

# GEM 1501 Problem Solving With Computers

## Lecture 6:

### Inefficiency and Intractability

Martin Henz

# Summary of Previous Lecture

- Correctness of algorithms
  - Kinds of errors
  - Sources of errors
  - Correctness
- Efficiency of algorithms
  - Example
  - Big-O
  - Recurrences

# Overview of Today's Lecture

- Growth of functions
- NP-complete problems
- Is  $P$  equal  $NP$ ?
- Even worse problems

# Complexity of Towers of Hanoi

- For  $N$  rings, the number of moves is  $2^N - 1$ .
- On a 1 GHz computer with one move per clock cycle, it would take 36 years to move 60 rings!
- 70 rings would take about 40000 years!
- 80 rings would take about 50 million years!

# What's the problem?

- The problem “Towers of Hanoi” is so hard, because the output (moves) is so long.
- How about problems with short answers?
- Let us focus on problems with yes/no answers!
- These problems are called “decision problems” .

# Example: Monkey Puzzle

- Assume an  $M \times M$  board with tiles.
- Arrange  $N = M^2$  tiles so that the edges match.
- Simple solution: Try all possible ways!
- This approach is “impractical”!
- Why?

# Growth of Functions

- Table of functions (10, 50, 300)
- Table for monkey puzzle (10, 50, 300)
- Log-log graph

# Intractable Problems

- Polynomial problems can be solved in time  $O(N^k)$  for some constant  $k$ .
- For exponential problems, there is no  $k$  such that the problem is in  $O(N^k)$ .
- For some problems, we don't know!

# Can we wait for faster computers?

- What if we use a computer 100 times faster?
- What if we use 1000 computers hooked together?
- Polynomial problems and exponential problems really behave differently!

# NP-Complete Problems

- For this class of problems, we don't know whether they are polynomial or exponential.
- We know that they can be solved in exponential time (upper bound).
- Solving many of them requires at least linear time (lower bound).

# Examples of NP-Complete Problems

- Monkey problem
- Traveling salesman problem
- Timetabling problem
- Satisfiability problem
- Coloring a map with three colors
- Hamiltonian path problem

# NP-Complete Problems have Short Certificates

- If the answer to the decision problem is “yes”, we are asking for a proof.
- One way to prove the answer is to provide a “certificate”.
- NP-Complete problems have small certificates (often linear) and the proof of the certificate is fast (also often linear).
- Examples:
  - Monkey problem
  - Timetabling problem
  - Satisfiability problem
  - Coloring a map with three colors

# What does “NP” stand for?

- NP stands for “non-deterministic polynomial”
- A “non-deterministic computer” is a computer that can guess the right answer.
- Another way of thinking about it: Each time we have a choice between  $K$  alternatives, we make  $K$  copies of the computer.

# NP-complete Problems Stand and Fall Together

- All NP-complete problems are equivalent!
- If we can solve one in polynomial time, we can solve all.
- How come?
- By transformation!
- Example: Transforming the Hamiltonian path problem into a traveling salesman problem.

# Is P Equal to NP?

- Question lies at the heart of “Algorithmics”
- Unsolved since it was posed in 1971
- Most people believe that  $P \neq NP$ , but no one knows for sure
- NP-completeness is good evidence for its probable intractability

# Update on Primality

- Book (second edition) says that primality testing is unsolved
- In 2002, Prof. Manindra Agarwal (IIT Kanpur) and two of his students, Nitin Saxena and Neeraj Kayal gave a polynomial algorithm ( $O((\log n)^{12} f(\log \log n))$  where  $f$  is a polynomial)

# Provably Intractable Problems

- Let us generalize checkers to an  $N \times N$  board.
- Let us say you have a strategy of playing checkers.
- The problem is to decide whether the strategy always wins.
- This problem is provably intractable.
- There will *never* be a program that can efficiently decide whether a given strategy will always win.

# Worse Than Exponential

- There are problems that need a runtime of  $2^{2^N}$ .
- Example: Presburger arithmetic
- These problems are called “double-exponential”.
- There are also “triple-exponential”.
- Problems that are not “K-exponential” are called “nonelementary”.

# Overview of Today's Lecture

- Growth of functions
- NP-complete problems
- Is  $P$  equal  $NP$ ?
- Even worse problems

# Next Week

- Midterm
- Noncomputability and undecidability