Eulerian tours and trails

These notes summarize the class lectures on Eulerian graphs.

A basic tool in the theory of Eulerian graphs is the following fact about trails. In what follows \( d_T(x) \) is the degree of a vertex relative to the subgraph defined by the edges and vertices of a trail \( T \).

**Theorem 1** Let \( T \) be a \((u,v)\)-trail in a general graph. If \( T \) is closed \((u = v)\), then \( d_T(x) \) is even for every vertex \( x \) of \( T \). If \( T \) is open \((u \neq v)\), then \( d_T(u) \) and \( d_T(v) \) are odd, but \( d_T(x) \) is even for all other vertices of \( T \).

**Definition 1** An Eulerian tour of a general graph \( G \) is a closed trail that visits every edge of \( G \). An Eulerian trail of \( G \) is an open trail that visits every edge of \( G \).

**Questions:**

1) Which graphs admit Eulerian tours? Which admit Eulerian trails?

2) How does one find Eulerian tours or trails if they exist?

Theorem 1 easily leads to a necessary condition for the existence of Eulerian trails. Consider a connected graph \( G \). A tour \( T \) of a graph \( G \) visits every edge and hence every vertex of \( G \). Hence, for every vertex \( x \) of \( G \) \( d_G(x) = d_T(x) \), and so by Theorem 1, it must be true that *Euler’s condition*:

\[
\text{For every vertex } x \text{ of } G, \quad d_G(x) \text{ is even.} \quad \text{(Euler’s condition)} \quad (1)
\]

In fact this condition is also sufficient for a graph to be Eulerian.

**Theorem 2** A general graph \( G \) admits an Eulerian tour if and only if the degree of every vertex is even and the edges of \( G \) all belong to one connected component.

**Remark.** The statement that a general graph satisfies Euler’s condition and that all its edges belong to one connected component is the same as saying that \( G \) is the union of a connected graph whose vertices all have even degrees with a (possibly empty) set of isolated vertices.
Proof of necessity in Theorem 2.

We repeat the argument given above, but this time taking care of the possibility of isolated vertices. Assume that $G$ admits an Eulerian tour $T$. Since all the vertices in a tour are connected and since the tour visits all the edges of $G$, the edges must all belong to a single component $H$ of $G$. Repeating the argument we gave above, $d_G(x) = d_T(x)$ for every vertex $x$ in $H$, implying by Theorem 1 that all vertices in $H$ have even degree. Any vertex of $G$ not in $H$ is isolated and hence has degree zero, which is even. Thus we have established that $G$ satisfies Euler’s condition (1) and that its edges belong to a single component of $G$.  

Proof of sufficiency in Theorem 2.

The proof here makes use of maximal trails. A maximal trail is a trail, containing at least one edge, that is not a proper subtrail of a longer trail. Maximal trails are easy to come by. For example, take an arbitrary, non-isolated vertex of a general graph $x$ and consider the collection of all trails that contain $x$. This collection is finite and it must contain at least one trail of longest possible length. Such a longest trail is maximal.

Our strategy for proving sufficiency in Theorem 2 is to establish two preliminary facts.

Lemma 1 Let $G$ satisfy Euler’s condition (1). A maximal trail in $G$ must be closed.

Lemma 2 Let $G$ be a connected general graph that satisfies condition (1). Let $S$ be a closed trail in $G$ that is not an Eulerian trail. Then the edge set of $S$ is properly contained in a longer trail, and hence $S$ is not maximal.

Before proving these lemmas, let us use them to complete the proof of sufficiency. Let $G$ satisfy (1) and assume that all the edges belong to one connected component. The isolated vertices of $G$, if any exist, play no part in an Eulerian tour, so we can eliminate them and assume simply that $G$ is connected, so that $G$ satisfies the hypotheses of Lemma 2. Take any vertex $x$ and let $T$ be a longest trail containing $x$. Then $T$ is non-trivial, that is contains edges, and $T$ is maximal. Lemma 1 implies that it is closed. Lemma 2 implies that it visits every edge of $G$. Hence $T$ is an Eulerian tour.  

\[ \diamond \]
We now prove the two lemmas.

**Proof of Lemma 1.** Let $T$ be any open trail in $G$, and let $v$ be the initial or terminal vertex of $T$. Then by Theorem 1, $d_T(v)$ is odd. By hypothesis, $d_G(v)$ is even and so there exist at least one edge $e$ incident on $v$ that is not already an edge of $T$. Adding this edge to $T$ creates a longer trail, so $T$ can not be maximal if it is open. Therefore maximal trails are closed.  

**Proof of Lemma 2.** Let $S$ be a closed trail in $G$ and assume that $S$ is not an Eulerian tour. Let $G'$ be the graph obtained by removing the edges of $S$ from $G$. $G'$ contains all the edges of $G$ not in $S$, so the edge set of $G'$ is non-empty. Also, for each $x$ in $G$, $d_{G'}(x) = d_G(x) - d_S(x)$. Since $d_G(x)$ is even by assumption and since $d_S(x)$ is even by Theorem 1, it follows that every vertex of $G'$ has even degree in $G'$.

Now, let $v$ be a vertex of $S$ incident to an edge $e$ not in $S$. Such a vertex $v$ must exist; otherwise $G$ itself would not be connected, contrary to assumption. This vertex thus belongs to a connected component of $G'$ and there is a non-trivial maximal trail in $G'$ that includes the edge $e$. Call this trail $S'$. It is closed by Lemma 1 because the vertices of $G'$ have even degrees. See the figure.

![Diagram](image_url)

The edges of $S$ and $S'$ are disjoint because $S'$ is a trail in $G'$ and $G'$ and $S$ share no common edges by definition of $G'$. Thus the closed walk obtained by first traversing the edges of $S$ from $v$ to $v$ and then traversing the edges of $S'$ from $v$ to $v$, defines a trail which is longer than $S$ and which contains $S$.  

Next we turn to the question of Eulerian trails. Let $G$ be a general graph that admits an Eulerian trail $T$ and let $u$ and $v$ be the endpoints of this trail. Recall that, by definition of an Eulerian trail, $u \neq v$. As before $d_G(x) = d_T(x)$ for all vertices $x$ of $G$. By Theorem 1, $d_G(u) = d_T(u)$ and $d_G(v) = d_T(v)$ are
odd, while \( d_G(x) = d_T(x) \) is even for all other vertices. Thus if \( G \) admits an Eulerian trail, exactly two of its vertices must have odd degree. Of course all the edges will also belong to one connected component defined by \( T \). The converse is also true. We state this result in the next theorem.

**Theorem 3** Existence of Eulerian trails. A general graph \( G \) admits an Eulerian trail if and only if all its edges belong to one connected component and it has exactly two vertices of odd degree.

**Proof:** We have already given the argument for the necessity of having two odd degree vertices and for the connectedness.

To argue sufficiency, we use a trick. Suppose all the edges of \( G \) are in one connected component and that exactly two vertices, call them \( u \) and \( v \) have odd degree. Add an edge \( e \) between \( u \) and \( v \) and call this augmented graph \( G' \). This addition will add one to the degrees of \( u \) and \( v \) and leave the degrees of all other vertices the same. Hence all the vertices of \( G' \) will have even degree. Also, the edges of \( G' \) will belong to one connected component. So we can apply Theorem 2 to conclude that \( G' \) admits an Eulerian tour. By removing the edge \( e \) from this tour one obtains an Eulerian trail that starts and ends at the vertices \( u \) and \( v \).

We next address the problem of finding algorithms to calculate Eulerian tours.

The first algorithm is inspired by the proof of Lemma 2. The idea is to find edge-disjoint closed trails and link them into longer trails until an Eulerian tour is constructed. To proceed, we formalize the procedure for finding a closed trail through a given point.

**Closed Trail Algorithm:**

**Input:** A connected graph \( G \) satisfying condition (1) and a vertex \( v \).

**Output:** A closed trail in \( G \) that starts and ends at \( v \).

**Algorithm:** Build a trail \( T \) by starting at \( v \) and adding edges on edges that are not already in \( T \) until the trail returns to \( v \).

It is not immediately clear that this algorithm works. One must show that by repeatedly choosing new edges one is forced eventually to return to
the initial vertex $v$. For this it is important the all the vertices have even degree. If there are vertices with odd degree, one can still build a trail by adding on new edges any way one likes. But the maximal trail one builds may not be closed, so long as edges are not repeated. We leave this proof as an exercise. The essential idea is contained in the proof of Lemma 1 from Theorem 1.

Given two, edge-disjoint closed trails $S$ and $\bar{S}$, we saw in the proof of Lemma 2 how we can combine them into a closed trail that traverses all the edges of $S$ and $\bar{S}$. Let us refer to this procedure as linking $S$ and $\bar{S}$. We allow the case in which $S = \emptyset$ is the empty trail with no vertices or edges; then we define the linking of $S$ and $\bar{S}$ to be simply $\bar{S}$.

**Euler Tour Algorithm:**

**Input:** A connected graph $G$ satisfying condition (1) and a vertex $x$.

**Output:** An Eulerian tour of $G$.

**Procedure:** Initialize $H := G$, $v := x$, $T := \emptyset$. Do

1. Apply the Closed Path Algorithm with input $H$ and $v$ to produce a closed trail $T'$. Link $T$ and $T'$ and relabel the larger closed trail $T$.

2. If $G - E(T)$ has no edges, stop. $T$ is an Eulerian tour. Otherwise, let $v$ be vertex of $T$ incident to an edge not in $T$, let $H$ be the component of $G - E(T)$ containing $v$, and return to step 1.

The proof that this algorithm works is contained essentially in the proof of Lemma 2. Taken together, the proof that the Closed Path Algorithm and the Euler tour algorithm work constitute a constructive version of the proof given above using Lemmas 1 and 2. The algorithms described above are not fully specified. To actually code them one must specify a protocol for which new edge to choose if there is more than one possibility.

One might ask if there is an algorithm which finds the Eulerian tour in one go, without producing shorter circuits, which then need to be spliced. There are such algorithms. Here is one.

**Fleury’s algorithm**

1. Input: A connected graph $G$ whose vertices all have even degree.
2. Output: An Eulerian circuit in $G$.

3. Notation: $R$ is the trail being built; $v$ is the current terminal vertex of $R$; $H$ is the graph of edges left to visit. Initialize by $H = G$ and by setting $v = x$, $R = \{x\}$ for any vertex $x$ in $G$.

4. Iteration. Add to $R$ any edge $e = vw$ incident to the current terminal vertex $v$ that is not a bridge of $H$, unless there is no other choice. Remove this edge $e$ from $H$ and any isolated vertices this removal creates. (Vertex $w$ becomes the new terminal vertex of $R$.) Iterate until $R$ traverses every edge of $G$.

One may summarize this algorithm as: never cross a bridge unless there is no other choice. As we will show in the proof of this algorithm, when it is necessary to cross a bridge, removal of this bridge from $H$ isolates the previous terminal vertex of $R$, which is then removed.

To see that this algorithm works one needs to show that the graph $H$ is connected at every step. If this is the case, the trail it produces will traverse every edge of the original graph $G$ and the degree of every vertex in the final trail will be the same as its degree in $G$. Since all vertices of $G$ have even degree, this final trail must be closed, by Theorem 1.

So assume that at some step $H$ is connected and $v$ is the terminal vertex of $R$. Notice that at every step, the vertices of $H$ have all even degree in $H$, except for the initial vertex $x$ and the current terminal vertex $v$ of $R$. Suppose that there are at least two edges of $H$ (that is, edges in $G$ which have not already been included in $R$) that are incident to $v$. We need to argue that two of these edges cannot simultaneously be bridges. A proof by contradiction will work. So assume that edges $e = vw$ and $f = vz$ incident to $v$ are bridges. Let $H_1$ be the component of $H - e$ containing the vertex $w$ and let $H_2$ be the component of $H - f$ containing the vertex $z$. These subgraphs must be disjoint. If they were not there would be a cycle containing both $e$ and $f$ in contradiction to the assumption that they are bridges. (Draw a picture!) One of these subgraphs will not contain the initial vertex $x$. Suppose it is $H_1$. If the bridge $e$ to $H_1$ is added to $R$ next, then eventually the algorithm must end at a vertex $y$ of $H_1$. This means that when $R$ arrives at $y$ for the final time it will have used all edges incident to $y$. But this is a contradiction. Since $y$ is not the initial vertex of $R$, the number of edges of $R$ incident to $y$ is odd, and it is assumed that all vertices have even degree.
We have shown that at most one edge of \( H \) incident to \( v \) can be a bridge. So unless there is only a single edge of \( H \) incident to \( v \), Fleury’s algorithm will subtract an edge from \( H \) that is not a bridge, so that \( H \) at the next step remains connected. If there is just one edge left in \( H \) incident to \( v \), \( v \) will be an end vertex of \( H \) and the edge a bridge; its removal will leave behind \( v \) as an isolated vertex and Fleury’s algorithm removes this isolated vertex from \( H \) also, leaving behind a connected graph.

The figure illustrates the algorithm; the dotted lines represented the Eulerian trail being built, the solid lines, edges of the current graph \( H \). Notice in going from the third graph to the fourth, the algorithm avoids taking the bridge to the left and thereby avoids ending prematurely at a tour of just the four leftmost vertices,