

Lecture 12: NP-Completeness

CS442: Great Insights in Computer Science
Michael L. Littman, Spring 2006

What is CS About?

- Something we'd like the computer to do.
- Formalize it into a problem:
 - What's the input?
 - What's the output?
 - What property should the output have?
- Design/ implement an algorithm to solve it.
- Repeat.

Example: Sorting

- Want to make searching a list fast.
- Formal Problem: Sorting.
 - Input: A list of n items and a method of comparison ("*less than or equal to*").
 - Output: A list of n items.
 - Property: Output is reordering (permutation) of input list. Each item less than or equal to the one after it in the list.



Factoring

- Input: A positive integer.
- Output: Two positive integers.
- Property: The product of the two output integers should equal the input, or "None" if input is prime.

FACTORISATION

6127049569075942128162558427465565
7260868460760319841482797982260315
1562636921130240334284467961571889
9328182959467198786178804369420568
8352551258319382990948318104995562
4510589912696573429067127442879129
8474527796793829737067915624465318
9754708742598658852436344373234057
2982308396576265269454435373897942
6311879135265107590764939899242733
1262743457541639963233300342784456
4182540621165072646771998059

Can you factor the above 402 digit number? It is composed of two large prime numbers. The first correct factorisation sent to webmaster@mathematik.com wins 100 US Dollars.

Why We Like It

- Once a problem is formalized, we can restrict our attention to it (another kind of reduction) without worrying about how it will be used.
- Strict rules about what constitutes a valid solution to the problem.
- We can recognize a right answer when we've got one.

Invalid Problems

- Most important thing to try to learn today: recognizing well-formulated vs. poorly formulated problems.
- Problem should have enough information to be complete and unambiguous.

Verbal Magnification

- Input: A word, x .
- Output: A word y .
- Property: y should have the same meaning as x only moreso, or "N/A" if no such word exists.
- Example: large => huge, tired => exhausted, hungry => starving, corn => N/A,
- Incomplete, uncheckable.

Examples: Valid or Not?

- Input: Two superheroes. Output / Property: Who would win if they fought?
- Input: Word, word list. Output / Property: Does the reverse of the word appear in the word list?
- Input: Photo. Output / Property: List of items pictured.
- Input: A Boolean formula with and/or/not/ T/ F. Output / Property: Is it T?
- Input: Two colors. Output / Property: Whether they match.

Decision Problems

- In one class of problems, the output is always just yes/no (T/F, 0/1).
- We call these problems *decision problems*.
- They might seem pretty limited, but they are actually quite powerful.

Median

- Input: A list l of numbers, a number x from the list.
- Property: True if x is the median of l (sorts to the $(n+1)/2$ position if n is odd and the $n/2$ position if it is even).

Interreducibility

- Two decision problems are called *interreducible* if the answer to either can be used to solve the other.
- How use sort, to solve median?
 - Sort l , and check the halfway position.
- How use median to sort?
 - Ask if each item is the median. Once found, split the list and recurse.

Example

- **List:** [5, 4, 11, 3, 10, 7, 6, 18, 16, 8, 1, 2, 9, 0, 15]
- median(7) = yes; split.
- [5, 4, 3, 6, 1, 2, 0], 7, [11, 10, 18, 16, 8, 9, 15]
- median(3) = yes, median(11) = yes; split.
- [1,2,0], 3, [5, 4, 6], 7, [10, 8, 9], 11, [18, 16, 15]
- medians: 1, 5, 9, 16; split.
- 0,1,2,3,4,5,6,7,8,9,10,11,15,16,18

Polynomial Equivalence

- It takes one call to sort to solve median.
- It takes $O(n \lg n)$ calls to median to sort.
- Since the running time in both cases is polynomial ($O(n^k)$ for some constant k), we can say that the two problems are equally hard (or easy) to solve.
- They are *polynomially equivalent* since a polynomial-time solution to one implies a polynomial-time solution to the other.

Summary

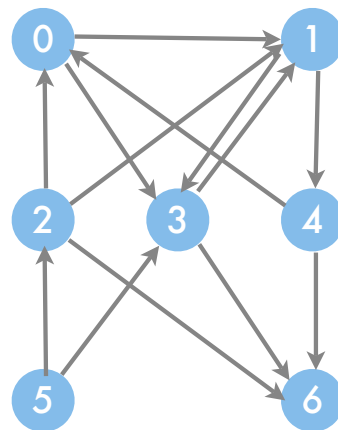
- Idea #1:
 - A problem should be formalized in terms of input, output, and property.
- Idea #2:
 - The property should be checkable by a computer program written specifically for the problem and using only the input and output.

Summary Continued

- Idea #3:
 - The output of a *decision problem* is a bit.
- Idea #4:
 - Problems can be *reduced* to each other: The solution to one problem solves another.
- Idea #5:
 - Problems are *polynomially equivalent* if each solves the other with polynomial # of calls.

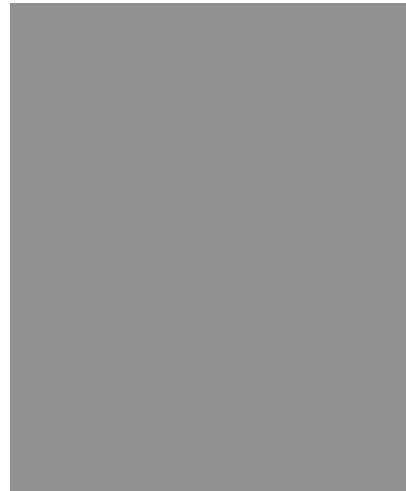
Hamiltonian Path

- Input: Graph with one source and one sink.
- Output: yes/no.
- Property: Is there a path that visits each node exactly once?
- Yes proof: The path.



Hamiltonian Path

- Input: Graph with one source and one sink.
- Output: yes/ no.
- Property: Is there a path that visits each node exactly once?
- Yes proof: The path.



"Yes Proof"

- A large class of decision problems have the property that a "yes" answer means that there's a small "proof" that can be quickly and automatically (in polynomial time) verified as being correct.
- It was named "NP" for "non-deterministic polynomial time".

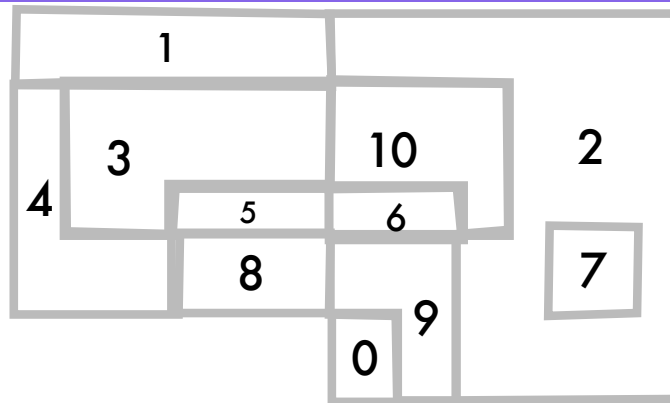
Some NP Examples

- Problem: Is x the median of list l ? Yes proof: The sorted order.
- Problem: Does G have a Hamiltonian Path? Yes proof: The path.
- Problem: Can this map be colored with 3 colors? Yes proof: The coloring.
- Problem: Can k coins make c cents? Yes proof: The number of each coin type.

NP and Puzzles

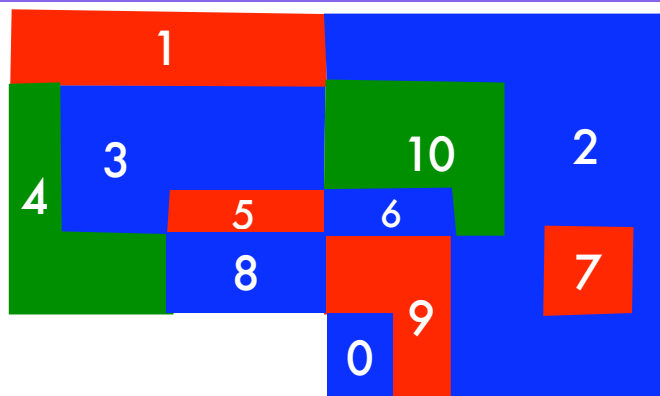
- Most puzzles can be thought of as NP problems. Why? Because the answer appears the next day. It might be hard to find the answer, but it's easy to check once you hear it.
- Sometimes I say NP stands for "Nice Puzzle" for this reason.

Map Coloring



- Each region gets a color. Regions that share an edge must be different colors (corners or ok).

Colored Map



Satisfiability: Key Problem

- Input: A Boolean formula (T/F/and/or/not/ n variables).
- Output: Yes/no
- Property: Yes means there is a way to set the variables to T/F so that formula is T.
- Yes proof: An assignment of each variable to a truth value.

SAT Example

(not x or y) and (w or y) and (not x or w) and
(y or x) and (w or x or not y) and
(w or not y) and (x or y) and
(not w or not x or y) and (not x or not y)

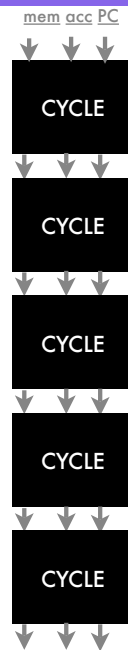
- Is there a way to assign T/F to w , x , and y to make the formula True?

Or, Another Way

- Let's think things through.
- If an NP problem has a yes answer, then there is a (polynomial time) program that can check the proof.
- So, really, every NP problem is just "Is there a proof that the answer is yes?"
- Or, "Is there you could tell my checker program that would make it answer "yes"?"

Remarkable Insight (Cook)

- But, a checker program is really just running a computer for some number of steps.
- And, each step the computer makes is really just a Boolean formula.
- So, every NP problem is really a satisfiability problem!
- Is there some initial contents of the memory that would make the checker program answer "yes"?



Why Is It So Cool?

- Well, every NP problem can be solved with a polynomial-time algorithm if SAT can. (Because we can convert the corresponding checker program into a big formula and ask if it is satisfiable.)
- Further, anything that is polynomially equivalent to SAT could also be used to solve all NP problems in polynomial time.
- We call these problems *NP-complete*.

Can We Solve Them All?

- We say a problem is in P (polynomial time) if there is a polynomial-time algorithm that solves it.
- Is SAT in P? If it is, then all NP problems are in P.
- If this were the case, we could say “P=NP”.
- On the other hand, if any problem in NP takes more than polynomial time, then all NP-complete problems take more than polynomial time.
- We can say “P not equal to NP”.
- And... we don't know.

P vs. NP-Complete

NP-complete

- Satisfiability
- Graph coloring
- Hamiltonian Path (visit all nodes)
- Subset Sum (subset of numbers sum to z ?)
- Traveling Salesperson Problem (TSP)

P: Polynomial time

- Median
- Prime
- String search
- Connectivity (go s to t)
- Eulerian Path (visit all edges)
- Coin sum (subset of coins sum to z ?)

Reductions

- To discover that problem x is NP-complete, need to show how a polynomial time solution to x would solve some NP-complete problem in polynomial time.
- Often devilishly clever.
- Since TSP is so famous in the popular press, let me show you how we know TSP is NP-complete (and therefore probably not solvable efficiently in the worst case)...

TSP Connections

TSP: Given an undirected graph with weighted edges, is there a tour of minimum weight? Given instance, is there a tour of weight $\leq W$? "NP-complete" means "no known polynomial-time algorithm" and "NP-hard" means "no known polynomial-time algorithm, but every NP problem can be reduced to it".

Cook showed that every NP problem is really just a satisfiability problem. Hamiltonian Path solver can actually figure out if a Boolean formula is satisfiable!

Underlying question: Given a Boolean formula, is there a Hamiltonian Path? Hamiltonian Path solver can actually figure out if a Boolean formula is satisfiable!

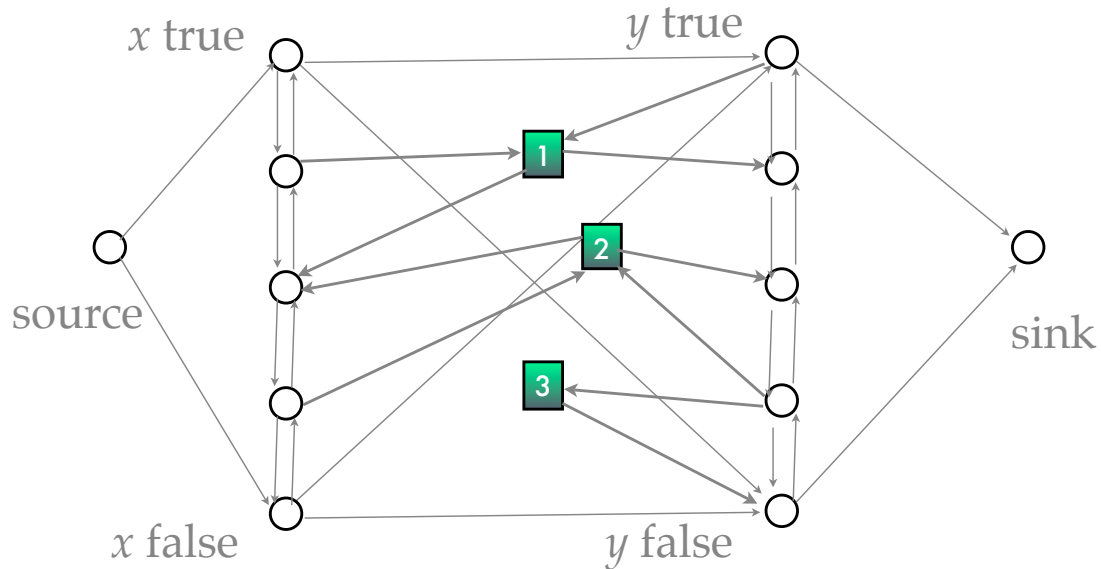
Hamiltonian Path: Given graph, is there a path that visits all nodes?

SAT to Hamiltonian Path

- We want an automatic procedure for taking any Boolean formula and creating a corresponding graph.
- The graph has a Hamiltonian Path if and only if the original formula was satisfiable.
- Therefore, the answer to the Hamiltonian Path problem is also the answer for the SAT problem.
- So, Hamiltonian Path solves SAT.

Concrete Example

$(x \text{ or } y) \text{ and } (\text{not } x \text{ or not } y) \text{ and } (y)$



Implication

- Every Hamiltonian Path provides a satisfying assignment of the original formula.
- Any satisfying assignment of the formula can be translated to a Hamiltonian Path.
- No family resemblance, but they are definitely related!
- Note: Used “CNF” form of the formula. A separate argument shows that any formula can be expressed in CNF.

Targum: Valid? NP? P?

- **Jumble:** Given a word list l and a jumble of letters w , can the letters of w be rearranged to form a word in l ?
- **Crossword clue:** Given a clue, and an answer word w , does w answer the clue?
- **Crossword grid:** Given answers for each slot, do they fit in the grid without conflicts?
- **Sudoku:** Given a partial grid, can it be completed properly?
- **Word find:** Does word w appear in grid G ?

Next Time

- Heuristics.
- Finish Hillis, Chapter 5.