Lecture 21: Intractability

Princeton CS Building West Wall, Circa 2001

Overview

What is an algorithm? Turing machine.

Which problems can be solved on a computer? Computability.

Which ALGORITHMS will be useful in practice? Analysis of algorithms.

Which PROBLEMS can be solved in practice? Intractability.

Properties of Algorithms

Q. Which ALGORITHMS are useful in practice?

A working definition: (Cobham 1960, Edmonds, 1962)

- Measure running time as a function of input size N.
- Efficient = polynomial time for all inputs.
- Inefficient = "exponential time" for some inputs.

Ex: Dynamic programming algorithm for edit distance takes N^2 steps.
Ex: brute force algorithm for TSP takes N! steps.

Theory: definition is broad and robust; huge gulf between polynomial and exponential algorithms.

Practice: exponents and constants of polynomials that arise are small $\Rightarrow$ scales to huge problems.

Exponential Growth

Exponential growth dwarfs technological change.

- Suppose each electron in the universe had power of today's supercomputers . . .
- And each works for the life of the universe in an effort to solve TSP problem via brute force.

Quantity	Number
Supercomputer instructions per second	10^{13}
Age of universe in seconds †	10^{17}
Electrons in universe †	10^{79}

† Estimated

- Will not help solve 1,000 city TSP problem with brute force.

$1000! \gg 10^{1000} \gg 10^{79} \times 10^{17} \times 10^{13}$

P

Definition of P: Set of all yes-no problems solvable in polynomial time on a deterministic Turing machine.

Problem	Description	Algorithm	Yes	No
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
COMPOSITE	Does x have a factor other than 1 and itself?	Agarwal-Kayal-Saxena (2002)	51	53
EDIT-DISTANCE	Is the edit distance between strings x and y less than 5?	Dynamic Programming	niether neither	acgggt ttttta
BACON	Is the Kevin Bacon number of actor x less than 5?	Breadth First Search	Julia Roberts	Akbar Abdi
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Remark. Algorithm typically also solves the related search problem.

Extended Church-Turing Thesis

Extended Church-Turing thesis:
- P = yes-no problems solvable in poly-time time on real computers.
- If computable by a piece of hardware in time T(N) for input of size N, then computable by TM in time $(T(N))^k$ for some constant k.

Evidence supporting thesis:
- True for all physical computers.
- k = 2 for random access machines.

Implication: to make future computers more efficient, only need to focus on improving implementation of existing designs.

Possible counterexample: quantum computers.
- Shor's factoring algorithm is poly-time on quantum computer.
- No poly-time algorithm known for classical computers.

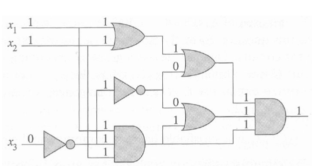
Properties of Problems

Which PROBLEMS won't we be able to solve in practice?
- No easy answers, but theory helps.

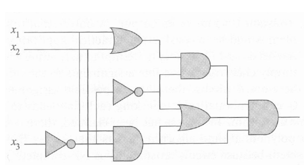
Two hard problems.
- Factorization: Given an integer, find its prime factorization.

$4901 = 13^2 \times 29$

- CIRCUIT-SAT: Is there a way to assign inputs to a given combinational circuit that makes its output true?



yes



no

Reference: CLRS

More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.
Biology: protein folding.
Chemical engineering: heat exchanger network synthesis.
Civil engineering: equilibrium of urban traffic flow.
Economics: computation of arbitrage in financial markets with friction.
Environmental engineering: optimal placement of contaminant sensors.
Financial engineering: find minimum risk portfolio of given return.
Genomics: phylogeny reconstruction.
Electrical engineering: VLSI layout.
Mechanical engineering: structure of turbulence in sheared flows.
Medicine: reconstructing 3-D shape from biplane angiocardiogram.
Operations research: optimal resource allocation.
Physics: partition function of 3-D Ising model in statistical mechanics.
Politics: Shapley-Shubik voting power.
Pop culture: Minesweeper consistency, playing optimal Tetris.
Statistics: optimal experimental design.

Q. Why do we believe these problems intrinsically hard to solve in practice?

Reduction

Reduction. Problem X reduces to problem Y if given an efficient subroutine for Y, you can devise an efficient algorithm for X.

- Cost of solving X $\le$ cost of solving Y + cost of reduction.
- May call subroutine for Y more than once.

Consequences:

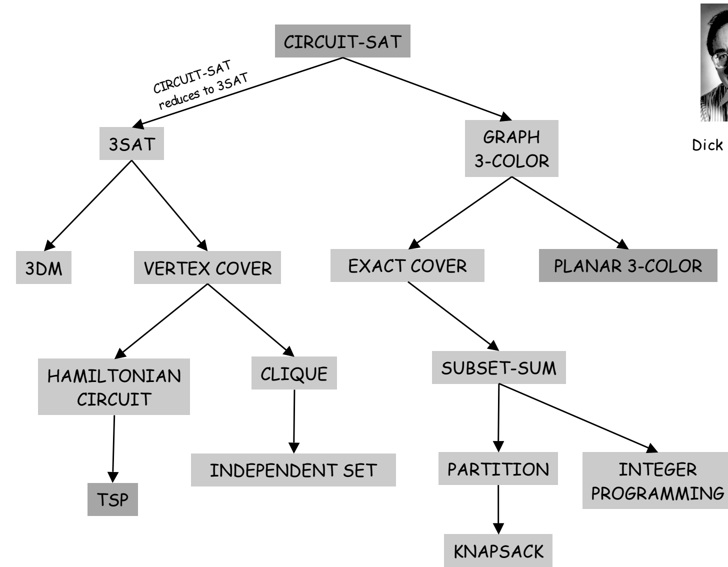
- Classify problems: establish relative difficulty between two problems.
- Design algorithms: given algorithm for Y, can also solve X.
- Establish intractability: if X is hard, then so is Y.

Example.

- X = COMPOSITE.
- Y = Factorization.

```
static boolean isComposite(int x) {
    int[] factors = factorize(x);
    return (factors.length > 1);
}
```

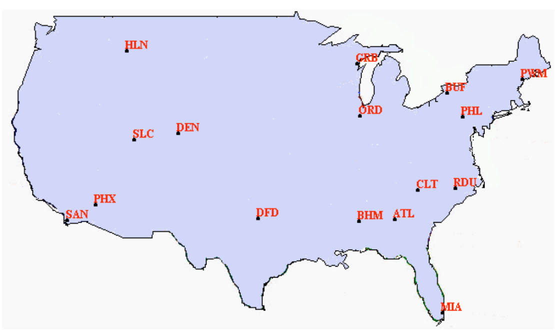

More Reductions



Dick Karp (1972)

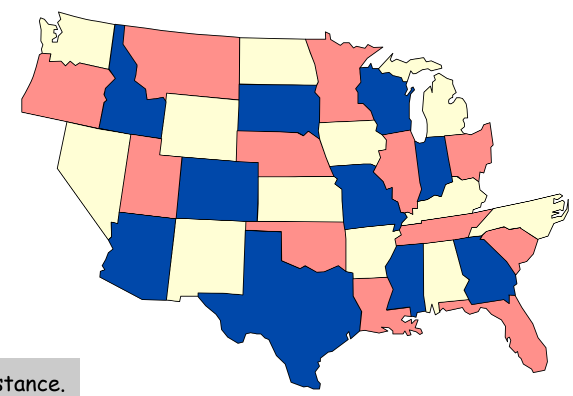
Some Hard Problems

TSP. What is the shortest tour that visits all N cities?



Some Hard Problems

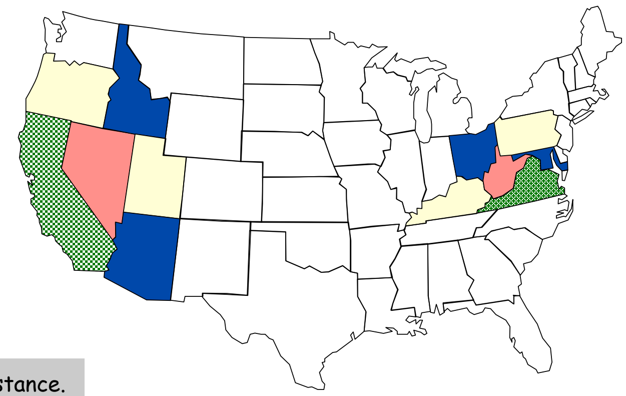
PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?



YES instance.

Some Hard Problems

PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

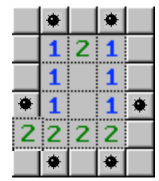


NO instance.

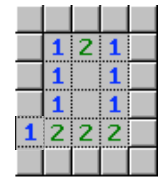
Minesweeper Consistency Problem

Minesweeper.
- Start: Blank grid of squares, some conceal mines.
- Goal: Find location of all mines without detonating any.
- Repeatedly choose a square.
  - if mine underneath, it detonates and you lose
  - otherwise, computer tells you # neighboring mines

MINESWEEPER. Given a state of what purports to be a N-by-N Minesweeper game, is it logically consistent?



yes



no

Reduction

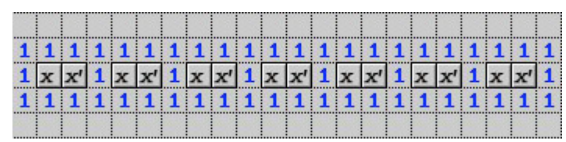
Reduction. Problem X reduces to problem Y if given an efficient subroutine for Y, you can devise an efficient algorithm for X.

Consequences:
- Classify problems: establish relative difficulty between two problems.
- Design algorithms: given algorithm for Y, can also solve X.
- Establish intractability: if X is hard, then so is Y.

Example.
- X = CIRCUIT-SAT.
- Y = MINESWEEPER.

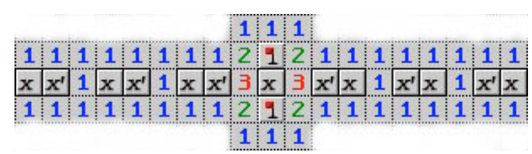
Minesweeper Consistency Problem

SAT reduces to MINESWEEPER.
- Build circuit by laying out appropriate minesweeper configurations.
- Minesweeper game is consistent if and only if circuit is satisfiable.



A Minesweeper Wire

Exactly one of x and x' is a bomb.

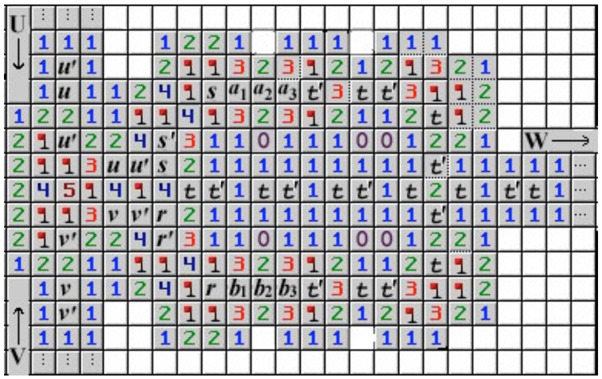


A Minesweeper NOT Gate

Minesweeper Consistency Problem

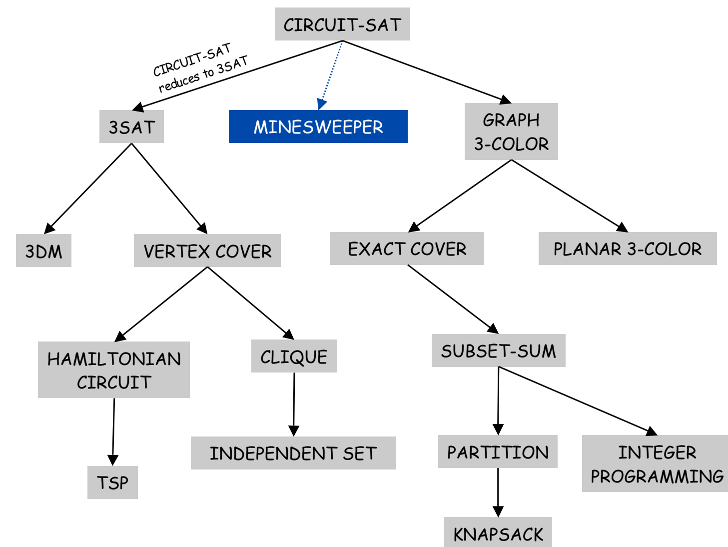
SAT reduces to MINESWEEPER.

- Build circuit by laying out appropriate minesweeper configurations.
- Minesweeper game is consistent if and only if circuit is satisfiable.



A Minesweeper AND Gate

One More Hard Problem



Nondeterminism

Nondeterministic machine. One that guesses the right answer, and only needs to check that the guessed answer is correct.

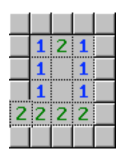
Ex 1: COMPOSITE.

- Guess a divisor d of x.
- Check that d divides x.

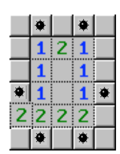
Input: x = 437,669
Guess: d = 809

Ex 2: MINESWEEPER.

- Guess an assignment of mines to squares.
- Check that each square adjacent to the required number of mines.



Input



Guess

running time = # steps to check

Observation. Checking seems much easier than solving from scratch.
Q. Is it really?

Complexity Classes

P. Set of yes-no problems solvable in poly-time on a deterministic TM.
EXP. Same as P, but in exponential-time.
NP. Same as P, but on non-deterministic Turing machine.

Cook-Levin Theorem (1960s). ALL NP problems reduce to SAT.

if we can solve SAT, we can solve any of them

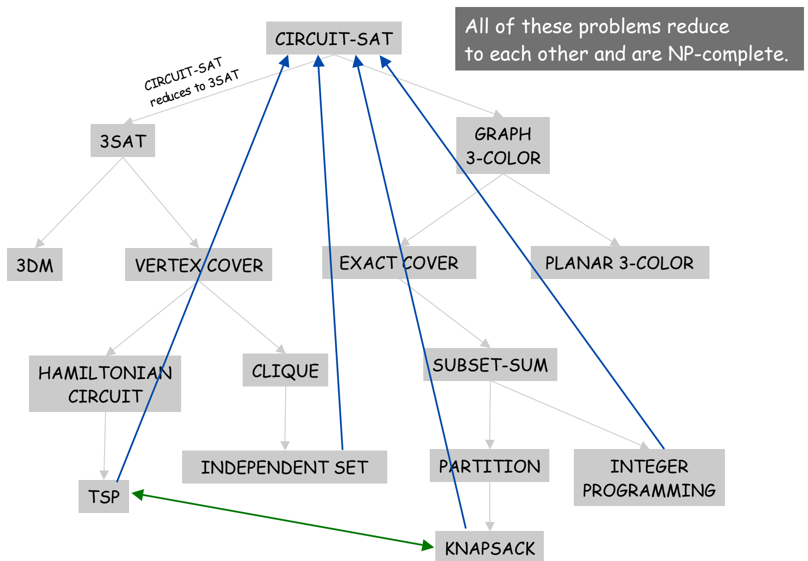
NP-complete. All NP problems to which SAT reduces.

if we can solve any of them, we can solve SAT

Implications.

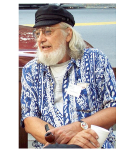
- If efficient algorithm for SAT, then P = NP.
- If efficient algorithm for any NP-complete problem, then P = NP.
- If no efficient algorithm for some NP problem, then none for SAT.

Cook's Theorem: Implications



The Main Question

Does P = NP?
- Is there a poly-time algorithm for SAT?
- Does nondeterminism help you solve problems faster?
- Clay $1 million prize.



Jack Edmonds, 1962

would break modern cryptography and collapse economy

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, . . .
If no: No efficient algorithms possible for 3-COLOR, TSP, . . .

Consensus opinion on P = NP? Probably no.

Implications of NP-Completeness

Classify problems according to their computational requirements.
- NP-complete: SAT, all Karp problems, thousands more.
- P: RELPRIME, COMPOSITE, LSOLVE.
- Unclassified: FACTOR is in NP, but unknown if NP-complete or in P.

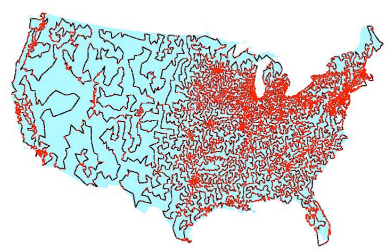
Computational universality.
- All known algorithms for NP-complete problems are exponential.
- If any NP-complete problem proved exponential, so are rest.
- If any NP-complete problem proved polynomial, so are rest.

Proving a problem is NP-complete can guide scientific inquiry.
- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: Istrail proves 3D problem NP-complete.

Coping With Intractability

Relax one of desired features.
- Solve the problem in polynomial time.
- Solve the problem to optimality.
- Solve arbitrary instances of the problem.

Complexity theory deals with worst case behavior.
- Instance(s) you want to solve may be "easy."
- Concorde algorithm solved 13,509 US city TSP problem.



(Cook et. al., 1998)

Coping With Intractability

Relax one of desired features.
- Solve the problem in polynomial time.
- Solve the problem to optimality.
- Solve arbitrary instances of the problem.

Develop a heuristic, and hope it produces a good solution.
- No guarantees on quality of solution.
- Ex: TSP assignment heuristics.
- Ex: Metropolis algorithm, simulating annealing, genetic algorithms.

Design an approximation algorithm.
- Guarantees to find a nearly-optimal solution.
- Ex: Euclidean TSP tour guaranteed to be within 1% of optimal.
- Active area of research, but not always possible!


Sanjeev Arora (1997)

Coping With Intractability

Relax one of desired features.
- Solve the problem in polynomial time.
- Solve the problem to optimality.
- Solve arbitrary instances of the problem.

can do any 2 of 3

Exploit intractability.
- Cryptography. (see next lecture)

Keep trying to prove P = NP.

Summary

Many fundamental problems are NP-complete.
- TSP, CIRCUIT-SAT, 3-COLOR.

Theory says we probably won't be able to design efficient algorithms for NP-complete problems.
- You will run into these problems in your scientific life.
- If you know about NP-completeness, you can identify them and avoid wasting time and energy.