Beauty and Joy of Computing

Limits of Computing

Ivona Bezáková

CS10: UC Berkeley, April 14, 2014

(Slides inspired by Dan Garcia's slides.)

Computer Science Research Areas

- Artificial Intelligence
- Biosystems & Computational Biolo
- Database Management Systems
- Graphics
- Human-Computer Interaction
- Networking
- Programming Systems
- Scientific Computing
- Security
- Systems
  - **Theory**
    - **Complexity theory**
    - …
  - …

[www.eecs.berkeley.edu/Research/Areas/]

Revisiting Algorithm Complexity

A variety of problems that:

- are tractable with efficient solutions in reasonable time
- are intractable
- are solvable approximately, not optimally
- have no known efficient solution
- are not solvable

Revisiting Algorithm Complexity

Recall:

- **running time** of an algorithm: how many steps does the algorithm take as a function of the size of the input

- various **orders of growth**, for example:

    - constant
    - logarithmic
    - linear
    - quadratic
    - cubic
    - exponential

Examples ?

Revisiting Algorithm Complexity

Recall:

- **running time** of an algorithm: how many steps does the algorithm take as a function of the size of the input

- various **orders of growth**, for example:

  - constant
  - logarithmic
  - linear
  - quadratic
  - cubic
  - exponential

(constant through cubic:) **Efficient**: order of growth is polynomial

Such problems are said to be "in **P**" (for polynomial)

Intractable Problems

- Can be solved, but not fast enough; for example
  - exponential running time
  - also, when the running time is polynomial with a huge exponent (e.g., $f(n) = n^{10}$)
  - in such cases, can solve only for small n…

Hamiltonian Cycle

Input: cities with road connections between some pairs of cities

Output: possible to go through all such cities (every city exactly once) ?

Notice: YES/NO problem
(such problems are called **decision problems**)

Hamiltonian Cycle

Input: cities with road connections between some pairs of cities

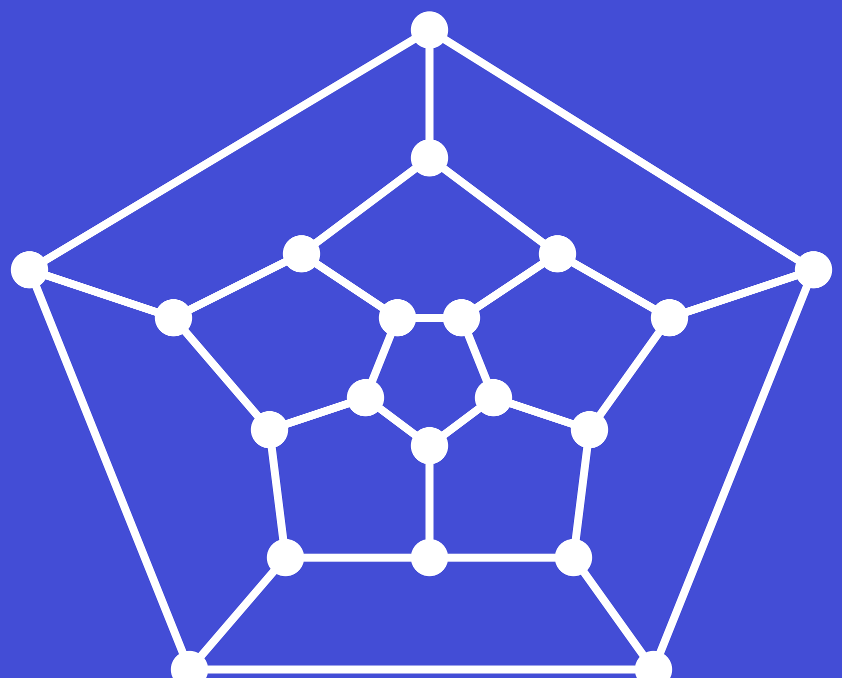
Output: possible to go through all such cities (every city exactly once) ?

PEER INSTRUCTION:

For this input, is there a Hamiltonian cycle ?

(a) Answer YES

(b) Answer NO

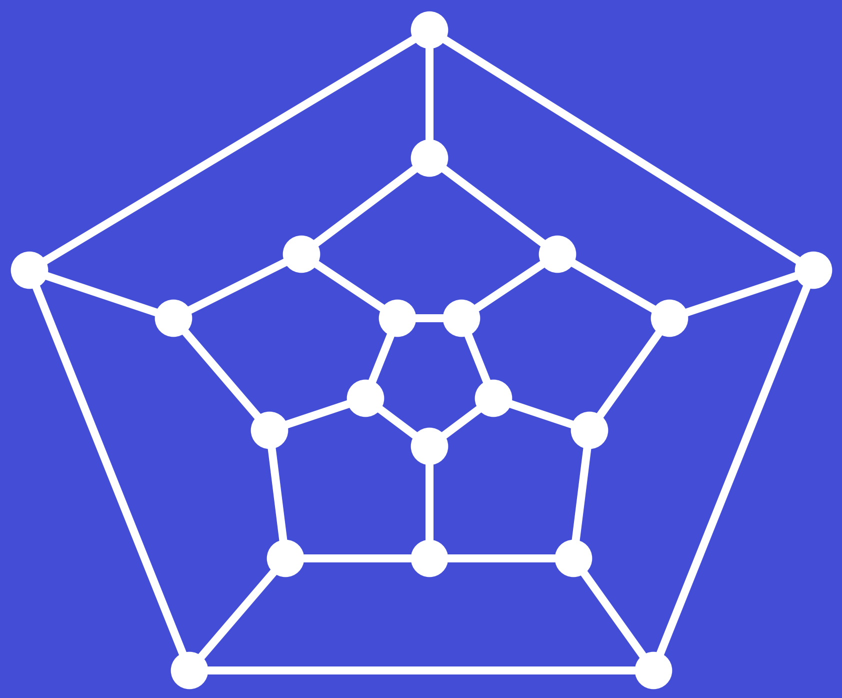


Hamiltonian Cycle

Input: cities with road connections between some pairs of cities

Output: possible to go through all such cities (every city exactly once) ?

What did you do to solve the problem ?



Traveling Salesman Problem

Input: cities with road connections between pairs of cities, roads have lengths

Output: find a route that goes through all the cities, returns to the origin, and minimizes the overall traveled length

PEER INSTRUCTION:

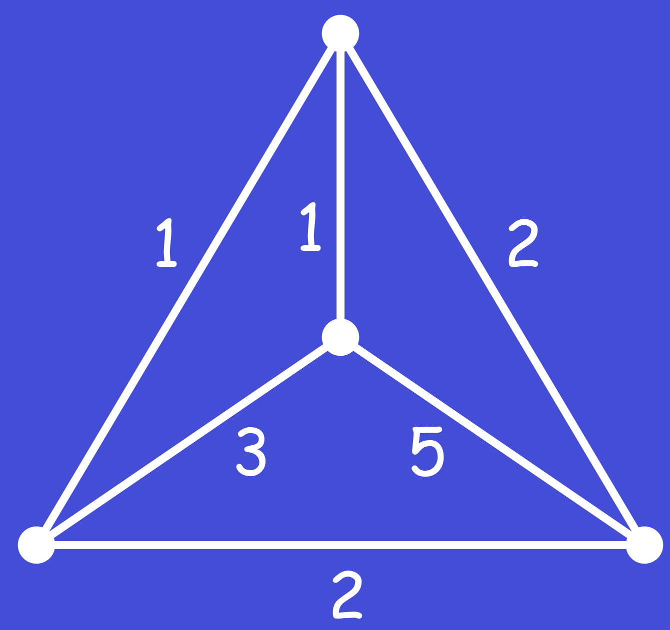
For this input, what is the shortest possible length ?

(a) total length 7

(b) total length 8

(c) total length 9

(d) total length 10



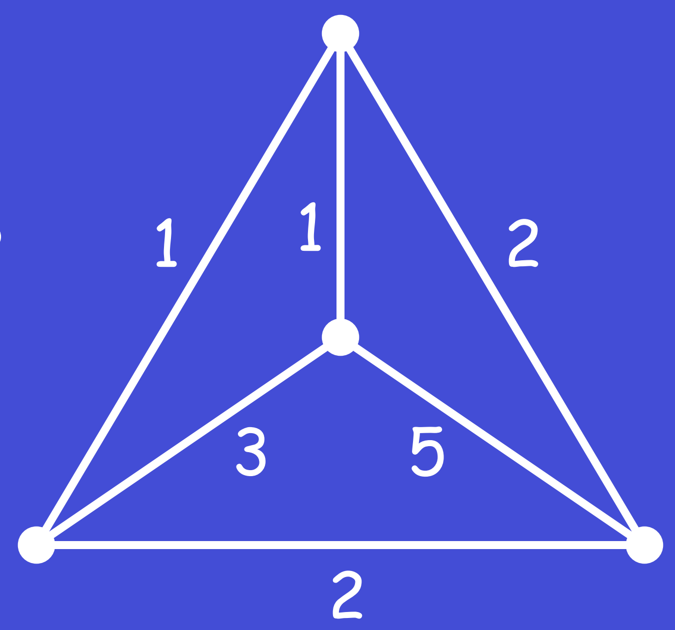
Traveling Salesman Problem

Input: cities with road connections between pairs of cities, roads have lengths

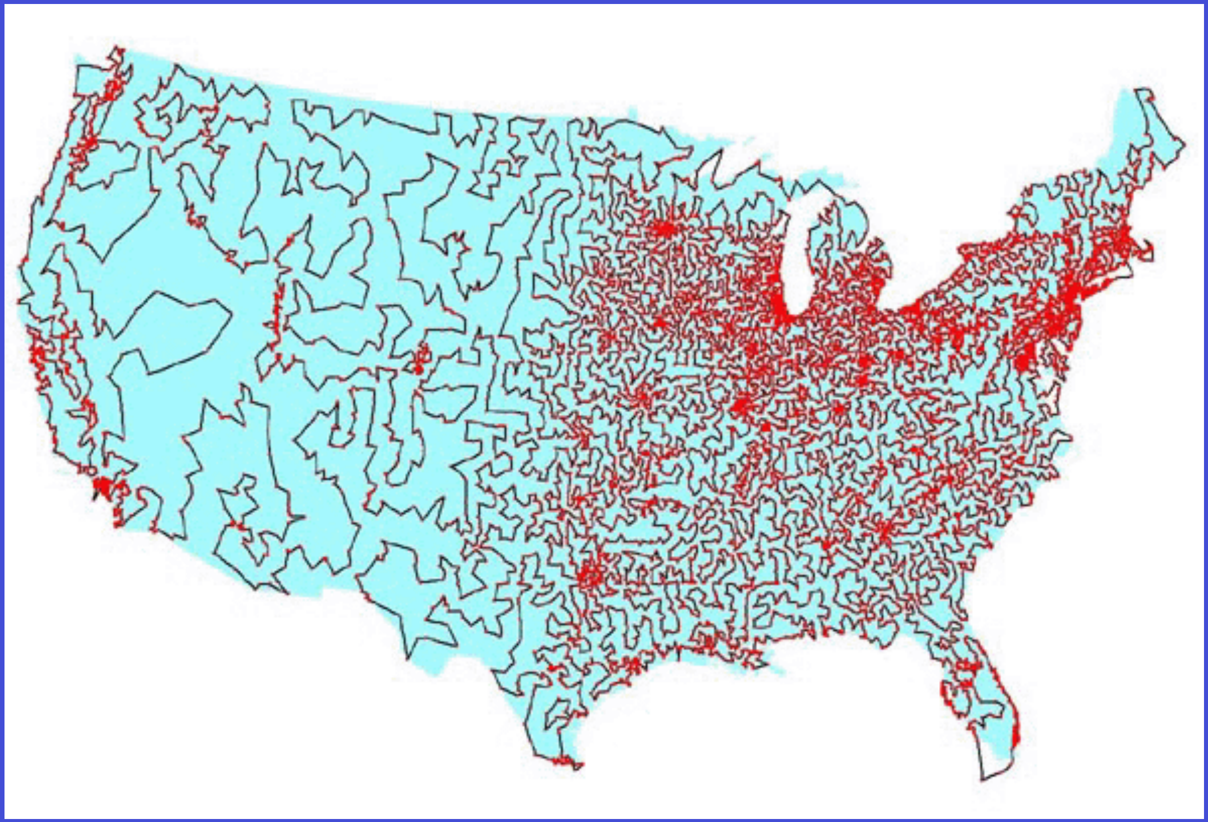
Output: find a route that goes through all the cities, returns to the origin, and minimizes the overall traveled length

Not a decision problem…

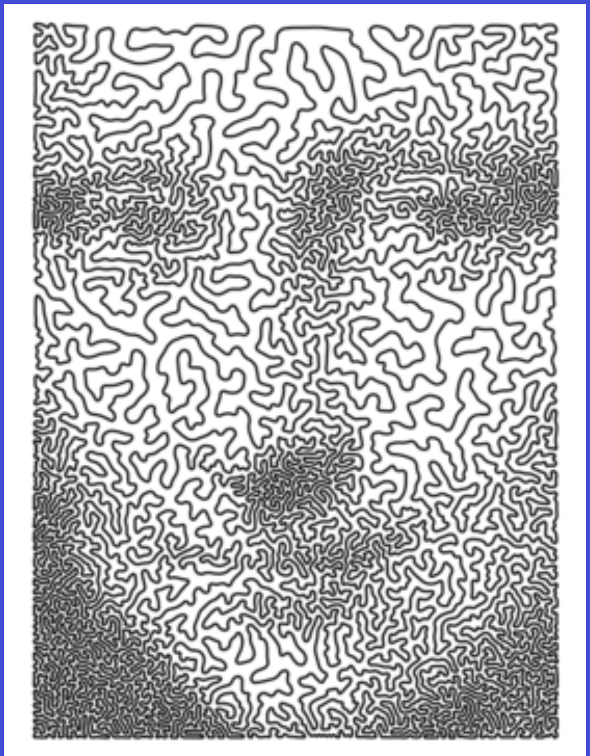
But we can ask:
Is there a route shorter than x ?



Traveling Salesman Problem



David Applegate, Robert Bixby, Vašek Chvátal, William Cook



Bob Bosch (TSP Art)

Hamiltonian Cycle vs Traveling Salesman

- suppose we have a magic device that solves the Traveling Salesman Problem

- can we use it to solve the Hamiltonian Cycle ?

This is called a **reduction**. Find a solution to one problem, then all others that reduce to it can be solved!

P vs NP

Recall: P = problems with polynomial-time algorithms

We do not know how to solve Hamiltonian Cycle or Traveling Salesman in polynomial time! (No efficient solution known.)

But…

If we "**guess**" a permutation of the cities, we can easily **verify** whether they form a cycle of length shorter than x.

NP = problems whose solutions can be efficiently verified

(N stands for non-deterministic [guessing]; P is for polynomial)

P vs NP

P = problems with polynomial-time algorithms

NP = problems whose solutions can be efficiently verified

The BIG OPEN PROBLEM in CS: **Is P = NP ???**

$1,000,000 reward

http://www.claymath.org/millennium-problems

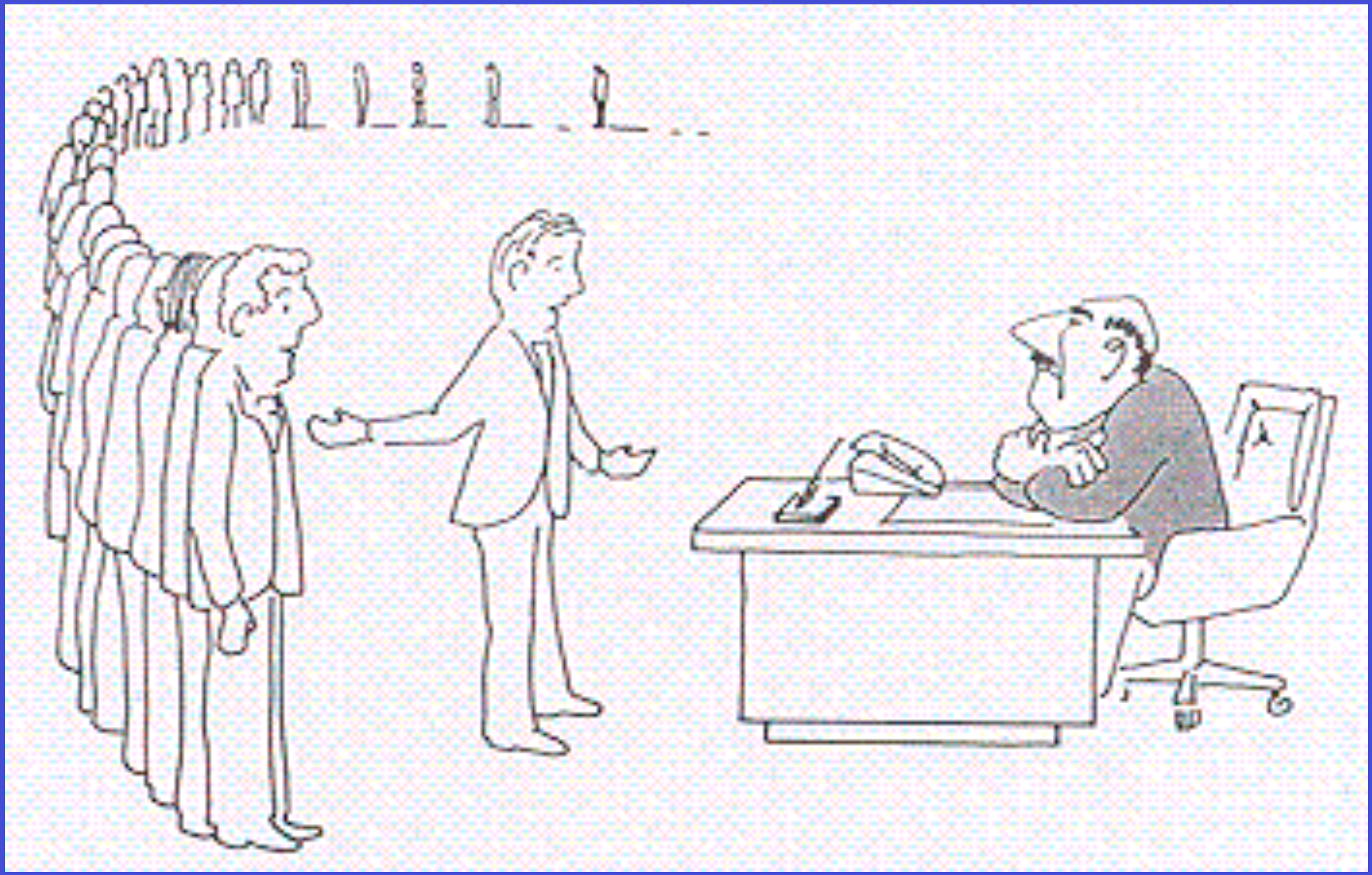
A problem is **NP-hard** if all problems in NP reduce to it.

I.e., efficiently solving an NP-hard problem gives efficient algorithms for all problems in NP!

An NP-hard problem is **NP-complete** if it is in NP.

Examples: Hamiltonian Cycle, Traveling Salesman Problem, …

NP-complete problem: what to do ?



What to tell your boss if they ask you to solve an NP-complete problem:
“I can’t find an efficient solution but neither can all these famous people.”

http://max.cs.kzoo.edu/~kschultz/CS510/ClassPresentations/NPCartoons.html

NP-complete problem: what to do ?

Another option: **approximate** the solution

- Seems unlikely to solve exactly but sometimes can get "close" to the optimum

- For example, traveling salesman:

    - If the input is a metric (satisfies the triangle inequality), then we can efficiently find a solution that is not worse than 1.5x optimum

Beyond NP: Unsolvable problems

Are there problems that, no matter how much time we use, we cannot solve ?

Some terminology:

- Decision problems: YES/NO answer
- Algorithm is a **solution** if it produces the correct answer in a finite amount of time
- Problem is **decidable** if it has a solution

Alan Turing

proved that not all problems are decidable!

Review: Proof by Contradiction

How many primes are there ?

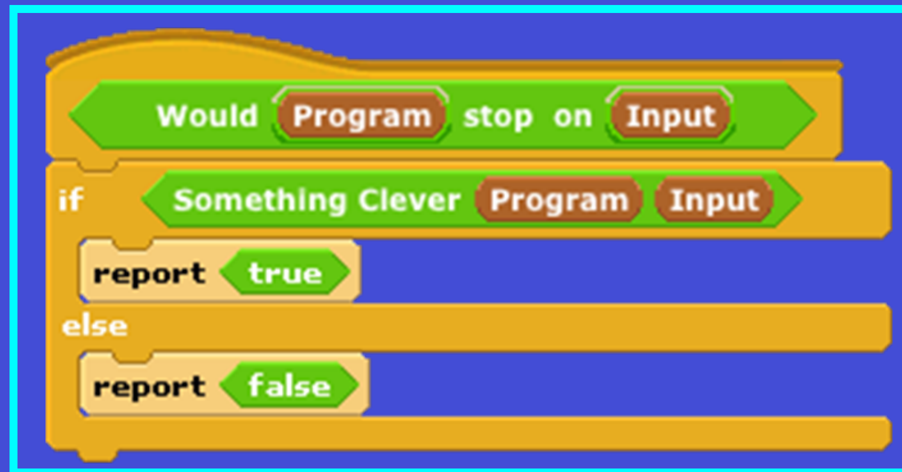
Euclid

www.hisschemoller.com/wp-content/uploads/2011/01/euclides.jpg

Beyond NP: The Halting Problem

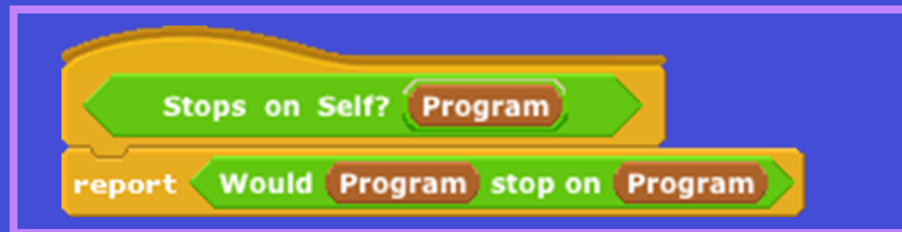
Input: a program and its input

Output: does the program eventually stop ?



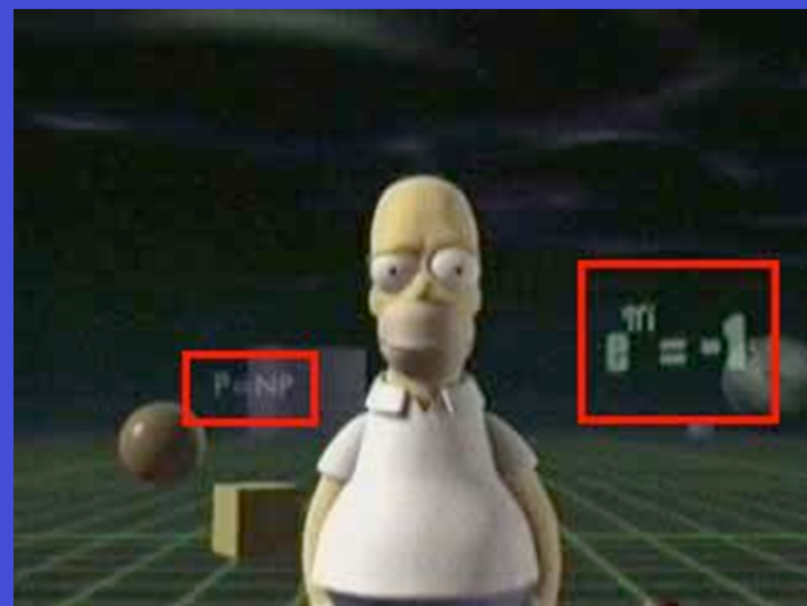
Turing's proof, by contradiction:

- Suppose somebody can solve it
- Write Stops on Self
- Write Weird
- Call Weird on itself…



Conclusions

- Complexity theory: important part of CS
- If given an important problem, rather than try to solve it yourself, see if others have tried similar problems
- If you do not need an exact solution, approximation algorithms might help
- Some problems are not solvable!



http://www.win.tue.nl/~gwoegi/P-versus-NP.htm

P = NP ?

Pavel Pudlák