Computability Theory I

Introduction

Guoqiang Li

Shanghai Jiao Tong University

Sep. 19, 2014

Instructor and Teaching Assistant

- Guoqiang LI

Instructor and Teaching Assistant

- Guoqiang LI
  - Homepage: http://basics.sjtu.edu.cn/∼liguoqiang
  - Course page: http://basics.sjtu.edu.cn/∼liguoqiang/teaching/comp14/index.htm
  - Email: li-gq@cs.sjtu.edu.cn
  - Office: Rm. 1212, Building of Software
  - Phone: 3420-4167

Instructor and Teaching Assistant

- Guoqiang LI
  - Homepage: http://basics.sjtu.edu.cn/∼liguoqiang
  - Course page: http://basics.sjtu.edu.cn/∼liguoqiang/teaching/comp14/index.htm
  - Email: li-gq@cs.sjtu.edu.cn
  - Office: Rm. 1212, Building of Software
  - Phone: 3420-4167
  - YSSY: many IDs…
  - weibo: http://www.weibo.com/flyinsail
- TA:

Instructor and Teaching Assistant

- Guoqiang LI
  - Homepage: http://basics.sjtu.edu.cn/~liguoqiang
  - Course page: http://basics.sjtu.edu.cn/~liguoqiang/teaching/comp14/index.htm
  - Email: li-gq@cs.sjtu.edu.cn
  - Office: Rm. 1212, Building of Software
  - Phone: 3420-4167
  - YSSY: many IDs…
  - weibo: http://www.weibo.com/flyinsail
- TA:
  - Mingzhang HUANG: mingzhanghuang@gmail.com
  - Xiuting TAO: taoxiuting@gmail.com
- Office hour: Wed. 14:00-17:00 @ SEIEE 3-327

What do you think you can learn from this course?

Aim of the Course

- Q: Can the course improve the skill of programming?

Aim of the Course

- Q: Can the course improve the skill of programming?
  - A: Nope!

Aim of the Course

- Q: Can the course improve the skill of programming?
    - A: Nope!
- Q: Can the course improve the ability of algorithms?

Aim of the Course

- Q: Can the course improve the skill of programming?
  - A: Nope!
- Q: Can the course improve the ability of algorithms?
  - A: Perhaps, seldom.

Aim of the Course

- Q: Can the course improve the skill of programming?
  - A: Nope!
- Q: Can the course improve the ability of algorithms?
  - A: Perhaps, seldom.
- The course may provide a view of computation, an overlook of what we are doing in computer science, and a basic study of theoretical computer science.

Aim of the Course

- Q: Can the course improve the skill of programming?
  - A: Nope!
- Q: Can the course improve the ability of algorithms?
  - A: Perhaps, seldom.
- The course may provide a view of computation, an overlook of what we are doing in computer science, and a basic study of theoretical computer science.
- It is rather a philosophy than a technique, although some parts are quite technically.

It May Answer

It May Answer

- A software company that is developing a compiler capable of checking if a program contains a loop.

It May Answer

- A software company that is developing a compiler capable of checking if a program contains a loop.
- A hardware company that is determined to design a computer that can solve problems that no existing computers can solve.

It May Answer

- A software company that is developing a compiler capable of checking if a program contains a loop.
- A hardware company that is determined to design a computer that can solve problems that no existing computers can solve.
- A service provider that is working on a theorem prover that is supposed to answer every question about numbers.

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762
- Trains: 1807
- Airplanes: 1903

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762
- Trains: 1807
- Airplanes: 1903
- Supersonic: 1947 $343.2m/s$

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762
- Trains: 1807
- Airplanes: 1903
- Supersonic: 1947 $343.2m/s$
- Circular velocity 1957 $7.9km/s$
- Earth escape velocity 1959 $11.2km/s$
- Solar system escape velocity 1977 $16.7km/s$

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762
- Trains: 1807
- Airplanes: 1903
- Supersonic: 1947 $343.2m/s$
- Circular velocity 1957 $7.9km/s$
- Earth escape velocity 1959 $11.2km/s$
- Solar system escape velocity 1977 $16.7km/s$
- Q: Can we achieve in arbitrarily fast velocity?

History of Velocity

Human beings are keen on speed, and cannot stop the step to chase moving as fast as possible.

- Wheels: Mid-4th millennium BC.
- Automobiles: 1762
- Trains: 1807
- Airplanes: 1903
- Supersonic: 1947 $343.2m/s$
- Circular velocity 1957 $7.9km/s$
- Earth escape velocity 1959 $11.2km/s$
- Solar system escape velocity 1977 $16.7km/s$
- Q: Can we achieve in arbitrarily fast velocity?
  - Grandfather paradox

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600
- Basic arithmetic: Al Khwarizmi (780 - 850)

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600
- Basic arithmetic: Al Khwarizmi (780 - 850)
- ENIAC: 1946

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600
- Basic arithmetic: Al Khwarizmi (780 - 850)
- ENIAC: 1946
- NP problem
- The curse of exponential time

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600
- Basic arithmetic: Al Khwarizmi (780 - 850)
- ENIAC: 1946
- NP problem
- The curse of exponential time
- Advanced algorithms: simplex, DPLL, antichain.

Computation

Human beings are also keen on computation, and cannot stop the step to chase computing as complex as possible.

- Decimal system: AD 600
- Basic arithmetic: Al Khwarizmi (780 - 850)
- ENIAC: 1946
- NP problem
- The curse of exponential time
- Advanced algorithms: simplex, DPLL, antichain.
- Q: Can we achieve in arbitrarily complex computation?

What problems can be solved by computers?

Computer science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra

Let us begin to learn some basic astronomical phenomena!

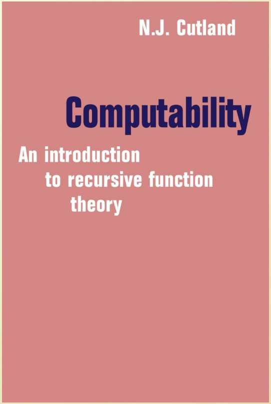
The technique part is quite similar to puzzles of wise men.

So, please have a fun!

Intuition is extremely important!

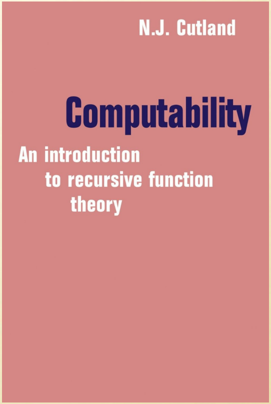
Reference Book

- Computability: An Introduction to Recursive Function Theory.
    - Nigel J. Cutland



Reference Book

- Computability: An Introduction to Recursive Function Theory.
    - Nigel J. Cutland
- plus extra reading materials.



Scoring Policy

- 10% Attendance.
- 20% Assignments.
- 70% Final exam.

Scoring Policy

- 10% Attendance.
- 20% Assignments.
  - Four assignments.
- 70% Final exam.

Scoring Policy

- 10% Attendance.
- 20% Assignments.
  - Four assignments.
  - Each one is 5 pts.
- 70% Final exam.

Scoring Policy

- 10% Attendance.
- 20% Assignments.
  - Four assignments.
  - Each one is 5 pts.
  - Work out individually.
- 70% Final exam.

Scoring Policy

- 10% Attendance.
- 20% Assignments.
  - Four assignments.
  - Each one is 5 pts.
  - Work out individually.
- 70% Final exam.
- There are also several homework. The answer may be given in exercise lectures, two or three times.

Special Requirements

A notebook and a pen.

Any questions?

0. Prologue

Effective Solutions

What problems can be solved by computers?

Famous Problems

- Diophantine equations
- Shortest path problem
- Travelling salesman problem (TSP)
- Graph isomorphism problem (GI)

Intuition

An effective procedure consists of a finite set of instructions which, given an input from some set of possible inputs, enables us to obtain an output through a systematic execution of the instructions that terminates in a finite number of steps.

Intuition

Intuition

Theorem proving is in general not effective.

Proof verification is effective.

Intuition

Theorem proving is in general not effective.

Proof verification is effective.

Unbounded search is in general not effective.

Bounded search is effective.

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?
- How is a problem instance (a graph) represented in a computer?

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?
- How is a problem instance (a graph) represented in a computer?
- How is the answer to a problem instance represented?

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?
- How is a problem instance (a graph) represented in a computer?
- How is the answer to a problem instance represented?
- How is an effective procedure formalized?

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?
- How is a problem instance (a graph) represented in a computer?
- How is the answer to a problem instance represented?
- How is an effective procedure formalized?

- Can every function from $\mathbb{N}$ to $\mathbb{N}$ be calculated by a C program?

Representation of Problem

- How does a computer solve the GI problem or the TSP Problem?
- How is a problem instance (a graph) represented in a computer?
- How is the answer to a problem instance represented?
- How is an effective procedure formalized?

- Can every function from $\mathbb{N}$ to $\mathbb{N}$ be calculated by a C program?
  - Negative.

Punchline

- In a formal theory of computability, every problem instance can be represented by a number and every number represents a problem instance.

Punchline

- In a formal theory of computability, every problem instance can be represented by a number and every number represents a problem instance.
- A problem is a function $f : \mathbb{N} \to \mathbb{N}$ from numbers to numbers.

Punchline

- In a formal theory of computability, every problem instance can be represented by a number and every number represents a problem instance.
- A problem is a function $f : \mathbb{N} \to \mathbb{N}$ from numbers to numbers.
- A problem is computable if it can be calculated by a program.

Everything is number!

Pythagoras

Decision Problem

Decision Problem

A problem $f : \mathbb{N} \to \mathbb{N}$ is a decision problem if the range $ran(f)$ of f is $\{0, 1\}$, where 1 denotes a 'yes' answer and 0 a 'no' answer.

A decision problem g can be identified with the set $\{n \mid g(n) = 1\}$.

Conversely a subset A of $\mathbb{N}$ can be seen as a decision problem via the characteristic function of A:

$$c_A(n) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{otherwise.} \end{cases}$$

Decision Problem as Predicate

A decision problem can be stated as a predicate $P(x)$ on number.

It relates to the problem-as-function viewpoint by the following characteristic function of $P(x)$:

$$c_P(n) = \begin{cases} 1, & \text{if } P(n) \text{ is valid,} \\ 0, & \text{otherwise.} \end{cases}$$

$$
\begin{aligned}
\text{Decision Problem} \quad &\Leftrightarrow \quad \text{Subset of } \mathbb{N} \\
&\Leftrightarrow \quad \text{Predicate on } \mathbb{N}
\end{aligned}
$$

Several Problems

Problem I

Is the function $tower(x)$ defined below computable?

$$tower(x) = \underbrace{2^{2^{\cdots^{2}}}}_{x}$$

Problem I

Is the function $tower(x)$ defined below computable?

$$tower(x) = \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{x}$$

Theoretically it is computable.

Problem II

Consider the function f defined as follows:

$$f(n) = \begin{cases} 1, & \text{if } n > 1 \text{ and } 2n \text{ is the sum of 2 primes,} \\ 0, & \text{otherwise.} \end{cases}$$

The Goldbach Conjecture remains unsolved. Is f computable?

Problem II

Consider the function f defined as follows:

$$f(n) = \begin{cases} 1, & \text{if } n > 1 \text{ and } 2n \text{ is the sum of 2 primes,} \\ 0, & \text{otherwise.} \end{cases}$$

The Goldbach Conjecture remains unsolved. Is f computable?

It is clearly computable even if we do not know what it is.

Problem III

Consider the function g defined as follows:

$$g(n) = \begin{cases} 1, & \text{if there is a run of exactly } n \text{ consecutive } 7'\text{s} \\ & \text{in the decimal expansion of } \pi, \\ 0, & \text{otherwise.} \end{cases}$$

It is known that π can be calculated by $4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\right)$.
Is g computable?

Problem III

Consider the function g defined as follows:

$$g(n) = \begin{cases} 1, & \text{if there is a run of exactly } n \text{ consecutive } 7'\text{s} \\ & \text{in the decimal expansion of } \pi, \\ 0, & \text{otherwise.} \end{cases}$$

It is known that π can be calculated by $4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\right)$. Is g computable?

We do not know whether it is computable or not.

Problem IV

Consider the function h defined as follows:

$$h(n) = \begin{cases} 1, & \text{if } n \text{ is the machine code of a } C \text{ program that} \\ & \text{terminates in all inputs,} \\ 0, & \text{otherwise.} \end{cases}$$

Problem IV

Consider the function h defined as follows:

$$h(n) = \begin{cases} 1, & \text{if } n \text{ is the machine code of a } C \text{ program that} \\ & \text{terminates in all inputs,} \\ 0, & \text{otherwise.} \end{cases}$$

This is the Halting Problem, a well known undecidable problem. In other words there does not exist any C program calculating h.

The only general approach to check if a function is defined on all numbers is to calculate it on all inputs.

Problem V

Consider the function i defined as follows:

$$i(x,n,t) = \begin{cases} 1, & \text{if on input } x, \text{ the machine coded by } n \\ & \text{terminates in } t \text{ steps,} \\ 0, & \text{otherwise.} \end{cases}$$

There could be a number of ways to interpret "t steps".

Problem V

Consider the function i defined as follows:

$$i(x,n,t) \quad = \quad \begin{cases} 1, & \text{if on input } x, \text{ the machine coded by } n \\ & \text{terminates in } t \text{ steps,} \\ 0, & \text{otherwise.} \end{cases}$$

There could be a number of ways to interpret "t steps".

The function i is intuitively computable.

Next Lecture

The examples try to suggest that in order to study computability one might as well look for a theory of computable functions.

Next Lecture

The examples try to suggest that in order to study computability one might as well look for a theory of computable functions.

We will begin with a machine model, register machine.

Homework

- home reading: diagonal method.
- home reading: Presburger arithmetic.