# Non-deterministic Turing machines

ian.mcloughlin@gmit.ie

## Polynomial time

**Definition**

An algorithm is said to be solvable in *polynomial time* if the number of steps required to complete the algorithm for a given input is $O(n^k)$ for some nonnegative integer $k$, where $n$ is the complexity of the input.

**Informally: P complexity class**

The P complexity class is the set of problems for which there exists, for each such problem, at least one algorithm to solve that problem in polynomial time.

## Polynomial time on a Turing machine

**Sorting** algorithms are usually compared in terms of comparisons.

**Other algorithms** might be compared in terms of something else, like iterations.

**With Turing machines** we can use the number of times we look up the state table.

**The size** of the input can be the length of the input on the tape initially.

**P complexity class**

The P complexity class is the set of languages for which there exists some Turing machine that decides the language in polynomial time.

## Non-deterministic Turing machine

**The usual** Turing machines are often called deterministic Turing machines.

**Deterministic** Turing machines have exactly one row in their state table for every combination of (non-terminal) state and tape symbol.

**This means** there is only one path to follow at a given point in time.

**Nondeterministic** Turing machines can have any number of rows for each state/symbol (including none).

**Essentially** they allow for parallel computation – they can branch into two or more paths at the same time.

## Non-deterministic Turing machine and languages

**Languages** are accepted by non-deterministic Turing machines, where an input string is accepted if any branch ends in the accept state.

**Deciders** – if a non-deterministic Turing machine always halts on all branches of computation, no matter what the input, then we say it decides the language it accepts.

**Any** language that is accepted (or decided) by a non-deterministic Turing machine has some deterministic Turing machine that accepts (or decides) it. So non-deterministic Turing machines don't really have any extra abilities over deterministic ones.

## Non-deterministic polynomial time

**Definition**

A decision problem is in the NP complexity class if it is decidable by a non-deterministic Turing Machine in polynomial time.

**P is a subset of NP**

Note that every determinisitic Turing machine is also a non-deterministic one, by our definitions. The P complexity class is a subset of NP because of this.

**Equivalent definition**

An equivalent definition of NP that you may come across is that NP is the set of languages $A$ that can be verified in polynomial time. By verified we mean that a deterministic Turing machine can accept a language $\{wc\}$ where $w$ is in $A$ and $c$ is some string, called the certificate for $w$.

## NP-complete problems

**Definition**

A problem is NP-hard if each problem in NP can be reduced to it in polynomial time.

**Reduction**

Reduction is a way of converting one problem into another, so that a solution to one is a solution to the other. By reducing decision problem A to decision problem B, we mean that we can transform inputs to A into inputs to B in such a way that a given input to A is accepted iff the corresponding input to B is.

**Definition**

A problem is NP-complete if it's in NP and is NP-hard.