5CCS2FC2: Foundations of Computing II

P vs NP: The Million Dollar Problem

Week 2

Dr Christopher Hampson

Department of Informatics

King's College London

Computable Functions

Computable Functions

- A **function** $f : \Sigma^* \to \Sigma^*$ on words from Σ^* is **computable** if there is a (deterministic) Turing Machine M_f such that

  (i) M_f **accepts** all input words $w \in \Sigma^*$

  (ii) When **started** in configuration $(q_{\text{init}}, \varepsilon, w)$ the machine eventually **terminates** in configuration $(q_{\text{accept}}, \varepsilon, f(w))$

> In other words, the machine **transforms** the contents of the tape from w to $f(w)$ and returns the tape head to the start of the word.

- **Computable Functions**

f is computable in polynomial-time	$\iff$	f is **computable** and M_f if terminates in **polynomial-time**

Polynomial Reductions and NP-completeness

- **Polynomial Reduction**
  - A **polynomial reduction** from a problem X to a problem Y is a function $f : \Sigma^* \to \Sigma^*$ **computable in polynomial-time** such that

$$w \in X \iff f(w) \in Y$$

$X \leq_p Y$ means X is **polynomially reducible** to Y

- **NP-hardness**
  - A problem X is said to be **NP-hard** if *every* problem in **NP** can be polynomially reduced to it.

$$Y \leq_p X \qquad \text{for all } Y \in \mathbf{NP}$$

(X is at least as hard as *every* NP-problem)

Proving NP-hardness

Theorem If Y is **NP**-hard and $Y \leq_p X$, then X is also **NP**-hard.

Proof:

Step 1) Suppose that:

(i) Y is **NP**-hard and **(ii)** $Y \leq_p X$

Step 2) By **(i)** we have that $Z \leq_p Y$ for all $Z \in$ **NP**

Proving NP-hardness

Step 3) By **(ii)** and **transitivity** we have that

$$Z \leq_p Y \leq_p X \qquad \text{for all } Z \in \textbf{NP}$$

Q.E.D.

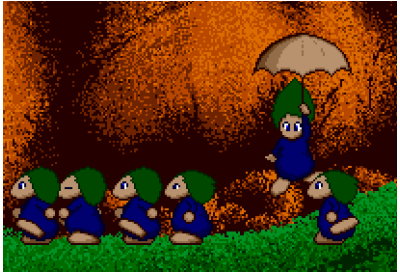
A typical approach to demonstrating that a problem is **NP**-hard is to show that **SAT** is reducible to it. *i.e.* that **SAT** $\leq_p X$.

- **NP-completeness**
  - A problem X is said to be **NP-complete** if
    - **(i)** X is NP-hard **(lower bound)**,
    - **(ii)** X also belongs to **NP** **(upper bound)**.

- **Some examples of NP-complete problems**
  - The Boolean Satisfiability Problem
  - The Clique problem
  - The Hamiltonian Cycle Problem
  - The Travelling Salesman Problem (TSP)
  - The Graph Colouring Problem
  - The Knapsack Problem

`https://en.wikipedia.org/wiki/List_of_NP-complete_problems`

- **Some examples of NP-complete problems**
  - Many Games and Puzzles
    - $(n \times n)$ Sudoku
    - Lemmings
    - Pokémon



End of Slides!