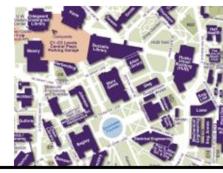
CSE 332:
NP Completeness

Richard Anderson, Steve Seitz

Winter 2014

Announcements

- Review session for the final
  - Saturday, noon, EEB 125
- Final
  - Monday, March 17, Johnson 102
    - 4:30 PM or 6:30 PM
    - 110 minutes, closed book



Your First Task

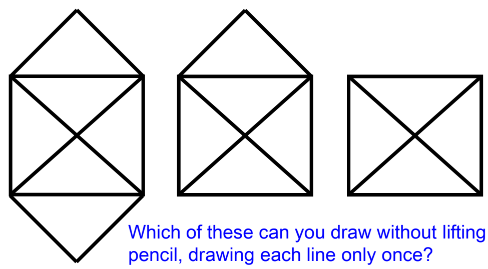
Your company has to inspect a set of roads between cities by driving over each of them.

Driving over the roads costs money (fuel), and there are a lot of roads.

Your boss wants you to figure out how to drive over each road exactly once.

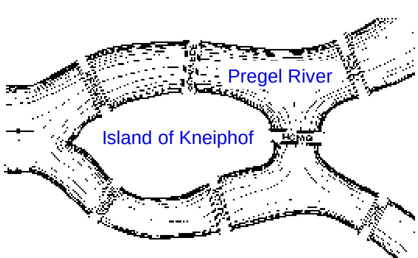
You get a bonus if, after inspecting the last road, the car is back where it started.

Try it with paper and pencil



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.

Euler Circuits and Tours

$e^{i\pi} = -1$

- 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 exists *iff* the graph is connected and each vertex has even degree (= # of edges on the vertex)
  - An Euler tour exists *iff* the graph is connected and either all vertices have even degree or exactly two have odd degree

Finding Euler Circuits

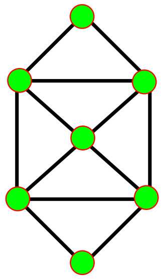
Given a connected, undirected graph G = (V,E), find an Euler circuit in G.

Euler Circuit Existence Algorithm:
Check to see that all vertices have even degree

Running time =

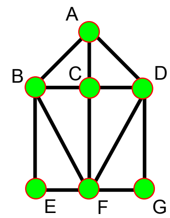
Euler Circuit Algorithm:
1. Do an edge walk from a start vertex until you are back at the start vertex. Mark each edge you visit, and do not revisit marked edges. You never get stuck because of the even degree property.
2. The walk is removed leaving several components each with the even degree property. Recursively find Euler circuits for these.
3. Splice all these circuits into an Euler circuit

Running time =



Euler Tour

For an Euler Tour, exactly two vertices are odd, the rest even. Using a similar algorithm, you can find a path between the two odd vertices.



Your Second Task

Your boss is pleased…and assigns you a new task.

Your company has to send someone by car to a set of cities.

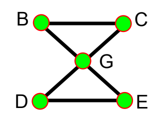
The primary cost is the exorbitant toll going into each city.

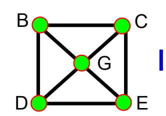
Your boss wants you to figure out how to drive to each city exactly once, returning in the end to the city of origin.

Hamiltonian Circuits

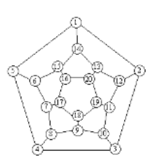
W. R. Hamilton (1805-1865)

- Euler circuit: A cycle that goes through each *edge* exactly once
- Hamiltonian circuit: A cycle that goes through each *vertex* exactly once (except the first=last one)
- Does graph **I** have:
  - An Euler circuit?
  - A Hamiltonian circuit?
- Does graph **II** have:
  - An Euler circuit?
  - A Hamiltonian circuit?
- Does the Hamiltonian circuit problem seem easier or harder?





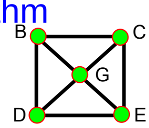
Hamilton's Icosian Game



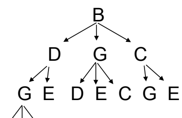
Finding Hamiltonian Circuits

- Problem: Find a Hamiltonian circuit in a connected, undirected graph G.
- 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*



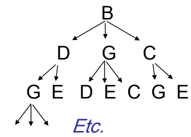
Etc. *Search tree* of paths from B

Analysis of our Exhaustive Search Algorithm

- Let the average branching factor of each node in this tree be b
- |V| vertices, each with ≈ b branches
- Total number of paths ≈ b·b·b … ·b

  =

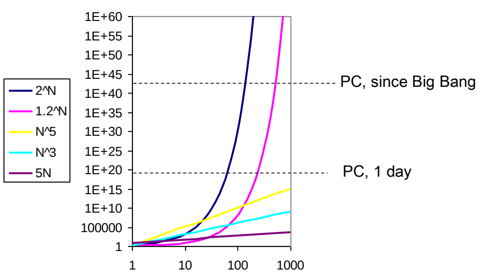
- Worst case → Exponential time!



Etc.

Search tree of paths from B

Exponential Time



Polynomial vs. 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

The Complexity Class P

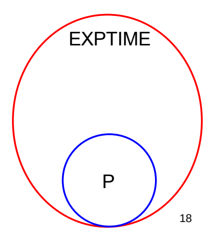
- The set P is defined as the set of all problems that can be solved in *polynomial worst case time*
  - Also known as the *polynomial time* complexity class
  - All problems that have some algorithm whose running time is $O(N^k)$ for some *k*
- Examples of problems in P: sorting, shortest path, Euler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

Problem Spaces

If a problem is not polynomial-time solvable (not in P), then it is an exponential-time problem.

Shorthand:

- P solutions are fast
- EXPTIME are slow
  - Sometimes viewed as “intractable”



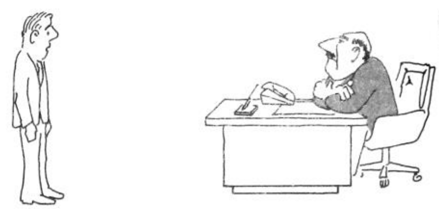
Saving your job

Try as you might, every solution you come up with for the Hamiltonian Circuit problem runs in exponential time.

You have to report back to your boss.

19

What you'd rather not say…



"I can't find an efficient algorithm, I guess I'm just too dumb."

20

What you'd like to say…

"I can't find an efficient algorithm, because no such algorithm is possible!"

But can you actually say this…?

21

When is a problem easy?

- We've seen some "easy" graph problems:
  - Graph search
  - Shortest-path
  - Minimum Spanning Tree
- Not easy for us to come up with, but easy for the computer, once we know algorithm.

22

When is a problem hard?

- Almost everything we've seen in class has had an efficient algorithm
- But of course, computers can't solve *every* problem quickly.
- In fact, there are perfectly reasonable sounding problems that no computer could ever solve in any reasonable amount of time (as far as we know!).
  - The Hamiltonian Circuit problem is one of these (as far as we know). More later on just how hard it is…

23

What you *can* say…

"I can't find an efficient algorithm, but neither can all these famous people."

24

When is a problem hopeless?

- Some problems are "undecideable" – no algorithm can be given for solving them.
  - The Halting Problem: is it possible to specify any algorithm, which, given an arbitrary program and input to that program, will always correctly determine whether or not that program will enter an infinite loop?
  - No! [Turing, 1936]
- We'll focus on problems that have a glimmer of hope…

Alan Turing (1912-1954)

25

A Glimmer of Hope

Suppose you have a problem that is at least decideable.

If the output can be checked for correctness in polynomial-time, then ***maybe*** a polynomial-time solution exists!

26

The Complexity Class NP

- *Definition*: NP is the set of all problems for which a given *candidate solution* can be *tested* in polynomial time
- Are the following in NP:
  - Hamiltonian circuit problem?
  - Euler circuit problem?
  - All polynomial time algorithms?

27

Why NP?

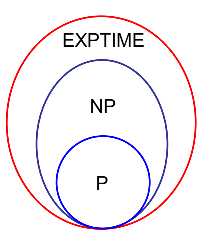
- NP stands for *Nondeterministic Polynomial time*
  - Why "nondeterministic"? Corresponds to algorithms that can guess a solution (if it exists) → the solution is then verified to be correct in polynomial time
  - Nondeterministic algorithms don't exist – purely theoretical idea invented to understand how hard a problem could be

28

Problem Spaces (revisited)

We can now augment our problem space to include NP as a superset of P.

Whenever someone finds a polynomial time solution to a problem currently believed to be in NP - P, it moves to P.



29

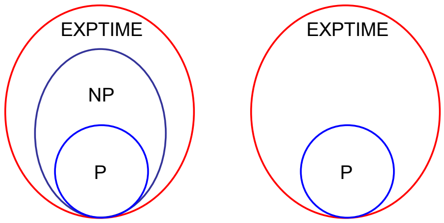
Your Chance to Win a Turing Award (and $$$)

- It is generally believed that P ≠ 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!
  - This is the fundamental open problem in theoretical computer science.
  - Nearly everyone has given up trying to prove it. Instead, theoreticians prove theorems about what follows once we assume P ≠ NP !

30

P = NP?

Perhaps instead P = NP, but that would seem to be even harder to prove…



P ≠ NP   P = NP

What you'd like to say (revisited)…

"I can't find an efficient algorithm, because no such algorithm is possible. And that means P ≠ NP, and now I'm super famous and rich, and I quit!"

Your Third Task

Your boss buys your story that others couldn't solve the last problem.

Again, your company has to send someone by car to a set of cities.

The primary cost is distance traveled (which translates to fuel costs).

Your boss wants you to figure out how to drive to each city exactly once, then returning to the first city, while staying within a fixed mileage budget C.

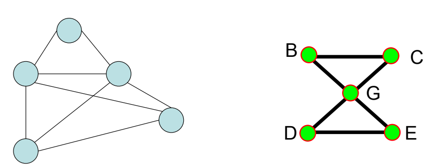
The Traveling Salesman Problem (TSP)

- This amounts to solving…
  …The Traveling Salesman Problem:
  - Given a complete (fully connected) weighted graph G, and an integer C,
  - is there a cycle that visits all vertices with cost ≤ C?
- One of the canonical problems in computer science.
- Note difference from Hamiltonian cycle: graph is complete, and we care about weight.

Transforming HC into TSP

- We can transform Hamiltonian Cycle into TSP.
- Given graph G=(V, E):
  - Assign weight of 1 to each edge
  - Augment the graph with edges until it is a complete graph G'=(V, E').
  - Assign weight of 2 to the new edges.
  - Let C = |V|.

Examples



Polynomial-time transformation

- G' has a TSP tour of weight |V| iff (if and only if) G has a Hamiltonian Cycle.
  - Proof: "obvious"
- What was the cost of transforming HC into TSP?
- In the end, because there is a polynomial-time transformation from HC to TSP, we say *TSP is "at least as hard as" Hamiltonian cycle.*

37

What you can say…

"I can't find an efficient algorithm, since all these famous people couldn't solve HC efficiently and I can prove that TSP is at least as hard."

38

Satisfiability

In 1971, Stephen Cook studied the Satisfiability Problem:

- Given a Boolean formula (collections of ANDs, ORs, NOTs) over a set of variables,
- is there a TRUE/FALSE assignment to the variables such that the Boolean formula evaluates to TRUE?

39

Satisfiability

…and he proved something remarkable:

Every problem in NP can be polynomially transformed to the Satisfiability Problem.

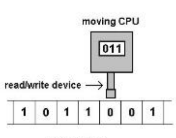
Thus, Satisfiability is at least as hard as every other problem in NP, i.e., it is the hardest NP problem.

We call it an "NP-complete" problem.

40

Turing Machines

The proof is intricate and depends on a computing abstraction called a Turing Machine…



which is generalized to allow guessing the answer.

41

The Steam Powered Turing Machine

In the mid-1980s, Alan Borning attributed a peculiar implementation of a Turing Machine to Larry Ruzzo, giving rise to this famous mural:

42

NP-completeness

- In fact, Satisfiability can be polynomially reduced to some other NP problems (and vice versa).
- These other problems are equivalent to Satisfiability, and so all other problems in NP can be transformed to them, as well.
- NP-complete problems thus form an equivalence set in NP (all equivalently hard, i.e., the hardest).
- Solving one would give a solution for all of them!
  - If any NP-complete problem is in P, then all of NP is in P

43

What's NP-complete

- Satisfiability of logic formulas
- All sorts of constraint problems
- All sorts of graph problems, including:
  - Hamiltonian Circuits
  - Traveling Salesman?
  - Graph coloring: decide if the vertices of a graph be colored using K colors, such that no two adjacent vertices have the same color.
- Not an overstatement to say that every area of computer science comes up against NP-complete problems.

44

One tweak and you could be in NP-complete

- It's amazing how "little" tweaks on a problem change the complexity:
  - Euler circuit is in P, but Hamiltonian circuit is NP-complete.
  - Shortest path between two points is computable in $O(|V|^2)$, but longest path is NP-complete.

45

Analyzing Your Hard Problem

- Your problem seems really hard.
- If you can transform an NP-complete problem into the one you're trying to solve, then you can stop working on your problem!
- …unless you really need that Turing award.

46

What you'd like to say (re-revisited)…

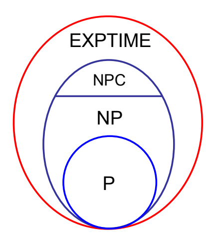
"I showed that TSP is NP-complete, and I found an efficient algorithm for it, and that means P = NP, and I've totally rocked the CS world!"

(Not likely!)

47

P, NP, NPC, and Exponential Time Problems

- All **currently known** algorithms for NP-complete problems run in **exponential** worst case time
- Diagram depicts relationship between P, NP, and EXPTIME (class of problems that **provably require** exponential time to solve)



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

48

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 some NP-Complete problems are also average-time NP-Complete!
2. Settle for fast algorithms that give near-optimal solutions: In traveling salesman, may not give the cheapest tour, but maybe good enough.
   But finding even approximate solutions to some NP-Complete problems are NP-Complete!

Coping with NP-Completeness

3. Just get the exponent as low as possible! Much work on exponential algorithms for satisfiability: in practice can often solve circuits with 1,000+ inputs
   But even $2^{n/100}$ will eventual hit the exponential curve!
4. Restrict the problem: Longest Path is easy on trees, for example. Many hard problems are easy for restricted inputs.

Great Quick Reference

Is this lecture complete? Hardly, but here's a good reference:

Computers and Intractability: A Guide to the Theory of NP-Completeness
by Michael S. Garey and David S. Johnson