TCS Profession and Communication Skills

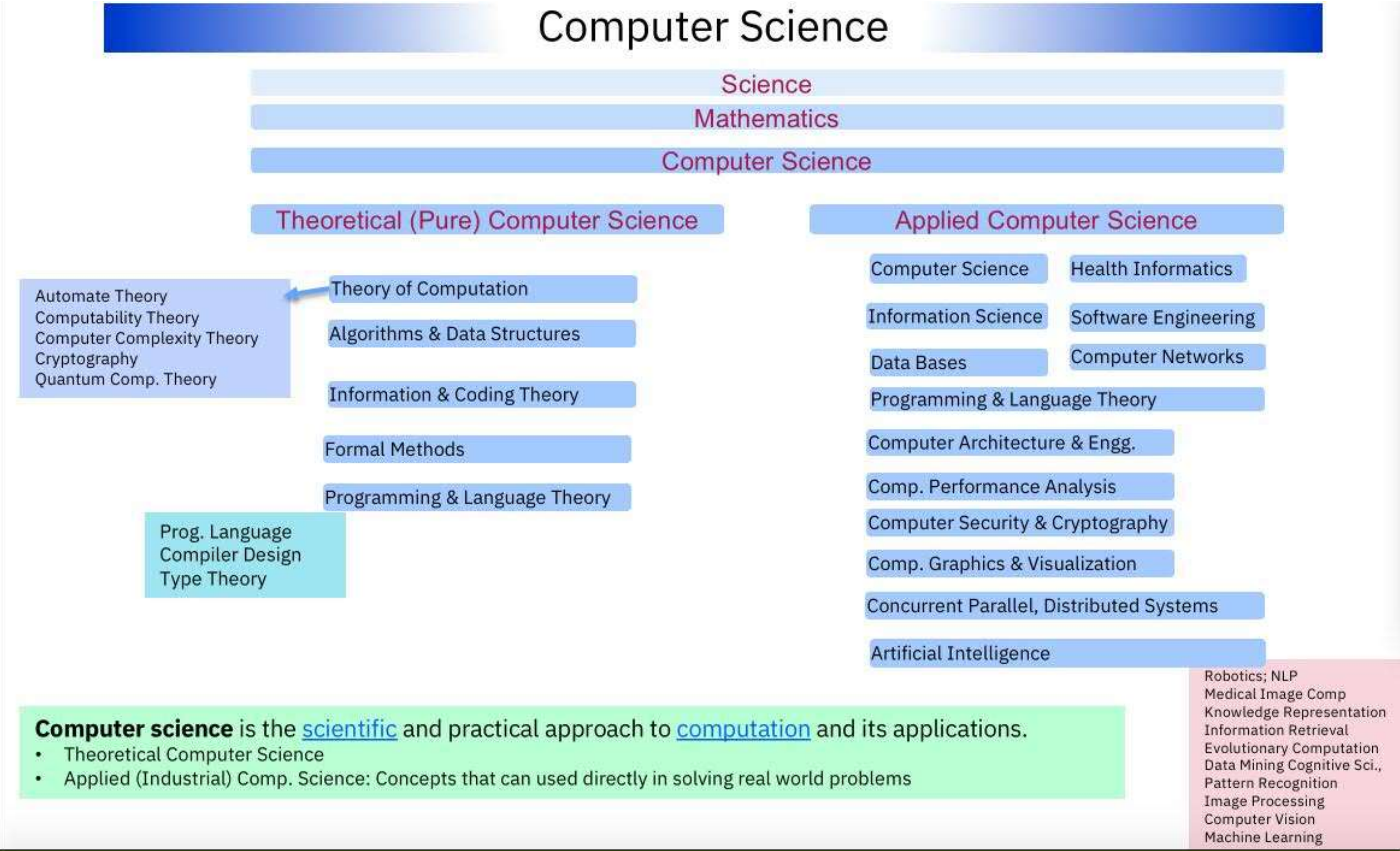
NITIN SAXENA (NITIN@CSE.IITK.AC.IN)

[*WITH HELP FROM INTERNET SOURCES]

2024; AUG 21, 23, 28, 30; SEP 4, 6

Is Theory Maths?

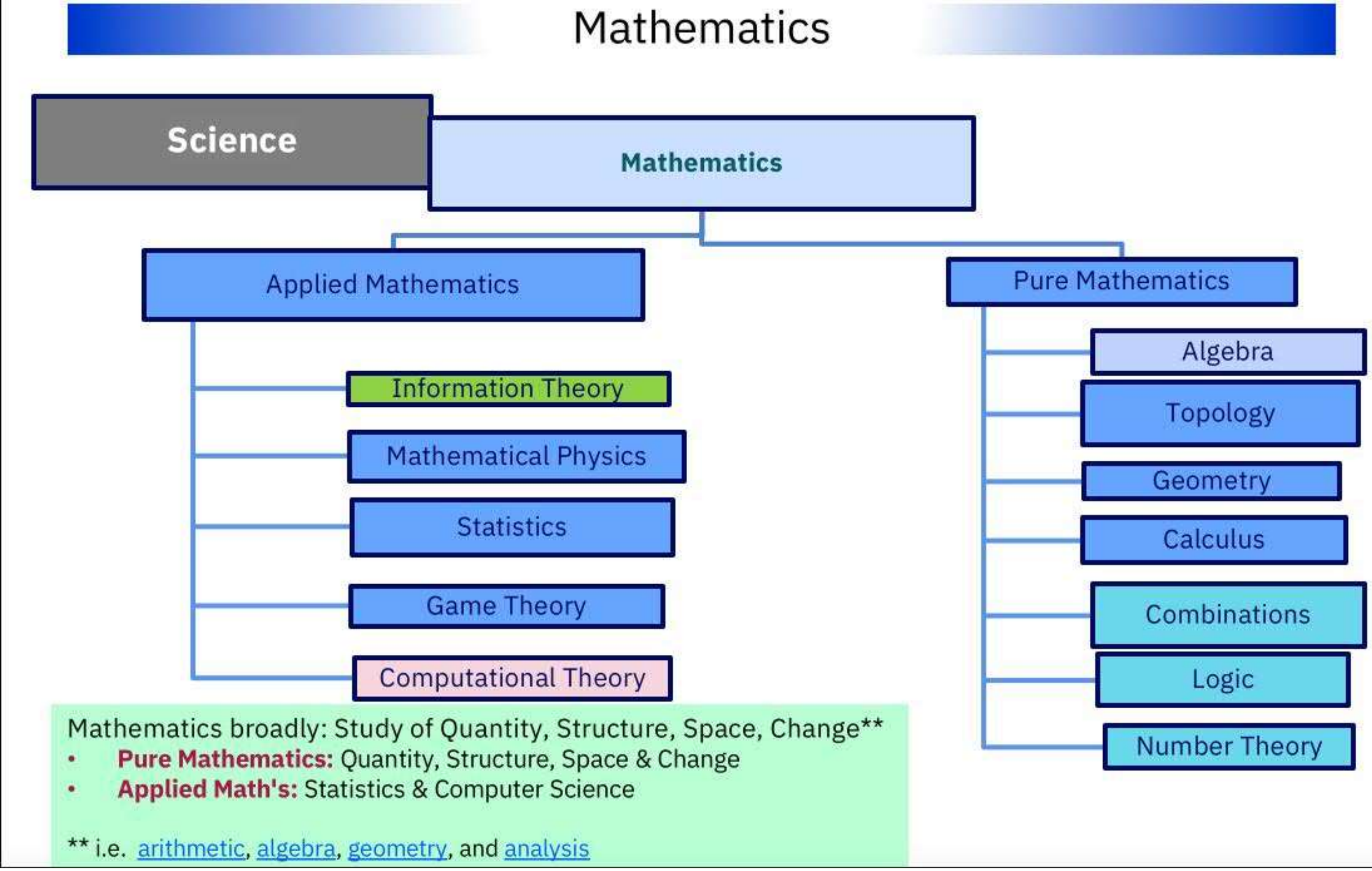
- Theory = Theoretical Computer Science (TCS)
- Theory is the art of
  - designing algorithms
  - guessing their efficacy
  - proving your guess
- No theory without compute
- Maths is a larger framework
  - doesn’t need compute
  - needs aesthetics!
  - different examples based on scientific areas



Is Maths Theory?

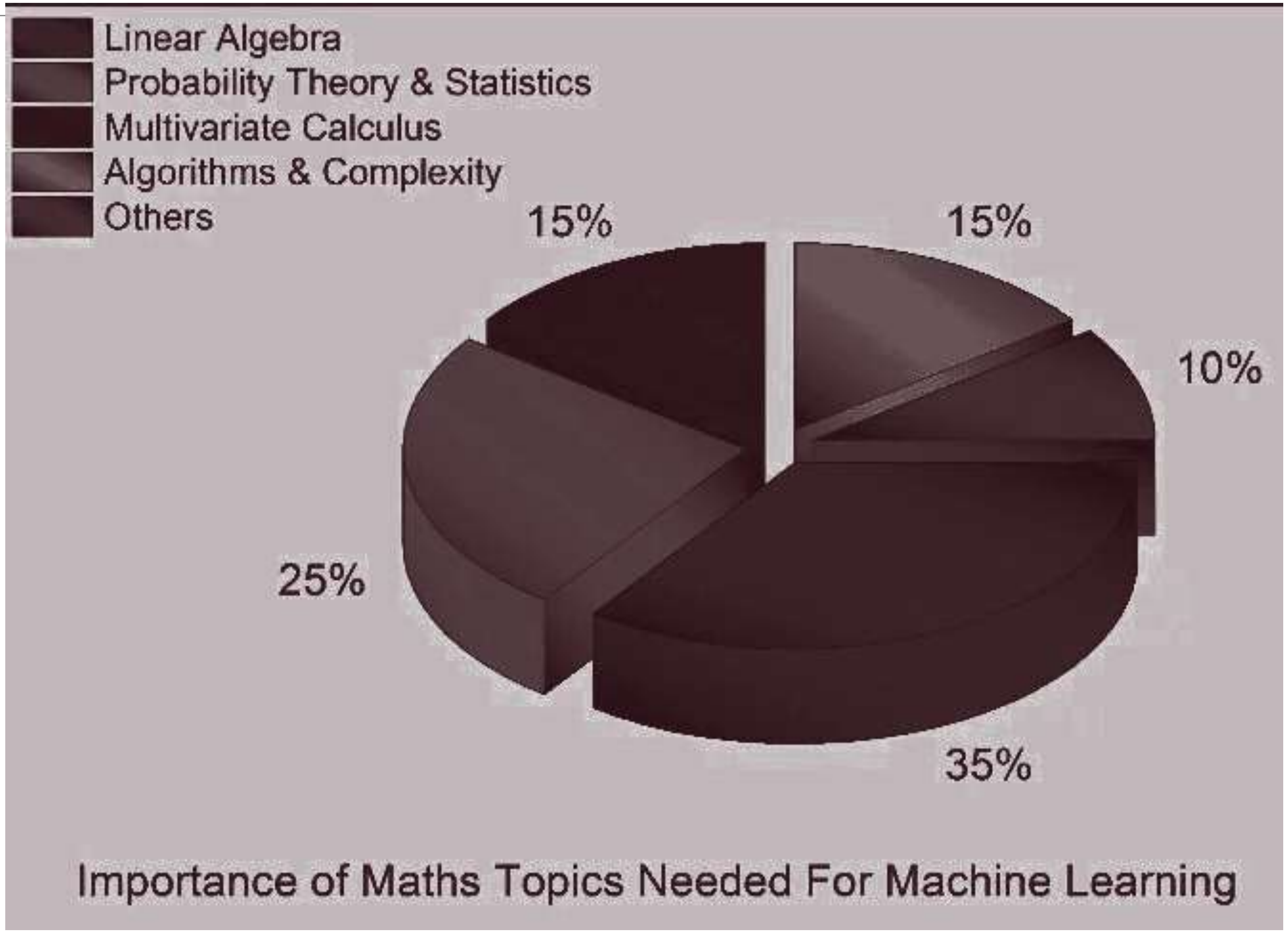
- Maths *may* interact with computation
  - but it largely exists on it own
  - = abstract or pure
- Mathematicians don't need computers
- Our motivation is *all* from computing
- Maths textbooks have a very precise and rigorous notation
- Theory manuscripts are written in a notation that has a computational feel!

- Theory enriches Maths and vice versa!

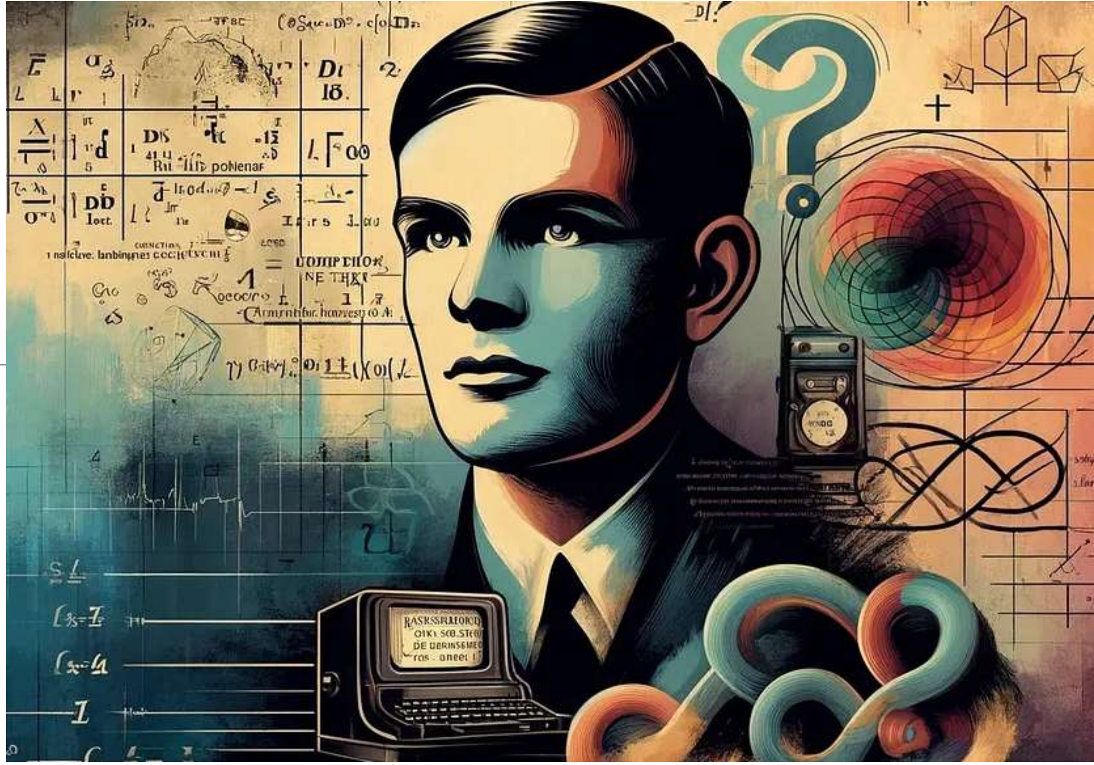


Is Maths necessary in CS?

- Linear Algebra → ML, Graphics
- Probability → Machine Learning
- Calculus → Deep learning, Complexity
- Number theory → Cryptography, Error-correcting code
- Geometry → Vision, Motion planning
- Algebra + Combinatorics → Communication, Storage
- Game theory → Matching, Auction, Trading, Recommendation system



Theory statements



- The Divine Dilemma: Can God Solve the Halting Problem?
- The halting problem is undecidable.
- Theorem: The halting problem is undecidable.
- Theorem [Turing, 1936]: The halting problem is undecidable.
- Define the concepts:
  - *Halting problem* is a decision problem that asks if a computer program will terminate, or run forever, given a description of the program and an input.
  - *Undecidable problem* is a decision problem that cannot be solved by an algorithm.

- Algorithm is a set of rules that must be followed when solving a particular problem.
- Define a *Turing Machine*
  - as a symbolic description "of rules that must be followed".
- Definition: An *algorithm* is a Turing machine.

Theory statements in Systems

- Mathematically *model* the real-life problem.
- Allocate T tasks to P processors.
- Define the concepts:
  - *Feasible*
  - *Optimal*
  - *Hard, Easy*
  - *NP-hard*

The Journal of Real-Time Systems, 4, 145–165 (1992)

Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy*

K.W. TINDELL, A. BURNS AND A.J. WELLINGS
Real Time Systems Research Group, Department of Computer Science, University of York, England

Abstract. A distributed hard real time system can be composed from a number of communicating tasks. One of the difficulties with building such systems is the problem of where to place the tasks. In general there are P^T ways of allocating T tasks to P processors, and the problem of finding an optimal feasible allocation (where all tasks meet physical and timing constraints) is known to be NP-Hard. This paper describes an approach to solving the task allocation problem using a technique known as *simulated annealing*. It also defines a distributed hard real-time architecture and presents new analysis which enables timing requirements to be guaranteed.

1. Introduction

Building real-time systems on distributed architectures presents engineers with a number of challenging problems. One issue is that of scheduling the communication media, another concerns the allocation of software components to the available processing resources. Distributed systems typically consist of a mixture of periodic and sporadic tasks, each with an associated deadline and possibly precedence constraints. Failure to meet the deadlines of critical tasks may lead to a catastrophic failure of the system, and consequently off-line analysis of allocation and processor scheduling is required to guarantee task deadlines.

In general, the three activities of task allocation, processor scheduling and network scheduling are all NP-hard problems (Burns 1991). This has led to a view that they should be considered separately. Unfortunately, it is often not possible to obtain optimal solutions (or even feasible solutions) if the three activities are treated in isolation. For example, allocating a task T to a processor P will increase the computational load on P, but may reduce the load on the communications media (if T communicates with tasks on P), hence the response time of the communication media is reduced, allowing communications deadlines elsewhere in the system to be met. The tradeoffs can become very complex as the hard real-time architecture becomes more expressive; a simple and scalable approach is required.

Previous approaches to solving the task allocation problem have mostly concentrated on graph theoretic algorithms [for example (Chen and Yur 1990; Chu and Lan 1987)] or heuristics [for example (Bannister and Trevedi 1983; Houstis 1990)]. Most have tried to maximize system throughput (i.e., minimize the computational and communication resource requirements for tasks in the system), often by reducing *bottlenecks*, resulting in allocations which may or may not be schedulable. However, these approaches do not take a global view; they rely on the observation that fast systems (i.e., ones which maximize system

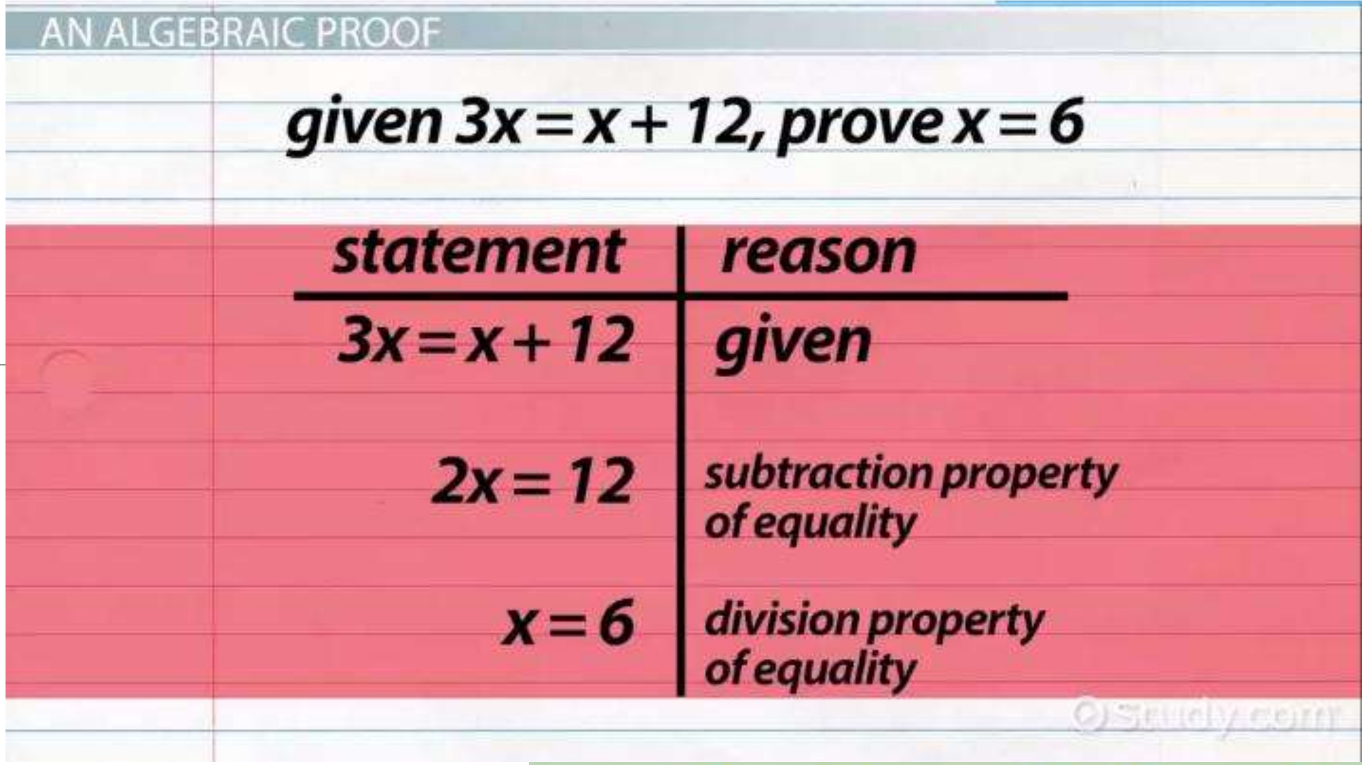
*This work was supported in part by British Aerospace (Commercial Aircraft) Ltd, and the UK Department of Trade and Industry.

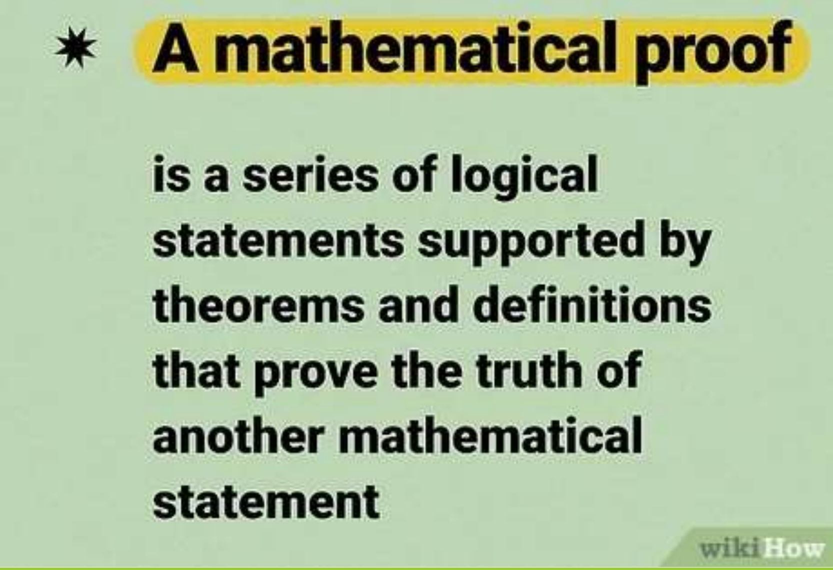
Assignment 1:

- Write 2-3 lines on an area of Theoretical Computer Science that you like. Why do you like it?
- Write 2-3 lines on an area of Mathematics that you like. Why do you like it?
- Write 2-3 lines on an application of Mathematics that you have heard about.
- Give 2 examples of improperly formulated “theory statements” that you came across in college or online or in the media. Next, write down their correctly formulated versions.

Prove your beliefs

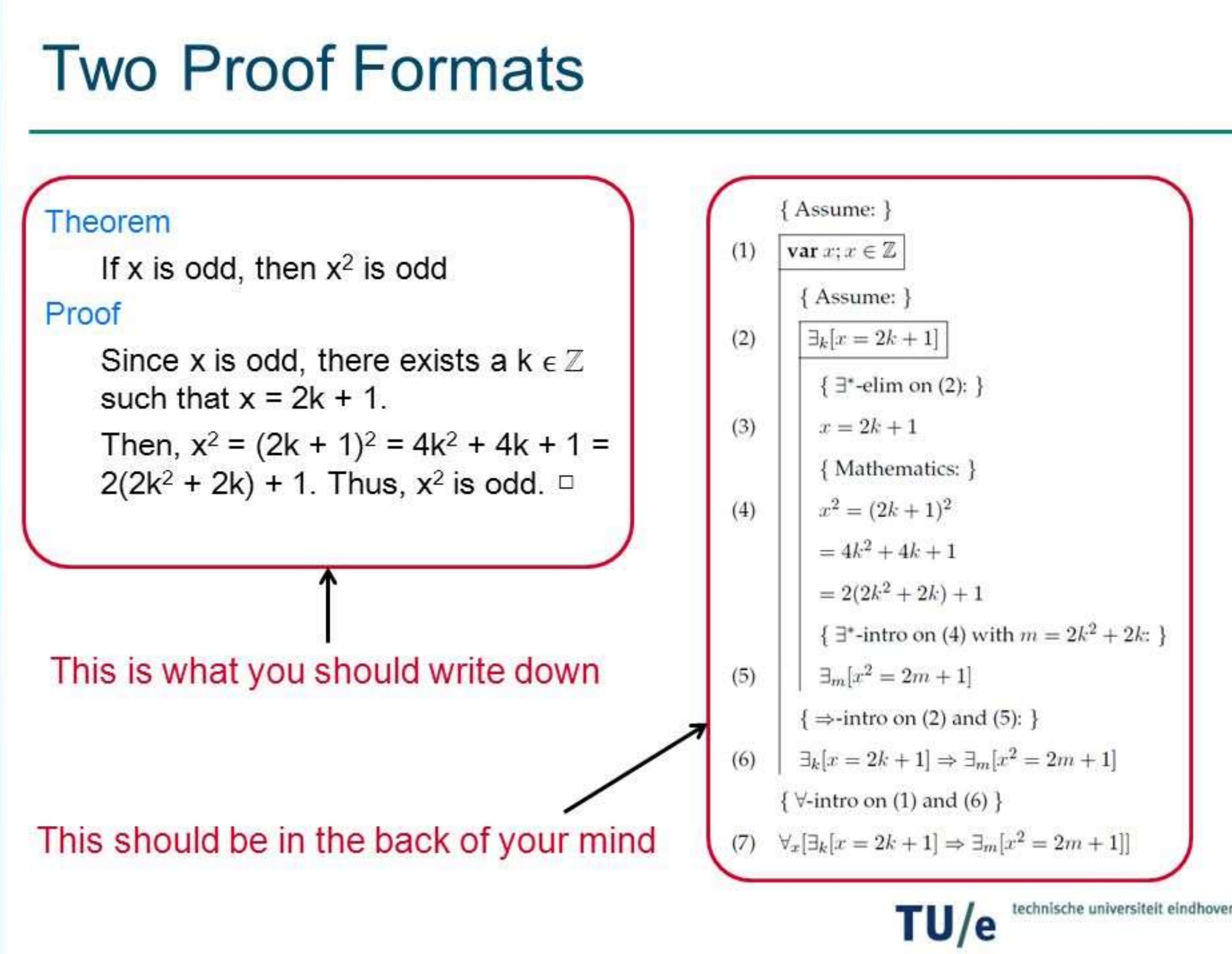
- Maths is written in a proof format.
- Begin: with what you are given.
- *Middle: is direct logical consequence.*
- End: with what was asked to be proved.
  - QED ∎
- Break up into *many* Theorems.





Prove your beliefs

- Proof is like a computer program.
- But, keep it readable
  - as much as possible!
- Figure on the right: Steps 1-7 show the logical evolution of statements.
  - much like simple steps of a computer
- Figure on the left: The overall idea gets conveyed with minimal notation.



Fake proofs

- Cancellation rule?
- Caution: Never divide by 0.
  - $\frac{1}{x}$ may be undefined in your ring.

$$
\begin{aligned}
a &= b \\
a^2 &= ab \\
a^2 - b^2 &= ab - b^2 \\
(a+b)(a-b) &= b(a-b) \\
a+b &= b \\
2b &= b \\
2 &= 1
\end{aligned}
$$

Fake proofs

- Product rule of derivative operator.
- The number of summands *non*-constant?
- Caution: Look at the whole function $x * x$
  - sum x "x times" is undefined, for real x

$$x^2 = \underbrace{x + x + x + \cdots + x}_{x \text{ times}}$$

$$\frac{d}{dx}(x^2) = \frac{d}{dx}(\underbrace{x + x + x + \cdots + x}_{x \text{ times}})$$

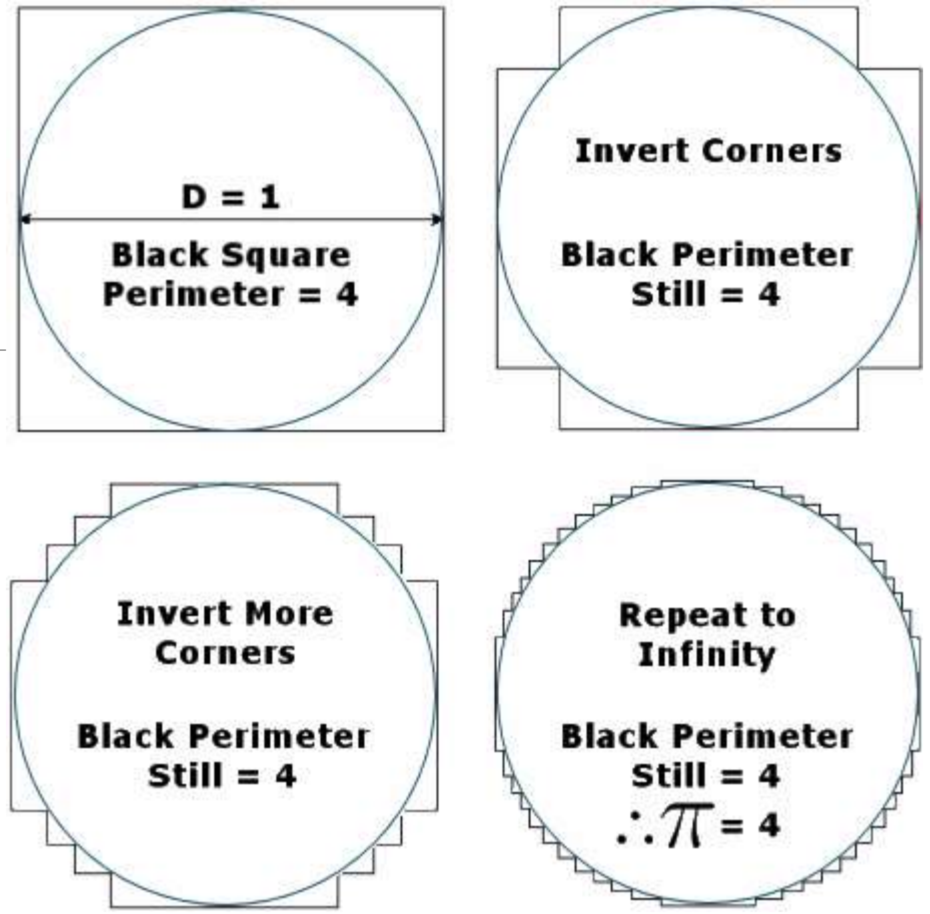
$$2x = \underbrace{1 + 1 + 1 + \cdots + 1}_{x \text{ times}}$$

$$2x = x$$

$$2 = 1$$

Fake proofs

- Perimeter = sum of parts.
- Infinitely many parts?
- Do the parts *converge* to the circle?
- Where's the *error* converging to?
- Caution: Part should be a *tangent*.
  - Discretization introduces large errors
  - Bound them!



Fake proofs by AI chatbots

What's a good proof that every prime is the sum of two squares?

One way to prove that every prime number can be written as the sum of two squares is to use the fact that every positive integer can be written as a sum of four squares. This is known as Lagrange's Four Square Theorem.

y positive integer
he sum of four
roceed by induction.
asy to check: 1 can
+ 0^2 + 0^2 + 0^2.
t every positive
l can be written as
uares. We need to
lso be written as the
es.

Can a rectangle and a circle have the same area?

No, a rectangle and a circle cannot have the same area unless the rectangle is not a true rectangle, but a shape that has curved sides and resembles a circle.

This is because the area of a rectangle is given by the formula A = l x w, where l represents the length of the rectangle, and w represents its width. On the other hand, the area of a circle is given by the formula A = πr^2, where π is the mathematical constant pi and r represents the radius of the circle.

Since the area of a rectangle is always a product of two positive numbers, whereas the area of a circle is always a product of π and the square of the circle's radius, it is impossible for them to have the same area.

Even if we take the square of the circle's radius to be a rational number, π will still be an irrational number, making it impossible for the areas to match exactly.

To prove that $(27)^{1/3}$ is irrational, we can use proof by contradiction.

Let's assume that $(27)^{1/3}$ is rational. This means it can be expressed as a ratio of two integers in the form $\frac{a}{b}$, where a and b are integers with no common factors other than 1, and b is not equal to 0.

So, we have $(27)^{1/3} = \frac{a}{b}$, where a and b have no common factors other than 1.

Now, let's cube both sides of the equation:

$$(27)^{1/3} = \left(\frac{a}{b}\right)^3$$
$$27 = \frac{a^3}{b^3}$$

Multiplying both sides by b^3, we get:

Assignment 2:

- Give examples of at least three rules that allow "logical consequences" inside a proof.
- Write a proof of the statement: "x is odd $\Longleftrightarrow x^2$ is odd".
- If we use GP (Geometric Progression formula) twice in the series: $\cdots + \left(\frac{1}{x}\right)^2 + \left(\frac{1}{x}\right) + 1 + x + x^2 + \cdots$, we get the sum=ZERO! Write this fake proof and discuss why it's fake.
- Describe the derivative of the function $f(x) = |x|$, for real-valued x. [Function $|x|$ gives the non-negative magnitude of x.]
- Recall the circle-perimeter (fake) proof done in the class. Correct it to calculate the value of π.

Proof Techniques

- **Deduction**
  - If $(\alpha \to \beta)$ and α, then β.
- **Induction**
  - If $(\alpha(k) \to \alpha(k+1))$ and $\alpha(0)$, then $\forall \ell, \alpha(\ell)$.
- **Contraposition**
  - If $(\alpha \to \beta)$, then $(\sim \beta) \to (\sim \alpha)$.
- **Contradiction**
  - If $(\alpha \to False)$, then $(\sim \alpha)$.
- **Diagonalization**
  - Draw a matrix; *contradict* the diagonal!

$$
\begin{array}{ll}
s_1 & = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & \quad \vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\ddots
\end{array}
$$

$$s = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots$$

Common ~~proof~~ techniques

Proof by intimidation Trivial!

Proof by cumbersome notation The theorem follows immediately from the fact that $\left|\bigoplus_{k\in S}\left(\mathfrak{K}^{\mathbb{F}^\alpha(i)}\right)_{i\in\mathcal{U}_k}\right| \preccurlyeq \aleph_1$ when $[\mathfrak{H}]_{\mathcal{W}} \cap \mathbb{F}^\alpha(\mathbb{N}) \neq \emptyset$.

Proof by inaccessible literature The theorem is an easy corollary of a result proven in a hand-written note handed out during a lecture by the Yugoslavian Mathematical Society in 1973.

Proof by ghost reference The proof my be found on page 478 in a textbook which turns out to have 396 pages.

Circular argument Proposition 5.18 in [BL] is an easy corollary of Theorem 7.18 in [C], which is again based on Corollary 2.14 in [K]. This, on the other hand, is derived with reference to Proposition 5.18 in [BL].

Proof by authority My good colleague Andrew said he thought he might have come up with a proof of this a few years ago...

Internet reference For those interested, the result is shown on the web page of this book. Which unfortunately doesn't exist any more.

Proof by avoidance *Chapter 3:* The proof of this is delayed until Chapter 7 when we have developed the theory even further. *Chapter 7:* To make things easy, we only prove it for the case $z = 0$, but the general case in handled in Appendix C. *Appendix C:* The formal proof is beyond the scope of this book, but of course, our intuition knows this to be true.

facebook.com/Mathematicx

Types of statements

- Conjecture: an unproved belief.
  - P≠NP .
- Axiom: an unprovable, defining, belief.
  - Peano's axioms [$s(\cdot)$ is called *successor*].
- Hypothesis: a testable prediction.
  - Riemann's hypothesis. Church-Turing thesis.
- Theorem: a formal statement with proof.
  - Prime number theorem.
- Corollary, Lemma, Claim, Proposition, Fact
  - diverse assertions from/towards a theorem.
- Algorithm (proved) *vs* Heuristic (unproved).

Peano Axioms for natural numbers

PA1 $\forall x(\neg(s(x) = 0))$

PA2 $\forall x \forall y(s(x) = s(y) \rightarrow x = y)$

PA3 $\forall x(x + 0 = x)$

PA4 $\forall x \forall y(x + s(y) = s(x + y))$

PA5 $\forall x(x \cdot 0 = 0)$

PA6 $\forall x \forall y(x \cdot s(y) = x \cdot y + x)$

PA7 $[A(0) \wedge \forall x(A(x) \rightarrow A(s(x)))] \rightarrow \forall x A(x)$

- **"Hypothesis is a tentative prediction or explanation of the relationship between two variables' It implies that there is a systematic relationship between an independent and dependent variable".**

Fundamental Theorem of Arithmetic: Every positive integer has a prime factorisation, unique up to the order of the factors

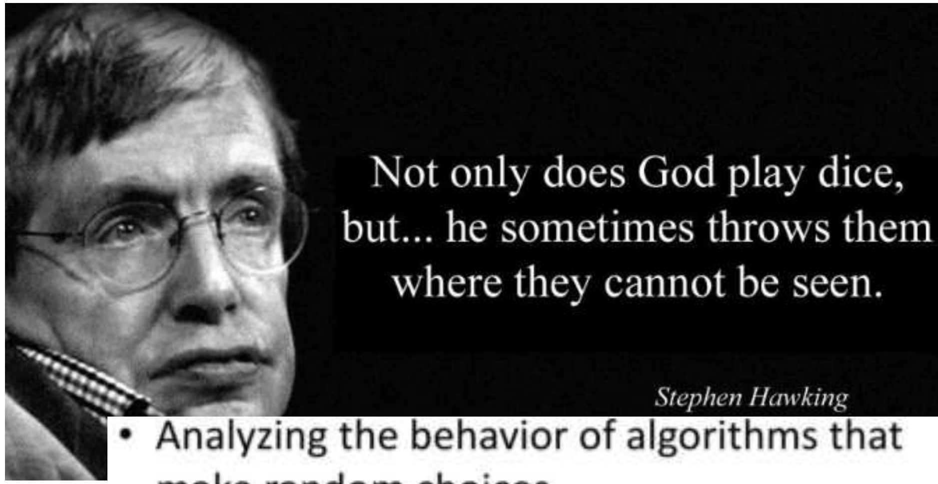
Fundamental Theorem of Algebra: Every nonconstant polynomial over the field of complex numbers has at least one root

Fundamental Theorem of Calculus: For every continuous function f on an interval $[a, b]$ the function $g(x) = \int_a^x f(t)\,dt$ is an antiderivative of f on (a, b)

Fundamental Theorem of Linear Algebra: The row space of a matrix is orthogonal to the nullspace of the matrix, and the dimensions add up to the number of columns of the matrix

Does God play dice?

- CS relies on probabilities.
  - Are they necessary?
- Random sampling is a powerful tool:
  - algorithms, systems testing, networks
  - prover-verifier protocol, proof-checking, cryptosystems
  - ML model with biased input distribution
- If I toss any coin the probability of Heads is ½.
- For an *un*biased coin, probability of Heads is ½.
- My ML model works very well on real data.
- My ML model decides x on dataset X with *mean-absolute-error* of 10%.



- Analyzing the behavior of algorithms that make random choices
  - Running time, performance
- Testing computer systems
  - Generating input/demand to test a system
- Modeling discrete structures
  - Understanding the structure of the internet or social networks

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Assignment 3:

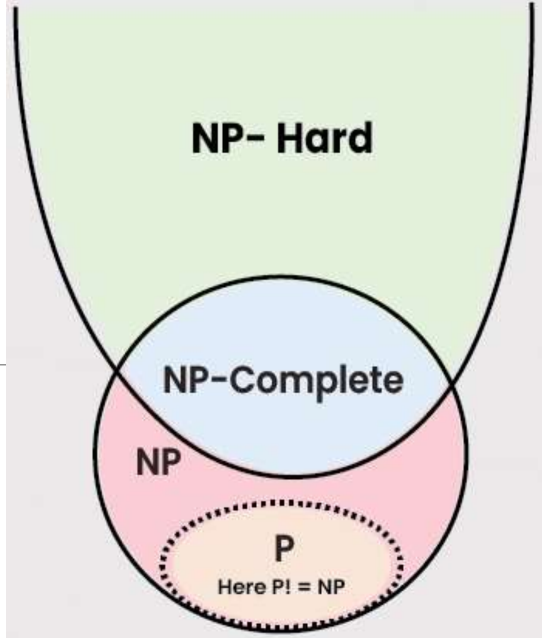
- Prove by induction: $\Sigma_{i\in[n]}\left(\frac{1}{i^2}\right) \leq \left(2-\frac{1}{n}\right)$.
- Prove by contradiction: $\sqrt{2}$ is irrational.
- Prove by diagonalization: There is **no** bijection from N (natural numbers) to R (reals).
- What is the difference between a conjecture and an axiom? Give examples.
- What is the difference between a hypothesis and a theorem? Give examples.
- If a drug is advertised as having 80% efficacy, does it mean that it will treat your illness with probability = 0.8? Discuss.

Is life Hard?

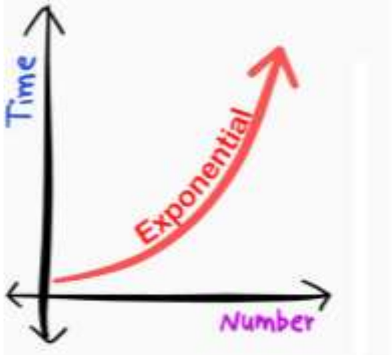
- What does easy or hard mean in CS?
  - Is it in theory or practice?
- Depends on the context:

1. Fibonacci heaps are hard to *analyse*.
2. Matrix multiplication is *suboptimal*.
3. Integer factoring is hard to *solve*.
4. Hamiltonian path is hard to *find*.
5. Cycles are hard to *count*.

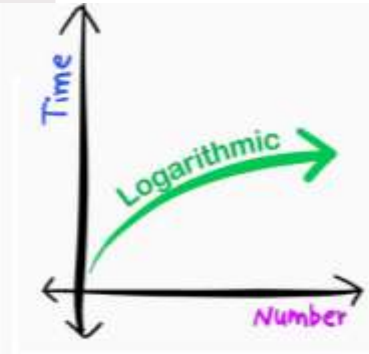
- Hardness is of diverse types:
  - Insightful Naming is important!



Procedure	Binary heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$O(\lg n)$

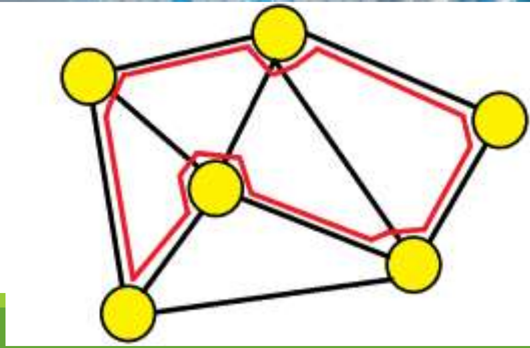


CLASSICAL

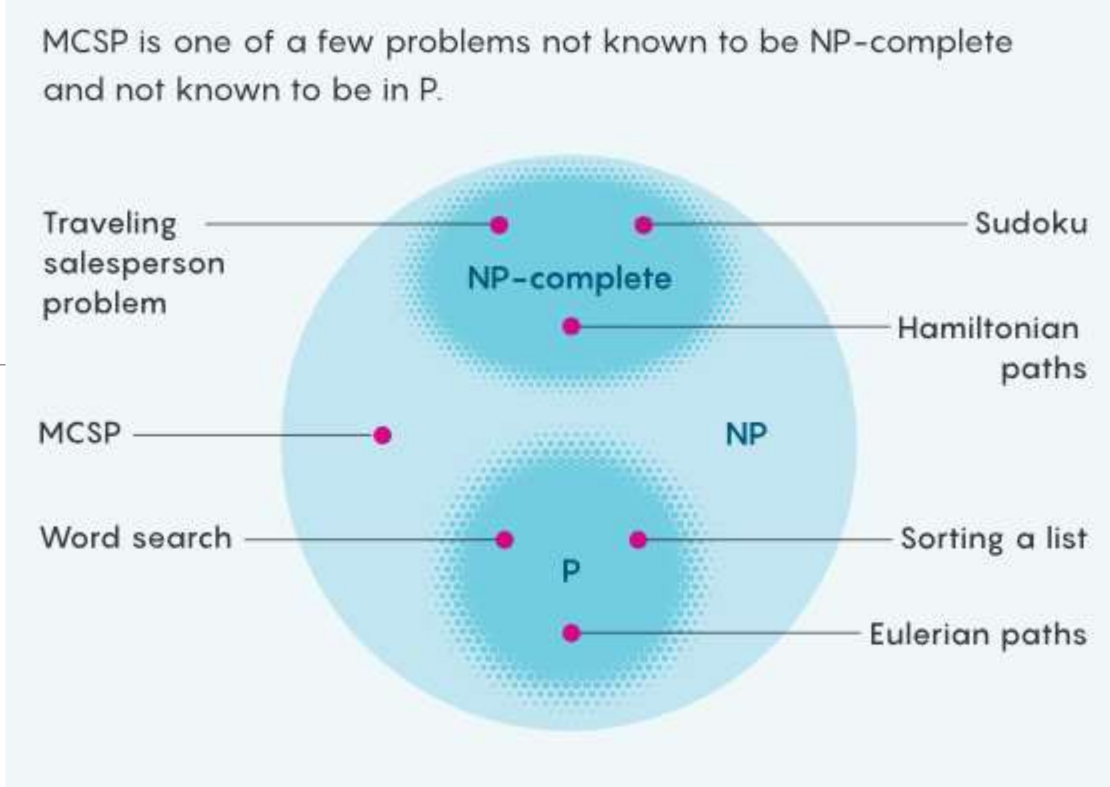


QUANTUM

There are only two hard problems in computer science: cache invalidation and naming things.
- Phil Karlton



Is Hardness bad?

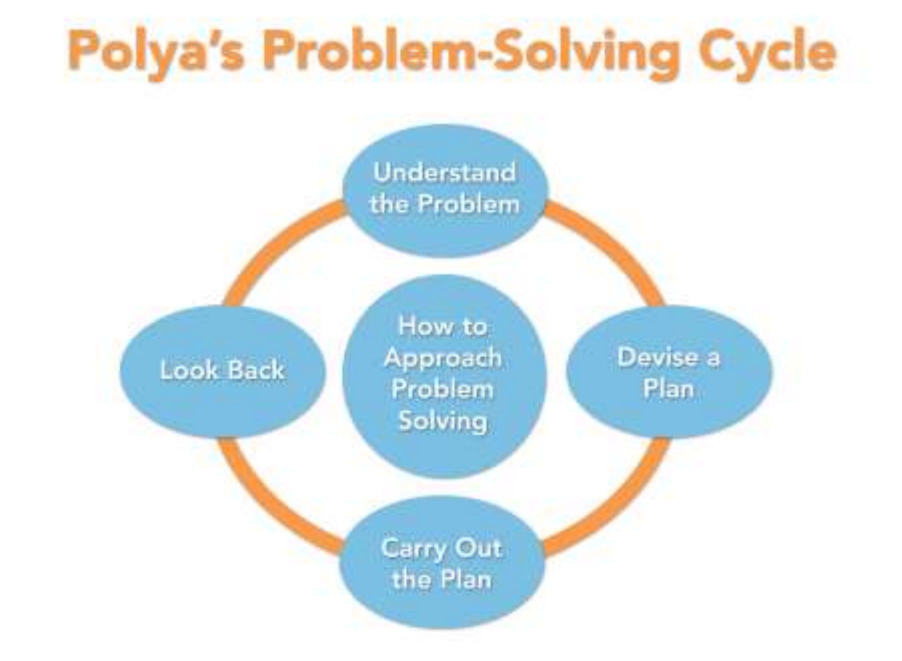


- Do scientists prove hardness only for sadistic fun?!
- We don't want to look for algorithms that don't exist.
- We can try solving for restricted inputs.
- Once we know a problem is hard, we could search for special-case heuristics
  - Approximation
  - Machine Learning, or Deep Learning
- Hard problems help design secure cryptosystems
  - IntegerFactoring, DiscreteLog → RSA
  - ShortestVector (SVP) → Lattice cryptosystem
  - SystemSolver → Multivariate Cryptography

- If a well-known hard problem Y reduces to X, then X is also *hard*.
- Else, we're *stuck* wondering about X.
  - problems of intermediate complexity.

How to survey?

- Recognize your area of interest (say **A**).
  - easier to pick from existing labels
  - internet, copilot, proceedings, journals, manuscripts, magazines
- Read few papers in **A**, make notes, give talks.
  - follow-up on the recent citations
- Identify a problem (say **P**) in area **A**.
- Specialize your search to **P** (and its *vicinity*).
  - read-up its state-of-the-art
  - give talks
  - follow-up on citations
  - be clear about what's done and what's open
  - talk to experts

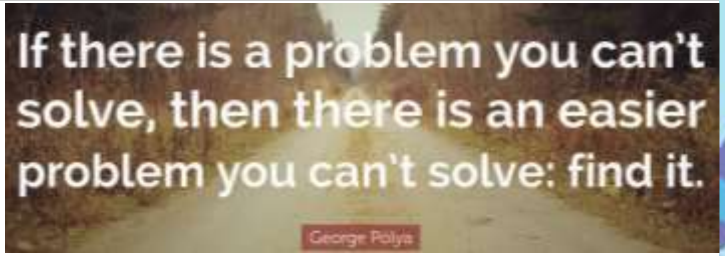


Assignment 4:

- State two computational problems that are easy. Give their complexities.
- State two computational problems that are hard. How hard are they?
- Do AI/ML solve truly hard problems? Discuss.
- (Optional) Do you know a problem that was considered hard, but turned out to have an easy proof?
- Is the professional sport of Chess hard, in a computational complexity sense?

How to initiate research?

- Recognize your area of interest (say **A**).
- Identify open problem **P** in area **A**.
  - Survey **P** and its *vicinity*.
- Guess an approach $\mathbb{S}$ – a first stab at the problem.
- Identify strength of $\mathbb{S}$.
- Identify weakness of $\mathbb{S}$.
- Look back:
  - does $\mathbb{S}$ explain the literature *better*?
  - does $\mathbb{S}$ solve a new *special-case*?
- Challenge the status-quo:
  - decomplicate and clarify!



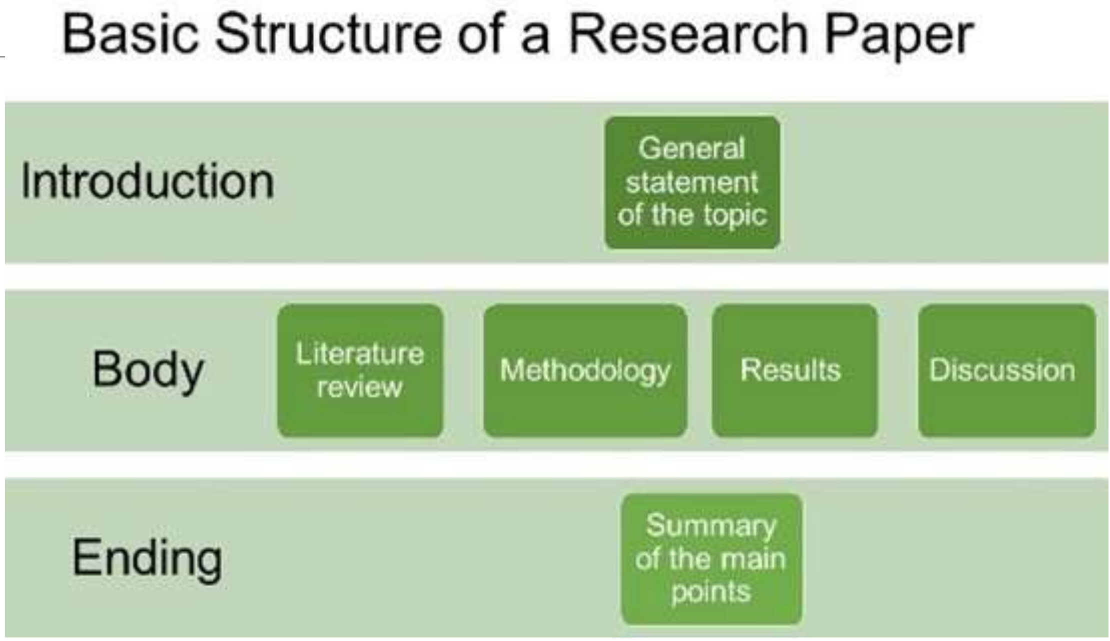
Time for a result or a preprint?

- Look back:
  - does approach $\mathbb{S}$ explain the literature *better*?
  - does $\mathbb{S}$ solve a new *special-case*?
- if you've a nontrivial result, double-check and
  - write a *draft*
  - do multiple iterations of *editing*!
- Double-check: Target audience eager to read it?
  - Is it a result you're proud of?
  - Is there a venue where people publish such results?
- Polish the preprint and submit it in a conference.
  - In CS, conference submissions are important

"It's all original research. I had no assistance when I looked it up on Wikipedia."

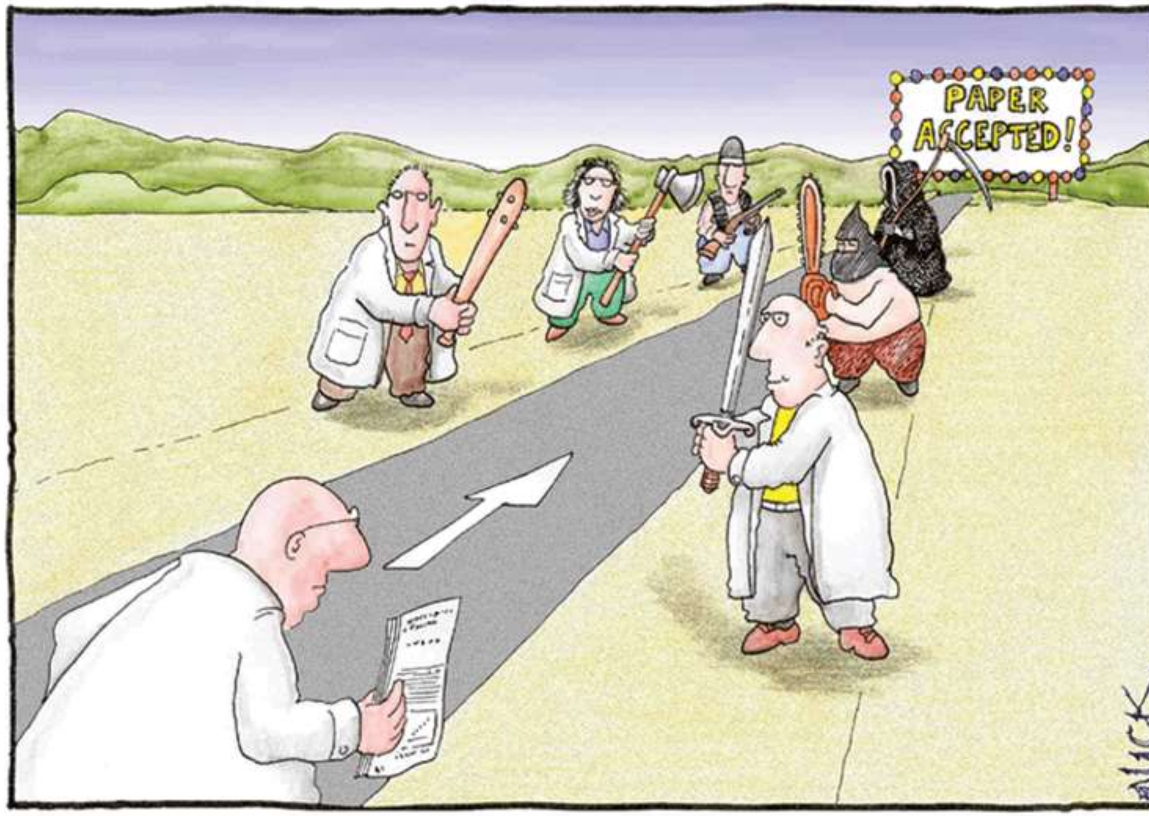
Write a paper for non-experts

- In CS, conference submissions are important.
- Assume your reader to be a non-expert
  - *impatient*: has little time to read
  - *critic*: may not share your excitement about the problem
  - *nitpick*: may not like if you don't follow the traditions of the conference
    - E.g. language, organization, paper-length, etc.
- Introduction: divide into subsections
  - motivation; history
  - results in simple words; main ideas
- Technical details: divide into sections
- Conclusion, Acknowledgements, References, Appendix



Incorporate feedback

- CS is a highly collaborative field.
  - esp. compared to other theory fields
- Get your colleagues to read the first section (/10 pages).
- Gather their candid feedback. Did they
  - understand the result?
  - understand the idea?
  - value the problem?
- Make amends to generally weaken the criticism.



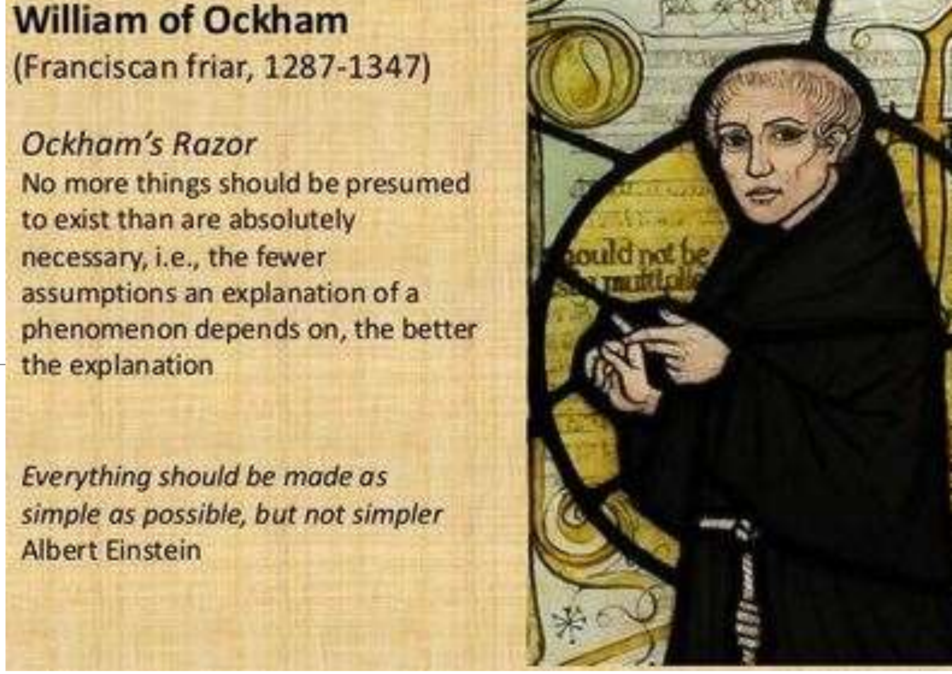
Most scientists regarded the new streamlined peer-review process as "quite an improvement."

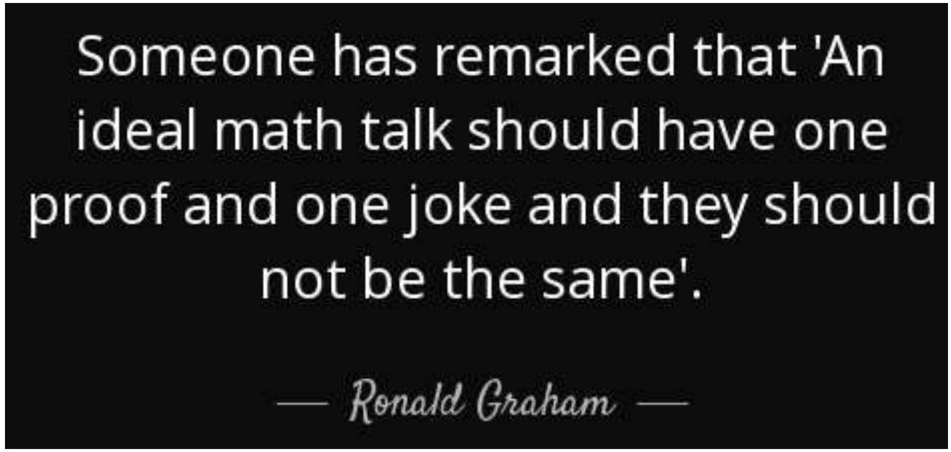
Assignment 5:

- Write 1-2 lines about your area of interest.
- Write 1-2 lines about an open problem in your area of interest. Identify a special-case.
- Regarding the special-case above: Suggest your title and abstract, of a research paper, that you hope to write one day.
- Regarding the paper title/abstract above: Show your work to your neighbour and get his/her feedback. Summarize it in 1-2 lines.

Present to non-experts

- Presentation ≠ Reading a paper !
- Pick the simplest, most impactful, result to highlight.
  - not the most general
  - not the most complicated
  - use minimal notation
- Focus on ideas, rather than equations or data.
  - discuss very easy, but failed, ideas
- Tentative breakup of time (resp. #slides out of 20):
  - survey: *30% (6)*
  - new result statements/ motivation/ interpretation: *40% (8)*
  - proof ideas: *25% (5)*
  - open questions & conclusion: *5% (1)*





Incorporate talk feedback

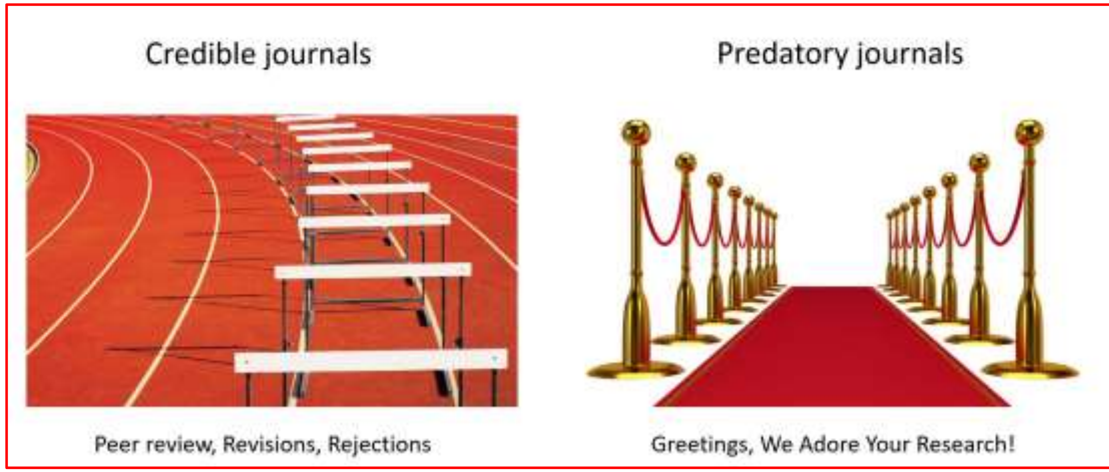
- Practice timing your talk.
  - Minutes: 5 (pitch), 15, 20, 30, 50, or 75 (keynote).
- Give a mock-talk to your colleagues.
  - ask someone to note the feedback down
- Gather from the audience's body-language: Did they
  - value the problem?
  - place the result in a larger context?
  - appreciate the idea?
- Make amends to generally weaken the criticism.
  - people's feedback may be contradictory; do your best edit!

How’s your paper reviewed?

- Conference specifies a standard format.
  - non-compliance leads to quick rejection by reviewers
- Three reviewers read your submission independently
  - double-blind review
  - brief report is written
- Review report has
  - reviewer confidence score
  - paper evaluation score
  - importance of the problem, clarity of presentation
  - interesting proof ideas; paper strength and weakness
- Unless the scores are very high, the program committee (PC) votes on your paper quality!
  - area politics?!

What makes a venue prestigious?

- Quality of the members of Program Committee (PC).
- Quality of the earlier papers: are they
  - cited/used widely?
  - mentioned in popular talks?
  - inspiring books/surveys/products?
- Quality of the peer review process: Is
  - a Revision demanded?
  - the Acceptance rate realistic?
- Quality of the publishing company:
  - does it have conferences of high-quality?
  - is it *open-source*?
  - shouldn't be *predatory*!



Assignment 6:

- List 2-3 prestigious venues in your area of interest; where you would want to publish in the future.
- List 2-3 do’s resp. don'ts in a CS talk.
- Regarding the paper title/abstract you wrote in the last assignment: Create two slides/pages of a fictional talk you might want to give.
- Regarding the talk-slides above: Show your work to your neighbor and get his/her feedback. Summarize it in 1-2 lines.