TRACTABLE & INTRACTABLE PROBLEM

Tractable Problem: A problem that is solvable by a polynomial-time algorithm.
The upper bound is polynomial.
Here are **examples** of tractable problems (ones with known polynomial-time algorithms):
– Searching an unordered list
– Searching an ordered list
– Sorting a list
– Multiplication of integers (even though there's a gap)
– Finding a minimum spanning tree in a graph (even though there's a gap)

Intractable Problem: a problem that cannot be solved by a polynomial-time algorithm. The lower bound is exponential.
From a computational complexity stance, intractable problems are problems for which there exist no efficient algorithms to solve them.
Most intractable problems have an algorithm that provides a solution, and that algorithm is the brute-force search.
This algorithm, however, does not provide an efficient solution and is, therefore, not feasible for computation with anything more than the smallest input.

Examples
Towers of Hanoi: we can prove that any algorithm that solves this problem must have a worst-case running time that is at least $2^n - 1$.
* List all permutations (all possible orderings) of n numbers.

POLYNOMIAL-TIME ALGORITHMS: A polynomial-time algorithm is an algorithm whose execution time is either given by a polynomial on the size of the input, or can be bounded by such a polynomial. Problems which can be solved by a polynomial-time algorithm are called "tractable" problems. As an example, most algorithms on arrays can use the array size, n, as the input size. In order to find the largest element in any array requires a single pass through the array, so the algorithm which does this is of O(n), or it is a "linear time" algorithm.

Sorting algorithms take O(n log n) or O(n2) time. Bubble sort takes linear time in the least case, but O(n2) time in the average and worst cases. Heapsort takes O(n log n) time in all cases. Quicksort takes O(n log n) time on average, but O(n2) time in the worst case.

As far as O(n log n) is concerned, it must be noted that the base of the logarithms is irrelevant, as the difference is a constant factor, which is ignored. All programming tasks we know have polynomial solutions. It is not due to the reason that all practical problems have polynomial-time solutions.

Rather, it is because the day-to-day problems are one for which there is no known practical solution.

NON-DETERMINISTIC POLYNOMIAL TIME ALGORITHMS: A nondeterministic computation is viewed as:

1. when a choice point is reached, an infallible oracle can be consulted to determine the right option.

2. When a choice point is reached, all choices are made and computation can proceed simultaneously.

A Non-deterministic Polynomial Time Algorithm is one that can be executed in polynomial time on a nondeterministic machine. The machine can either consult an oracle in constant time, or it can spawn an arbitrarily large number of parallel processes, which is obviously a nice machine to have.