# 5CCS2FC2: Foundations of Computing II

# P vs NP: The Million Dollar Problem

## Week 2

**Dr Christopher Hampson**

*Department of Informatics*

*King's College London*

# Asymptotic Notation

# Asymptotic Notation

- **Big Oh Notation (upper bounds)**
  - A function $g : \mathbb{N} \to \mathbb{N}$ **eventually dominates** a function $f : \mathbb{N} \to \mathbb{N}$ if there is some constant $c > 0$ such that

$$\boxed{\exists N \in \mathbb{N} \left( \forall n > N;\ f(n) \ \leq\ c \cdot g(n) \right)}$$

# Asymptotic Notation

- **Big Oh Notation (upper bounds)**

$$f(n) = O(g(n)) \quad \Longleftrightarrow \quad f(n) \text{ is eventually dominated by } g(n)$$

# Asymptotic Notation

- **Big Omega Notation (lower bounds)**

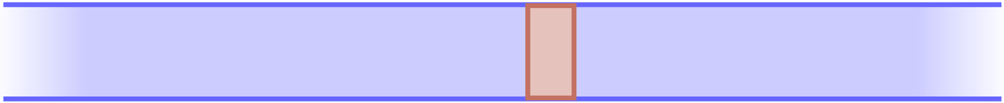
$$f(n) \in \Omega\big(g(n)\big) \quad \Longleftrightarrow \quad f(n) \text{ eventually dominates } g(n)$$

# Asymptotic Notation

- **Big Theta Notation** (tight bounds)

$$\boxed{\Theta\big(g(n)\big) \;=\; O\big(g(n)\big) \;\cap\; \Omega\big(g(n)\big)}$$

# Asymptotic Notation



$\Theta(1)$ $\Theta(\log_2(n))$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n\log_2(n))$ $\Theta(n^2)$ $\Theta(n^3)$ $\cdots$ $\Theta(2^n)$ $\Theta(n!)$

- **Decision Problems**
  - Every **problem**, for which the possible answers are **YES** or **NO** (or, alternatively, **True** or **False**) can be interpreted as a language,

$$L_P = \left\{ w \in \Sigma^* \; : \; \begin{array}{c} w \text{ is an instance of the problem } P \text{ whose} \\ \text{answer is YES} \end{array} \right\}$$

**(for some suitable choice of alphabet $\Sigma$)**

# Complexity Classes P and NP

# Complexity Classes P and NP

- **Time Complexity**

  - Given a (terminating) machine $M$, define a function $\boxed{t_M : \Sigma^* \to \mathbb{N}}$

$$t_M(w) := \begin{cases} \text{number of steps required to} \\ \text{terminate on input } w \in \Sigma^* \end{cases}$$

# Complexity Classes P and NP

- **Time Complexity**
  - We define the **worst-case** time complexity as $\boxed{T_M(n) : \mathbb{N} \to \mathbb{N}}$

$$\boxed{T_M(n) := \max \{ \, t(w) : \text{all words } w \text{ of length } n \, \}}$$

# Complexity Classes P and NP

- **(Deterministic) Polynomial time**
  - A langauge $L$ is said to be solvable in **polynomial time** if there exists a **deterministic Turing Machine** $M$ such that:

    **(i)** $M$ **accepts** $L$,

    **(ii)** $T_M(n) \in O(n^k)$ is eventually dominated by a **polynomial function**

$$\textbf{P} = \{ \text{ all problems decidable in polynomial time } \}$$

## Complexity Classes P and NP

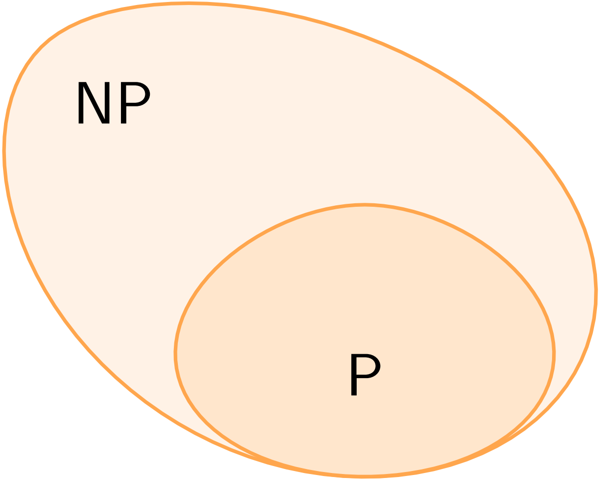
- **Non-deterministic Polynomial time**
  - A langauge $L$ is said to be solvable in **non-deterministic polynomial time** if there exists a ***non-deterministic* Turing Machine** $M$ such that:

    **(i)** $M$ **accepts** $L$,

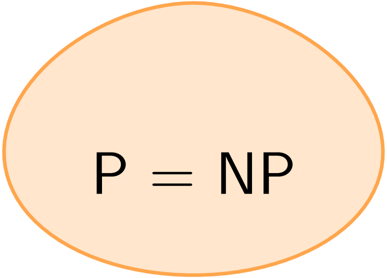
    **(ii)** $T_M(n) \in O(n^k)$ is eventually dominated by a **polynomial function**

$$\textbf{NP} = \left\{ \begin{array}{c} \text{all problems decidable in} \\ \text{non-deterministic polynomial time} \end{array} \right\}$$

# Complexity Classes P and NP



## Complexity Classes P and NP



# End of Slides!