

Step up a level in abstraction. From now on, we'll talk about programming at an algorithmic level, not at a Turing Machine level, but anything we talk about can be computed by a Turing Machine.

## 1 P

Suppose you have a problem of size  $N$ , for example sort a list of length  $N$ , multiply two matrices of size  $N$  by  $N$ , find the maximum of a list of  $N$  numbers, etc. *The problem is said to be in the set  $P$  if there is an algorithm, that could be run on a Turing Machine, to compute a solution to the problem in polynomial time.*

In other words,  $P$  is the set of all problems that can be computed by TMs in polynomial time.

### 1.1 Examples

Sorting a list - clearly you could do that in  $\leq N^2$  operations. Matrix multiplication - can be done in  $N^3$  operations. List maximum - can be done in  $N$  operations.

## 2 NP

### 2.1 Boolean Satisfiability

Given a boolean expression  $E$  in  $N$  variables,  $V_1, V_2 \dots V_N$  in the Product of Sums form. Is there a set of inputs  $V_1, V_2 \dots V_N$  such that the expression evaluates to True?

Can you think of a way to do this in polynomial time?

Consider a related problem. Suppose someone gives you  $V_1, V_2 \dots V_N$  and asks "does this work?"

Can you solve this in polynomial time?

### 2.2 Non-Deterministic Turing Machines

These are completely analogous to NDFSMs. In other words, there can be multiple possible transitions from one given state to a next state.

### 2.3 NP

Suppose you have a problem  $X$ . If you can make an algorithm of the following form, then  $X \in NP$ .

Algorithm:

1) Non-deterministically (i.e. using an NDTM) generate a possible solution for  $X$ .

2) Verify the solution (in polynomial time, using a DTM).

More formally,  $NP$  is the set of all languages that can be recognized by an  $NDTM$ .

Equivalently,  $NP$  is the set of all languages that have a polynomial verification algorithm.

### 2.3.1 Example: The Travelling Salesman Problem (TSP)

Given a set of  $N$  cities and distances between them, is there a circuit of length  $L$  that starts at one city, visits all the cities exactly once, and returns to the starting city?

TSP  $\in NP$  — Given a path, it's easy to check if its length is  $L$ .

### 2.3.2 Example: The Clique Problem

Given a graph with  $N$  vertices, is there a clique of size  $K$ ? (I.e. is there a subgraph of  $K$  vertices which are all connected to each other?)

This problem is in  $NP$  — Given a set of  $K$  vertices, it's easy to see if they're all connected.

## 3 NP-completeness

Remark:  $P \subseteq NP$

It is not known if  $NP \subseteq P$ .

There is a set of problems, known as *NP-complete problems* that have the property that if any one of them is in  $P$ , then all  $NP$  problems are in  $P$ .

Boolean Satisfiability is one such problem.

The Traveling Salesman Problem is another.

The Clique Problem is a third.

The key point about  $NP$ -complete problems is this: Let  $X$  be an  $NP$ -complete problem. Let  $Y$  be any problem in  $NP$  (not necessarily  $NP$ -complete). Then, you can reduce  $Y$  to  $X$  in polynomial time.