

Time Complexity - P and NP Classes.

The very first discussion of the P vs. NP problem took place by Kurt Gödel and John von Neumann in 1956. Oddly enough, Gödel thought that P would be equal to NP. The P vs. NP question is one of the most profound and sought after question in (theoretical) computer science.

Complexity Class P

The complexity class P is the class of all problems that have a solution time decidable by a polynomial function of the size of that problem by a deterministic algorithm. The P stands for Polynomial time. Class P problems can be solved in time $O(N^C)$ for some constant C such that N is the input size of the problem. A problem is assigned to class P if the number of steps needed to solve it is bounded by some power of the problem's size. An example could be, sorting items in a list. Another example is finding an item within a list simply by scanning all items until the desired item is found. Each of these examples are solved in $O(N^2)$ and $O(N)$ time respectively.

Polynomial Time Decidable

In the philosophical sense, polynomial time problems are tractable. A problem that requires $\theta(n^{1000})$ time, however, is intractable but it is not practical and rare to find a problem that requires such time. Reasonable computation models have also shown that when a problem is solved in polynomial time for a given computation model then that problem can be solved in polynomial time, in other reasonable computation models. So, all deterministic computational models can simulate any other with only a polynomial difference in running time such as n^4 vs. n^{10} . Even though as a programmer one would want to make their algorithms as efficient as possible regardless of the measure of time gain, the focus of computability theory is the comparison between polynomial and non-polynomial time bounded solutions. A division between easy problems and hard problems had to be made somewhere. So, we're looking at distinguishing $\{n^2, n^3, n^{99}, \dots\}$ set of problems from $\{n, 2^n, 100^n, n^n, \dots\}$ set of problems. Class P problems can be solved exponentially, for instance in using a brute-force search. However, through a deeper understanding of the problem, class P problems in such a case, would reveal polynomial time bounded solutions.

Complexity Class NP

The complexity class NP is the class of all problems that are verifiable in polynomial time or solvable in polynomial time by a nondeterministic algorithm. In fact, NP actually stands for Nondeterministic Polynomial time. If given an answer to the problem we could then verify that it is correct or not correct in polynomial time. Another way to look at it, is that a problem is in NP if the number of steps to verify the solution is bounded by some power of the problem's size but with the ability to perfectly guess the solution when faced with a choice of more than 1 option. All class P problems are class NP problems but not vice versa. Sometimes problems do not lend themselves to a more efficient solution than doing a brute-force search. We do not know for sure whether or not these problems have polynomial time bounded solutions.

Polynomial Time Verification

In general, researchers believe that $P \neq NP$, because ever since 1971, extensive and rigorous research has been conducted without success, to find a polynomial time algorithm for certain NP problems. *Reasonably, class P problems can be solved quickly, where as class NP problems have solutions that can be verified quickly.* Ultimately, researchers are unable to prove either $P \neq NP$ or $P = NP$. Additionally, study has shown that the complexity of several problems are related,

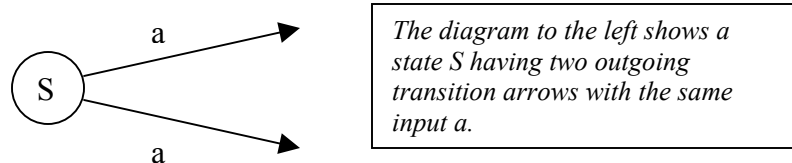
in that an entire class of problems can use a polynomial time solution. One of the unique aspects of NP problems are that if solution to NP problem is somehow known then showing that the solution is correct can be reduced to a polynomial verification. If indeed, $P \neq NP$ then solutions to NP problems would require an exhaustive search in the worst case. The Hamiltonian cycle problem, which is a directed path in a directed graph, goes through each vertex exactly once. There currently does not exist a polynomial time algorithm for a Hamiltonian cycle but a solution can be polynomially verified. Another problem that is only polynomially verifiable is testing the compositeness of a number, because all one needs is to properly guess a divisor.

Nondeterminism vs. Determinism

Now is actually a good time to talk about nondeterminism and determinism. Nondeterminism is a generalization of determinism. Nondeterminism does not add power for Turing machines, thus two Turing machines that perform the same computation, where one Turing machine is deterministic and the other is nondeterministic are equivalent in power.

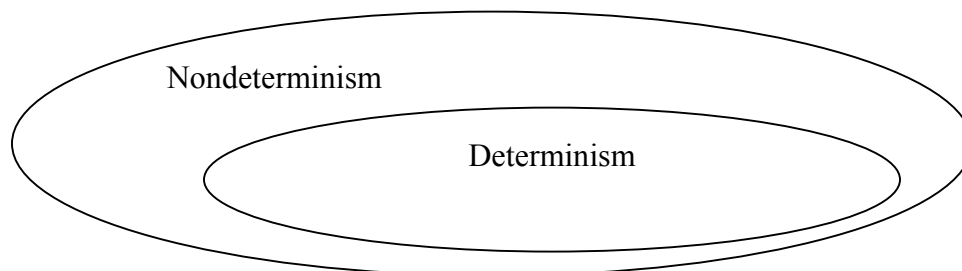
Nondeterminism

A nondeterministic machine allows for more than one outgoing transition arrow for the same input. This machine can either compute all pathways or branches in parallel, or it can make a choice at any given state that has such a condition.



Determinism

A deterministic machine is a machine that at any given state during execution, after reading the next input symbol, the next state is unambiguously known or determined. Every deterministic machine is a special kind of nondeterministic machine but the converse is not true.



Links

- <http://theory.lcs.mit.edu/>
- <http://www.cis.ohio-state.edu/~gurari/theory-bk/theory-bk.html>
- <http://www.cs.bu.edu/fac/lnd/toc/>
- <http://www.cs.unb.ca/~alopez-o/comp-faq/faq.html>