How Algorithms Are Analyzed in computer science

The slowest algorithms require an exponential number of steps in relation to the number of input values. The fastest algorithms can be executed in some constant number of steps, and they aren’t affected by the number of input values. [\n]

[extend]

In algorithm design, the number of input values is represented by the variable n, and sometimes additional variables are used for algorithms whose running times depend on the sizes of more than one set of input values. Exponential algorithms typically run in some order of O(C^n) time complexity, where C is a constant and n is the variable number of input values. [\n]

For example, a simplistic brute-force algorithm for finding a password has to process n combinations of 256 ASCII characters, so it runs in O(256^n) time. [\n]

One of the most important areas of algorithm research is the problem of P versus NP, or polynomial-time algorithms versus nondeterministic polynomial-time algorithms. There is a [a][$1 million prize](http://www.newyorker.com/tech/elements/a-most-profound-math-problem)[/a] being offered to anyone who can prove that P is or is not NP. This prize has been on offer for several decades, and so far, none of the smartest computer scientists have figured out a way to prove it. If it can be proved that they’re the same, then functions such as the brute-force password crack could run in polynomial time complexity, or O(n^C), where C is any constant. [\n]

[link]http://www.newyorker.com/tech/elements/a-most-profound-math-problem[link]

[/extend]