

# 1 Complexity Zoo

## 1.1 TIME[ $f(n)$ ]

Informally: problems that can be solved in  $f(n)$  time.

**Definition 1.1.** Given some function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , TIME[ $f(n)$ ] are the set of problems solvable within  $O(f(n))$  atomic steps on a deterministic Turing machine. Where  $n$  is the size of the input.

## 1.2 NTIME[ $f(n)$ ]

Informally: problems that can be solved nondeterministically in  $f(n)$  time.

**Definition 1.2.** Given some function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , NTIME[ $f(n)$ ] are the set of problems solvable within  $O(f(n))$  atomic steps on a nondeterministic Turing machine.

## 1.3 SPACE[ $f(n)$ ]

Informally: problems that can be solved in  $f(n)$  space.

**Definition 1.3.** Given some function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , SPACE[ $f(n)$ ] are the set of problems solvable using a tape of length  $O(f(n))$  on a deterministic Turing machine. Where  $n$  is the size of the input.

## 1.4 NSPACE[ $f(n)$ ]

Informally: problems that can be solved non-deterministically in  $f(n)$  space.

**Definition 1.4.** Given some function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , NSPACE[ $f(n)$ ] are the set of problems solvable using a tape of length  $O(f(n))$  on a non-deterministic Turing machine. Where  $n$  is the size of the input.

## 1.5 SIZE[t]

Informally: problems that can be solved by a circuit of size  $t(n)$

Formally:

**Definition 1.5.** A language  $L$  is in SIZE[t] if there is a  $t(n)$ -size circuit family  $\{C_n\}_{n \in \mathbb{N}}$  s.t.  $\forall x \in \{0, 1\}^n, x \in L \iff C_n(x) = 1$ .

## 1.6 P

Informally: all problems that can be solved in polynomial time.

**Definition 1.6.**

$$\mathbf{P} = \bigcup_{k \geq 0} \text{TIME}[n^k]$$

Descriptive Complexity definitions:

**Definition 1.7.**

$$\mathbf{P} = \text{FO}(\text{LFP})$$

(First Order logic extended with the Least Fixed Point operator, with successor. A high level, handwavy description of the LFP operator is the added ability to recursively define FO formulas.)

**Definition 1.8.**

$$\mathbf{P} = \text{SO}(\text{Horn})$$

(Second Order logic restricted with Horn. SO logic allows you to quantify over subsets/relations/functions on the domain, and Horn means all ‘clauses’ are really implications with literal in the conclusion and all literals positive.)

Circuit Complexity definitions:

**Definition 1.9.**

$$\mathbf{P} = \mathbf{P} - \text{uniform}$$

$\mathbf{P}$ –uniform = Set of circuit families  $\{C_n\}_{n \in \mathbb{N}}$  for which there is a Turing Machine that on input  $1^n$  outputs the description of  $C_n$  in polynomial time.

**Definition 1.10.**

$$\mathbf{P} = \mathbf{L} - \text{uniform}$$

$\mathbf{L}$ –uniform = Set of circuit families  $\{C_n\}_{n \in \mathbb{N}}$  for which there is a Turing Machine that on input  $1^n$  outputs the description of  $C_n$  using logarithmic space.

Notable Problems in  $\mathbf{P}$ :

- 2-SAT
- 2-Colourability
- Reachability

## 1.7 NP

Informally: all problems that can be solved in nondeterministic polynomial time.

**Definition 1.11.**

$$\mathbf{NP} = \bigcup_{k \geq 0} \text{NTIME}[n^k]$$

Turing Machine definition:

**Definition 1.12.**

$$x \in \mathbf{NP} \iff \exists w : \|w\| \leq p(\|x\|) \text{ s.t. } M(x, w) = 1$$

In terms of a verifier:

Informally: The set of decision problems where a solution can be verified in polynomial time.

Descriptive Complexity Definition:

**Definition 1.13.**

$$\mathbf{NP} = \text{SO}\exists$$

(Existential Second Order)

Notable Problems in **NP**:

- SAT
- 3-Colourability
- TSP
- Subset sum

## 1.8 coNP

Turing Machine definition:

**Definition 1.14.**

$$x \in \mathbf{coNP} \iff \forall w : \|w\| \leq p(\|x\|) \text{ s.t. } M(x, w) = 1$$

In terms of a verifier:

Informally: The set of decision problems where a solution can be refuted in polynomial time.

## 1.9 FPT

Informally, the set of problems that can be solved in polynomial time for some fixed parameter.

**Definition 1.15.** The set of problems that can be parameterised by  $k$  and can be solved in  $f(k)n^c$ , where  $f(x)$  is only dependent on  $k$ , and  $c$  is an independent constant.

**P** is contained within **FPT**.

If a problem is in **FPT**, then for any fixed  $k$  that problem is in **P**.

**FPT** is also known as **W[0]**

Notable Problems in **FPT**:

- Vertex Cover

## 1.10 W[1]

**Definition 1.16.** The class of parametrized problems that admit a parametrized reduction to the following problem: Given a nondeterministic single-tape Turing machine, decide if it accepts within  $k$  steps.

N.B This is short acceptance

**Definition 1.17.** The class of parametrized problems that admit a parametrized reduction to the following problem: Given a Boolean circuit  $C$ , with a mixture of fanin-2 and unbounded-fanin gates. There is at most 1 unbounded-fanin gate along any path to the root, and the total depth (fanin-2 and unbounded-fanin) is constant. Does  $C$  have a satisfying assignment of Hamming weight  $k$ ?

N.B This is Weighted 3-SAT.

Notable Problems in **W[1]**:

- Short Acceptance
- Weighted 3-SAT
- Clique (of size  $k$ )
- Independent set (of size  $k$ )

**1.11**   **W[2]**

**1.12**   **W[i]**

**1.13**   **FPTAS**

**1.14**   **PTAS**

**1.15**   **L**

Informally: all problems that can be solved using logarithmic space (excluding the input)

**Definition 1.18.**

$$\mathbf{L} = \text{SPACE}[\log n]$$

This means you effectively have the input and then a fixed number of counters/pointers (up to the size of the input)

Notable Problems in **L**:

- Planar Graph Isomorphism

**1.16**   **NL**

Informally: all problems that can be solved using nondeterministic logarithmic space (excluding the input)

**Definition 1.19.**

$$\mathbf{NL} = \text{NSPACE}[\log n]$$

This means you effectively have the input and then a fixed number of counters/pointers (up to the size of the input)

**Definition 1.20.**

$$\mathbf{NL} = \text{SO}(\text{Krom})$$

**Definition 1.21.**

$$\mathbf{NL} = \mathbf{coNL}$$

Notable Problems in **NL**:

- Reachability
- Unreachability

## 1.17 PSPACE

### 1.18 $\Sigma_2^P$

**Definition 1.22.**

$$\Sigma_2^P = \mathbf{NP}^{\mathbf{NP}}$$

Turing Machine definition:

**Definition 1.23.**

$$x \in \Sigma_2^P \iff \exists w : \|w\| \leq p(\|x\|) \forall u : \|u\| \leq p(\|x\|) \text{ s.t. } M(x, w, u) = 1$$

### 1.19 $\Sigma_i^P$

**Definition 1.24.**

$$\Sigma_i^P = \mathbf{NP}^{\Sigma_{i-1}^P}$$

Turing Machine definition:

**Definition 1.25.**

$$x \in \Sigma_i^P \iff \exists u_1 \forall u_2 \dots Q_i u_i M(x, u_1, \dots, u_i) = 1$$

$|u_j| \leq p(x)$  and  $Q_i = \forall/\exists$  if  $i$  is even/odd.

**Definition 1.26.**

$$\Sigma_i^P = \text{co} - \Pi_i^P$$

### 1.20 $\Pi_2^P$

**Definition 1.27.**

$$\Pi_2^P = \text{coNP}^{\mathbf{NP}}$$

Turing Machine definition:

**Definition 1.28.**

$$x \in \Pi_2^P \iff \forall w : \|w\| \leq p(\|x\|) \exists u : \|u\| \leq p(\|x\|) \text{ s.t. } M(x, w, u) = 1$$

### 1.21 $\Pi_i^P$

**Definition 1.29.**

$$\Pi_i^P = \mathbf{coNP}^{\Sigma_{i-1}^P}$$

Turing Machine definition:

**Definition 1.30.**

$$x \in \Pi_i^P \iff \forall u_1 \exists u_2 \dots Q_i u_i M(x, u_1, \dots, u_i) = 1$$

$|u_j| \leq p(x)$  and  $Q_i = \exists/\forall$  if  $i$  is even/odd.

**Definition 1.31.**

$$\Pi_i^P = \mathbf{co} - \Sigma_i^P$$

### 1.22 PH

**Definition 1.32.**

$$\mathbf{PH} = \bigcup_i \Sigma_i^P$$

### 1.23 $P^{SAT}$

### 1.24 $\mathbf{NP}^{SAT}$

$\mathbf{NP}$  with a SAT oracle, equivalent to  $\Sigma_2^P$

### 1.25 $\mathbf{P/poly}$

Circuit definition:

**Definition 1.33.**

$$\mathbf{P/poly} = \bigcup_c \mathbf{SIZE}[n^c]$$

Turing Machine definition: decision problems solvable by a polynomial-time

Turing machine that receives an 'advice string' that is polynomial in size.

More formally:

**Definition 1.34.**

$$\mathbf{P/poly} = \bigcup_{c,d} \mathbf{TIME}[n^c]/n^d$$

## 1.26 EXP

Informally: Problems that take exponential time to solve.

**Definition 1.35.**

$$\mathbf{EXP} = \bigcup_c \text{TIME}[2^{n^c}]$$

## 1.27 NC<sup>0</sup>

Constant size circuits.

## 1.28 NC<sup>1</sup>

Contains Parity.

## 1.29 NC<sup>2</sup>

Contains Reachability.

## 1.30 NC<sup>i</sup>

**Definition 1.36.**

$\mathbf{NC}^i$  = all languages decidable by an  $\mathbf{L}$ -uniform circuit family of polynomial size and depth  $O(\log^i n)$

## 1.31 NC

**Definition 1.37.**

$$\mathbf{NC} = \bigcup_i \mathbf{NC}^i$$

## 1.32 TFNP

Conventional complexity classes are concerned with decision problems, i.e., given a graph  $G$  and some number  $k$  determine whether or not  $G$  has a clique of size  $k$ .

This loses its meaning when the answer is always ‘yes’ - for example, does this bimatrix game have a mixed Nash equilibrium?

**Definition 1.38.** TFNP is the set of binary relations  $R(x, y)$  such that for every  $x$  there exists at least one  $y$  (which is at most polynomially larger than  $x$ ) such that  $R(x, y)$  holds. Algorithms that solve problems in this class take an input  $x$  and produce some  $y$  such that  $R(x, y)$  holds, in polynomial time.



### 1.33 PPAD

An example of a problem in TFNP is the following:

**Problem 1.1.** END OF THE LINE. We are given a graph  $G$  that is a disjoint union of directed paths - every vertex has at most one predecessor and at most one successor. This graph may be exponentially sized, but is given implicitly as a Turing machine computing the predecessor and successor of each node (if they exist, otherwise signals that this node is a sink/source). Given a source in  $G$ , can we find a sink, or any other source, in polynomial time?

Of course, a sink always exists, but simply moving along successor edges may visit all (exponentially many) nodes of a graph.

**Definition 1.39.** PPAD is the set of problems in TFNP reducible to END OF THE LINE.

A notable problem in PPAD, which turns out to be PPAD-complete, is to compute a mixed Nash equilibrium of some bimatrix game.

## 2 Graph Theory

### 2.1 Definitions

The graph  $H$  is a minor of the graph  $G$  if a copy of  $H$  can be obtained by deleting vertices and deleting or contracting edges of  $G$ . The contraction of the edge  $(x, y)$  is achieved by replacing  $x$  and  $y$  with a new vertex  $z$  such that  $N(z) = N(x) \cup N(y)$ .

$H$  is a topological minor of  $G$  if some sequence of subdivisions of  $H$  results in an induced subgraph of  $G$ . An edge  $(x, y)$  is subdivided by adding a vertex  $z$  such that  $N(z) = \{x, y\}$

### 2.2 Graphs

Claw

### 2.3 Graph Classes