Computational complexity (P, NP and NPC)

**Definition (from Dictionary.com)**

Computational complexity is the mathematical characterization of the difficulty of a mathematical problem which describes the resources required by a computing machine to solve the problem.

**Explanation**

The computational complexity of a problem is used to describe the amount of computational resources needed to solve said problem. Computational complexity relates to the complexity of all possible algorithms that could be used to solve the same problem. While different resources can be examined like time, memory or electricity, the most common approach to define computational complexity is by calculating the number of basic steps used by the machine in the process of computing (the definition of a basic step is not Unambiguous, and might differ from case to case).

Computational complexity is often described as a function of the input, using the big O notion. If, for example, the complexity of a problem is in a linear relation to the length of the input (n) we can say that the computational complexity of this problem is O(n). in this case if the input becomes twice longer, the algorithm will require approximately twice the amount of steps to solve the problem. If the complexity of a problem is O(n2), doubling the length of the input will require the algorithm to perform 4 (22) more steps.

Computational problems can be divided into groups based on their complexity. Problems from the class P are problems that have a known solution with a polynomial time complexity. The class P is part of the class NT which contains problems for which a given solution can be verified by an algorithm with polynomial time complexity. The question whether all the problems in NP are also in P (i.e. whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer) is an open question, but the presumed answer to this question is no. Another interesting class of problems, also a part of NP is the class NPC. NPC problems all have no known polynomial solution (and as such are not in P), and are seemingly unrelated, but it has been proven that all problems in NPC are reducible from one another, meaning that if a polynomial solution can be found to one NPC problem, it will result in a polynomial solution to all the problems in NPC.

**Biological relevance**Problems in computational biology can also be described by their computational complexity. When comparing DNA, RNA or protein sequences, for example, we might want to try and find the longest sub-sequence two sequences have in common. The complexity of a dynamic programming based algorithm to this problem is O(nXm), n and m being the lengths of the sequences. As O(nXm) is a polynomial time, we can say that this problem is in class P. however the more generalized problem, which aims to find the longest sub-sequence **n** sequences have in common, has no known polynomial time solution and is in part of the class NP but not a part of the class P.