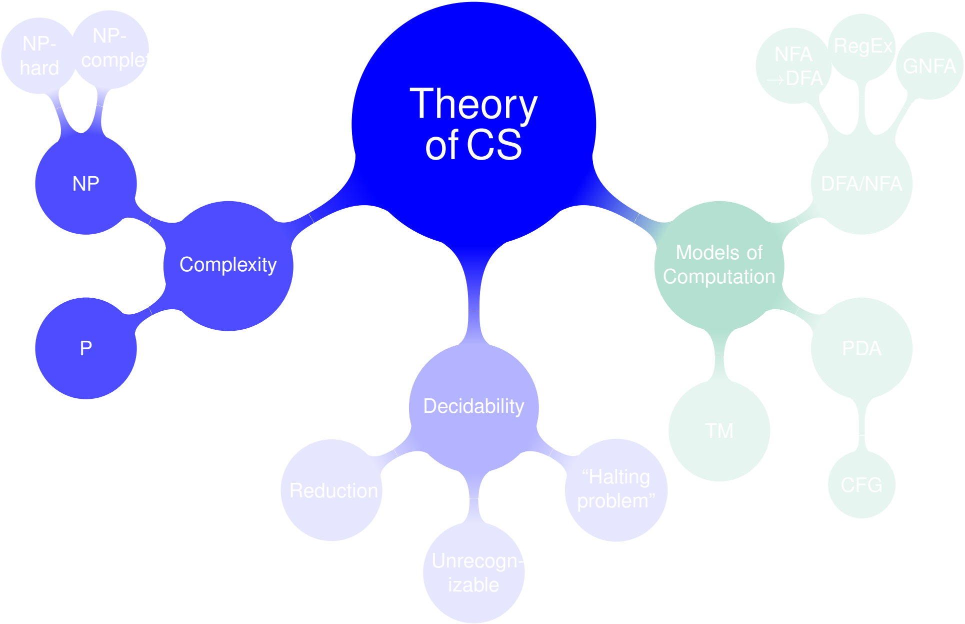
Complexity

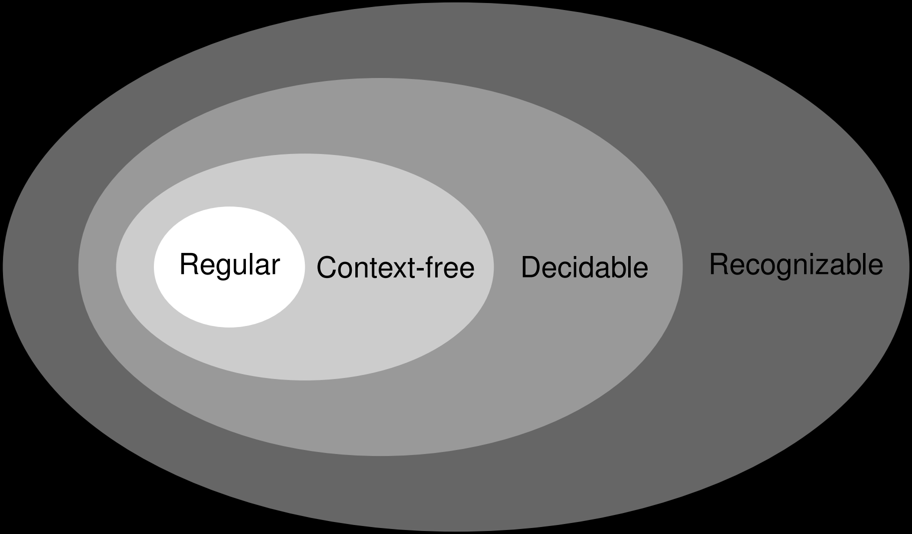
Dr Kamal Bentahar

School of Computing, Electronics and Mathematics
Coventry University

Lecture 8


Theory of CS
Complexity
NP
P
NP-hard
NP-complet
Decidability
Reduction
Unrecogn-izable
"Halting problem"
Models of Computation
DFA/NFA
NFA →DFA
RegEx
GNFA
TM
PDA
CFG

The “computation universe” discovered so far. . .



Time Complexity

- Being **decidable** means that an **algorithm** exists to decide the problem.
- However, the algorithm may still be *practically* ineffective because of its **time** and/or **space** cost.

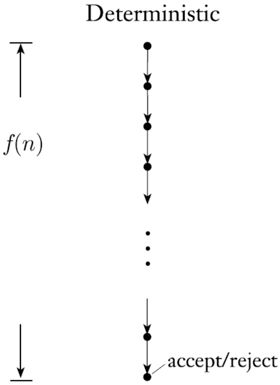
Shorthands:

- **DTM:** Deterministic Turing Machine (single-tape).
- **NDTM:** Non-deterministic Turing Machine (single-tape).

Running time / Time complexity

The **running time** or **time complexity** of a DTM that always halts is the maximum number of steps $f(n)$ that it makes on any input of length n.

We say that it "runs in time $f(n)$"; or that it is "an $f(n)$-time TM."

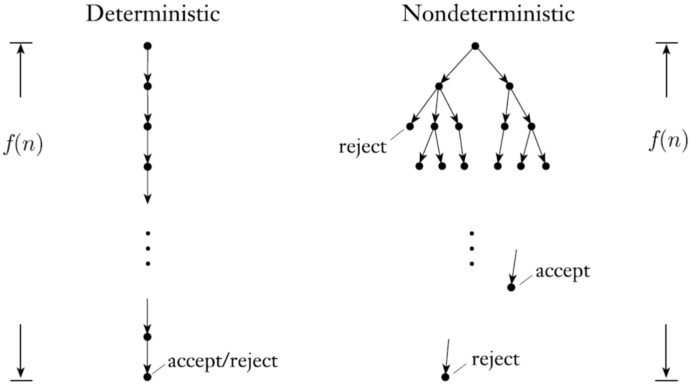


Running time / Time complexity

The **running time** or **time complexity** of a DTM that always halts is the maximum number of steps $f(n)$ that it makes on any input of length n.

For an NDTM consider **all the branches** of its computation.

We say that it "runs in time $f(n)$"; or that it is "an $f(n)$-time TM."



Big-O notation cheat-sheet

- Constant $O(1)$
- Polynomial $O(n), O(n^2), \ldots, O(n^k), \ldots$ $(k \geq 1)$

  Also, $O(n^a \log^b n)$ for any positive constants a and b.
- Exponential $O(2^n), O(3^n), \ldots, O(b^n), \ldots$ $(b > 1)$
- Factorial $O(n!)$
- $O(n^n)$

In general

$$O(n^a) \leq O(n^a \log^b n) \leq O(n^{a+1})$$

Formal definition of big-O notation

$f(n) = O(g(n))$ means that, when n becomes "sufficiently large," $f(n)$ becomes bounded by a multiple of $g(n)$.

Big-O notation

Let f and g be functions

$$f, g \colon \mathbb{N} \to \mathbb{R}^+$$

We say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$,

$$f(n) \leq c\, g(n).$$

Alternatively, we can calculate the limit of the ratio

$$\frac{f(n)}{g(n)} \to \text{constant} < \infty \quad \text{as } n \to \infty$$

We say that $g(n)$ is an **(asymptotic) upper bound** for $f(n)$.

Example

- $n + 1 = O(n)$ because

$$\frac{n+1}{n} = 1 + \frac{1}{n} \quad \to 1 \text{ as } n \to \infty$$

- $5n^3 + 43n - 69 = O(n^3)$ because

$$\frac{5n^3 + 43n - 69}{n^3} = 5 + \frac{43}{n^2} - \frac{69}{n^3} \quad \to 5 \text{ as } n \to \infty$$

- $n^2 + 2^n = O(2^n)$ because

$$\frac{n^2 + 2^n}{2^n} = \frac{n^2}{2^n} + 1 \quad \to 1 \text{ as } n \to \infty$$

Example

- $n + 1 \neq O(1)$ because

$$\frac{n+1}{1} = n + 1 \quad \to \infty \text{ as } n \to \infty$$

- $5n^3 + 43n - 69 \neq O(n)$ because

$$\frac{5n^3 + 43n - 69}{n} = 5n^2 + 43 - \frac{69}{n} \quad \to \infty \text{ as } n \to \infty$$

- $n^2 + 2^n \neq O(n^2)$ because

$$\frac{n^2 + 2^n}{n^2} = 1 + \frac{2^n}{n^2} \quad \to \infty \text{ as } n \to \infty$$

Time complexity class

Interesting results:

Complexity relationships among TM variants

Let $t(n)$ be a function, where $t(n) \geq n$.

Then:

- Every $t(n)$ time **multi-tape** TM has an equivalent $O(t^2(n))$ time DTM.
- Every $t(n)$ time NDTM has an equivalent $O(2^{O(t(n))})$ time DTM.

This suggests studying the difference between **deterministic** and **non-deterministic** time.

The class **P**

Deterministic Time complexity class

Let $t\colon \mathbb{N} \to \mathbb{R}^+$ be a function.
Define the **deterministic-time complexity class** $\mathit{TIME}(t(n))$ to be the set of languages that are decidable by an $O(t(n))$ time DTM.

The class **P**

P is the class of languages that are decidable in **polynomial time** on a DTM.

$$\mathbf{P} = \mathit{TIME}(1) \cup \mathit{TIME}(n) \cup \mathit{TIME}(n^2) \cup \mathit{TIME}(n^3) \cup \cdots .$$

The class **NP**

Nondeterministic Time complexity class

Let $t\colon \mathbb{N} \to \mathbb{R}^+$ be a function.
Define the **nondeterministic-time complexity class** $NTIME(t(n))$ to be the set of languages that are decidable by an $O(t(n))$ time NDTM.

The class **NP**

NP is the class of languages that are decidable in polynomial time on a NDTM.

$$\mathbf{NP} = NTIME(1) \cup NTIME(n) \cup NTIME(n^2) \cup NTIME(n^3) \cup \cdots .$$

We will see later that **NP** can also be characterised as the class of languages that have polynomial time "**verifiers.**"

P examples – path between two vertices in a graph

$$PATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph that has a directed path from } s \text{ to } t\}.$$

A polynomial time algorithm for PATH

Input: $\langle G, s, t\rangle$, where G is a directed graph with vertices s and t.
Output: *true* if there is a path between s and t; *false* otherwise.

```
Place a mark on vertex s.
repeat
    Scan all the edges of G.
    If an edge (a, b) is found going from a marked vertex a to an unmarked
    vertex b, then mark vertex b.
until no additional vertices are marked
If t is marked then accept else reject
```

P examples – relatively prime numbers

$$RELPRIME = \{\langle x, y\rangle \mid x \text{ and } y \text{ are relatively prime}\}.$$

E : *Euclidean algorithm* for computing the *greatest common divisor*

Input: $\langle x, y\rangle$, where x and y are natural numbers.
Output: $\mathrm{gcd}(x, y)$

```
repeat
        x ← x mod y
        Exchange x and y.
until y = 0
return x.
```

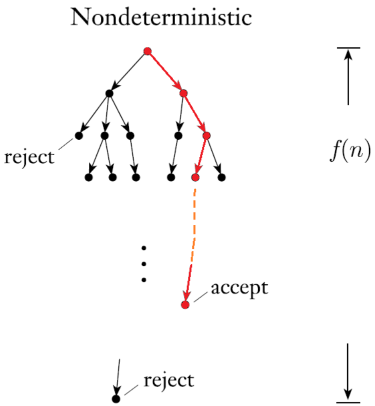
Algorithm that solves RELPRIME, using E as a subroutine

On input $\langle x, y\rangle$, where x and y are natural numbers:

1. Run E on $\langle x, y\rangle$.
2. If the result is 1, *accept*. Otherwise, *reject*."

NP – Solution vs Verification

- **Solving**: finding/searching for a solution.
- **Verifying**: confirming that a proposed solution is correct.



NP – Solution vs Verification

- **Solving**: finding/searching for a solution.
- **Verifying**: confirming that a proposed solution is correct.

Example

Given a candidate for a tour around England, we just need to check if it:

1. contains all the required cities
2. uses no city more than once
3. finishes at its starting point
4. uses only valid routes

NP – verifiers

Verifiers

A verifier for a language L is an algorithm V, where

$$L = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some } \textit{certificate} \text{ string } c\}.$$

- A **polynomial time verifier** runs in polynomial time in the length of w.
- A language is **polynomially verifiable** if it has a polynomial time verifier.

NP examples – Subset Sum Problem

$$\textit{SUBSETSUM} = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_n\} \text{ is a set of positive integers and } t = y_1 + \cdots + y_\ell \text{ for some } \{y_1, \ldots, y_\ell\} \subseteq S\}.$$

Verifier: "On input $\langle\langle S, t\rangle, c\rangle$:

1. Test whether c is a subset of S.
2. Test whether the sum of the numbers from c is t.
3. If both pass, *accept*. Otherwise, *reject*."

Alternatively, polynomial time NDTM: "On input $\langle S, t\rangle$:

1. Non-deterministically select a subset c of the numbers in S.
2. Test whether the sum of the numbers from c is t.
3. If the test passes, *accept*. Otherwise, *reject*."

NP examples – Cliques in a graph

Cliques

A **clique** in an undirected graph is a subgraph, wherein every two vertices are connected by an edge.
A k**-clique** is a clique that contains k vertices.

$$CLIQUE = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}.$$

Verifier for $CLIQUE$: "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a subgraph with k vertices in G.
2. Test whether G contains all the required edges to make c a clique.
3. If both pass, *accept*. Otherwise, *reject*."

Alternatively, polynomial time NDTM: "On input $\langle G, k\rangle$, where G is a graph:

1. Non-deterministically select a subset c of k vertices of G.
2. Test whether G contains all the required edges to make c a clique.
3. If yes, accept; otherwise, reject."

Summary

P — Polynomial-time

Class of languages that are decidable in polynomial time by a DTM.

$$\mathbf{P} = \bigcup_{k \geq 0} TIME(n^k).$$

NP — Nondeterministic Polynomial time

Class of languages that are decidable in polynomial time by a NDTM.

$$\mathbf{NP} = \bigcup_{k \geq 0} NTIME(n^k).$$

Also: Class of languages that have polynomial time (DTM) verifiers.

$\mathbf{P} = \mathbf{NP}$ or $\mathbf{P} \neq \mathbf{NP}$?