UNIT 14B
The Limits of Computing: P and NP

Last Lecture

- Review of the big O notation
- Tractable vs. intractable decision problems
  - Polynomial-time vs. super-polynomial
- Examples of intractable decision problems: Money Puzzle, Traveling Salesman

Today's Lecture

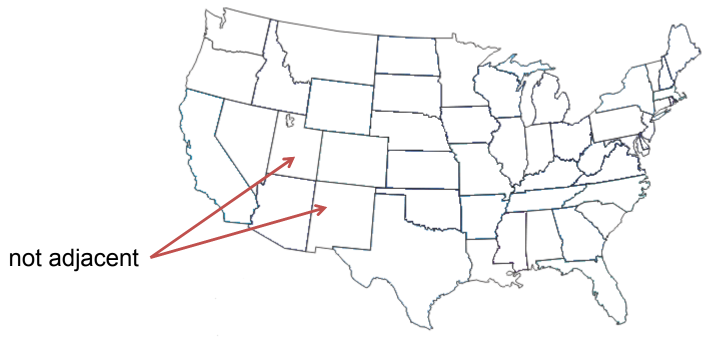
- More examples of intractable problems
- Introduce the class NP (nondeterministic polynomial time)
- The P vs. NP question
- NP-completeness

Map Coloring

- Given a map of N territories, can the map be colored using **K** colors such that no two adjacent territories are colored with the same color?

Map Coloring

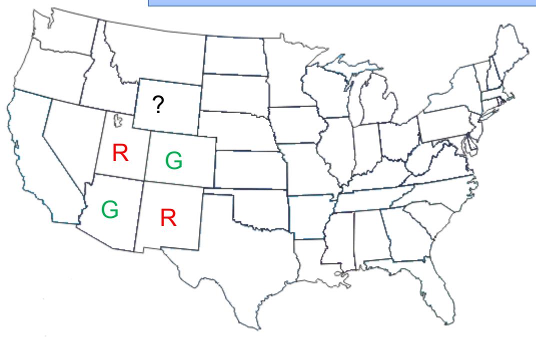
- Given a map of N territories, can the map be colored using **K** colors such that no two adjacent territories are colored with the same color?



2-Coloring

- Given a map of N territories, can the map be colored using **2** colors such that no two adjacent territories are colored with the same color?

Yes, only if the map contains no point that is the junction of an odd number of territories. We can find a yes or no answer quickly.



4-Coloring

- Given a map of N territories, can the map be colored using 4 colors such that no two adjacent territories are colored with the same color?
- Theorem: Answer is always yes for 4 colors.

7

3-coloring (analysis)

How many possible 3-colorings are there?

8

3-coloring (analysis)

- Given a map of N territories, can the map be colored using **3** colors such that no two adjacent territories are colored with the same color?
  - Pick a color for territory 1 (3 choices)
  - Pick a color for territory 2 (3 choices)
  - ...
- There are $3 * 3 * \ldots *3= 3^N$ possible colorings.

Verifiability

- No known tractable algorithm to decide, however it is easy to verify a solution.

Satisfiability

- Given a Boolean formula with N variables using the operators AND, OR and NOT:
  - Is there an assignment of Boolean values for the variables so that the formula is true (satisfied)?
    Example: (X AND Y) OR (NOT Z AND X)
  - Truth assignment: X = true, Y = true, C = false.
- How many assignments do we need to check for N variables?
  - Each symbol has 2 possibilities ... 2^N assignments

The Big Picture

- Intractable problems are solvable if the amount of data (*N*) that we're processing is small.
- But if *N* is not small, then the amount of computation grows exponentially and the solutions quickly become intractable (i.e. out of our reach).
- Computers can solve these problems if *N* is not small, but it will take far too long for the result to be generated.
  - We would be long dead before the result is computed.

Decision Problems

- We have seen four examples of decision problems with simple brute-force algorithms that are intractable.

– The Monkey Puzzle	O(N!)
– Traveling Salesperson	O(N!)
– Map Coloring	$O(3^N)$
– Satisfiability	$O(2^N)$

Are These Problems Tractable?

- For any one of these problems, is there a single tractable (polynomial) algorithm to solve any instance of the problem?
  Haven't been found so far.
- Possible reasons:
  - These problems have undiscovered polynomial-time solutions.
  - These problems are intrinsically difficult – we cannot hope to find polynomial solutions.
- Important discovery: Complexities of some of these problems are linked. If we can solve one we can solve the other problems in that class.

P and NP

- The class **P** consists of all those decision problems that can be solved on a deterministic sequential machine in an amount of time that is polynomial in the size of the input Polynomial decidability
- The class **NP** consists of all those decision problems whose positive solutions can be verified in polynomial time given the right information, or equivalently, whose solution can be found in polynomial time on a non-deterministic machine.

from Wikipedia Polynomial verifiability

15110 Principles of Computing, Carnegie Mellon University 15

Decidability vs. Verifiability

P = the class of problems that can be decided (solved) quickly

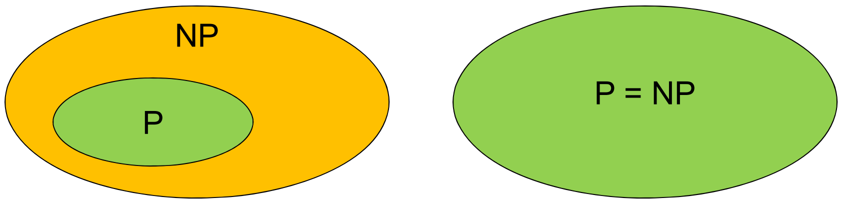
NP = the class of problems for which candidate solutions can be verified quickly

15110 Principles of Computing, Carnegie Mellon University 16

Example

- Finding the Minimum in an Array
  Verifiable in polynomial time? YES
  Solvable in polynomial time? YES
- Map Coloring
  Verifiable in polynomial time? YES
  Solvable in polynomial time? ?
- If a problem is in P, it must also be in NP.
- If a problem is in NP, is it also in P?

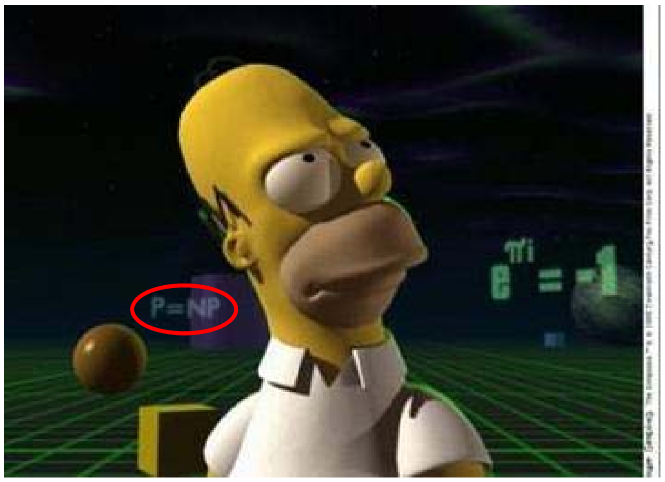
Two Possibilities



If P ≠ NP, then some decision problems can't be solved in polynomial time.
If P = NP, then all computable problems can be solved in polynomial time.

The Clay Mathematics Institute is offering a \$1M prize for the first person to prove P = NP or P ≠ NP.
(http://www.claymath.org/millennium/P_vs_NP/)

Watch out, Homer!

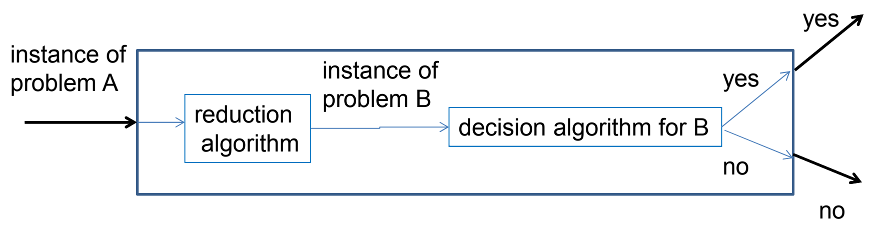


NP-Complete Problems

- An important advance in the P vs. NP question was the discovery of a class of problems in NP whose complexity is related to the whole class [Cook and Levin, ‘70]: if one of these problems is in P then NP = P.

Reductions

- Consider 2 problems A and B: Suppose we are trying to solve A and have a decision algorithm for B.



- Reduction algorithm should be polynomial time and the reduction should be such that A and B give the same result in all cases

21

NP-completeness

- A problem A is NP-complete if
  - A is in NP
  - Every other problem in NP is polynomial time reducible to A (there is an efficient way to transform each problem in NP to A).

22

NP-Complete

- The class **NP-Complete** consists of all those problems in NP that are least likely to be in P.
  - Monkey puzzle, Traveling salesperson, map coloring, and satisfiability are all in NP-Complete.
- Every problem in NP-Complete can be transformed to another problem in NP-Complete.
  - If there were some way to solve one of these problems in polynomial time, we should be able to solve all of these problems in polynomial time.

Example: Reduction

- A Boolean formula is in conjunctive normal form, called a cnf-formula if it comprises several clauses connected with ANDs. Following is a 3cnf-formula (each clause has 3 literals):

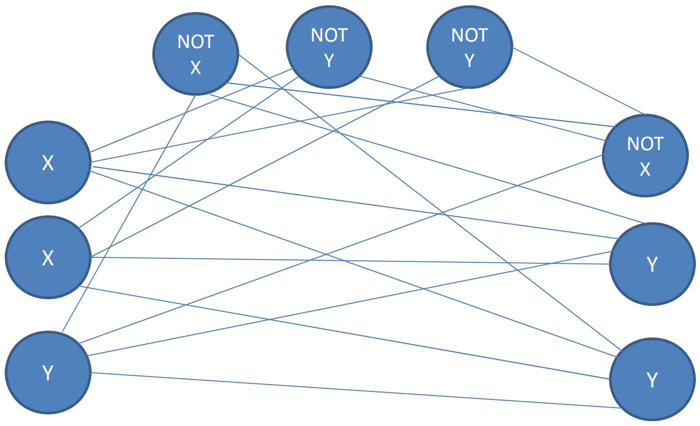
(X OR X OR Y) AND ((NOT X) OR (NOT X2) OR NOT Y)) AND (NOT X OR Y OR Y)

- A *k-clique* in a graph is a subgraph with *k* nodes wherein every two nodes are connected by an edge.
- We can show that satisfiability of a 3cnf-formula with *k* clauses can be reduced to finding a *k-clique* in a “related” undirected graph, which is constructed following certain rules

Example: Reduction (cont'd)

- Reducing 3SAT to Clique:
  - Organize nodes into k groups of size 3 where each such group corresponds to a clause in the associated clause (use literals as labels of nodes)
  - No edge between nodes in the same group
  - No edge between two nodes with contradictory labels

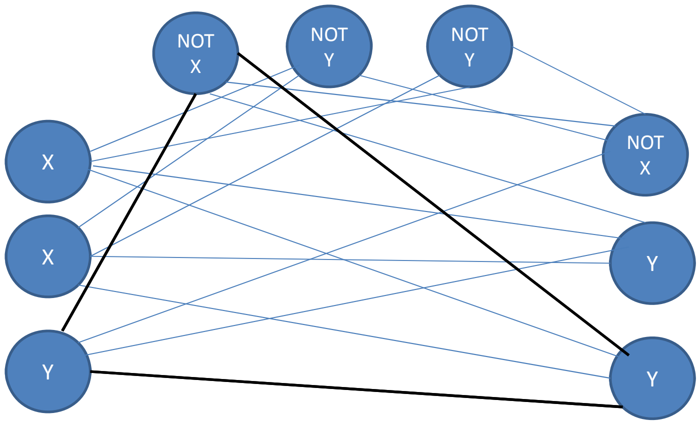
Example: Reduction (cont'd)



(X OR X OR Y) AND ((NOT X) OR (NOT Y) OR NOT Y)) AND (NOT X OR Y OR Y)

Clause 1 Clause 2 Clause 3

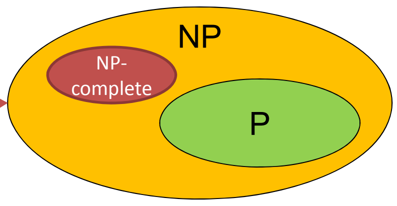
Example: Reduction (cont’d)



$$\underbrace{\text{(X OR X OR Y)}}_{\text{Clause 1}} \text{ AND } \underbrace{\text{((NOT X) OR (NOT Y) OR NOT Y))}}_{\text{Clause 2}} \text{ AND } \underbrace{\text{(NOT X OR Y OR Y)}}_{\text{Clause 3}}$$

27

Why is NP-completeness of Interest?



Theorem: If any NP-complete problem is in P then all are and P = NP.

Most believe P ≠ NP. So, in practice NP-completeness of a problem prevents wasting time from trying to find a polynomial time solution for it.

28

NP-completeness in Practice

- Thousands of problems have been proved NP-complete.
- If you have a problem that is in NP, and you don't know a polynomial time algorithm for it, it may be reasonable to assume (prove) that it is NP-complete until proved otherwise.
  - E.g. many optimization problems in industry.

What's Next?

- Are all computational problems solvable by computer?
  - NO!
    There are some that we can't solve no matter how much time we give the computer, no matter how powerful the computer is.