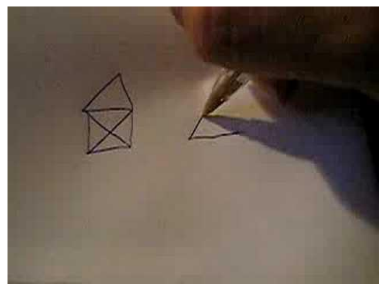
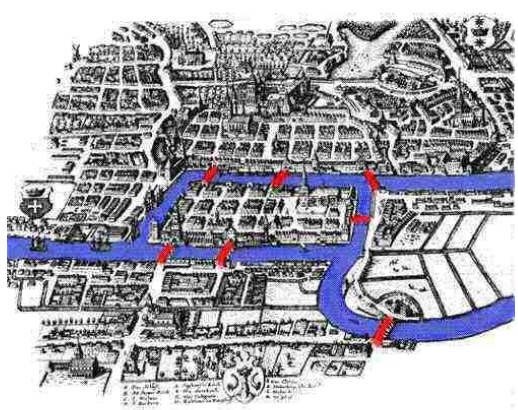
Ma/CS 6a

Class 8: Eulerian Cycles



By Adam Sheffer

The Bridges of Königsberg



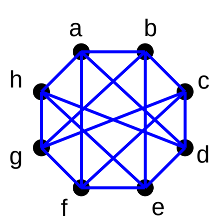
- Can we travel the city while crossing every bridge exactly once?

How Graph Theory Was Born

Leonhard
Euler 1736

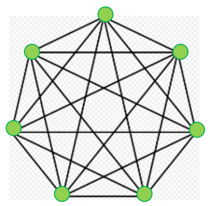
Eulerian Cycle

- An ***Eulerian path*** in a graph G is a path that passes through every edge of G exactly once.
- An ***Eulerian cycle*** is an Eulerian path that starts and ends at the same vertex.

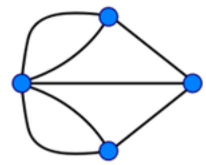


$$E = a \to b \to c \to d \to e \to f \to g \to h \to a \to d \to h \to e \to b \to g \to c \to f \to a$$

Is There an Eulerian Cycle in the Graph?



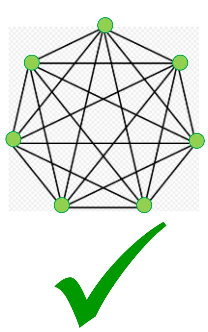
K_7

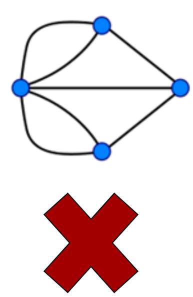


Königsberg

What Graphs Contain an Eulerian Cycle?

- **Claim.** An undirected connected (possibly not simple) graph $G = (V, E)$ contains an Eulerian cycle if and only if every vertex of V has an even degree.





Proving One Direction

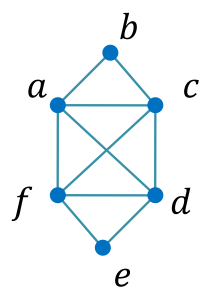
- Assume that G contains an Eulerian cycle and prove that the degrees are even:
- Choose an arbitrary Eulerian cycle and traverse it starting a vertex $s \in V$.
- For every $u \in V$, let k_u denote the number of times that we visit u.
  - Each time we visit u, we enter through one edge and leave through another. Thus, $\deg(u) = 2k_u$.
  - The only exception is $\deg(s) = 2k_s - 2$.

Proving the Other Direction

- Assume that every vertex has an even degree and prove that there exists an Eulerian cycle.
  - We prove the claim by presenting an algorithm that always finds such a cycle.

The Algorithm – Part 1

- Choose an arbitrary vertex $v \in V$ and start a path from it.
  - At each step, choose an edge, cross it, and throw it away from the graph.
  - Stop only when returning to v.



$$a \to f \to d \to a$$

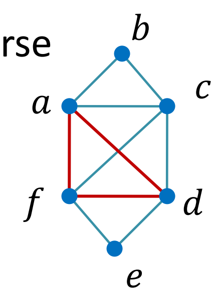
Correctness

- **Claim.** As long as we did not return to v, we cannot get stuck:
- **Proof.** For any vertex $u \in V \setminus \{v\}$, we claim that we cannot get stuck in u:
  - Before every visit of u, it has an even degree.
  - While visiting u, it has an odd degree.
  - It is impossible to visit u when it has degree 0.

The Algorithm – Part 2

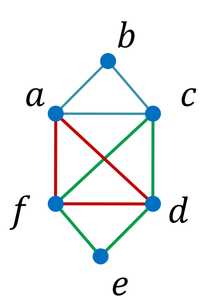
- If the cycle that we found contains every edge of the graph, we are done.
- Otherwise, one of the vertices that we visited still has a positive degree.
  - Because the graph is connected!
- Choose such a vertex and traverse the graph as before, until we return to our starting point.

$$f \to e \to d \to c \to f$$



The Algorithm – Part 2 (cont.)

- We now have two edge-disjoint cycles, with at least one common vertex between them.



$$a \to f \to d \to a$$

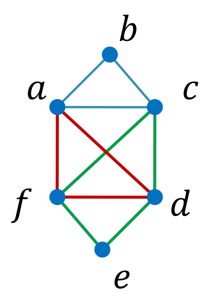
$$\uparrow$$

$$\underbrace{f \to e \to d \to c \to f}$$

$$a \to f \to e \to d \to c \to f \to d \to a$$

The Algorithm – Part 3

- Repeat part 2 until no edges remain in the graph.



$$a \to f \to e \to d \to c \to f \to d \to a$$

$$+$$

$$c \to b \to a \to c$$

$$=$$

$$a \to f \to e \to d \to c \to b \to a \to c \to f \to d \to a$$

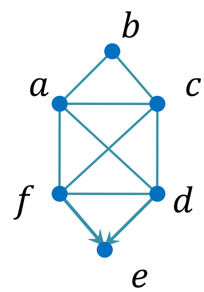
Correctness of the Algorithm

- Already proved: The algorithm cannot get stuck while constructing a cycle.
- Since G is connected, if there are remaining edges at the beginning of a step, at least one edge must be connected to the existing cycle.
- Termination. At the end of each step we obtain a cycle with a larger number of edges. Thus, the process terminates after $\leq |E|$ steps.

Eulerian Cycles in Directed Graphs

- Does the even degree condition still hold in a directed graph?

NO!



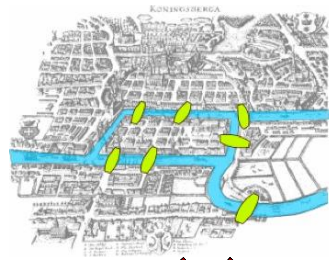
- What should the new condition be?
  - *Indegree* = *Outdegree*.

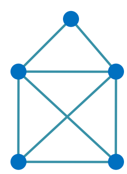
Eulerian Paths

- **Claim.** A connected undirected graph contains an Eulerian path (which is not a cycle) if and only if it contains exactly two vertices with odd degrees.

- **Recall.** An *Eulerian path* is an Eulerian cycle that does not necessarily start and end in the same vertex.

Example: Eulerian Paths



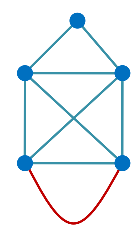


Proof – One Direction

- Assume that a (non-cycle) Eulerian path exists and show that exactly two vertices have an odd degree:
  - Similarly to the previous proof, we travel along the path.
  - A vertex that we visit k times has degree $2k$.
  - If we visited the first/last vertex k times, it is of degree $2k - 1$.

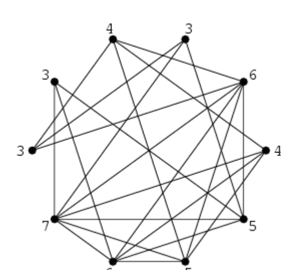
Proof – The Other Direction

- Assume that exactly two vertices have an odd degree and prove that an Eulerian path exists:
  - Add a new edge between these two vertices.
  - We obtain a graph with no odd degrees. Thus, it contains an Eulerian cycle.
  - Removing the new edge turns this cycle to an Eulerian path in the origianl graph.



Covering with Several Paths

- **Problem.** Let $G = (V, E)$ be a connected (not necessarily simple) graph with exactly $2k$ vertices of an odd degree. Prove that the edges of the graph can be covered by k edge-disjoint paths.



Solution

- By induction on k:
  - Basis: when $k = 1$, this is the case of a graph containing an Eulerian path.
  - Step: Assume for $2k - 2$ and prove for $2k$.
  - Add an edge e between two vertices with odd degrees. The resulting graph has $2k - 2$ vertices with odd degrees.
  - Induction hypothesis: the edges of the graph can be covered by $k - 1$ edge disjoint paths.
  - Removing e may split one path into two, resulting in k edge disjoint paths

Running Times

- Given a graph $G = (V, E)$ with
$$n = |V| + |E|.$$
  - Algorithm ***A*** computes something on G in $10^5 n^2$ steps.
  - Algorithm ***B*** computes the same thing in $10^{-2} n^3$ steps.
- Which algorithm is better?
  - Algorithm ***A*** is better when $n \geq 10^7$.
  - *Asymptotic computational complexity* – we only care about large values of n.

Asymptotic complexity – Shortest Paths

- Given a graph $G = (V, E)$ and vertices $s, t \in V$, we wish to find the shortest path between s and t.
  - Go over every possible path in G. There could be about $|V|!$ such *simple* paths.
  - BFS from s – finds the shortest path in at most $c(|V| + |E|)$ time (for some constant c).
- Which is better?
  - Since G is simple, $|E| < |V|^2$.
  - $|V|^2$ is ***MUCH*** better than $|V|!$.

Checking Some Values of $|V|$

|V|	$10^6|V|^2$	|V|!
1	10^6	1
5	$2.5 \cdot 10^7$	120
10	10^8	$3.6 \cdot 10^6$
50	$2.5 \cdot 10^9$	$\sim 3 \cdot 10^{64}$
100	10^{10}	$\sim 9 \cdot 10^{157}$
1000	10^{12}	$\sim 4 \cdot 10^{2567}$

What Can a Computer Do?

Intel Core i7 2600K	128,300 MIPS at 3.4 GHz	37.7	9.43	2011
Intel Core i7 Extreme Edition 3960X (Hex core)	177,730 MIPS at 3.33 GHz	53.3	8.89	2011
Fujitsu K computer (88,128 cores)	**10,000,000,000 MIPS** at 2 GHz	113,471.314	56.736	2011
AMD FX-8350	97,125 MIPS at 4.2 GHz	23.1	2.9	2012
Intel Core i7 3770k	106,924 MIPS at 3.9 GHz	27.4	6.9	2012
Intel Core i7 3630QM	113,093 MIPS at 3.2 GHz	35.3	8.83	2012
Intel Core i7 4770k	127,273 MIPS at 3.9 GHz	32.0	8.0	2013

$$10^{16} \cdot \underbrace{(3 \cdot 10^{8})}_{\text{Second per year}} = 3 \cdot 10^{24}.$$

In This Course

- We will not seriously analyze running times of algorithms.
- We will consider a running time to be "reasonable" if it is polynomial in n (the size of the input).
- For example, $cn!, c2^{n}, c2^{\sqrt{n}}$ are ***not*** reasonable running times.

The End

I need more jokes for last slides! Send me stuff!