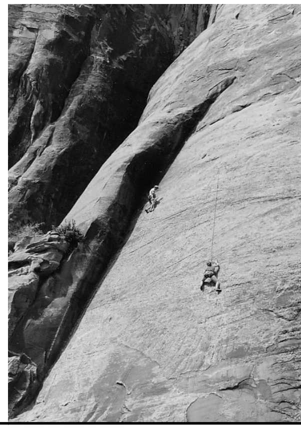
CSE 326: Data Structures
Lecture #20

Really, *Really* Hard Problems

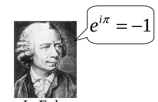
Henry Kautz

Winter Quarter 2002



Today’s Agenda

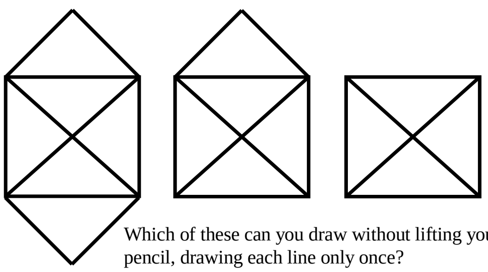
- Solving pencil-on-paper puzzles
  - A “deep” algorithm for Euler Circuits
- Euler with a twist: Hamiltonian circuits
- Hamiltonian circuits and NP complete problems
- The NP =? P problem
  - Your chance to win a Turing award!
  - Any takers?
- Weiss Chapter 9.7



L. Euler
(1707-1783)

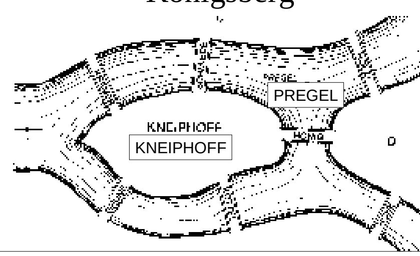
W. R. Hamilton
(1805-1865)

It’s Puzzle Time!



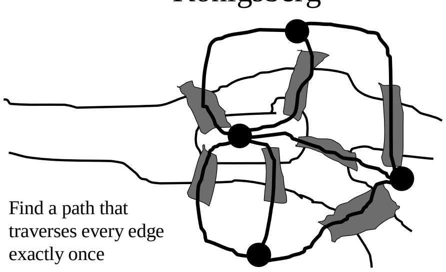
Which of these can you draw without lifting your pencil, drawing each line only once?
Can you start and end at the same point?

Historical Puzzle: Seven Bridges of Königsberg



Want to cross all bridges but…
Can cross each bridge only once (High toll to cross twice?!)

A “Multigraph” for the Bridges of Königsberg



Find a path that traverses every edge exactly once

Euler Circuits and Tours

- 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*
- Named after Leonhard Euler (1707-1783), who cracked this problem and founded graph theory in 1736
- Some observations for undirected graphs:
  - An Euler circuit is only possible if the graph is connected and each vertex has even degree (= # of edges on the vertex) [Why?]
  - An Euler tour is only possible if the graph is connected and either all vertices have even degree or exactly two have odd degree [Why?]

Euler Circuits and Tours

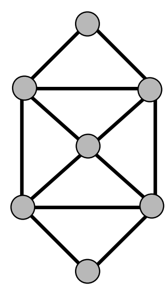
- Euler tour: a path through a graph that visits *each edge exactly once*
- Euler circuit: an Euler tour that *starts and ends at the same vertex*
- Named after Leonhard Euler (1707-1783), who cracked this problem and founded graph theory in 1736
- Some observations for undirected graphs:
  - An Euler circuit is only possible if the graph is connected and each vertex has even degree (= # of edges on the vertex)
    - Need one edge to get into vertex and one edge to get out
  - An Euler tour is only possible if the graph is connected and either all vertices have even degree or exactly two have odd degree
    - Could start at one odd vertex and end at the other

Euler Circuit Problem

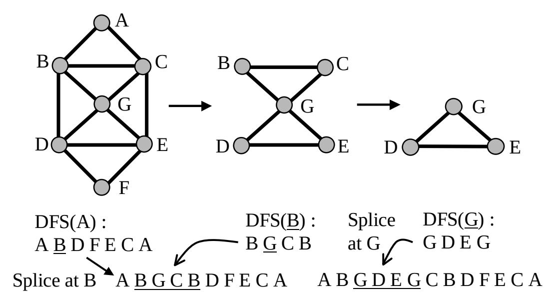
- Problem: Given an undirected graph G = (*V*,*E*), find an Euler circuit in G
- Note: Can check if one exists in linear time (how?)
- Given that an Euler circuit exists, how do we *construct* an Euler circuit for G?
- *Hint*: Think deep! We've discussed the answer in depth before…

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*|)

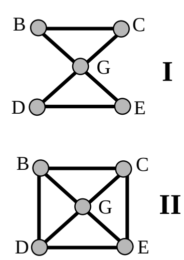


Euler Circuit Example



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?

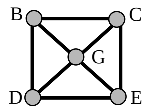


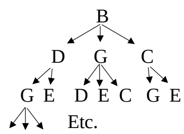
Finding Hamiltonian Circuits in Graphs

- Problem: Find a Hamiltonian circuit in a graph G = (*V*,*E*)
  - Sub-problem: Does G contain a Hamiltonian circuit?
  - No known easy algorithm for checking this…
- One solution: 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*
- $|V|$ vertices, each with $\approx B$ branches
- Total number of paths $\approx B \cdot B \cdot B \ldots \cdot B$ $= O(B^{|V|})$
- Worst case → Exponential time!





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

Review: 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 $\Omega(B^N)$ for any constant $B > 1$

The Complexity Class P

- The set P is defined as the set of all problems that can be solved in *polynomial worse case time*
  - Also known as the *polynomial time* complexity class
  - All *problems* that have some *algorithm* whose running time is $O(N^k)$ for some *k*
- Examples of problems in P: tree search, sorting, shortest path, Euler circuit, *etc*.

The Complexity Class NP

- *Definition*: 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:
  - Hamiltonian circuit problem: *Why is it in NP?*

The Complexity Class NP

- *Definition*: 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:
  - Hamiltonian circuit problem: *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)

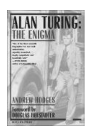
Why NP?

- 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
  - Are any other problems in P also in NP?

More Revelations About NP

- Are any other problems in P also in NP?
  - YES! *All* problems in P are also in NP
    - Notation: $P \subseteq NP$
    - If you can solve a problem in polynomial time, can definitely verify a solution in polynomial time
- Question: Are all problems in NP also in P?
  - Is $NP \subseteq P$?

Your Chance to Win a Turing Award: P = NP?



Alan Turing (1912-1954)

- Nobody knows whether $NP \subseteq P$
  - Proving or disproving this will bring you instant fame!
- It is generally believed that $P \neq NP$, *i.e.* there are problems in NP that are not in P
  - But no one has been able to show even one such problem!
  - Practically all of modern complexity theory is premised on the *assumption* that $P \neq NP$
- A very large number of useful problems are in NP

NP-Complete Problems

- The "hardest" problems in NP are called *NP-complete problems (NPC)*
- Why "hardest"? A problem X is NP-complete *iff*:
  1. X is in NP and
  2. *Any* problem Y in NP can be *converted to an instance of* X in polynomial time, such that *solving X also provides a solution for Y*

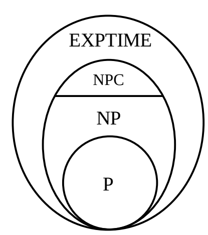
  In other words: Can use algorithm for X as a *subroutine* to solve Y
- Thus, if you find a poly time algorithm for just one NPC problem, all problems in NP can be solved in poly time
  - Example: The Hamiltonian circuit problem can be shown to be NP-complete (not so easy to prove!)

Searching Really Big Graphs

- Any kind of search (DFS, BFS, A*) is polynomial in the size of the graph (number of vertices)
- But a search problem *might* be NP-complete in terms of a *small description* of a *very large graph*
- *Example: Blocks World* 
  - O(|V|) to find a shortest path between any two vertices
  - But if given only the initial and final states (size of these descriptions is ≈ number of blocks), problem is NP-complete

P, NP, and Exponential Time Problems

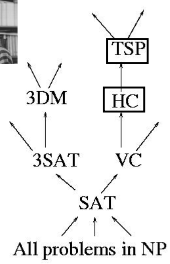
- All *currently known* algorithms for NP-complete problems run in *exponential* worst case time
  - *Finding a polynomial time algorithm for any NPC problem would mean:*
- Diagram depicts relationship between P, NP, and EXPTIME (class of problems that *provably require* exponential time to solve)



It is believed that
$P \neq NP \neq EXPTIME$

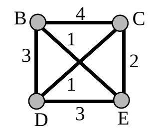
The Graph of NP-Completeness

- Stephen Cook first showed (1971) that satisfiability of Boolean formulas (SAT) is NP-complete
- Hundreds of other problems (from scheduling and databases to optimization theory) have since been shown to be NPC
- How? By showing an algorithm that converts a known NPC problem to your pet problem in poly time → then, your problem is also NPC!

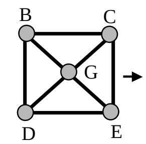


Showing NP-completeness: An example

- Consider the **Traveling Salesperson (TSP) Problem**: Given a *fully connected, weighted* graph G = (V,E), is there a cycle that visits all vertices exactly once and has total cost ≤ K?
- TSP is in NP (why?)
- Can we show TSP is NP-complete?
  - Hamiltonian Circuit (HC) is NPC
  - Can show TSP is also NPC if we can convert any input for HC to an input for TSP in poly time



Cycle with cost ≤ 8 → BDCEB

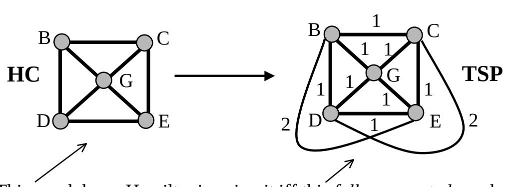


Convert to input for TSP

Input for HC

TSP is NP-complete!

- We can show TSP is also NPC if we can convert any input for HC to an input for TSP in polynomial time. Here's one way:



This graph has a Hamiltonian circuit iff this *fully-connected* graph has a TSP cycle of total cost ≤ K, where K = |*V*| (here, K = 5)

Coping with NP-Completeness

1. *Settle for algorithms that are fast on average:* Worst case still takes exponential time, but doesn't occur very often. *But <u>some</u> NP-Complete problems are also average-time NP-Complete!*
2. *Settle for fast algorithms that give near-optimal solutions:* In TSP, may not give the cheapest tour, but maybe good enough. *But finding even approximate solutions to <u>some</u> NP-Complete problems is NP-Complete!*
3. *Just get the exponent as low as possible!* Much work on exponential algorithms for Boolean satisfiability: in practice can usually solve problem with 1,000+ variables
   - Hot Application: Microprocessor Design Verification

Calendar

- Coming Up – Specialized Data Structures
  - Search Trees for Spatial Data (Class notes)
  - Binomial Queues (Ch 6.8)
  - Randomized Data Structures (Ch 10.4.2, 12.5)
  - Huffman Codes (10.1.2)
- Friday, March 8th – Practice homework
  - Not to be turned in – a solution set will be handed out on the last day of class
  - Doing this assignment will be a very good way to prepare for the midterm!
- Homework #7 (Mazes) due Wednesday, March 13th
  - NO late assignments accepted after Friday, March 15th – we mean it!
- Friday, March 15th – Last day of class – party – demos – celebration
- Monday, March 18th, 2:30 – 4:20 pm – Final Exam