COMP2048
Theory of Computing

Tutorial 2 - Mathematics for Computer Scientists

Warm-up and Recall

- *Let f be a map from a set A to its power set $\wp(A)$ then $f: A \to \wp(A)$ is not surjective and $|A| \ll |\wp(A)|$.*

$$
\begin{array}{lcl cccccccccccc l}
f(1) & = & \{ & 1, & & & & & & & & & & & \} \\
f(2) & = & \{ & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10, & 11, & \ldots \} \\
f(3) & = & \{ & & 2, & 3, & 4, & & 6, & & 8, & & 10, & & \ldots \} \\
f(4) & = & \{ & 1, & & 3, & 4, & 5, & & 7, & & 9, & & 11, & \ldots \} \\
f(5) & = & \{ & 1, & 2, & & 4, & 5, & 6, & 7, & & 9, & & 11, & \ldots \} \\
f(6) & = & \{ & & & 3, & 4, & & 6, & 7, & & 9, & 10, & & \ldots \} \\
f(7) & = & \{ & 1, & & & & 5, & & 7, & & 9, & & & \ldots \} \\
f(8) & = & \{ & & & 3, & 4, & & & 7, & 8, & & & 11, & \ldots \} \\
f(9) & = & \{ & 1, & 2, & & & 5, & 6, & & & 9, & 10, & & \ldots \} \\
f(10) & = & \{ & 1, & 2, & & 4, & 5, & 6, & & & 9, & 10, & 11, & \ldots \} \\
f(11) & = & \{ & 1, & 2, & & 4, & & 6, & & & 9, & & 11, & \ldots \} \\
\vdots & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

$$
\begin{array}{lcl cccccccccccc l}
T & = & \{ & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10, & 11, & \ldots \}
\end{array}
$$

A set S is finite if there is a bijection {1,...,n} ↔ S for some n ≥ 0. In that case, we say |S| = n.

S is infinite if it is not finite.

A set S is countably infinite if there is a bijection f : N ↔ S.

S is uncountable if it is not countable.

Warm-up and Recall

- *Let f be a map from a set A to its power set $\wp(A)$ then $f: A \rightarrow \wp(A)$ is not surjective and $|A| \ll |\wp(A)|$.*

$$
\begin{array}{lllllllllllllll}
f(1) & = \{ & 1, & & & & & & & & & & & & \} \\
f(2) & = \{ & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10, & 11, & \dots & \} \\
f(3) & = \{ & & 2, & 3, & 4, & & 6, & & 8, & & 10, & & \dots & \} \\
f(4) & = \{ & 1, & & 3, & 4, & 5, & & 7, & & 9, & & 11, & \dots & \} \\
f(5) & = \{ & 1, & 2, & & 4, & 5, & 6, & 7, & & 9, & & 11, & \dots & \} \\
f(6) & = \{ & & & 3, & 4, & & 6, & 7, & & 9, & 10, & & \dots & \} \\
f(7) & = \{ & 1, & & & & 5, & & 7, & & 9, & & & \dots & \} \\
f(8) & = \{ & & & 3, & 4, & & & 7, & 8, & & & 11, & \dots & \} \\
f(9) & = \{ & 1, & 2, & & & 5, & 6, & & & 9, & 10, & & \dots & \} \\
f(10) & = \{ & 1, & 2, & & 4, & 5, & 6, & & & 9, & 10, & 11, & \dots & \} \\
f(11) & = \{ & 1, & 2, & & 4, & & 6, & & & 9, & & 11, & \dots & \} \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}
$$

$$
T = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, \dots \}
$$

CHALLENGE

By definition: An alphabet Σ is finite.

Let L = Σ+ is a Regular Expression (language).

Strings generated from L is { a, aa, aaa, aaaa, …}.

Proposition: Every language is either finite or countably infinite.

Let P is the set of the programs that can recognise (solve) some of the strings generated from L.

Proof: P is uncountably infinite.

Warm-up and Recall

- *Let f be a map from a set A to its power set $\wp(A)$ then $f: A \to \wp(A)$ is not surjective and $|A| \ll |\wp(A)|$.*

$f(1) = \{$	1,												$\}$
$f(2) = \{$	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	$\ldots$	$\}$
$f(3) = \{$		2,	3,	4,		6,		8,		10,		$\ldots$	$\}$
$f(4) = \{$	1,		3,	4,	5,		7,		9,		11,	$\ldots$	$\}$
$f(5) = \{$	1,	2,		4,	5,	6,	7,		9,		11,	$\ldots$	$\}$
$f(6) = \{$			3,	4,		6,	7,		9,	10,		$\ldots$	$\}$
$f(7) = \{$	1,				5,		7,		9,			$\ldots$	$\}$
$f(8) = \{$			3,	4,			7,	8,			11,	$\ldots$	$\}$
$f(9) = \{$	1,	2,			5,	6,			9,	10,		$\ldots$	$\}$
$f(10) = \{$	1,	2,		4,	5,	6,			9,	10,	11,	$\ldots$	$\}$
$f(11) = \{$	1,	2,		4,		6,			9,		11,	$\ldots$	$\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	

$T = \{$	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	$\ldots$	$\}$

A set S is finite if there is a bijection {1,...,n} ↔ S for some n ≥ 0. In that case, we say |S| = n.
S is infinite if it is not finite.
A set S is countably infinite if there is a bijection f : N ↔ S.
S is uncountable if it is not countable.

Georg Cantor (in 1873) used the "diagonalization" to solve the problems about measuring the sizes of infinite sets. However, this technique is more powerful than that.

Decidability

- A decision problem is a problem which has a yes or no answer on a given input.

- Example: “Is N countable infinite?” is a decision problem.

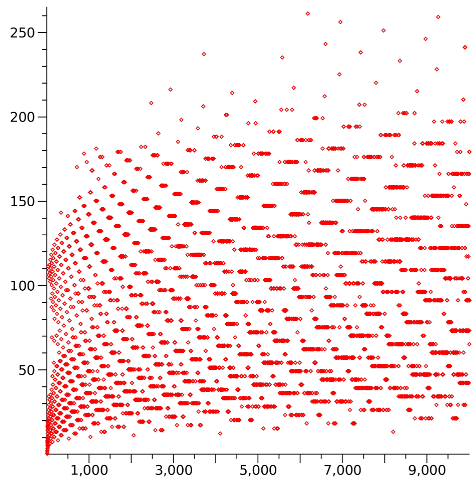
Decidability

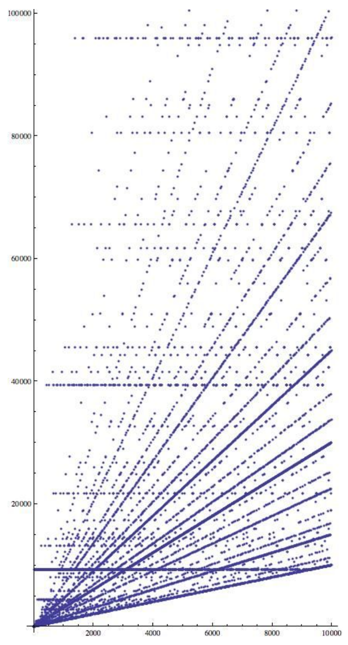
- A decision problem is a problem which has a yes or no answer on a given input.
- A decision problem is decidable if and only if there exists an algorithm that decides it (that is, an algorithm that determines yes or no correctly on any valid input).
- Otherwise, we say that the decision problem is undecidable.
- Example:
  - "Given a graph (set of vertices and edges between the vertices), how many vertices does it have?"
  - "Given a program P, does P terminate on input I?"

Decidability

- Does the following example on an input n terminate?

```
while (n > 1){
    if ( (n % 2) == 0){
        n = n/2;
    } else {
        n = 3 * n + 1;
    }
}
```





https://en.wikipedia.org/wiki/Collatz_conjecture

The Halting Problem

- The Halting Problem is undecidable. That is, the problem of "Given a program P and an input I, determine if P halts on input I" is undecidable.
- To show problems are undecidable, there are two common approaches:
  - Prove from first principles that the problem is undecidable.
  - Prove by reducing the problem to a known undecidable problem.
- Typically we use the second approach. Abstractly, we show problem $Q1$ is reducible to problem $Q2$:
  - Given an algorithm for deciding $Q2$, we could use it to decide $Q1$.
  - We prove:
    - If $Q2$ is decidable, then $Q1$ is decidable.
  - Taking the contrapositive:
    - If $Q1$ is undecidable, then $Q2$ is undecidable.

The Halting Problem

- We prove:
  - If $Q2$ is decidable, then $Q1$ is decidable.
- Taking the contrapositive:
  - If $Q1$ is undecidable, then $Q2$ is undecidable.
- Assume towards contradiction that there is an algorithm B that decides a new problem $P0$.
- Show that we can create a new algorithm A such that it decides the Halting Problem (or another problem we know is undecidable).
- Conclude by Turing's theorem that such a B should not have existed since the Halting Problem is undecidable and hence $P0$ is undecidable.

The Dollar Game

Surprise #3: There is a number associated to any graph G, called the *Euler number*, which plays a key role in analyzing whether or not the game is winnable. The Euler number g of a graph is defined as #edges − #vertices + 1. Let N denote the total number of dollars in the game at any time. Then (see [2, Theorem 1.9]):

- If N is at least g, then the game is always winnable.
- If N is less than g, then the game may or may not be winnable. (See Surprise #6 below for a stronger and more precise result.)

http://people.math.gatech.edu/~mbaker/pdf/g4g9.pdf

Fermat’s Little Theorem

- The tables below show the powers of nonzero numbers mod 7.

k	1	2	3	4	5	6
1^k mod 7	1	1	1	1	1	1
2^k mod 7	2	4	1	2	4	1
3^k mod 7	3	2	6	4	5	1
4^k mod 7	4	2	1	4	2	1
5^k mod 7	5	4	6	2	3	1
6^k mod 7	6	1	6	1	6	1

- For prime p and every integer a such that $a \not\equiv 0 \; mod \; p$ then $a^{p-1} \equiv 1 \; mod \; p$.
- We can see that the equation always generate an unique result from 1 to p-1.

- Can we use this property to make a practical use in Computer Science?

Fermat’s Little Theorem

- Suppose Alice and Bob want to exchange a key (number) through a public network channel.

LET’S THINK ABOUT USING MODULAR ARITHMETIC

WE JUST NEED ALICE AND BOB HAVE THE SAME KEY…

…NO MATTER WHAT IS THE VALUE OF THAT KEY

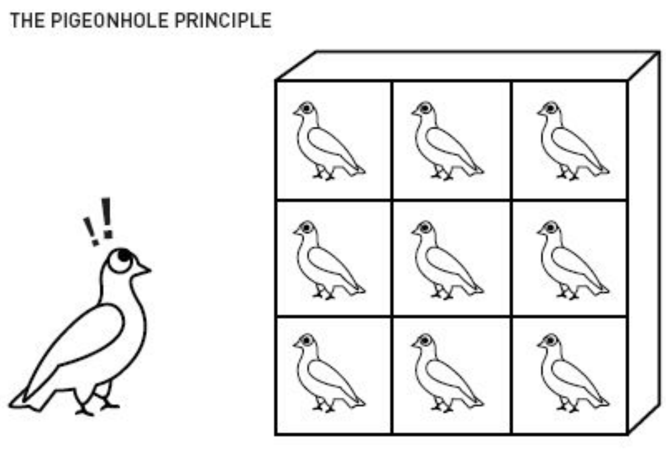
RSA ALORITHM REQUIRES BOTH ALICE AND BOB…

…HAVE TO GENERATE THEIR OWN PRIVATE KEY AND PUBPLIC KEY…

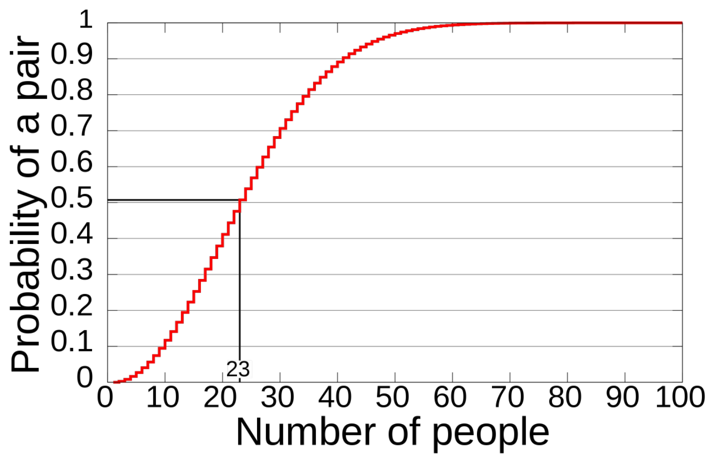
…BUT WHAT ABOUT ALICE AND BOB ARE BUILDING THE SAME KEY?

Laboratory

- Imagine a group of people. How big do you think the group would have to be before there's more than a 50% chance that two people in the group have the same birthday (assume a year has 365 days)?



Laboratory

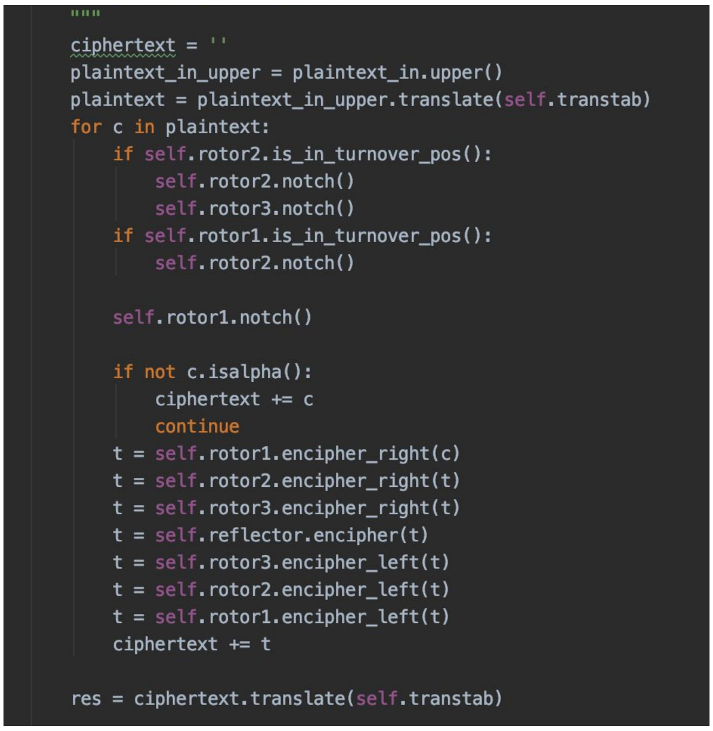


- If your code breaking algorithm runs more than a half of the total possibilities that the keys can have, then it won’t work ☺

Laboratory

- If you have troubles on doing laboratory and really need a hint? Take a look on modular arithmetic.

https://forms.gle/EqLcqNupwBfUv6Yp7



```python
"""
ciphertext = ''
plaintext_in_upper = plaintext_in.upper()
plaintext = plaintext_in_upper.translate(self.transtab)
for c in plaintext:
    if self.rotor2.is_in_turnover_pos():
        self.rotor2.notch()
        self.rotor3.notch()
    if self.rotor1.is_in_turnover_pos():
        self.rotor2.notch()

    self.rotor1.notch()

    if not c.isalpha():
        ciphertext += c
        continue
    t = self.rotor1.encipher_right(c)
    t = self.rotor2.encipher_right(t)
    t = self.rotor3.encipher_right(t)
    t = self.reflector.encipher(t)
    t = self.rotor3.encipher_left(t)
    t = self.rotor2.encipher_left(t)
    t = self.rotor1.encipher_left(t)
    ciphertext += t

res = ciphertext.translate(self.transtab)
```