What are the main theoretical barriers to resolving P vs NP?

An Extended Project Qualification Dissertation

Camron Short

Candidate Number: 6439
Centre Number: 19216

Defining the problem

Big O-Notation

When shopping, there are multiple ways to collect all the items on your list. You might start at one end of the store and work through it methodically, which is often the most efficient - but only if your list is already sorted by store layout. Without planning, you might find yourself doubling back or criss-crossing the store multiple times. These differences in method lead to drastically different total effort. Similarly, the time taken depends not just on the method, but also on who is doing the shopping and how many items are on the list. These human and environmental factors make it impossible to measure an algorithm's speed purely in seconds. To address this, computer scientists use **Big O notation**, a mathematical way to describe how the number of steps an algorithm takes grows with the size of the input. For example, an algorithm with complexity $O(n)$ will perform at most a number of operations proportional to the length of the input list (n). An algorithm with $O(n^2)$ complexity might check each item against every other, leading to significantly more steps. These expressions describe the algorithm's **asymptotic behaviour** - that is, how it scales as inputs grow very large. When this growth follows a polynomial pattern like $O(n)$ or $O(n^2)$, we say it runs in **polynomial time**, which is generally considered efficient and tractable.

Sets and Set Notation

Earlier, we used a shopping list to explore algorithmic strategies. Mathematically, that list can be viewed as a **set** - a well-defined collection of distinct objects, such as items to buy. In computer science and mathematics, Sets are fundamental. A set can contain numbers, items, states, algorithms, or even other sets. For example, the set of paths through a store might include every possible route you could take. The set of solutions to a problem includes all algorithms that solve it. We describe relationships between sets and their elements using **Set notation**:

- $\in$: "is an element of" (e.g., $3 \in \{1, 2, 3\}$ means 3 is in the set)
- $\notin$: "is not an element of" (e.g., $4 \notin \{1, 2, 3\}$)
- $\subseteq$: "is a subset of" (e.g., $\{1, 2\} \subseteq \{1, 2, 3\}$)
- $\cup$: union (e.g., $A \cup B$ contains all elements in A or B)
- $\cap$: intersection (e.g., $A \cap B$ contains only elements in both A and B)
- $\setminus$: set difference (e.g., $A \setminus B$ contains elements in A but not in B)