has more than a single vertex and all vertices have even order.

Suppose at most one connected component has more than a single vertex and all vertices have even order. Is there an Eulerian circuit?

In other words, are our necessary conditions also sufficient?

We can ignore connected components with only one vertex since they have no edges.

So we can assume our graph is connected and non-empty.

First idea: Start anywhere and wander around (nearly) aimlessly.

Don't repeat any edges, but otherwise choose edges at random until none are available.

Illustration

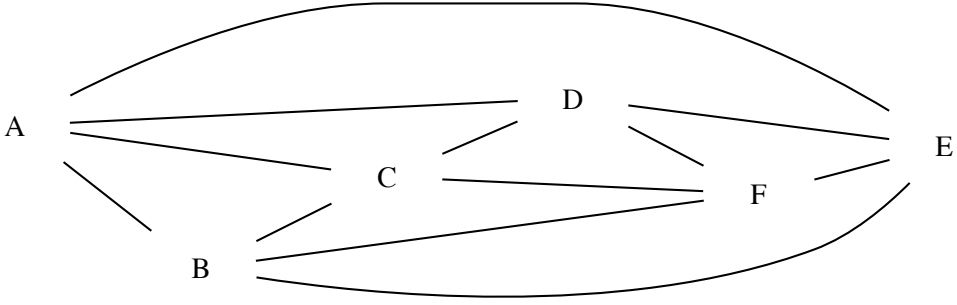

Figure 1: Searching for an Eulerian circuit

Illustration

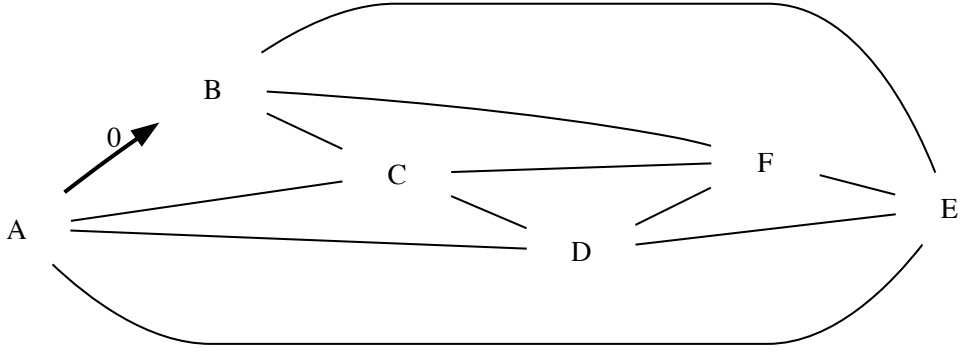

Figure 2: Searching for an Eulerian circuit

We randomly pick A as the initial vertex and the edge to B as the initial edge.

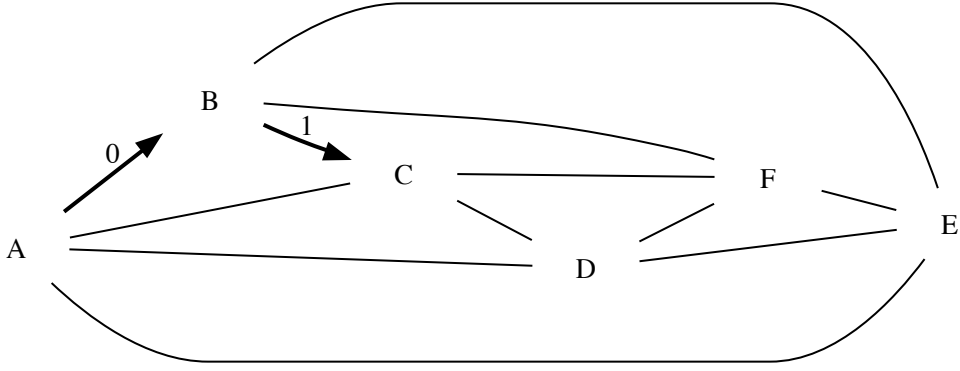

Figure 3: Searching for an Eulerian circuit

We can continue (there are still edges from B available) so we do.

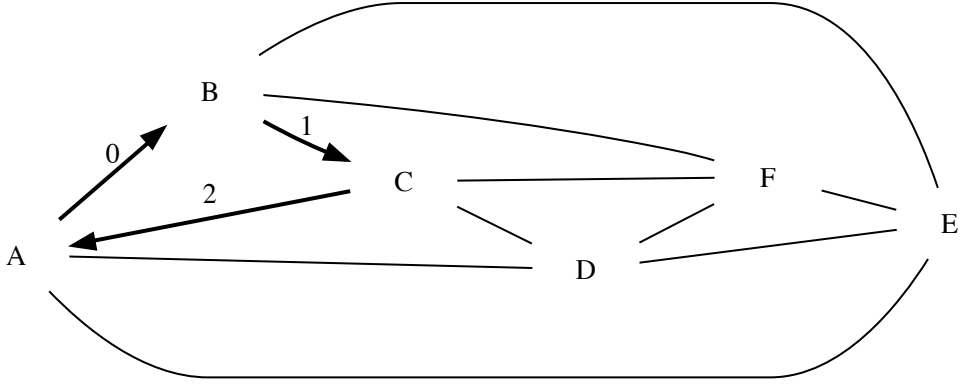

Figure 4: Searching for an Eulerian circuit

Back where we started, but that's okay. There are still unused outgoing edges so we can continue.

Illustration

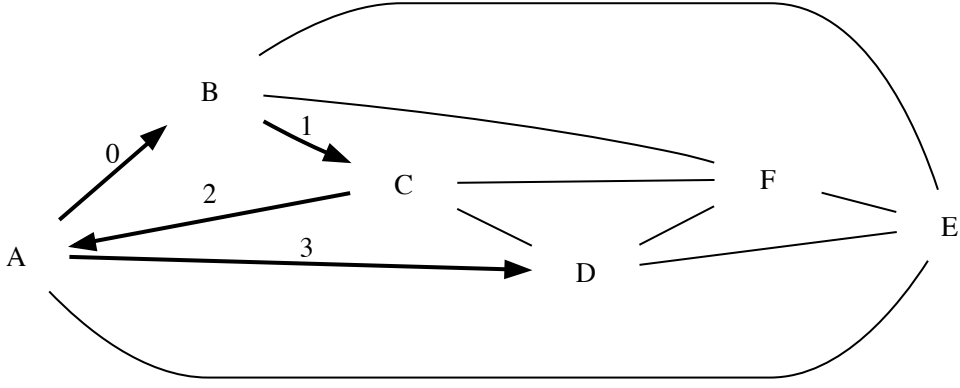

Figure 5: Searching for an Eulerian circuit

Keep wandering around.

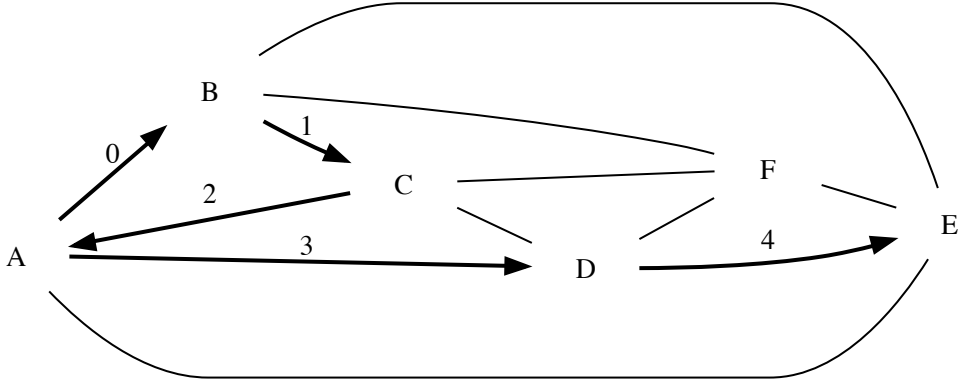

Figure 6: Searching for an Eulerian circuit

Still wandering.

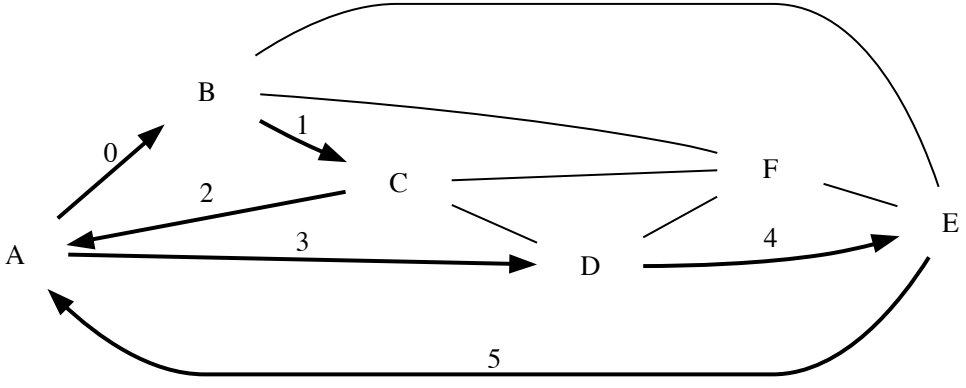

Figure 7: Searching for an Eulerian circuit

We're back where we started and have nowhere to go. There are no available edges left at A.

Dead ends, backtracking

We reached a dead end, i.e. a point where our trail can't be continued. What can we do?

There is no edge from our current vertex which hasn't already been traversed. Are there such edges from some other vertex?

If not then we're done. We have an Eulerian trail!

It must be a circuit because all vertices were of even degree.

Unfortunately here there are other vertices with unused edges.

Second idea: backtracking. Go back to the most recently visited such vertex and make a different choice.

Illustration, continued

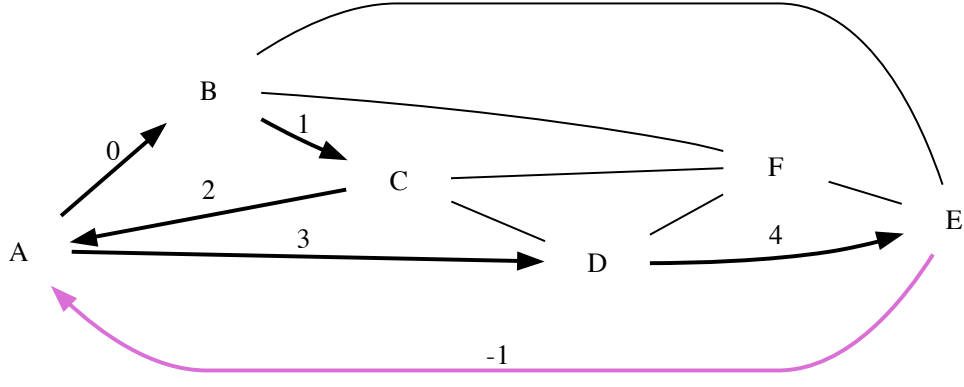

Figure 8: Still searching for an Eulerian circuit

Go back to E, the most recent vertex where we have available edges.

Illustration, continued

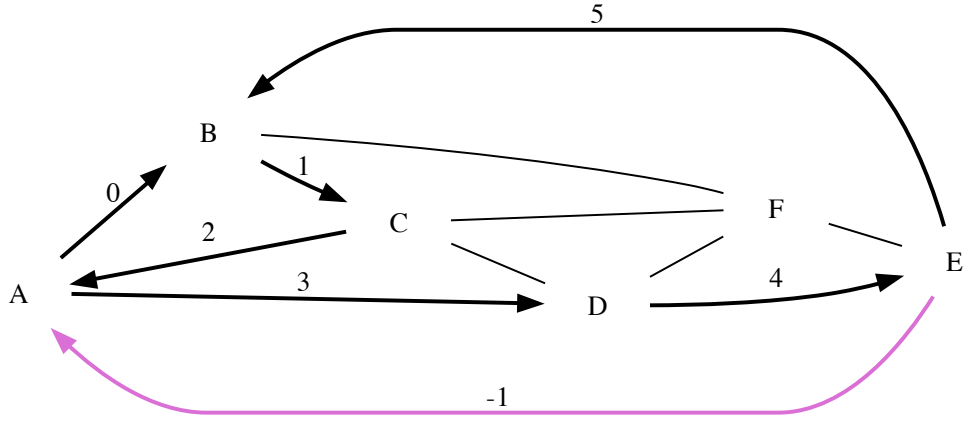

Figure 9: Still searching for an Eulerian circuit

Wander around some more, from E

Illustration, continued

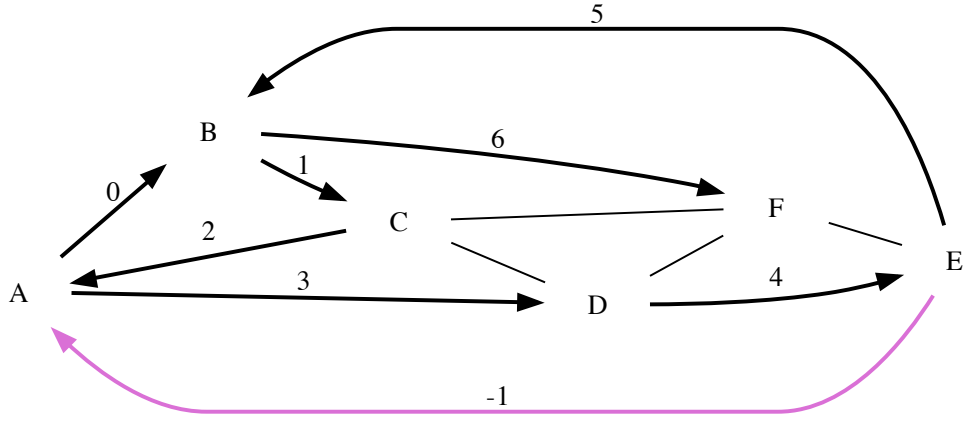

Figure 10: Still searching for an Eulerian circuit

Keep wandering.

Illustration, continued

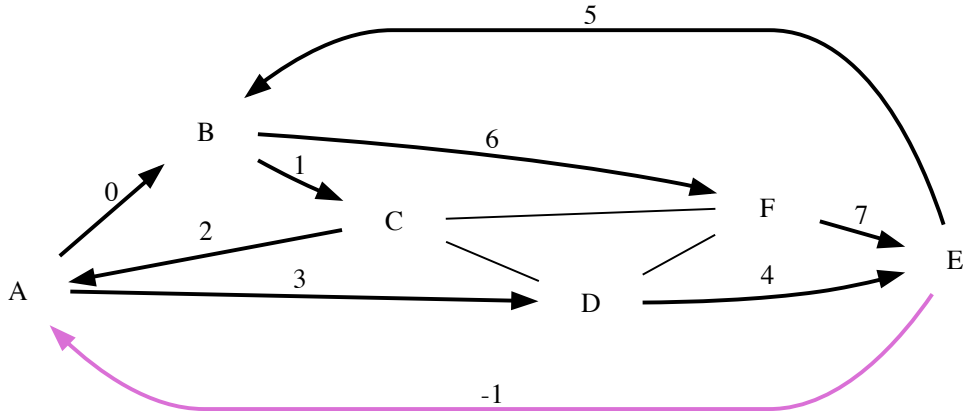

Figure 11: Still searching for an Eulerian circuit

More aimless wandering. Now we're stuck again though.

Illustration, continued

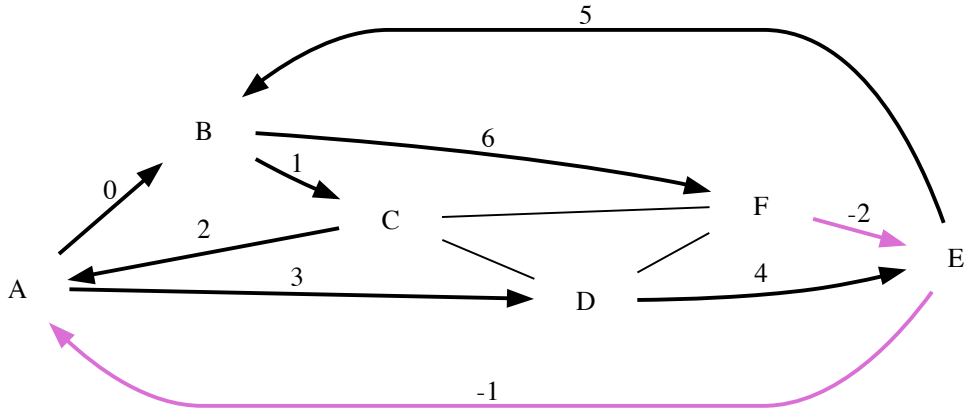

Figure 12: Still searching for an Eulerian circuit

Backtrack to F, the most recent vertex where we still have available edges.

Illustration, continued

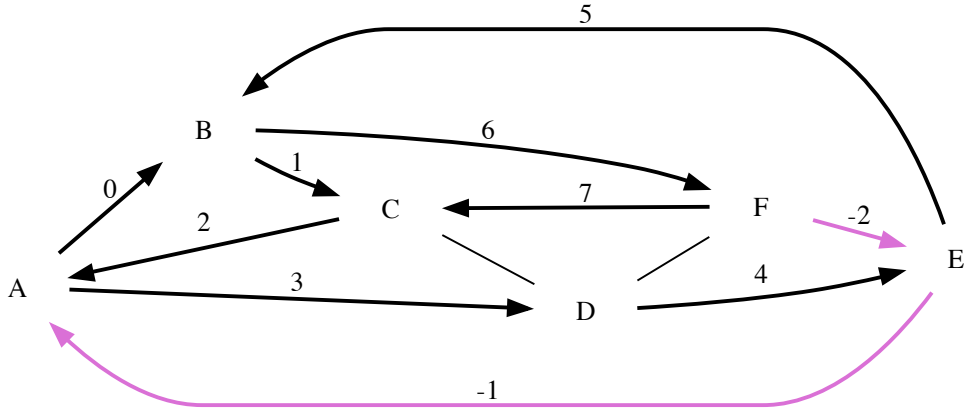

Figure 13: Still searching for an Eulerian circuit

Wander some more.

Illustration, continued

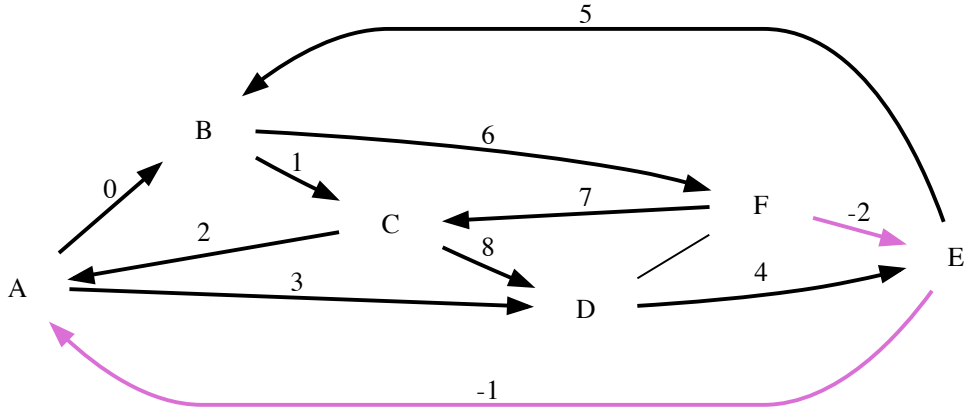

Figure 14: Still searching for an Eulerian circuit

Keep on wandering

Illustration, continued

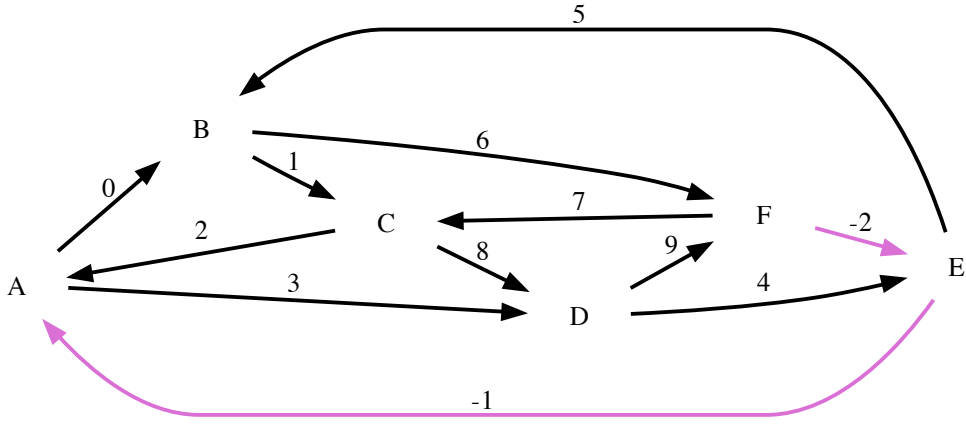

Figure 15: Still searching for an Eulerian circuit

We're stuck again, but this time it's okay.

Illustration, continued

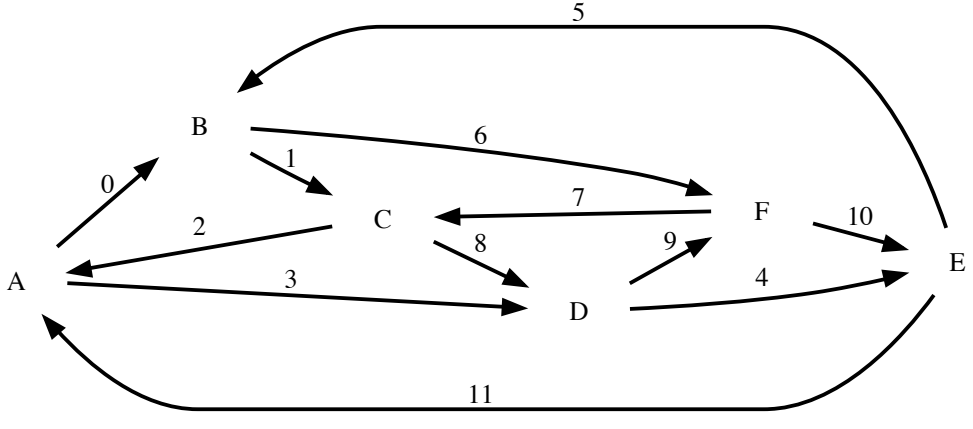

Figure 16: Found an Eulerian circuit

Two variants of the algorithm

There are two variants of this method, which give slightly different circuits:

- Keep the initial vertex of the trail fixed throughout.
  - When we backtrack we splice in a new circuit in the middle of the old one.
  - At intermediate stages we have two trails, black and purple in the diagram, which are joined at the end.
- When we have to backtrack we removed edges from the end, and tack them on at the beginning, changing the initial vertex.
  - There's always one trail.
  - The purple edges in the diagram aren't waiting to be added at the end; they're the beginning of the trail.
  - The numbering on all edges needs to be updated, not just the backtracked ones.

The two versions give trails which differ by a rotation, i.e. taking a segment from the end and adding it to the beginning.

The numbering on the previous diagram and the proof in the notes correspond to the first version. The next diagram, and the proof I'll give below, correspond to the second.

Illustration, continued

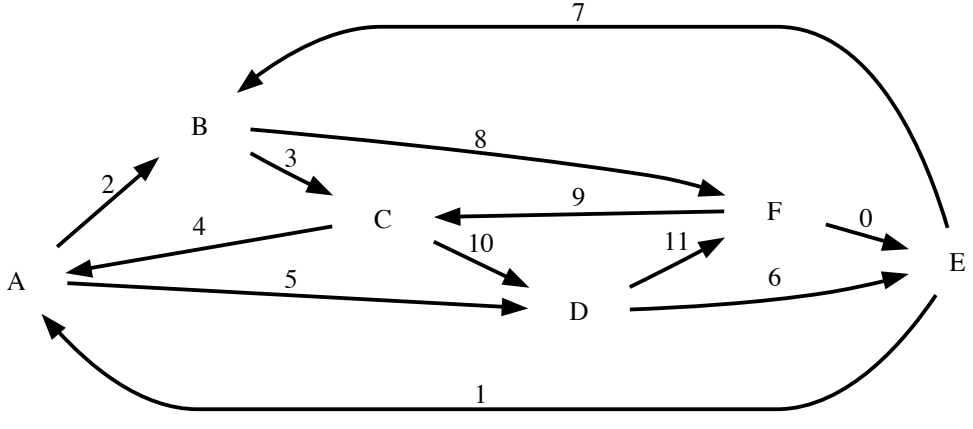

Figure 17: Found another Eulerian circuit

Why did this work?

Questions:

- Okay, this worked for this graph, but does it work in general?
- If so, why?

This is a non-deterministic algorithm, but unlike most non-deterministic algorithms this one works for *all* choices we have to make, not just some. That makes life easier.

We can prove that it works if we can show that

- it eventually terminates, and
- if it terminates then it terminates successfully.

Rephrasing the algorithm

At each step our state consists of a vertex and a trail. Initially the vertex is any vertex and the trail is empty. Afterwards the vertex is the final vertex of the trail.

If there are unused edges, i.e. one not part of the trail, at our current vertex we append one of them to the trail.

If there is no unused edge there and the trail is not a circuit, we halt.

If there is no unused edge there and the trail is a circuit we check whether some other vertex has unused edges, and halt if the answer is no.

Otherwise there is no unused edge at the current vertex, the trail is a circuit, and there is some other vertex on the trail which has unused edges. We then rotate the circuit so that that vertex is the initial and final vertex of the circuit, then append one of the unused edges.

Does this work, in general?

At every step we halt or append an edge to our trail.

The number of edges in the graph is finite and the trail never repeats one, so the algorithm must terminate. Does it terminate successfully?

There are two cases where we halt:

- There are no unused edges at the final vertex and the trail is not a circuit. This would be an unsuccessful termination, so we need to show this doesn't happen.
- There are no unused edges at any vertex of the trail, and the trail is a circuit. We need to show this is always a successful termination.

Every vertex was assumed to have even degree. The trail is a subgraph with even degree at all vertices, except the degree at the endpoints is odd if it's not a circuit.

The subgraph with the remaining edges then has odd degree at those vertices. Zero is not odd, so there's at least one unused edge at the final vertex.

In other words, the first case above can't happen.

Successful termination

Suppose there are no unused edges at any vertex of the trail, and the trail is a circuit.

Is there an unused edge whose endpoints are not on the trail?

The graph is connected, so if there were we could find a walk from a vertex on the trail to an endpoint of that edge.

Where does this walk leave the trail? It must be at a vertex on the trail where there was an unused edge, but there are no such vertices.

So there is no unused edge anywhere in graph, and the trail is Eulerian.

We've just established that an undirected graph without self loops has an Eulerian circuit if and only if it satisfies the following two conditions:

- At most one connected component has more than one vertex.
- All vertices have even degree.

Algorithms and proofs

If you want to show that something exists by giving an algorithm to find it you need to give a proof that the algorithm terminates successfully.

Some proofs that an object exists can be turned into an algorithm for finding it.

The algorithm above is adapted from the proof in the notes.

The proof there starts by saying there must be a trail of maximal length, and then showing that this trail must be Eulerian.

The first part is easy and the second part is done by showing how to extend a non-Eulerian trail.

The proof is a proof by contradiction, but we can still turn it into an algorithm!

The other case

If there's an Eulerian trail either all vertices have even degree and it's a circuit or two have odd degree and it's not.

Is it true that every connected-ish graph with exactly two vertices of odd degree has an Eulerian trail?

Yes. You can adapt the proof we just gave, but it's tedious and not completely straightforward.

Or you can resort to trickery, to reduce it to the case we just solved.

First attempt:

- Create a new graph, with an extra edge joining the vertices of odd degree.
- The new graph has all vertices of even degree.
- Apply the previously proved result to get an Eulerian circuit for the new graph.
- Rotate, if necessary, to put the extra edge at the end of the circuit.
- Remove that edge to get an Eulerian trail for the original graph.

The other case, continued

The trick on the previous slide almost works. Take a moment to see if you can see

- What's wrong with it
- How to fix it

The problem is that there might already be an edge between the two vertices of odd degree. We're not allowed to add another because we're not considering multigraphs.

We could fix it by redoing everything for multigraphs. That works, but is painful.

Second attempt:

- Create a new graph, with an extra vertex and two extra edges joining it to each of the vertices of odd degree.
- The new graph has all vertices of even degree.
- Apply the previously proved result to get an Eulerian circuit for the new graph.
- Rotate, if necessary, to put the extra vertex as the endpoints of the circuit.
- The two extra edges are now the first and last edge of the circuit.
- Remove those edges to get an Eulerian trail for the original graph.

Necessary and sufficient conditions for an Eulerian trail

We've now established that an undirected graph without self loops has an Eulerian trail if and only if it satisfies the following two conditions:

- At most one connected component has more than one vertex.
- At most two vertices have odd degree.

Also, if all vertices have even degree then all Eulerian trails are circuits and if two have odd degree than no Eulerian trail is a circuit. There can't be only one vertex of add degree.

So we have a complete understanding of Eulerian trails. Unfortunately we don't have, and can't have, a similar understanding of Hamiltonian paths.