CS 126 Lecture T6: NP-Completeness

Outline

- **Introduction: polynomial vs. exponential time**
- P vs. NP: the holy grail
- NP-Completeness: Cook's Theorem
- NP-Completeness: Reduction
- Conclusions

Where We Are

- T1 - T4:
  - **Computability**: whether a problem is solvable at all
  - Bad news: "most" problems are not solvable!
- T5 - T6:
  - **Complexity**: how long it takes to solve a problem
  - Bad news: many hard problems take so long to solve that they are almost as bad as non-solvable!
- Tuesday:
  - **Examples** of "fast" vs. "slow" algorithms
- Today:
  - **Classes** of problems depending on how "hard" they are

The "Good" vs. the "Bad"

- A given problem can be solved by many different algorithms, but some algorithms are far more efficient than others.

EFFICIENT:

"polynomial" time (ex: N^2) for ALL inputs

INEFFICIENT:

"exponential" time (ex: 2^N) for SOME inputs

“Efficient” vs. “Inefficient” Examples

- Sorting: *O(N*LogN)*
- TSP: *O(N!)*

* TRAVELING SALESPERSON

A salesperson needs to visit N cities.
Is there a route of length less than d?

- Who cares?
  - How long does it take to do TSP(1000)?
  - How big is 1000!?

Some Numbers

10^8	PC instructions/second
10^12	supercomputer instructions/second
10^9	seconds/year
10^13	age of universe in years (estimated)
10^79	number of electrons in the universe (estimated)

Exponential growth dwarfs technological change:

Suppose each electron in the universe had the computing power of today's supercomputers. If they worked together for the estimated life of the universe, they couldn't solve the traveling salesperson problem on 1000 cities (using the obvious algorithm).

$$1000! > 2^{1000} > 10^{300} >> 10^{79} * 10^{13} * 10^{9} * 10^{12}$$

Outline

- ~~Introduction: polynomial vs. exponential time~~
- **P vs. NP: the holy grail**
- NP-Completeness: Cook's Theorem
- NP-Completeness: Reduction
- Conclusions

Polynomial Time

P:

The set of all problems solvable in polynomial time on a deterministic Turing machine

- Why is this definition important ?

[Strong] version of Church-Turing thesis:
P is the set of all problems solvable in polynomial time on a real computer

Nondeterministic Polynomial Time

NP:

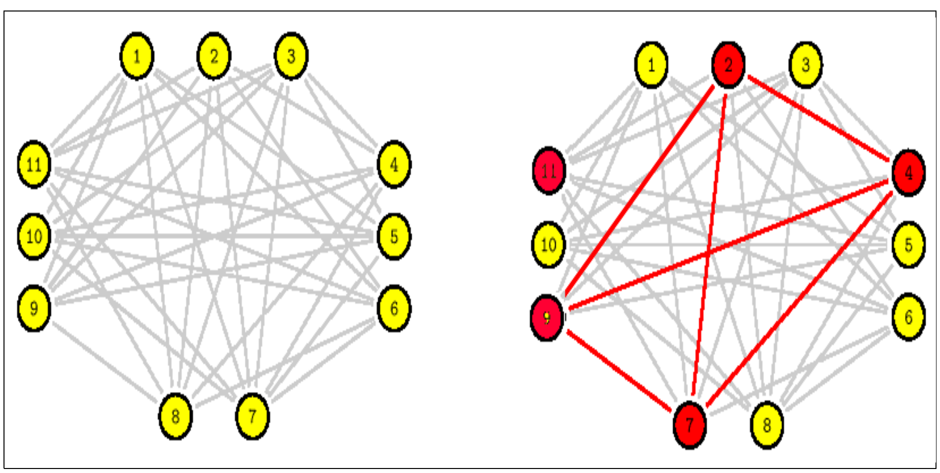
The set of all problems solvable in polynomial time on a nondeterministic Turing machine

- For a problem in NP, a machine can efficiently VERIFY that a given solution is correct

Ex:

factoring: does 15243198749 have factors >1? can verify that 12347 is a factor by dividing

Another NP Example: CLIQUE



Given N people, does there exist a group of size k such that every pair of people in the group know each other?

Another NP Example: Satisfiability

Is there a way to assign truth values to a given logical formula that makes it true?

Ex:

satisfiability: can verify that
$(x' + y + z)(x + y' + z)(y + z)(x' + y' + z')$
is 1 if x, y, z, are 1, 1, 0 (resp.)

P vs. NP

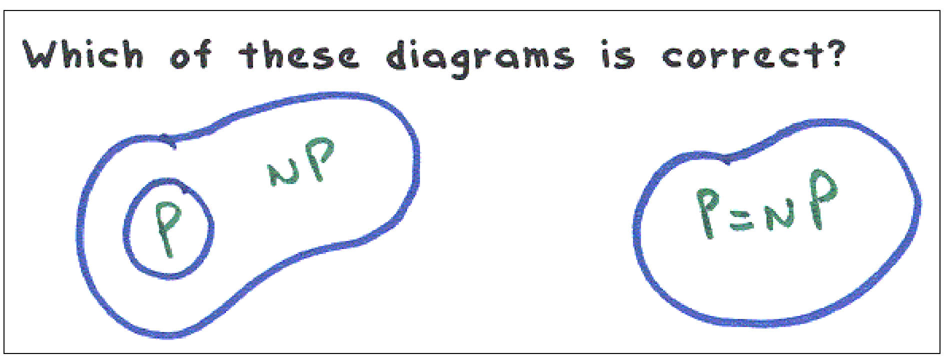
If a machine can guess (and is lucky),
it can solve a problem in NP quickly.
Actual computers can simulate Lucky Guessing,
in exponential time, by trying every possibility

[Possible exception ??
Quantum computers]

Possible Exception: Quantum Computing

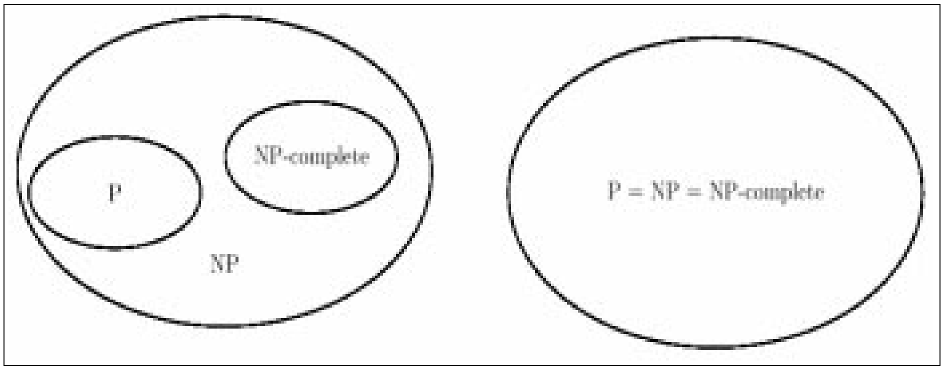
- Quantum mechanics: "coherent superposition"
  - A photon can be "here" and "there" simultaneously
  - An atom can be in two electronic states simultaneously
  - In general, a "qubit" can be 0 and 1 simultaneously!
  - A k-bit quantum register can store 2^k values simultaneously!
- Quantum computing
  - A single quantum instruction, effected by a laser pulse, for example, can transform a quantum register from one multi-state to another in one step
  - A classical computer needs 2^K steps or 2^K parallel registers to match this power
- Non-determistic TM: no more power than TM, but a lot faster than a determistic TM

P = NP? (The Holy Grail)



- Nondeterminism (Lucky Guessing) seems powerful, but no one has been able to PROVI that it helps for any particular problem.

NP-Completeness



NP-Complete

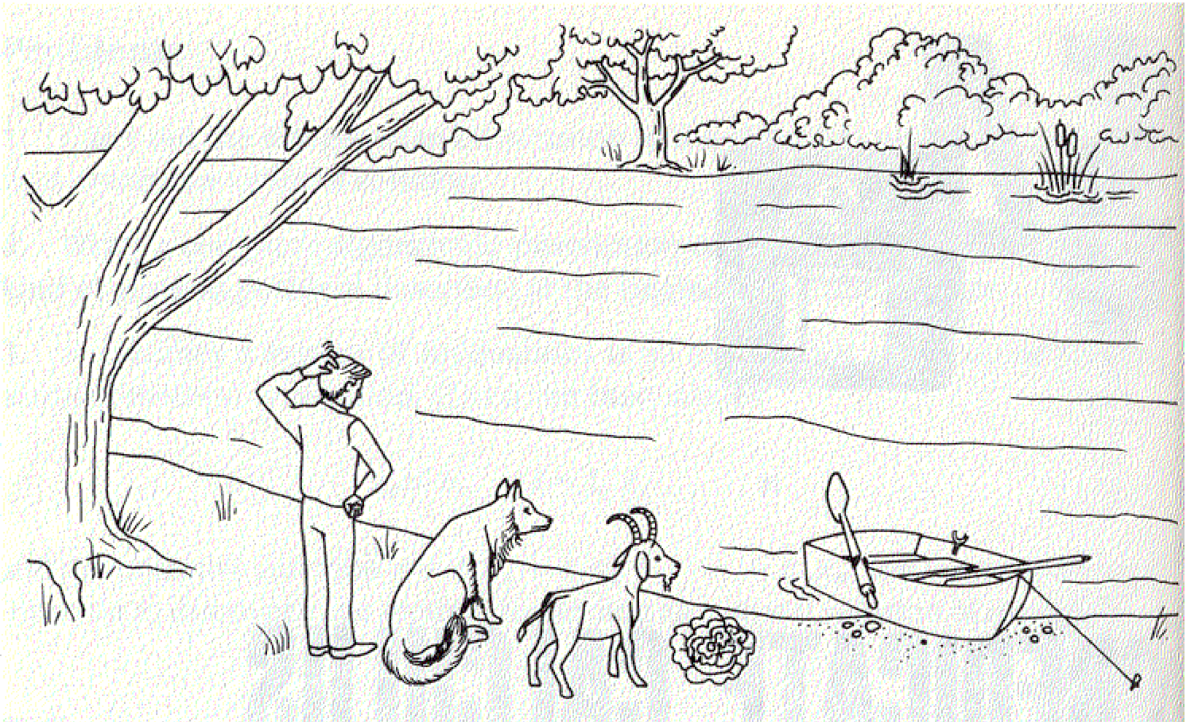
A problem in NP with the property that if it can be solved efficiently, then P=NP.

(Lucky Guessing doesn't help.)

Outline

- ~~Introduction: polynomial vs. exponential time~~
- ~~P vs. NP: the holy grail~~
- **NP-Completeness: Cook's Theorem**
  - **A digression in logic**
  - **The very first NP-Complete problem**
- NP-Completeness: Reduction
- Conclusions

A Puzzle



A Digression in Logic

- Classical logic had its origin in Aristotle
- Turing Machine was invented to settle whether logic satisfiability was solvable
- FSAs and PDAs were developed as simplifications of TMs
- History: perfect reversal of our presentation

Propositional Logic and Satisfiability Proof

Representation:
`Th`: *Today is Thursday*
`Fr`: *Tomorrow is Friday*
`Th` *and* `Fr` *can be* `0` *or* `1`

Given:
`Th`
`Th -> Fr`

Prove:
`Fr`

Proof:
Assume `Fr’`,
`Th * (Th->Fr) * Fr’`
`= Th * (Th’+Fr) * Fr’`

There is no assignment of Th and Fr that can make this formula true, so assumption must be wrong.

- Like the boolean algebra that we have learned
- Extension to “predicate calculus” to make it more powerful
- A powerful language for describing real world processes
- A darling artificial intelligence tool

A More Complex Example: The Puzzle

Representation:

$M_i=0$, if Man is on left bank at time i
$M_i=1$, if Man is on right bank at time i
Similarly define W_i, G_i, and C_i for Wolf, Goat, and Cabbage.

Given:

$M_0=W_0=G_0=C_0=0$
$M_i'W_i'G_i'C_i' \rightarrow M_{i+1}W_{i+1}'G_{i+1}C_{i+1}'$
$M_i'W_i'G_i'C_i \rightarrow M_{i+1}W_{i+1}G_{i+1}'C_{i+1}'$
$M_iW_iG_i'C_i \rightarrow M_{i+1}'W_{i+1}'G_{i+1}'C_{i+1}'$
. . . . . . **(many more similar rules)**

Prove:

$M_k=W_k=G_k=C_k=1$
(for some sufficiently large k)

Proof:

Similar as previous slide, assignment of M_i, W_i, G_i, C_i gives solution

What's the Relevance of This Puzzle? Propositional and Predicate Calculi as Descriptions of Computational Processes

- The puzzle is really a computational process
  - The initial locations of the man, wolf, goat, and cabbage are the input state
  - The movement rules are a program:
    + for each current state,
    + non-deterministically apply one of the applicable rules
    + transform to next state
  - The final locations: the desired output state
- If we can find a variable assignment to make the corresponding logic formula true, we have found a solution to the problem

Cook's Theorem

- A non-deterministic TM with its input is like a puzzle
- We can encode it with a logic formula like we did
- If we can find a variable assignment to make the formula true, we have found a solution to the puzzle, namely a simulation of the TM that solves the problem
- Therefore, if we can solve SATISFIABILITY quickly, then we can find solutions to non-determistic TMs quickly
- Any NP problem can be solved by a non-determistic TM by definition
- Therefore, if we can solve SATISFIABILITY quickly, we can solve any NP problem quickly
- SATISFIABILITY is the very first problem proven to be NP-Complete: a landmark theorem!

In Other Words ...

- An NP problem =
  An instance of non-deterministic TM =
  A SATISFIABILITY problem
- A solution to an NP problem =
  A successful simulation of the non-deterministic TM =
  A solution to the SATISFIABILITY problem
- Therefore, if we can solve SATISFIABILITY quickly, we can solve any NP problem quickly
- Now that we have found our first NP-Complete problem, are there others?

Outline

- ~~Introduction: polynomial vs. exponential time~~
- ~~P vs. NP: the holy grail~~
- ~~NP-Completeness: Cook's Theorem~~
- **NP-Completeness: Reduction**
  - **The basic idea: to show a problem is NP-Complete, we show it's "harder" than SATISFIABILITY**
- Conclusions

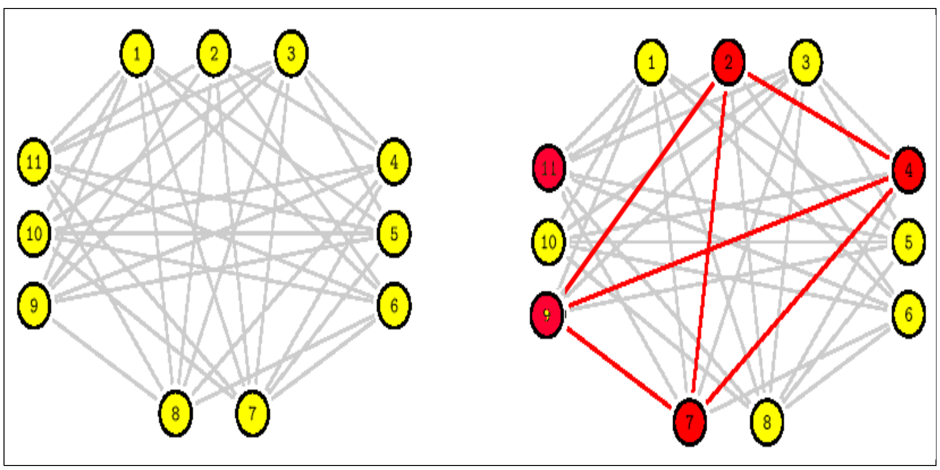
Reduction

For specific problems A and B, we can often show
If A can be solved efficiently, then so can B
(if so, we say that B "REDUCES TO" A)

- To prove a problem A to be NP-complete
  - * prove it to be in NP
  - * prove that some NP-complete problem B reduces to A

- That is, if A can be solved efficiently, then
  - * B can be solved efficiently
  - * so can every problem in NP

An NP-Complete Example: CLIQUE

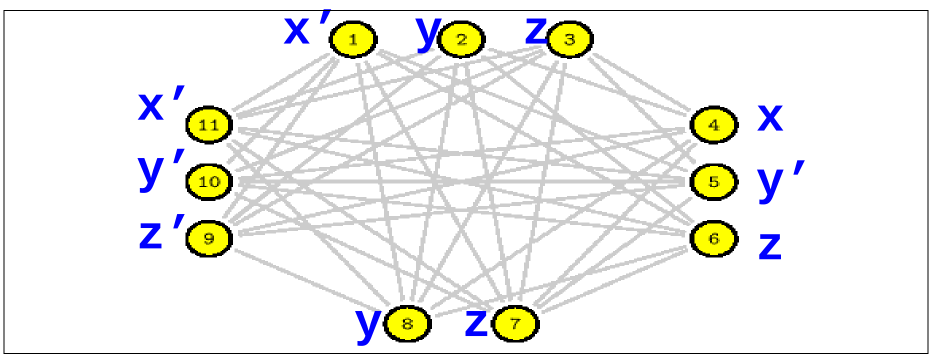


Given N people, does there exist a group of size k such that every pair of people in the group know each other?

Proving CLIQUE Is NP-Complete

- We have already shown **CLIQUE is NP**
- Now we will show **SATISFIABILITY reduces to CLIQUE**
- Given an instance of SAT,
  we construct an instance of CLIQUE
  that has a solution if and only if
  the SAT instance is satisfiable
- (A note)
  - We have seen that any logic formula can be expressed as a **sum-of-products** form
  - Any logic formula can also be expressed as a **product-of-sums form**

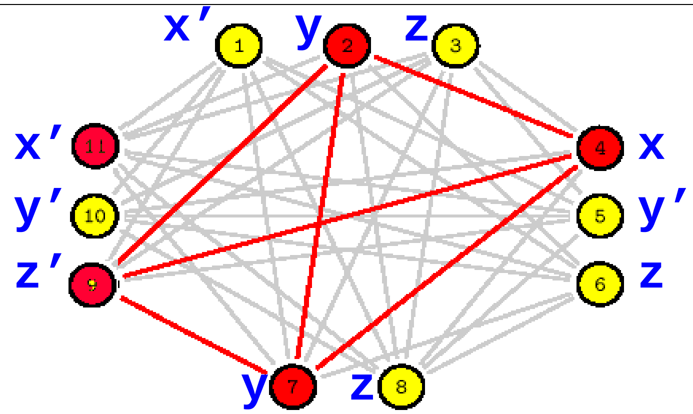
Transforming SAT to CLIQUE



- Associate a person with each variable
  occurrence in each clause
- Two people "know" one another EXCEPT if
  * they come from the same clause
  * they represent t and t' for some t

ex: (x' + y + z)(x + y' + z)(y + z)(x' + y' + z')

Solution to CLIQUE = SOLUTION to SAT



$(x'+y+z)(x+y'+z)(y+z)(x'+y'+z)$

- Solution to SAT **==>** solution to CLIQUE
- Solution to CLIQUE **==>** solution to SAT
- So, CLIQUE is NP-Complete

More NP-Complete Problems

- Thousands of problems have been shown to be NP-complete in this way.
- If any one of these important problems can be solved efficiently, they all can. ! (Moreover, so can any problem in NP).

More NP-Complete Problems

* TRAVELING SALESPERSON

  A salesperson needs to visit N cities.
  Is there a route of length less than d?

* SCHEDULING

  A set of jobs of varying length need to be done on two identical machines before a certain deadline. Can the jobs be arranged so that the deadline is met?

* SEQUENCING

  A set of four-character fragments have been obtained by breaking up a long string into overlapping pieces. Can the fragments be reconstituted into the long string?

Outline

- ~~Introduction: polynomial vs. exponential time~~
- ~~P vs. NP: the holy grail~~
- ~~NP-Completeness: Cook's Theorem~~
- ~~NP-Completeness: Reduction~~
- **Conclusions**

Implications of NP-Completeness

Either:

Conventional machines can do as well as machines capable of Lucky Guessing, but we don't know how to make them do so. (P=NP)

Or:

Lucky Guessing DOES help, but is a fiction on conventional machines, since none of the NP-complete problems can be solved in polynomial time. (P!=NP)

Not many people believe that P=NP
...but it's possible.

Proof that a problem is NP-complete is usually taken as a signal to abandon hope of finding an efficient solution.

Coping With NP-Completness

* Hope that the worst case doesn't occur
  (try to simulate Lucky Guessing)

* Change the problem
  (try for an approximate solution)

* Exploit NP-completeness
  (example: cryptography)

* Keep trying to prove that P=NP!

What We Have Learned Today

- What are P, NP, NP-Complete problems? What are their relationships?
- What's Cook's Theorem?
- What's reduction?