*Article*

# Note for the P versus NP Problem

**Frank Vega** [1]

[1] GROUPS PLUS TOURS INC., 9611 Fontainebleau Blvd, Miami, FL, 33172, USA; vega.frank@gmail.com

**Abstract:** $P$ versus $NP$ is considered as one of the most fundamental open problems in computer science. This consists in knowing the answer of the following question: Is $P$ equal to $NP$? It was essentially mentioned in 1955 from a letter written by John Nash to the United States National Security Agency. However, a precise statement of the $P$ versus $NP$ problem was introduced independently by Stephen Cook and Leonid Levin. Since that date, all efforts to find a proof for this problem have failed. Another major complexity class is $NP$-complete. It is well-known that $P$ is equal to $NP$ under the assumption of the existence of a polynomial time algorithm for some $NP$-complete. We show that the Monotone one-in-three 3-satisfiability ($M-1IN3-3SAT$) is $NP$-complete and $P$ at the same time.

**Keywords:** computational algorithm; complexity classes; completeness; polynomial time; reduction

**MSC:** 68Q15; 68Q17; 68Q25

## 1. Introduction

$P$ versus $NP$ is one of the most important and challenging problems in computer science [1]. It asks whether every problem whose solution can be quickly verified can also be quickly solved. The informal term "quickly" here refers to the existence of an algorithm that can solve the task in polynomial time [1]. The general class of problems for which such an algorithm exists is called $P$ or "class $P$" [1].

Another class of problems called $NP$, which stands for "nondeterministic polynomial time", is defined by the property that if an input to a problem is a solution, then it can be quickly verified [1]. The $P$ versus $NP$ problem asks whether $P$ equals $NP$. If it turns out that $P \neq NP$, which is widely believed to be the case, it would mean that there are problems in $NP$ that are harder to compute than to verify [1]. This would have profound implications for various fields, including cryptography and artificial intelligence [2].

Solving the $P$ versus $NP$ problem is considered to be one of the greatest challenges in computer science [1]. A solution would have a profound impact on our understanding of computation and could lead to the development of new algorithms and techniques that could solve many of the world's most pressing problems [1]. The problem is so difficult that it is considered to be one of the seven Millennium Prize Problems, which are a set of seven unsolved problems that have been offered a 1 million prize for a correct solution [1].

## 2. Materials and methods

$NP$-complete problems are a class of computational problems that are at the heart of many important and challenging problems in computer science. They are defined by the property that they can be quickly verified, but there is no known efficient algorithm to solve them. This means that finding a solution to an $NP$-complete problem can be extremely time-consuming, even for relatively small inputs. In computational complexity theory, a problem is considered $NP$-complete if it meets the following two criteria:

1. **Membership in NP**: A solution to an $NP$-complete problem can be verified in polynomial time. This means that there is an algorithm that can quickly check whether a proposed solution is correct [3].
2. **Reduction to NP-complete problems**: Any problem in $NP$ can be reduced to an $NP$-complete problem in polynomial time. This means that any $NP$-problem can be transformed into an $NP$-complete problem by making a small number of changes [3].

If it were possible to find an efficient algorithm for solving any one $NP$-complete problem, then this algorithm could be used to solve all $NP$ problems in polynomial time. This would have a profound impact on many fields, including cryptography, artificial intelligence, and operations research [2]. Here are some examples of $NP$-complete problems:

- **Boolean satisfiability problem (SAT)**: Given a Boolean formula, determine whether there is an assignment of truth values to the variables that makes the formula true [4].
- **Subset sum problem**: Given a set of positive integers and target $T$, determine whether there is a subset of the integers which sum to precisely $T$ [4].

These are just a few examples of the many $NP$-complete problems that have been studied and have a close relation with our current result. In this work, we show there is an $NP$-complete problem that can be solved in polynomial time. Consequently, we prove that $P$ is equal to $NP$.

## 3. Results

Formally, an instance of **Boolean satisfiability problem (SAT)** is a Boolean formula $\phi$ which is composed of:

1. Boolean variables: $x_1, x_2, \ldots, x_n$;
2. Boolean connectives: Any Boolean function with one or two inputs and one output, such as $\wedge$(AND), $\vee$(OR), $\neg$(NOT), $\Rightarrow$(implication), $\Leftrightarrow$(if and only if);
3. and parentheses.

A truth assignment for a Boolean formula $\phi$ is a set of values for the variables in $\phi$. A satisfying truth assignment is a truth assignment that causes $\phi$ to be evaluated as true. A Boolean formula with a satisfying truth assignment is satisfiable. The problem $SAT$ asks whether a given Boolean formula is satisfiable [4].

A literal in a Boolean formula is an occurrence of a variable or its negation [3]. A Boolean formula is in conjunctive normal form, or $CNF$, if it is expressed as an AND of clauses, each of which is the OR of one or more literals [3]. A Boolean formula is in 3-conjunctive normal form or $3CNF$, if each clause has exactly three distinct literals [3].

For example, the Boolean formula:

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

is in $3CNF$. The first of its three clauses is $(x_1 \vee \neg x_1 \vee \neg x_2)$, which contains the three literals $x_1$, $\neg x_1$, and $\neg x_2$.

We define the following problem:

**Definition 1.** ***Monotone one-in-three 3-satisfiability (M-1IN3-3SAT)***

*INSTANCE: A $3CNF$ formula with monotone clauses (meaning the variables are never negated).*

*QUESTION: Is there exists a truth assignment such that each clause contains exactly one true literal?*

*REMARKS: $M-1IN3-3SAT \in$ NP–complete [4].*

Finally, we deduce our main goal:

**Theorem 1.** $M-1IN3-3SAT \in P$.

**Proof.** Suppose we have the following sequence of variables in a given instance of $M-1IN3-3SAT$:

$$x_1, \ldots, x_n.$$

For each variable $x_i$ in the $3CNF$ formula, we define the functions $f$, $g$ and $h$ as,

- $f(x_i)$ is the number of unordered clauses $(x_i \vee x_j \vee x_k)$ such that this one belongs to the $3CNF$ formula whenever $j > i$ and $k > i$ at the same time;

https://doi.org/10.33774/coe-2024-pmlqm-v3 **ORCID:** https://orcid.org/0000-0001-8210-4126 

- $g(x_i)$ is the number of unordered clauses $(x_i \vee x_j \vee x_k)$ such that this one belongs to the $3CNF$ formula whenever $j < i$ and $k < i$ at the same time;
- $h(x_i)$ is the number of unordered clauses $(x_i \vee x_j \vee x_k)$ such that this one belongs to the $3CNF$ formula whenever either $j > i$ and $k < i$ or $j < i$ and $k > i$ at the same time.

We define a state as a quadruple $(i, s, r, t)$ of integers. This state represents the fact that,

$$\text{“}\textit{the subset of variables } x_1, \ldots, x_i$$

$$\textit{with } s \textit{ satisfied clauses}$$

$$\textit{where } -m \leq r \leq m \textit{ and } -m \leq t \leq m''\text{,}$$

where $m$ is the amount of clauses into the $3CNF$ formula. Each state $(i, s, r, t)$ has two next states:

1. $(i+1, s + h(x_{i+1}) + f(x_{i+1}) + g(x_{i+1}), r + f(x_{i+1}) + h(x_{i+1}), t - g(x_{i+1}) - h(x_{i+1}))$, implying that $x_{i+1}$ is included in the subset and it is evaluated as true;
2. $(i+1, s, r - g(x_{i+1}), t + f(x_{i+1}))$, implying that $x_{i+1}$ is included in the subset and it is evaluated as false.

Starting from the initial state $(0, 0, 0, 0)$, it is possible to use any graph search algorithm (e.g. **Breadth-first search (BFS)** [3]) to search the state $(n, m, 0, 0)$. Certainly, we satisfy all the clauses if they exactly contain one true literal just adding 1 by the true literal from the left most position (otherwise subtracting 1 by the false literal from the left most position), subtracting 1 by the false literal from the right most position (otherwise adding 1 by the true literal from the right most position) and simultaneously adding and subtracting 1 by the true literal from the middle position placed within each clause. The run-time of this algorithm is at most linear in the number of states. The number of states is at bounded by $n \cdot 4 \cdot m^3$ times and therefore, the whole time required is $O(n \cdot m^3)$. This is our software programming implementation in Python [5]. □

## 4. Conclusion

A proof of $P = NP$ will have stunning practical consequences, because it possibly leads to efficient methods for solving some of the important problems in computer science [1]. The consequences, both positive and negative, arise since various $NP$-complete problems are fundamental in many fields [2]. But such changes may pale in significance compared to the revolution an efficient method for solving $NP$-complete problems will cause in mathematics itself [1]. Research mathematicians spend their careers trying to prove theorems, and some proofs have taken decades or even centuries to be discovered after problems have been stated [1]. A method that guarantees to find proofs for theorems, should one exist of a “reasonable” size, would essentially end this struggle [1].

## References


1. Cook, S.A. The P versus NP Problem, Clay Mathematics Institute. https://www.claymath.org/wp-content/uploads/2022/06/pvsnp.pdf, 2022. Accessed 1 February 2024.
2. Fortnow, L. The status of the P versus NP problem. *Communications of the ACM* **2009**, *52*, 78–86. https://doi.org/10.1145/1562164.1562186.
3. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; The MIT Press, 2009.
4. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1 ed.; San Francisco: W. H. Freeman and Company, 1979.
5. Vega, F. ALMA | MONOTONE 1-IN-3 3SAT Solver. https://github.com/frankvegadelgado/alma, 2024. Accessed 1 February 2024. SHA commit: df13ce0c6c1f8c9bde2433b21e19991cafd191bd.

## Short Biography of Authors

**Frank Vega** is essentially a Back-End Programmer and Mathematical Hobbyist who graduated in Computer Science in 2007. In May 2022, The Ramanujan Journal accepted his mathematical article about the Riemann hypothesis. The article "Robin's criterion on divisibility" makes several significant contributions to the field of number theory. It provides a proof of the Robin inequality for a large class of integers, and it suggests new directions for research in the area of analytic number theory.



https://doi.org/10.33774/coe-2024-pmlqm-v3 **ORCID:** https://orcid.org/0000-0001-8210-4126 Content not peer-reviewed by Cambridge University Press. **License:** CC BY 4.0