

## UNIT 14B

### The Limits of Computing: P and NP

1

## 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

2

## Today's Lecture

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

3

## 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?



15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

4

## 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?



15110 Principles of Computing, Carnegie Mellon University - CORTINA

5

## 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.



15110 Principles of Computing, Carnegie Mellon University - CORTINA

6

## 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.

This does not imply that it is easy to find the 4-colorings that work.

7

## 3-coloring (analysis)

How many possible 3-colorings are there?



Exercise:

How many possible colorings are there for 3 regions?  $3^3$

How many possible colorings are there for 4 regions?  $3^4$

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  $3 * 3 * \dots * 3 = 3^N$  possible colorings.

15110 Principles of Computing, Carnegie Mellon University - CORTINA

9

## Verifiability

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

10

## 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:  $(A \text{ AND } B) \text{ OR } (\text{NOT } C \text{ AND } A)$
  - Truth assignment:  $A = \text{True}, B = \text{True}, C = \text{False}$ .
- How many assignments do we need to check for  $N$  variables?
  - Each symbol has 2 possibilities ...  $2^N$  assignments

15110 Principles of Computing, Carnegie Mellon University - CORTINA

11

## 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.



15110 Principles of Computing, Carnegie Mellon University - CORTINA

12

## 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  $O(2^N)$

15110 Principles of Computing, Carnegie Mellon University - CORTINA

13

## 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 these problems are linked. If we can solve one we can solve the other problems in that class.

14

## 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

## 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



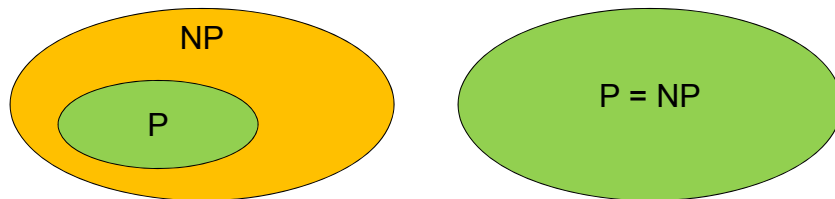
## Example

- Finding the Minimum in an Array  
Verifiable in polynomial time?   
Solvable in polynomial time?
- Map Coloring  
Verifiable in polynomial time?   
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?

15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

17

## Two Possibilities



If  $P \neq NP$ , 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  $P \neq NP$ .

([http://www.claymath.org/millennium/P\\_vs\\_NP/](http://www.claymath.org/millennium/P_vs_NP/))



15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

18

# Watch out, Homer!



15110 Principles of Computing, Carnegie  
Mellon University - CORTINA

19

## NP-completeness

- A problem A is NP-complete if
  - A is 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).

20

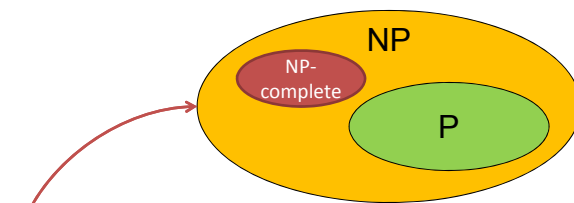
## NP-Complete

- 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.

15110 Principles of Computing, Carnegie Mellon University - CORTINA

21

## Why is NP-completeness of Interest?



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

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

22

## 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.

23

## 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.

24