GB13624 - Maths for Computer Science

Lecture 1 – Introduction to Proofs

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2024-10-02

Last updated October 2, 2024

Lecture 1 – Outline

In this lecture, we introduce the concept of **mathematical proofs**:

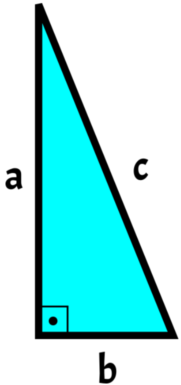
- **Section 1:** What are proofs, and why we need them;
- **Section 2:** Some proof methods;
- **Section 3:** Logical formulas and satisfiability;

This lecture covers the textbook's chapters 1, 2 and 3.

Part 1: Introduction to Proofs.

What is a proof?

Some concepts are easy to understand, but not easy to show that they are true.



- Pythagoras Theorem:

$$a^2 + b^2 = c^2$$

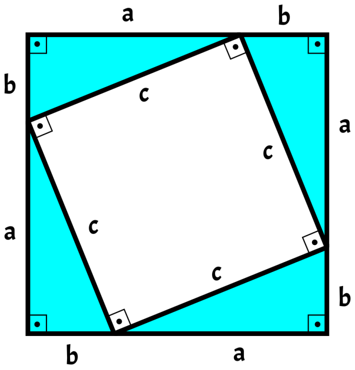
- It is easy to show this is true for **any one triangle**.
- But how do you show it is is true for **all** triangles?

The proof of the Pythagoras theorem is not obvious: there are more than 100 different proofs!

What is a proof?

One Pythagoras Proof

- **Proof**: by geometric construction
- Arrange four identical triangles;
- Show that internal angles are right;
- Internal square area: c^2
- External square area: $(a+b)^2$
- $(a+b)^2 = c^2 + 4(\text{area triangle})$
- $(a+b)^2 = c^2 + 4(\frac{ab}{2})$
- $a^2 + 2ab + b^2 = c^2 + 2ab$
- $a^2 + b^2 = c^2$

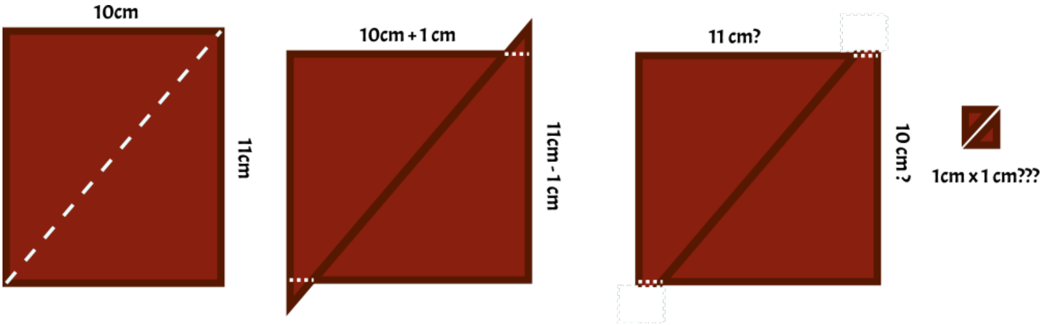


$\blacksquare$

The **Key Idea** of this proof is that a, b and c can be assigned to any right triangle.
But how do we find a new proof?

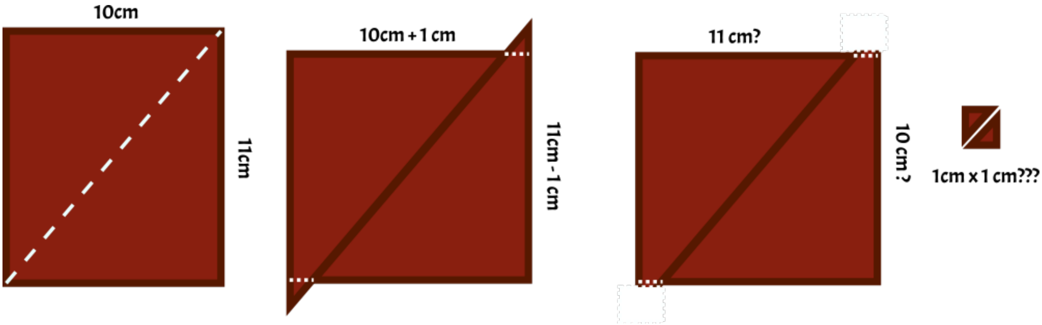
False Proofs – Infinite Chocolate!

- Be Careful! – It can be very hard to detect a wrong step in a proof.
- A wrong proof can be used to say something impossible is true.
- What is wrong in the proof below?



False Proofs – Infinite Chocolate!

- Be Careful! – It can be very hard to detect a wrong step in a proof.
- A wrong proof can be used to say something impossible is true.
- What is wrong in the proof below? Always check your assumptions!



Proofs and Computer Science

Why are proofs important for Computer Science?

- Proofs can be used **to show that a program is correct.**
  (or to show that a program is incorrect)

Examples

- Prove that the output of a program is correct for any input.
- Prove that certain input will cause a bug or crash in a program.
- Prove that a program finishes in N steps;

Proofs and Computer Science

Example:

Can you prove that the program below is correct (or incorrect)?

- To prove correctness, must prove for **any** input a, b, c
- To prove incorrectness, it is enough to show **one** input

```
int triangle_type(int a, int b, int c)
  // a, b, c are the length of the sides of a triangle;
  if (a == b)
    if (b == c)         return "all sides are equal";
    else                return "two sides are equal";
  else if (b == c)      return "two sides are equal";
  else                  return "all sides are different";
```

Proof Methods

How do we prove something?

For every nonnegative integer n, the value of $n^2 + n + 41$ is prime.

We could try to test values of n one by one:

$$n = 0; n^2 + n + 41 = 41, \text{prime}; n = 1; n^2 + n + 41 = 43, \text{prime}; n = 2; 47, \text{prime}; \ldots, \\ n = 20; 461, \text{prime...}$$

- When do we stop?
- ($n = 40; 41 \times 41$, is not prime...)

We need better ways to prove things!

Part 2: Proof Methods

Proof Concepts

Just like computer programs, we can explain proofs using natural language, or using formal structures.

The use of formal structures help us make sure that discussions about proofs are **accurate**, and help avoid confusions.

Let's review some concepts that are used when describing and discussing proofs.

Proof Concepts – Propositions

A proposition is a statement that is either True or False, and nothing else.

Proposition

- $2 + 3 = 5$
- $1 + 1 = 3$
- $513 \times 435 = 223165$
- There is no human taller than 3 meters.
- It rained on October, 3rd, 2020, 10:00 in Tokyo.
- Emacs is better than Vim.

Not proposition

- What is 2×8?
- Please give me cake.
- It is raining now.

Proof Concepts – Predicates

A predicate is a kind of proposition where the truth value depends on one or more variables:

- $P(n)$: n is a prime number;
- $L(N)$: The name N has five letters;
- $M(x, y)$: x and y are members of the same group;

Do not confuse predicates and number expressions!

Numeric expressions have numeric values, predicates have True or False values.

- $p(x) = x^2 + 3x + 1$. — $p(x)$ is a numeric expression;
- $P(X)$: $p(x+1) = p(x) + x + 1$. — $P(X)$ is a predicate about an expression;
- $K : P(X)$ is True for any $x \geq 0$. — K is a proposition;

Proof Concepts

Implication (IF)

An implication is a particular type of predicate that is very common in proofs:

$$P \implies Q$$

What does it mean?:

- $P \implies Q$: "If P is true, Q is true";
- $P \implies Q$: "When P is true, Q is true";

Note that this is a predicate. We could also write: $I(P, Q) : P \implies Q$

Proof Concepts

If and only iff (IFF)

Be careful! The implication only defines the value of Q when P is True:

- $P \implies Q$ If P is True, Q is True
- $P \implies Q$ If P is False, Q can be TRUE OR FALSE

Another common predicate is **If and only if.** It states that the two propositions must be true together.

- $P \iff Q$;
- IFF(P,Q);
- $P \implies Q$ AND $Q \implies P$;

Proof Concepts

Common Notation

The following notation is useful for creating logic predicates:

- And: X AND Y, $X \wedge Y$, X && Y
- Or: X OR Y, $X \vee Y$, X || Y
- Not: NOT(X), $\neg X$, $\overline{X}$
- If: IF X THEN Y, $X \implies Y$
- If and only if: IFF, $\iff$, $(X \implies Y) \wedge (Y \implies X)$
- Exists: $\exists n$ in $X(n)$; Every: $\forall n$ in $X(n)$ (We will talk about these soon)

There are more symbols that are used! Check the reference if you come across an unknown symbol.

Proving things using Predicates

We can use prepositions and predicates to organize our proofs into a "Logic Calculus". This is not very different from writing a computer program.

For every nonnegative integer n, the value of $n^2 + n + 41$ is prime.

We can define the expression $p(n) = n^2 + n + 41$, and the predicate:
$P : \forall n \in \mathbb{N}, n \geq 0, p(n)$ is prime

Then we can try to derivate new predicates to show that the predicate above is True in all situations.

Inference Rules

To show that something is true using logic predicates, we organize them in "well known" constructs, just like in programming.

These constructs are "inference rules", they usually have formal names, and are shaped as follows:

$$\frac{P, Q, R}{X}$$

The above structure means: "if we show that propositions P, Q, and R are true, then X must be true".

Proving an Implication

Direct Proof by Modus Ponens inference rule

The *Modus Ponens* inference rule says that:

$$\frac{P, P \implies Q}{Q}$$

"If P is true, and P implies Q is true, then Q must be true";

So to prove Q, we have to prove that P, and then that P **implies** Q is also true.

To prove the implication directly, we can assume that P is true, then show that Q must be true, step by step.

Proving an Implication

Direct Proof

Prove the implication: IF $0 \leq x \leq 2$, THEN $-x^3 + 4x + 1 > 0$

Proof.

- Let's assume $0 \leq x \leq 2$ is true;
- We can rewrite $-x^3 + 4x$ as $x(2 - x)(2 + x)$
- If $0 \leq x \leq 2$, then $x \geq 0$, $(2 - x) \geq 0$, and $(2 + x) \geq 0$
- $x \times (2 - x) \times (2 + x) \geq 0$
- $x(2 - x)(2 + x) + 1 > 0$
- $-x^3 + 4x + 1 > 0$

$\square$

Proving an Implication
Contrapositive

Another way to prove an implication is to "prove the contrapositive". This means using the following inference rule:

$$\frac{\text{NOT}(Q) \implies \text{NOT}(P)}{P \implies Q}$$

So:

- If we show that "when Q is false, then P must be false"
- This is equivalent to show that "When P is true, Q must be true";

Proving an Implication

Contrapositive

Theorem: if r is irrational (P), then $\sqrt{r}$ is also irrational (Q).

Proof.

We prove the contrapositive: If $\sqrt{r}$ is rational ($\neg Q$), then r is also rational ($\neg P$).

- If $\sqrt{r}$ is rational, then $\sqrt{r} = \frac{m}{n}$.
- m and n are integers (definition of rational numbers)
- Square both sides: $r = \frac{m^2}{n^2}$.
- m^2 and n^2 are also integers, so r is rational.

$\square$

Proving "If and only If"

Remember that "If and only If" can be defined as:

$$\frac{P \implies Q, Q \implies P}{P \iff Q}$$

So to prove $P \iff Q$, we can first prove the implication from P to Q, and then prove the implication from Q to P.

This is useful to show equivalence between two mathematical statements.

Proof By Cases

Example

Let's say you are refactoring code, and you want to profe that the two code samples below are equivalent. How would you do it?

Code 1

```
If (X > 0 OR (X <= 0 AND Y > 100))
  print("Hello!")
```

Code 2

```
If (X > 0 OR Y > 100)
  print("Hello!")
```

Proof By Cases

Definition

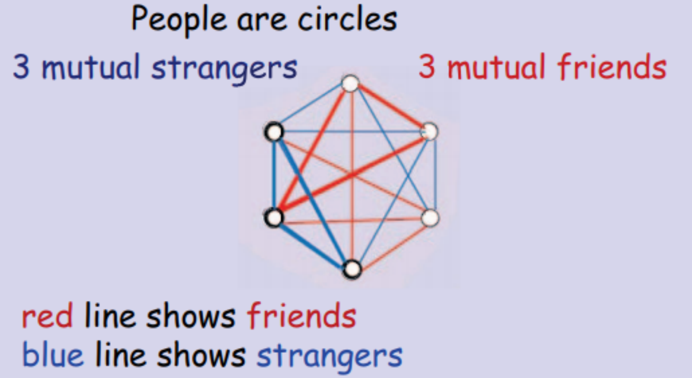
Proof By Cases, is a proof technique that uses the idea of "divide and conquer".

You break one complicated problem into easier, smaller sub-problems.

Important! When you create the cases, make sure that all possible cases are covered!

Example: Friends and Strangers

Theorem: In a group of 6 people, where **every pair** is either a friend or a stranger, then we **always** have at least a set of 3 mutual friends or a set of 3 mutual strangers.



Example: Friends and Strangers

Proof.

The proof is by case analysis. Let "A" be one of the six people. There are two cases:

1. Among the 5 other people, at least 3 are friends with A;
2. Among the 5 other people, at least 3 are strangers with A;

Let's assume case (1). Let's call the three friends B, C, D. There are two subcases:

- A. B-C, C-D, or B-D are friends. We have now 3 mutual friends with A and the pair here.
- B. B-C, C-D and B-D are strangers. This makes a 3 mutual strangers set with the three pairs.

This means that in case 1, the theorem holds. It is easy to see that case 2 is symmetrical to case 1. □

A WRONG Proof By Cases

Theorem: $2a^2 > a$, for all $a \in \mathbb{Z}$.

Proof.

The proof is by case analysis.

1. Case 1: $a > 0$;
   - $2a^2$ is equal to $2a \times a$
   - Since $a > 0$ and $a \in \mathbb{Z}$, then $a \geq 1$
   - $2 \times 1 \times 1 > 1$
2. Case 2: $a < 0$
   - Since $a < 0$ and $a \in \mathbb{Z}$, then $a \leq 1$
   - For any negative a, a^2 is positive, so $a^2 > a$.

Because the theorem holds for case (1) and case (2), it holds for all possible cases. $\square$

What is wrong with this proof?

Proof By Contradiction

Definition

"Proof by Contradiction" is a technique where you show that **the negative of the theorem implies a false fact to be true**.

For a simple example: "If gravity did not exist, then we would all be flying. Since we are not flying, then gravity must exist."

Sometimes, it can be easy to create a proof by contradiction by finding a good counter-example. Other times, we have to find an absurd consequence of the negative.

Use "Proof by Contradiction" to prove the following theorem:
Theorem: $\sqrt{2}$ is an irrational number.

Careful!!! Contradiction $\neq$ Contrapositive

Proof by Contradiction

Example

Proof.

We use proof by contradiction, and assume $\sqrt{2}$ is rational.

1. $\sqrt{2} = \frac{m}{n}$; $m, n \in \mathbb{Z}$; $n \neq 0$, and m, n have no common factors.
2. $n\sqrt{2} = m$ and squaring both sides give $2n^2 = m^2$.
3. m^2 is even (because $n^2 = \frac{m^2}{2}$)
4. If m^2 is even, then m is even too. So $m = 2k$ for some integer k.
5. So, $2n^2 = (2k)^2$, which leads to $n^2 = 2k^2$.
6. Following the logic of (3) and (4), n^2 is even, and n is even too.
7. However, if m and n are even, it is a contradiction with (1).

$\square$

Well Ordering Principle

Definition

The Well Ordering Principle (WOP) is a very useful principle in mathematics, that can also look a little bit "obvious":

Every non-empty set of
Non-negative Integer Numbers ($\mathbb{Z}^+$)
has one smallest element

Well Ordering Examples

- What is the smallest age among students in Tsukuba?
- What is the smallest number of coins that adds to 876 yens?
- What are the smallest integers m and n so that $x = \frac{m}{n}$?

Well Ordering Principle Proof Example

We can re-write the proof that $\sqrt{2}$ is irrational using WOP.

Proof.

1. $\sqrt{2} = \frac{m}{n}$; $m, n \in \mathbb{Z}$; $n \neq 0$;
2. By WOP, there is a **smallest** m and n so that $\sqrt{2} = \frac{m}{n}$
3. $n\sqrt{2} = m$ and squaring both sides give $2n^2 = m^2$.
4. m^2 is even (because $n^2 = \frac{m^2}{2}$)
5. If m^2 is even, then m is even too. So $m = 2k$ for some integer k.
6. So, $2n^2 = (2k)^2$, which leads to $n^2 = 2k^2$.
7. Following the logic of (4) and (5), n^2 is even, and n is even too.
8. If m and n are even, then $\sqrt{2} = \frac{m/2}{n/2}$, and $m/2$, $n/2$ are smaller than m, n, contradicting the WOP.

Why is the WOP useful?

General form for a WOP proof

The WOP gives us a general framework to produce proofs by contradiction:

- Structure your theorem around predicate $P(n)$, where $n \in \mathbb{N}$.
- Define a set C of counter examples, so that $C ::= \{n \in \mathbb{N} | P(n) \text{ is false}\}$.
- By WOP, consider the minimum element $m \in C$.
- Find a contradiction, for example:
  - if m exists, then it implies in the existence of a smaller element $m' < m, m' \in C$.
  - if m exists, then actually $P(m)$ is true, and m is not actually in C.
- Therefore, the minimum element m does not exist, the counter example set C does not exist, and $P(n)$ is true for all n.

WOP Proof examples:

Let's see two quick examples of proofs using WOP. Try doing these two proofs by yourself first:

- **Theorem:** Every $n > 1, n \in \mathbb{N}$ is a product of prime numbers.
- **Theorem:** For every $n \in \mathbb{N}$, $P(n) : n + 8 = 5a + 3b; a, b \in \mathbb{N}$.
  (for every n, $n + 8$ is composed of a sum of 3s and 5s)

WOP Proof example I: Prime factors

Theorem: Every integers bigger than 1 is a product of prime numbers.

Proof.

Proof by contradiction using the WOP.

- Assume, by WOP, that m is the smallest $\mathbb{N}$ that is not a product of prime numbers.
- Obviously m is not a prime, so $m = a_1 a_2 \ldots a_n$, where a_i is not prime.
- Is a_i a product of prime numbers?
  - If a_i is a product of prime numbers, then $a_i = p_1 p_2 \ldots p_n$, and m is now a product of prime numbers (contradiction)
  - If a_i is not a product of prime numbers, then m is not the **smallest** product of prime numbers. (contradiction)

□

WOP Proof example II: Postal Numbers

Theorem:

For every n, $n + 8$ is composed of 3s and 5s.

Proof.

Proof by contradiction using the WOP

- First, we quickly verify that $P(n)$ is true for 0..8
- By WOP, we assume that there is some minimum $m > 8$ where $P(m)$ is false.
- If $P(m)$ is false, then $m + 8$ cannot be composed of 3s and 5s.
- If m is minimum, then $P(m - 8)$ is true, and m is composed of 3s and 5s.
- If m is composed of 3s and 5s, then $m + 8$ is $m + 3 + 5$, and $P(m)$ is true! (Contradiction)

$\square$

Why Mathematical Language?

Human language can be imprecise, so we have mathematical language that can be more specific:

"Go to the supermarket to buy 1 milk pack. If they have eggs, buy 12."

Which of the following is correct?

- If the supermarket has eggs, buy 1 milk pack and 12 eggs.
- If the supermarket has eggs, buy 12 milk packs.

To avoid this imprecision, we prefer to use mathematical language when talking about logical relationships and proofs.

Predicate Calculus and Logical Operators

The mathematical language that we use in this lecture is called *Predicate Calculus*. Predicate calculus connects **Predicates** and **Propositions** using logical operators.

Many of the logical operators you already know from boolean logic:

- **AND**, **OR**, **XOR**, **NOT**, etc...

There are a few more unusual logical operators too:

- **IMPLIES**, **IFF**, **FOR ALL**, **EXISTS**

Predicate Calculus and Truth Tables

To evaluate a formula in predicate calculus, we can use Truth Tables, which describe every possible truth value to each proposition.

Example: P AND Q IMPLIES R

P	Q	R	P AND Q	P AND Q IMPLIES R
TRUE	**TRUE**	**TRUE**	**TRUE**	**TRUE**
TRUE	**TRUE**	FALSE	**TRUE**	FALSE
TRUE	FALSE	**TRUE**	FALSE	**TRUE**
TRUE	FALSE	FALSE	FALSE	**TRUE**
FALSE	**TRUE**	**TRUE**	FALSE	**TRUE**
FALSE	**TRUE**	FALSE	FALSE	**TRUE**
FALSE	FALSE	**TRUE**	FALSE	**TRUE**
FALSE	FALSE	FALSE	FALSE	**TRUE**

Logic Operators: "For All" and "Exists"

Two of the operators we mentioned are special, and deserve some special attention:

Operator: For all

For a predicate $P(x)$, FOR ALL $P(x)$ is True if $P(x)$ is true for **every** value of x. It is equivalent to a chain of "AND"s:

$$F(P(x)) : \forall x_i \in X, P(x_0) \wedge P(x_1) \wedge \ldots \wedge P(x_n)$$

Operator: Exists

For a predicate $P(x)$, EXISTS $P(x)$ is True if $P(x)$ is true for **any** value of x. It is equivalent to a chain of "OR"s:

$$E(P(x)) : \exists x_i \in X, P(x_0) \vee P(x_1) \vee \ldots \vee P(x_n)$$

Validity and Satisfiability

The logical operators "exists" and "for all" are closed linked to the concepts of "Validity" and "Satisfiability":

- **A logical formula is Valid if**: The formula evaluates for true for every possible assignment of every variable.

  For example: $P \land$ NOT $P \implies Q$ is valid.

- **A logical formula is Satisfiable if**: The formula evaluates for true for at least one possible assignment of variables.

  For example: $P \lor (Q \land R)$ is satisfiable

Equivalence

Comparison of Two Formulas

Another related concept is Equivalence. We say that two logical formulas are equivalent, if their result is identical for every variable assignment.

For example: NOT $(P \vee Q)$ is equivalent to NOT $P \wedge$ NOT Q

(DeMorgan's Law)

The equivalence of two formulas is useful when rewriting code, and showing that two different pieces of code have the same result.

Verifying Validity, Satisfiability, and Equivalence

How can we verify that some logical formulas are (Valid | Satisfiable | Equivalent)?

One way to do that is to examine the truth tables of the formulas:

- **A formula is satisfiable**: At least one line in the truth table evaluates to TRUE.
- **A formula A is valid**: The formula $\neg A$ is NOT satisfiable.
- **Two formulas (X,Y) are equivalent**: The formula $(A \wedge B) \vee (\neg A \wedge \neg B)$ is valid.

However, there is a problem: The size of the truth table is 2^n, where n is the number of variables!

The Satisfiability Problem

Consider the problem of simplifying a computer program: Given a program defined as a logical formula A, we want to find a smaller formula B that has the same functionality.

We can test if a certain B is equivalent to A by testing if the expression $A \iff B$ is **valid**. Alternatively, We can test that B is **not** equivalent to A by testing if NOT $(A \iff B)$ is **satisfiable**. If we can find only one variable assignment where A and B are not equal, then we can discard the program candidate B.

This kind of analysis is useful for making programs run faster, or for creating simpler and cheaper hardware.

The Satisfiability Problem

Proving equivalences

The basic algorithm for proving equivalence in a SAT problem is to test each combination of variables (each line in the truth table). As we discussed before, the number of lines is 2^n, so this can take a very long time.

Interestingly, if we KNOW one set of variables that satisfy the formula, it is very quick to test it. Just evaluate the formula.

This characteristic of SAT: "Very slow to find the answer, very fast to check the answer", is one of the key characteristics of NP-hardness. If you can find a quick solution to the SAT problem, you would become a very famous computer scientist!

Conclusion

Important Ideas from this lecture

- Proofs are sequences propositions that establish the truth or falsehood of an statement.
- Proof Techniques are organized ways to construct a proof;
  - Proof By Cases;
  - Contradiction;
  - Well Ordering Principle, etc;
- Predicate Logic use logical operators to show the truth or falsehood of a predicate;
  - Concepts of Validity and Satisfiability;
- There is a close relationship between proving an statement, and proving the correctness of a computer program

Reminder: Exercise sheet at manaba

- The homework for this lecture is on manaba;
- You have to submit your homework before the next lecture;
- If you start the homework now, you can ask questions during the lecture time;
- You can discuss the exercise with other students, but your homework is **individual**

Slide Credits

These slides are based on "Mathematics For Computer Science (Spring 2015)", by Albert Meyer and Adam Chlipala, MIT OpenCourseWare. `https://ocw.mit.edu`.

Individual images in some slides might have been made by other authors. Please see the following slides for information about these cases.

Image Credits I

1. Friends or Strangers Image from "Math for Computer Science" MIT-OCW slides