Unsolvable Computer Problems

Changlin Li

July 24, 2013

What This Talk is About

1. Impossible computer problems
2. Hard computer problems
3. Why this is annoying
4. Why this is good

The Basics of Computing

1. The algorthm: finite sequence of steps that terminate
2. Algorithmic run-time
3. A notion of computability

The History

1. David Hilbert (1862-1943)
    1.1 23 Problems
    1.2 The *Entscheidungsproblem* (Decision Problem)
    1.3 Can all mathematical problems[1] be solved by an algorithm?
    1.4 "We must know, we shall know"
2. Kurt Gödel
    2.1 Completeness Theorem and Incompleteness Theorems
    2.2 Two Incompleteness Theorems
3. Alan Turing
    3.1 The father of computability
    3.2 The *bombe* (computer for breaking Enigma)

[1] Not really all, only those which can be expressed using *first-order logic*.

The Progression of Ideas

1. Primitive Recursion
    - 1.1 Successor function
    - 1.2 Constant functions
    - 1.3 Identity functions
    - 1.4 Composition
    - 1.5 Primitive recursion[2]
2. General Recursion
3. Computability (λ-calculus)

[2] $g, h \in \mathcal{C}$ and $n \geq 1$, then $f \in \mathcal{C}$ if $f(0, \overline{x}) = g(\overline{x})$ and $f(x_1 + 1, \overline{x}) = h(x_1, f(x_1, \overline{x}), \overline{(}x))$

Turing's Breakthrough

1. Infinite tape
2. Blank and 1
3. State of program

Impossible Problems

1. Most famous one is the *Halting Problem*
2. Can I determine if a program is completely broken?
3. What about if it's constant?

Annoyingly Impossible Problems

1. It is impossible to determine how to fix a program
2. Some programs even impossible to fix!

And The Big Whopper

Any implementation-independent question is impossible to solve

Figure: Well. . . yeah

But It Doesn't Matter!

Any implementation-independent question is impossible to solve

Figure: Lol

Modern Computation

1. Practically impossible vs theoretically impossible
2. Big-Oh notation

Fibonacci Numbers

1. Default definition $F(n) = F(n-1) + F(n-2)$. TERRIBLE! $(O(2^N))$
2. Smarter, store the repetitive values (dynamic programming) $(O(n))$
3. Even smarter, matrix multi $\begin{bmatrix} F_{k+2} \\ F_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix}$

   $(O(\log n))$

Miracle Sort[3]

```
Start with an array in memory.
loop:
    Check to see whether it's sorted.
    Yes? We're done.
    No? Wait a while and check again.
end loop
```

[3] From http://stackoverflow.com/questions/2609857/are-there-any-worse-sorting-algorithms-than-bogosort-a-k-a-monkey-sort

How Expedia and Orbitz Lie

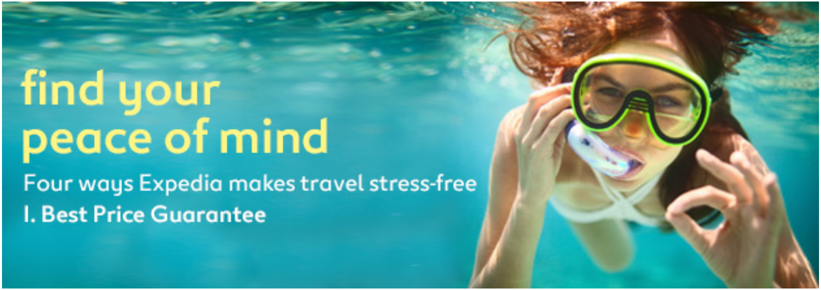

Figure: Expedia's Guarantee

The Traveling Salesman Problem

1. What if you want to make a tour of all the cities *only once*?
2. You're screwed!
3. Best algorithm so far: $O(n^2 2^n)$ a.k.a. “crazy bad”

When Practical Impossibility is Good!

Crypography and the idea of one-way functions

Competition Part 1

Imagine three people: Nabyl, German, and Juan. Nabyl wants to send messages to German using email. Juan though, happens to entirely control the email system. This means that he can modify and delete messages at will. More importantly, Juan is evil. He enjoys messing with emails that pass through his system.

Nabyl and German have one last meeting before one of them has to leave the office. For the next month, they can only communicate via email. They are resigned to the fact that Juan can delete their emails before they ever reach their intended recipient. However, they want to be able to make sure that Juan cannot impersonate one of them and that Juan cannot modify their emails without their knowledge. How can they devise a scheme to make this possible?

MACs

Message Authentication Codes

Competition Part 2

Nabyl and German realize that one or the other may request money during the time they are apart. Now they have a system set up to make sure that fraudulent requests by Juan are rejected. However, there remains the possibility that Juan could see one a request for money and, in order to bankrupt one of the receiver of the request, could simply duplicate the message and send it to that person over and over again. How can Nabyl and German guard against this?

Replay Attacks

One time codes

1. OAuth
2. SSL/TLS
3. And many more. . .

Competition Part 3

After they've left the office, Nabyl and German want to make sure that Juan can't read any of their messages. Unfortunately their only method of communicating (which includes how they might communicate any sort of encryption scheme to each other) is via Juan-controlled email (JCE^{TM}). How can they make sure that their messages can be read over the JCE and communicate how they decide to enforce their plan over JCE as well?

Public Key Cryptography

1. Diffie-Hellman
2. RSA

RSA

1. Create two prime numbers p and q
2. Compute $n = pq$
3. Compute $x = (p - 1)(q - 1)$
4. Compute e (coprime to x)
5. Compute $d^{-1} = e \mod x$
6. Encrypt message m with $m^e \mod n$
7. Decrypt ciphertext c with $c^d \mod n$