

## CME 305: Discrete Mathematics and Algorithms

Instructor: Professor Amin Saberi (saberi@stanford.edu)

January 8, 2009

# Lecture 1: Eulerian Circuits

Today we'll talk about one of the first known applications of graph theory, Eulerian circuits. We'll start by giving some basic definitions, prove Euler's result, then give an application from bioinformatics.

A **graph**  $G(V, E)$  is a set  $V$  of **vertices** and a set  $E$  of **edges**. We usually denote  $|V|$  as  $n$ , and  $|E|$  as  $m$ . In an **undirected** graph, an edge is an unordered pair of vertices. An ordered pair of vertices is called a **directed** edge. If we allow **multi-sets** of edges, i.e. multiple edges between two vertices, this is known as a **multigraph**. A self-loop or **loop** is an edge between a vertex and itself. An undirected graph without loops or multiple edges is known as a **simple** graph. In this class we will assume graphs to be simple unless otherwise stated.

If vertices  $a$  and  $b$  share an edge, we say that they are **adjacent** and write  $a \sim b$ . If vertex  $a$  is one of edge  $e$ 's endpoints,  $a$  is **incident** to  $e$  and we write  $a \in e$ . The **degree** of a vertex is the number of edges incident to it, denoted  $d_i$  for vertex  $v_i \in V$ . The **degree sequence** of graph is the sequence  $\{d_1, d_2, \dots, d_n\}$ .

A **walk** is a sequence of vertices  $v_1 \dots v_k$  such that  $\forall i \in 1..k-1, v_i \sim v_{i+1}$ . A **path** is a walk where  $v_i \neq v_j, \forall i \neq j$ , i.e. a walk that visits each vertex at most once. A **closed walk** is a walk where  $v_1 = v_k$ . A **cycle** is a closed path, i.e. a path combined with the edge  $(v_k, v_1)$ . A graph is **connected** if there exists a path between each pair of vertices.

A closed walk (circuit) on a graph  $G(V, E)$  is an **Eulerian circuit** if it passes through each edge in  $E$  exactly once. We call a graph **Eulerian** if it admits an Eulerian circuit.

The problem of finding Eulerian circuits is perhaps the oldest in graph theory. It originated with Euler in the 18th century; the problem was whether one could take a walk in Königsberg and cross each of the four bridges exactly once. Motivated by this, Euler was able to prove the following theorem:

**Theorem 1 (Euler, 1736)** *A graph  $G(V, E)$  is Eulerian if and only if  $G$  is connected and the degree of every vertex in  $G$  is even.*

### Proof:

Clearly,  $G$  must be connected, otherwise we will be unable to traverse all edges. As for even vertex degree:

" $\Rightarrow$ ": assume  $G$  has an Eulerian circuit. Every time the walk enters a vertex, it must leave it afterwards, so there must be an even degree or it will get "stuck". For the first vertex, we

leave it first and enter it last, so it must also have even degree.

“ $\Leftarrow$ ”: assume all the nodes in  $G$  have even degrees. Pick any vertex  $v \in V$  and start walking through the edges of the graph. Given that all the degrees of the nodes in  $G$  are even, we can only get stuck at  $v$ . Remove visited edges from  $G$ . If no edges remain, we are done. Otherwise, pick any remaining vertex with non-zero degree and restart traversal. Repeat this process until every edge has been removed.

We now have a collection of circuits  $c_1, \dots, c_k$ . We can merge them into one large circuit by traversing  $c_1$  until the first vertex  $v$  that's part of a not-yet traversed path  $c_j$ . We then start traversing  $c_j$ , following the same rule. We resume traversing  $c_1$  once  $c_j$  has finished. Note that since  $G$  was connected, we will be able to reach all circuits in this fashion. ■

Euler's theorem gives necessary and sufficient condition for whether a graph is Eulerian which can be easily checked in linear time. Note that this is a “constructive” proof: we explicitly defined an algorithm to find an Eulerian circuit in the proof. In addition, the algorithm is close to optimal, given that its running time is at most  $2|E|$  and  $|E|$  is an obvious lower bound on the number of operations any algorithm can achieve.

We can extend the result to find a necessary and sufficient condition for Eulerian paths, which is a walk (not necessarily closed) that visits each edge exactly once:

**Claim 1**  *$G$  has an Eulerian path if and only if it is connected and only two of its vertices have an odd degree.*

We can also extend this to directed graphs.

**Claim 2** *Eulerian circuits in directed graphs*

*Let  $G(V, E)$  be a directed graph.  $G$  has an Eulerian circuit if and only if*

- *$G$  is connected*
- *$\forall v \in V, \text{indegree}(v) = \text{outdegree}(v)$*

A seemingly similar structure was defined by Sir William Rowan Hamilton in the 19th century, but concerns visiting vertices once rather than edges. A cycle  $C$  in  $G$  is **Hamiltonian** if it visits every vertex in  $V$  exactly once.

This was originally invented and marketed by Hamilton as a board game, where the object was to find a Hamiltonian cycle around the edges of a dodecahedron. The puzzle failed financially, but the concept lives on.

A famous example of a Hamiltonian cycle problem is the *Knight's tour*, which asks whether one can move a knight in a chessboard while visiting each vertex exactly once and returning to the starting vertex. The graph representation consists of 64 vertices corresponding to the squares of the board, with an edge between vertices if there is a legal knight move between the two squares. Is there a Hamiltonian path (cycle) in the chess board graph? The answer is yes. Gauss enumerated the number of different Hamiltonian cycles in such a graph. It is a difficult problem...

As opposed to the Eulerian circuits, Hamiltonian paths are remarkably difficult to find. We will show later in the class that there exists no polynomial running time algorithm to find a Hamiltonian cycle in a general graph unless  $P = NP$ . For this particular problem, the best known running time is superpolynomial.

## Application: DNA Sequencing

An excellent reference for this topic is [2]. Recall that DNA consists of two complementary strands of nucleotides. There are 4 different nucleotides, namely  $A$ ,  $G$ ,  $C$  and  $T$  (for those wondering,  $U$  only appears in RNA, replacing  $T$ ). Our goal is to be able to read a DNA sequence. What's the best way to do it? One possible way (one that we will not study), due to Frederick Sanger, is called the Dideoxy termination method, or chain termination method. For this he we awarded the 1980 Nobel price in Chemistry.

A simpler technique, invented by Professor Patrick Brown from Stanford University, is based on *DNA arrays*. Basically the original DNA sequence is cut into pieces of a given length, tagged with a fluorescent agent, and exposed to an array of known sequences of the same length, so that the DNA pieces will hybridize with complementary sequences in the array. We can then tell whether particular sequences are present based on the strength of florescence in each square of the array. In this example we'll treat presense or absense as a binary variable, but in reality the florescence will be graded.

Assuming that this process is perfect, if we initially had a sequence of length  $n$  and we cut it into sub-sequences of length  $l$ , we see get each  $n - l + 1$  sub-sequences activated in the array. Given that we have 4 different nucleotides, there are  $4^l$  different sub-sequences of length  $l$ .

Given all the substrings of length  $l$  of a string  $S$  of length  $n$ , how can we reconstruct the original or the shortest string  $S_{opt}$ ? One was is to view this problem as finding a Hamiltonian path in a graph  $G(V, E)$ .

Put the set of vertices  $V$  as the set of all detected substrings of length  $l$  in the array. For each substring  $s_i$ , find all its possible successors (substrings  $s_j$  such that the first  $l - 1$  entries of  $s_j$  are identical to the last  $l - 1$  entries of  $s_i$ ) and add the directed edge  $(s_i, s_j)$  to  $E$ . Any Hamiltonian path in such graph defines a possible reconstruction of the original sequence.

This is a nice representation, except for the fact that finding a Hamiltonian path is an *NP*-Hard problem. There a better way of formulating the problem so that we can solve it more efficiently; we can pose it as finding an Eulerian circuit in a graph.

Instead of vertices, the detected subsequences represent edges in our new graph  $G'(V', E')$ . The nodes this time are the sequences of length  $l-1$  that appear in the detected subsequences. There are at most  $n-l+2$  such sequences. Put an directed edge between two vertices if the corresponding linking sequence is in the set of subsequences. In other words, if  $v_i$  and  $v_j$  are two nodes, then  $(v_i, v_j)$  is an edge if and only if the last  $l-2$  entries of  $v_i$  correspond to the first  $l-2$  entries of  $v_j$  AND the sequence obtained by concatenating  $v_i$  with the last entry of  $v_j$  is one of the given subsequences. For instance, if we had the sub-sequence ATCGA (here  $l=5$ ), then the nodes ATCG and TCGA would be connected.

Given that there are  $6 \times 10^9$  nucleotides in human's DNA, one can expect errors to happen. Even more likely is to have several repetitions of a given substring. To solve this problem, in practice two or more different spectrums of the same molecule (ie we use more than one value of  $l$ ) and we compare them.

## References

- [1] Lovasz, Laszlo. *Discrete Mathematics*. Springer-Verlag New York
- [2] N.C.Jones, P.A.Pevzner. *Bioinformatics Algorithms, An Introduction to*. The MIT Press