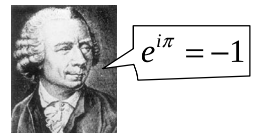
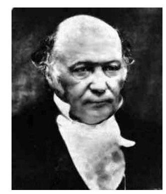
Lecture 26: Really, really hard problems: P versus NP

- <u>Today's Agenda:</u>
  - Solving 4^{th} grade pencil-on-paper puzzles
    - A "deep" algorithm for Euler Circuits
  - Euler with a twist: Hamiltonian circuits
  - Hamiltonian circuits and NP complete problems
  - The NP =? P problem
    - Your chance to win a Turing award!
    - Any takers?
- Covered in Chapter 9 in the textbook

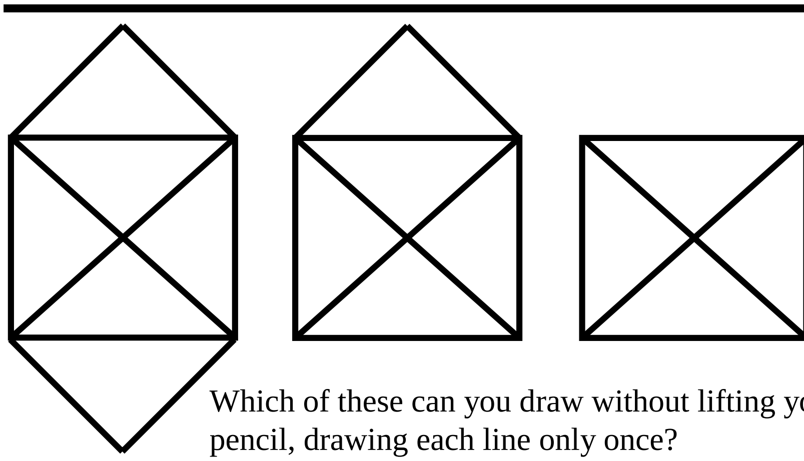


L. Euler
(1707-1783)



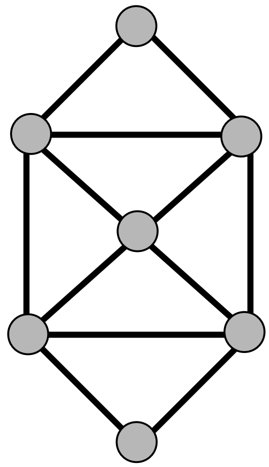
W. R. Hamilton
(1805-1865)

It's Puzzle Time!



Which of these can you draw without lifting your pencil, drawing each line only once?
Can you start and end at the same point?
(end: memories of 4^{th} grade days…)

Graph representation of the puzzle



Line segments = edges
Junctions = vertices

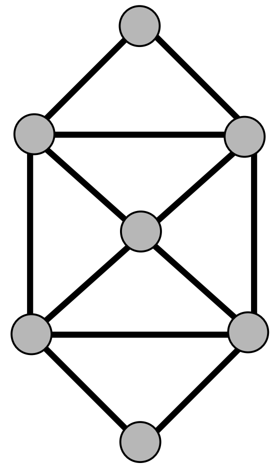
Can you traverse all edges exactly once, starting and finishing at the same vertex?

Euler Circuits

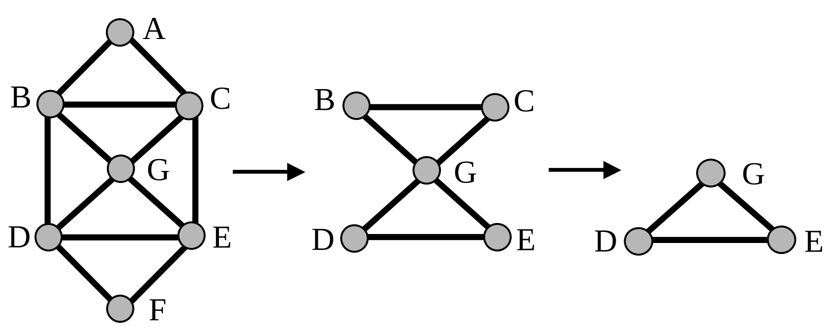
- Euler tour: a path through a graph that visits each edge exactly once
- Euler circuit: an Euler tour that starts and ends at the same vertex
- Observations:
  - An Euler circuit is only possible if the graph is connected and each vertex has even degree (# of edges onto vertex)
  - Why?
  - At every vertex, need one edge to get in and one edge to get out!

Finding Euler Circuits: DFS and then Splice

- Given a graph G = (*V*,*E*), find an Euler circuit in G
  - Can check if one exists in O(|*V*|) time (check degrees)
- Basic Euler Circuit Algorithm:
  1. Do a depth-first search (DFS) from a vertex until you are back at this vertex
  2. Pick a vertex on this path with an unused edge and repeat 1.
  3. Splice all these paths into an Euler circuit
- Running time = O(|*V*| + |*E*|)



Euler Circuit Example



DFS(A) :
A <u>B</u> D F E C A

DFS(<u>B</u>) :
B <u>G</u> C B

Splice at B A <u>B G C B</u> D F E C A

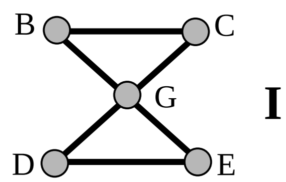
Splice at G

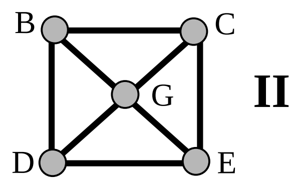
DFS(<u>G</u>) :
G D E G

A B <u>G D E G</u> C B D F E C A

Euler with a Twist: Hamiltonian Circuits

- Euler circuit: A cycle that goes through each *edge* exactly once
- Hamiltonian circuit: A cycle that goes through each *vertex* exactly once
- Does graph **I** have:
  - An Euler circuit?
  - A Hamiltonian circuit?
- Does graph **II** have:
  - An Euler circuit?
  - A Hamiltonian circuit?





Finding Hamiltonian Circuits in Graphs

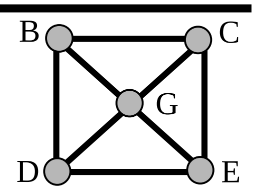
- Problem: Find a Hamiltonian circuit in a graph G = (V,E)
  - Sub-problem: Does G contain a Hamiltonian circuit?
  - Is there an easy (linear time) algorithm for checking this?

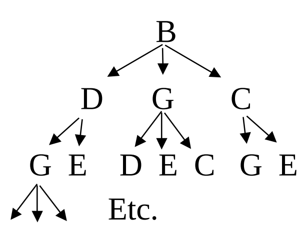
Finding Hamiltonian Circuits in Graphs

- Problem: Find a Hamiltonian circuit in a graph G = (*V*,*E*)
  - Sub-problem: Does G contain a Hamiltonian circuit?
  - No known easy algorithm for checking this…
- <u>One solution</u>: Search through *all paths* to find one that visits each vertex exactly once
  - Can use your favorite graph search algorithm (DFS!) to find various paths
- This is an *exhaustive search* (“brute force”) algorithm
- Worst case → need to search all paths
  - How many paths??

Analysis of our Exhaustive Search Algorithm

- Worst case → need to search all paths
  - How many paths?
- Can depict these paths as a *search tree*
- Let the average branching factor of each node in this tree be B (= average size of adjacency list for a vertex)
- $|V|$ vertices, each with ≈ B branches
- Total number of paths ≈ B·B·B … ·B = <u>$O(B^{|V|})$</u>
- Worst case → <u>Exponential time!</u>





Search tree of paths from B

How bad is exponential time?

N	log N	N log N	N^2	2^N
1	0	0	1	2
2	1	2	4	4
4	2	8	16	16
10	3	30	100	1024
100	7	700	10,000	1,000,000,000,000,00,000,000,000,000,000
1000	10	10,000	1,000,000	Fo'gettaboutit!
1,000,000	20	20,000,000	1,000,000,000,000	ditto
1,000,000,000	30	30,000,000,000	1,000,000,000,000,000,000	mega ditto plus

Polynomial versus Exponential Time

- Most of our algorithms so far have been O(log N), O(N), O(N log N) or O(N^2) running time for inputs of size N
  - These are all *polynomial time* algorithms
  - Their running time is O(N^k) for some $k > 0$
- Exponential time B^N is asymptotically *worse than any* polynomial function N^k for any k
  - For any k, N^k is o(B^N) for any constant $B > 1$
- Polynomial time algorithms are generally regarded as "fast" algorithms – these are the kind we want!
- Exponential time algorithms are generally inefficient – avoid these!

The "complexity" class P

- The set P is defined as the set of all problems that can be solved in <u>polynomial worse case time</u>
  - Also known as the polynomial time complexity class – contains problems whose time complexity is $O(N^k)$ for some k
- <u>Examples of problems in P</u>: searching, sorting, topological sort, single-source shortest path, Euler circuit, etc.

The "complexity" class NP

- <u>Definition</u>: NP is the set of all problems for which a given *candidate solution* can be *tested* in polynomial time
- Example of a problem in NP:
  - <u>Our new friend, the Hamiltonian circuit problem</u>: Why is it in NP?
    - Given a candidate path, can test in linear time if it is a Hamiltonian circuit – just check if all vertices are visited exactly once in the candidate path (except start/finish vertex)

Why NP?

- NP stands for Nondeterministic Polynomial time
  - Why "nondeterministic"? Corresponds to algorithms that can search all possible solutions in parallel and pick the correct one → each solution can be checked in polynomial time
  - Nondeterministic algorithms don't exist – purely theoretical idea invented to understand how hard a problem could be
- Examples of problems in NP:
  - Hamiltonian circuit: Given a candidate path, can test in linear time if it is a Hamiltonian circuit
  - Sorting: Can test in linear time if a candidate ordering is sorted
  - Sorting is also in P. Are any other problems in P also in NP?

Next Class:

More on P and NP

Review for Finals

Mini end-of-the-quarter party

To Do:

Programming Assignment #2 (Due next class)