

Today's talk:

What: Computability and Complexity

Why: it is the foundation of computer science
most of us (me included) don't know too much
about it

Why should we: it gives you boundaries and under-
standing

Problem: this normally fills a course (or two, three)

Problem: it is quite difficult (at least for me)

What is this all about?

1. Why don't compilers warn me of infinite loops?

2. Is *bubble sort* a good search algorithm?

go through list, swap items if	4 7 3 5
one bigger than other, repeat	4 3 5 7
passes until list sorted	3 4 5 7

1. Computability: it is not possible.

2. Complexity: the problem could be solved in $O(n \log n)$.

bubble sort runs in $O(n^2)$. *mergesort* runs in $O(n \log n)$.

Questions

Computability: What cannot be computed at all?

Complexity: What cannot be computed efficiently?

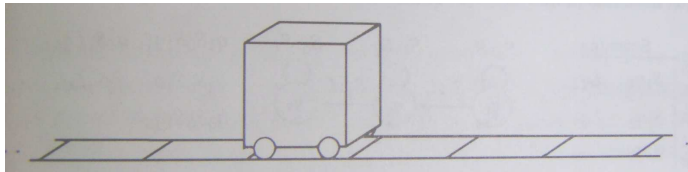
Computation

Computer platform: Need a *universal* model of computation.

Computation model: Different models, equivalent computational power:

- Turing machine (Turing)
- Lambda calculus (Church)
- Recursive functions (Gödel, Kleene, Herbrand)
- Post systems (Post)
- Register addressable machines (Shepherdson, Sturgis)

Turing machine: Idealised machine for carrying out computations



- tape marked into squares, reading head sits on some square
- each square is blank or has a stroke
- the computer reads a square
- writes a blank or a stroke
- optionally moves himself to another square

Programming a Turing machine: list of instructions that say what to do when reading some input when being in some stage.

Church-Turing thesis: Turing machine can compute all that is effectively computable, it is a universal computation device.

Sidestep: Enumerable sets

Enumerable sets: members can be enumerated

Enumeration: achieved by *enumeration function*

- *example:* set of positive numbers: $1, 2, 3, \dots$
- *example:* set of integers: $0, 1, -1, 2, -2, \dots$

Unintuitive results: measure infinite sets

- as many fractions in $[0, 1]$ as in $[0, 2]$

Not all sets are enumerable

- Cantor's diagonalisation proof
- Set of real numbers is not enumerable
- There is no bijection between natural numbers and real numbers
- Many many more real numbers than rational numbers

Uncomputability: Mechanical computation is limited

- Turing machine can compute all that can be computed.
- The number of Turing machines is enumerable (countably infinite).
- The number of functions is not enumerable (many many more).
- There must be functions that cannot be computed.
- *example*: the halting problem.

Incompleteness theorem: Axiomatic reasoning is limited

- Gödel: any consistent (sufficiently powerful) formal system is incomplete.
- You cannot use a mechanical operation to prove or disprove all formulas.
- Undecidability of first-order predicate logic.
- *example*: Cantor's continuum hypothesis.

Complexity

How difficult are problems: complexity of problem, not of solutions

Reductions amongst problems: problems can be translated to other problems: if we solve one of them, we solve all of them.

Complexity classes: we can make classes of problems that share computational characteristics.

P: solved by a Turing machine in polynomial time.

NP: solved by a non-deterministic Turing machine in polynomial time.

NP complete problems: a problem that is in NP and to which each NP problem can be reduced.

Satisfiability: for some formula, is there a value assignment that makes the statement true.

Hamiltonian cycle problem: find a cycle through a graph that visits each node exactly once.

Travelling salesman problem: find a Hamiltonian cycle with least weight.