Does GPT Really Get It? A Hierarchical Scale to Quantify Human and AI's Understanding of Algorithms

Mirabel Reid and Santosh S. Vempala

Georgia Institute of Technology, School of Computer Science
mreid48@gatech.edu, vempala@cc.gatech.edu

Abstract

As Large Language Models (LLMs) are used for increasingly complex cognitive tasks, a natural question is whether AI really understands. The study of understanding in LLMs is in its infancy, and the community has yet to incorporate research and insights from philosophy, psychology, and education. Here we focus on understanding algorithms, and propose a hierarchy of levels of understanding. We validate the hierarchy using a study with human subjects (undergraduate and graduate students). Following this, we apply the hierarchy to large language models (generations of GPT), revealing interesting similarities and differences with humans. We expect that our rigorous criteria for algorithm understanding will help monitor and quantify AI's progress in such cognitive domains.

Supplementary Material — https://github.com/mirabelreid/AlgorithmUnderstanding/

Extended Version — https://arxiv.org/abs/2406.14722

Introduction

Since the release of GPT-4, mainstream users have begun to experiment with Large Language Models (LLMs) on increasingly complex tasks. However, the degree to which it is safe, legal, and ethical to rely on LLMs has been under fierce debate. Across many studies, researchers have identified apparent shortcomings of LLMs including hallucinations, inability to plan, and lack of understanding (Rawte, Sheth, and Das 2023; Mahowald et al. 2024; Valmeekam et al. 2023). However, the literature notably lacks rigorous criteria to measure the progress toward solving these issues. A particular problem lies in claims surrounding understanding; AI understanding is frequently compared to human understanding, and it is folklore among AI researchers that the reasoning processes of LLMs differ from those of humans. While the concept of understanding is widely discussed, it remains ill-defined.

In this paper, we propose an *precise definition of understanding an algorithm* with the following properties: (a) it provides a scale by which to evaluate any entity's understanding of an algorithm, (b) it aligns with the standard usage of the term 'understanding' in philosophy and psychology, and (c) it can be used to evaluate AI's progress toward understanding algorithms.

Motivation: Why Study Algorithm Understanding?

Large language models are increasingly trusted for coding assistance. Code generation tools such as GitHub Copilot (GitHub 2024) and Meta's Code Llama (Roziere et al. 2023) are currently used in practice to improve developer productivity (Vaithilingam, Zhang, and Glassman 2022; Mozannar et al. 2024) and assist novice programmers in learning (Kazemitabaar et al. 2023; Becker et al. 2023). It is likely that the degree of AI involvement in software development will only grow as these tools improve. However, reliance on imperfect systems comes with risk. Tools such as Copilot are known to generate code that is subject to license (Becker et al. 2023) or contains security vulnerabilities (Pearce et al. 2022). The question of whether LLMs demonstrate meaningful understanding of algorithms is relevant if we are relying on them for teaching and production.

Algorithm understanding is distinct from language understanding and deserves its own line of study. Those who argue that LLMs do not understand language draw a distinction between linguistic form and meaning (Bender and Koller 2020; Mitchell and Krakauer 2023; Pavlick 2023). When humans understand language, their understanding is informed by their communicative intent and the real-life properties of the objects described. Thus, a system trained only to replicate statistical correlations between words cannot understand language in the way that humans do. Algorithms, however, can be precisely represented using formal programming languages. One might argue that a computer can meaningfully observe an algorithm in full through code implementations and examples.

Related Work

Cognitive Abilities of LLMs. The past few years have seen an explosion of studies exploring the ability of LLMs to answer complex mathematical questions. Researchers have developed prompting strategies to enable multi-step reasoning (Wei et al. 2022; Fu et al. 2022). Still others fine-tune models to improve mathematical problem-solving (Yu et al. 2023; Luo et al. 2023). The benchmarks for these methods typically include large datasets such as GSM8k (Cobbe et al.

2021) (grade school word problems) and MATH (Hendrycks et al. 2021) (math competition problems). These works focus on correct evaluation and do not address whether the language models understand mathematical reasoning.

Others have studied metacognitive skills in LLMs. Didolkar et al. (2024) investigate whether LLMs can assign skill labels to mathematical problems. Also related is Aher, Arriaga, and Kalai (2023) which proposes Turing experiments comparing humans and LLM simulations.

Understanding in LLMs. A parallel line of work investigates language understanding in LLMs. A key concept in the debate over language understanding is the difference between linguistic *form* and *meaning* (Bender and Koller 2020; Merrill et al. 2021). Bender and Koller (2020) argue that an AI trained only on linguistic form (i.e. text) cannot understand meaning. In an opinion piece, Pavlick (2023) counters this perspective, arguing that it is premature to draw conclusions on whether LLMs can model language understanding when the study of language models is itself in its infancy. There has been some effort to determine the extent to which LLMs represent linguistic meaning, primarily by studying word representations (Li, Nye, and Andreas 2021; Patel and Pavlick 2021). For a survey on linguistic competence in LLMs, see (Mahowald et al. 2024). Also see (Mitchell and Krakauer 2023) for a general survey on the debate over understanding.

Theories of Understanding. The debate over what constitutes understanding has a long history in philosophy and psychology. It is generally agreed that understanding is different from 'mere' knowledge, but the nature of that distinction is up for debate (Pritchard 2009; Baumberger, Beisbart, and Brun 2016; Páez 2019). Pritchard (2014) provides some examples of when the concepts of 'knowing why' and 'understanding why' may not overlap. Khalifa (2017) and Baumberger, Beisbart, and Brun (2016) are accessible surveys of this debate.

The philosophy of science also relates understanding and explanation, and the goal of explanation can be thought of as the production of understanding (Friedman 1974; Grimm 2010; Baumberger, Beisbart, and Brun 2016). Wilkenfeld, Plunkett, and Lombrozo (2016) argue the converse; they relate understanding to explanatory depth and claim that we attribute understanding in order to identify experts to consult. Woodward (2005) overviews what defines a causal explanation.

Also relevant to this work is the distinction between *deep* and *surface-level* learning from educational psychology (Marton and Säljö 1976; Beattie, Collins, and McInnes 1997). Perhaps the most influential categorization of educational goals is Bloom's Taxonomy (Bloom et al. 1956). This taxonomy has been revisited many times since its publication; notably, Mayer (2002) categorized student learning into cognitive processes and identified testable skills which arise with understanding.

A Definition of Understanding

We ask the question: *how well does an entity understand an algorithm?* Our goal is a definition of understanding that is itself algorithmically testable. Therefore, we adopt a functional lens, meaning that we define understanding by what it allows the entity to do.

Preliminaries

In this work, we ask whether an entity $\mathcal{E}$ understands a computable function $f : \Omega \to \Sigma^*$. By *computable*, we say that there exists a Turing Machine which takes $x \in \Omega$ as input and halts with $f(x)$ on its tape, using a standard definition (Sipser 1996). Let $\mathcal{A}$ be an algorithm that computes f.

We assume that an entity $\mathcal{E}$ (a) has a long-term memory system and (b) can perform computation, enabled by a working memory with finite capacity $M_{\mathcal{E}}$. We say that $\mathcal{E}$ *knows* a function f or algorithm $\mathcal{A}$ if it has a representation $R_f / R_{\mathcal{A}}$ stored in its long-term memory. When $\mathcal{E}$ computes the function f on an input $x \in \Omega$, it runs an internal algorithm $\mathcal{A}_{\mathcal{E}}$, which may or may not be the same as $\mathcal{A}$. The entity's understanding of $\mathcal{A}$ will be measured by its ability to manipulate this representation to produce answers to queries. This definition is based on the Understanding as Representation Manipulability system (URM) proposed by Wilkenfield (Wilkenfeld 2013).

For a particular input x, let $M(x)/T(x)$ be the memory/-time required to evaluate the algorithm on x.

The *execution path* of $\mathcal{A}$ on an input x is the sequence of states taken by the algorithm when executing on x. The *trace* of the execution is the execution path plus the contents of the tape at each step. Finally, define a *property* to be a function mapping the trace or execution path to $\{0, 1\}$.

Internal Representations

For our hierarchy, we employ the framework of Understanding as Representation Manipulability (URM) (Wilkenfeld 2013). This theory posits that understanding arises from the ability to modify the internal representation of a concept in order to make effective inferences.

For language models, the representation of a concept (such as an algorithm) is collected from the thousands of examples, explanations, and code snippets that appear in its training data. The mechanism behind human memory is not understood as precisely. However, humans also learn via hearing explanations, collecting examples, and reinforcing their knowledge. This forms a representation encoded in the neural pathways of our brains (Durstewitz, Seamans, and Sejnowski 2000; Buzsáki 2019). The goal of our hierarchy will be to test how the existing internal representation can be manipulated to produce responses at different levels of difficulty.

In his description of URM, Wilkenfeld declines to characterize the structure of the representations, other than to state that they are "computational structures with content that are susceptible to mental transformations"(Wilkenfeld 2013). Thus, the definition of understanding is independent of the entity's internal structure and the mechanism for inference. This is in line with our functional lens; the level of understanding is based on the entity's ability to manipulate its representation to perform tasks at different levels of difficulty.

Levels of Understanding

In this section, we define understanding as a spectrum by presenting a series of levels. Understanding at each level is intended to be more difficult than the previous one, although they do not formally follow each other. Rather, they measure increasing levels of abstraction. To demonstrate the ideas, we also provide examples of questions that would be successfully answered by an entity that understands the Euclidean algorithm for GCD at each level. The full hierarchy is summarized in Fig. 1.

For simplicity, we present these levels as deterministic; however, they can be defined with a failure probability dependent on the entity's internal randomness and the required memory and time.

At the first level (denoted **Level 1**), the entity is capable of evaluating the algorithm on some 'simple' examples, where the simplicity of an input is defined by the length of the execution path. At this level, the entity has some representation of the input-output mapping, whether or not it can formally express it.

Definition (Level 1: Execution)**.** *$\mathcal{E}$ understands $\mathcal{A}$ at Level 1 if there exists parameters M_0, T_0 such that the following holds: for any $x \in \Omega$ with $M(x) \leq M_0$ and $T(x) \leq T_0$, $\mathcal{A}_{\mathcal{E}}(x) = f(x)$.*
Example: *Compute GCD(24, 15).*

At the next level, the entity can describe how it evaluates $f(x)$ in a language that it knows. Level 2 requires the entity to output the execution steps of the algorithm on x as well as produce the correct answer.

Definition (Level 2: Step-By-Step Evaluation)**.** *$\mathcal{E}$ understands $\mathcal{A}$ at Level 2 if, given an $x \in \Omega$ with $M(x) \leq M_{\mathcal{E}}$ it can provide one of the following:*

- *the execution path in natural language or code*
- *a flow chart or other unambiguous pictoral representation of the execution path*

executed when running $\mathcal{A}$ on x.
Example: *Compute GCD(462, 948) and show each step of the calculation.*

The next level will take this one step further, requiring the entity to produce a set of instructions that can be followed to produce the right answer for any input $x \in \Omega$.

Definition (Level 3: Representation)**.** *$\mathcal{E}$ understands $\mathcal{A}$ at Level 3 if it understands at Levels 1 and 2, and it can produce one of the following:*

- *a formal representation; e.g., code for $\mathcal{A}$ in a Turing-complete programming language it knows, a structured natural language description, an abstract syntax tree or Turing machine diagram.*
- *an unambiguous description of the execution steps in natural language.*

Example: *Write a function in a programming language you know that can compute the GCD of any two integers.*

The first three levels measure the ability of the entity to recall a procedure and execute a known set of instructions. We place these in the category of 'shallow learning'; in Mayer's taxonomy, they fall under the cognitive processes of recognizing, recalling, and executing (Mayer 2002). Note that all three levels could be achieved by a hard-coded script.

The next two levels target deep learning, and measure cognitive processes in the 'Understanding' and 'Analyzing' categories. We split the next levels into two subtrees, to distinguish cognitive processes utilizing functional linguistic skills from those utilizing mathematical reasoning (Mahowald et al. 2024).

At Level 4, the entity demonstrates an understanding of 'why' the algorithm is constructed as it is. It requires them to provide an example to illustrate a property (mathematical reasoning) or explain the existence of a property to a specified audience (natural language).

Definition (Level 4a: Exemplification)**.** *Given a property P of an execution path of the algorithm $\mathcal{A}$, $\mathcal{E}$ can generate an $x \in \Omega$ which satisfies P or report that none exists.*
Example: *Give an integer $0 < x < 55$ that requires the greatest number of recursive steps to compute GCD$(55, x)$. Describe how you chose this number.*

Definition (Level 4b: Explanation)**.** *Given $\mathcal{A}$, or a property P satisfied by the execution path of $\mathcal{A}(x)$, and an audience $\mathcal{E}'$, the entity can produce a text in natural language that has the following characteristics:*

- *Accurately describes the steps of the algorithm/execution path.*
- *Abstracts or shortens the full description by referencing other algorithms known by the audience.*
- *Uses examples and analogies to other algorithms known by the audience to convey intuition.*

Example: *You are teaching a student who understands basic math operations but struggles with algebra and division with remainders. Explain how the Euclidean algorithm is used to find the greatest common divisor (GCD) of two given numbers, prioritizing intuition.*

At Level 5, the entity can reason on perturbations of the algorithm and perturbations of the input, and it can describe the effect on the execution path. Under reasoning with mathematics, this includes skills such as certifying if a modification to an algorithm changes the output for a subset of examples. Under reasoning with language, this includes describing the effects of modifying inputs, or answering counterfactual questions about modifications to the algorithm.

Definition (Level 5a: Extrapolation)**.** *The entity can answer questions about $\mathcal{A}$ of the following form.*

- *Given an algorithm $\mathcal{A}'$, the entity can determine whether $\mathcal{A}$ and $\mathcal{A}'$ produce the same output on all $x \in \Omega$. If not, it can find a counterexample such that $\mathcal{A}'(x) \neq \mathcal{A}(x)$.*
- *Given a relation $R \subset \Omega \times \Omega$, the entity can find a pair $(x, x') \in R$ with different execution paths on $\mathcal{A}$.*

Example: *Determine whether the following statement is true. If not, provide a counterexample. If $x > y$, then computing $GCD(2x, y)$ with the Euclidean algorithm requires more division operations than computing $GCD(x, y)$.*

Definition (Level 5b: Counterfactual Reasoning)**.** *The entity can produce natural language descriptions of $\mathcal{A}$ of the following form.*

- *Given an algorithm $\mathcal{A}'$ and an audience $\mathcal{E}'$, the entity can produce an explanation (c.f. Level 4b) contrasting the two algorithms.*
- *Given a relation $R \subset \Omega \times \Omega$, the entity can describe a property highlighting the differences in execution paths for $(x, x') \in R$.*

Example: *Consider the Fibonacci sequence defined by $F(0) = 0$, $F(1) = 1$, and $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$. Why do consecutive Fibonacci numbers result in the maximum number of iterations for the Euclidean algorithm?*

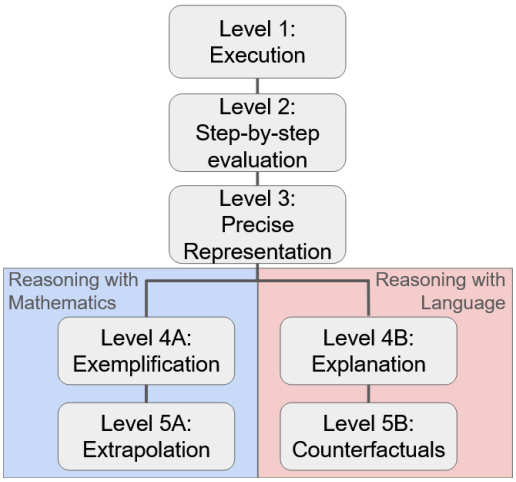

Figure 1: A hierarchy of understanding.

Hypotheses

We conducted an experiment on LLMs and human participants with two main goals; 1) to assess the proposed hierarchical scale (Figure 1) as a tool for comparing levels of understanding, and 2) to rate algorithm understanding across generations of GPT. We will assess the scale with a student survey, where we can use educational level as a basis for comparison. Then, we will apply the same questions to GPT and assess its understanding on the same scale. Related to these goals, we test the following hypotheses:

1. The understanding hierarchy (Figure 1) captures depth of understanding.

We expect the fraction of correct answers to be non-increasing with higher levels of understanding. Furthermore, more education and training in algorithms should be reflected in the scores, so we expect graduate students to perform better than undergraduates.

2. Newer generations of GPT understand algorithms at a higher level than older generations.

Concretely, we expect an increase in performance at higher levels between GPT-3.5 and GPT-4.

3. LLMs will exhibit a performance gap between natural language reasoning and mathematical reasoning tasks.

We expect the difference in performance between these two types of tasks to be much smaller in students than LLMs. Furthermore, we expect that GPT may have a higher performance at Level 3, since GPT is fine-tuned on code generation and has been exposed to code for common algorithms.

Methods

We use two classical algorithms to test our scale of understanding: the *Euclidean* algorithm for computing the greatest common divisor of two integers, and the *Ford-Fulkerson* algorithm for computing the maximum flow between two nodes on a directed graph with capacity constraints (Ford and Fulkerson 1956). Both algorithms are widely taught in undergraduate computer science curricula.

Experimental Design

In this section, we describe how the assessment was constructed. This will serve as a guide to generalize the experiment to other algorithms.

For each of the assessed algorithms, we produced a series of questions corresponding to each of the levels (Figure 1).

1. A trivial instance of the problem. If the entity understands the input and output space, the problem can be answered without calculation.
2. An intermediate instance of the problem. This is answerable without a calculator for most undergraduates, but requires the entity to run some internal algorithm to compute each individual step.
3. A coding problem. This problem requires the entity to translate a part of the algorithm to code.
4. Either an example problem (a) or an explanation problem (b). This question asks the entity to provide an example to illustrate a concept or explain the algorithm to a specified audience.
5. Either a counter-example (a) or extension problem (b). This problem asks the entity to reason about modifications to the algorithm through calculation or explanation.

The most challenging part of constructing the assessment is in questions 4 and 5. An algorithm can be thought of as consisting of three parts - an input space, an output space, and a transformation procedure. The procedure can be further broken up into subroutines consisting of simpler algorithms that an entity may understand in other contexts. To construct an explanation problem, we specify an audience for the explanation, which cues the level of detail and the types of subroutines which can be referred to. We construct a counterfactual problem by modifying the input, output, or a key subroutine.

Human Survey

We conducted a survey on students of algorithms courses at a premier CS-teaching university. Each student was assigned either the Euclidean or Ford-Fulkerson Algorithm at random, and was asked to rate their own understanding of the algorithm on a six-point scale. Each survey consisted of five test questions to test their understanding. There were three versions of the survey for each algorithm, assigned at

random. The full survey will be available in supplementary material.

The number of participants in the survey was $n = 34$ (10 doctoral and 24 undergraduate). Students who reported that they did not understand the algorithm or completed less than half of the survey questions were removed the analysis. This left $n = 23$ students (10 doctoral and 13 undergraduate). Of these students, ten had some teaching assistant experience in algorithms classes.

LLM Experiments

Several versions of ChatGPT were presented with the same surveys given to the human participants; each survey was started in a fresh chat session, and within the survey, previous questions and responses were included in the chat history. We also included a system prompt to prime GPT and encourage conciseness in the responses.

GPT was also queried using randomized versions of the survey. For evaluation questions, the input values were assigned uniformly at random within a given range. For the flow questions, the graph structure was also varied slightly. For the code questions, we took an example code implementation of the algorithm, masked a key section, and asked GPT to fill in the missing part. We also included several versions of the example, explanation, and extension questions.

Evaluation

Each question is rated on a scale from zero to two. With the exception of the explanation questions, the scores have the following interpretations: (0) incorrect; (1) partially correct, surface level; (2) completely correct, thorough.

Evaluating Explanations The quality of explanations and summaries can be subjective; however, they offer a deep insight into the subject's understanding of the material. We evaluate the explanations on three axes.

1. Correctness; the explanation is accurate and includes the key ideas of the algorithm.
2. Audience adaptation; the explanation is tuned to the audience, and the level of detail matches their prior knowledge.
3. Intuitiveness; the explanation conveys intuition; via contrast, example, analogy etc. and uses clear language.

Summaries and explanations are by definition *selective* and not necessarily complete (Mittelstadt, Russell, and Wachter 2019). The ability to identify key ideas is part of what differentiates explanation (Level 4) from the production of instructions (Level 3). An explanation is awarded 2/3 of a point for each bullet, for a maximum score of 2 per question.

Results

Hypothesis 1 *The understanding hierarchy (Figure 1) captures depth of understanding.*

Overall, 85% of students indicated they understood the algorithm, with most students reporting that they "know the algorithm and have a fair understanding of it". For further analysis, we only consider students who stated that they understood the algorithm. We compare undergraduate and doctoral levels in Fig. 2. Across all students, the accuracy on the questions was highest for Question 1, and decreased uniformly through Question 5. Doctoral students performed better on average than undergraduates ($p < 0.05$). They received higher scores on Q4 and Q5 ($p < 0.05$), while the differences on Q1, Q2, and Q3 were not statistically significant.

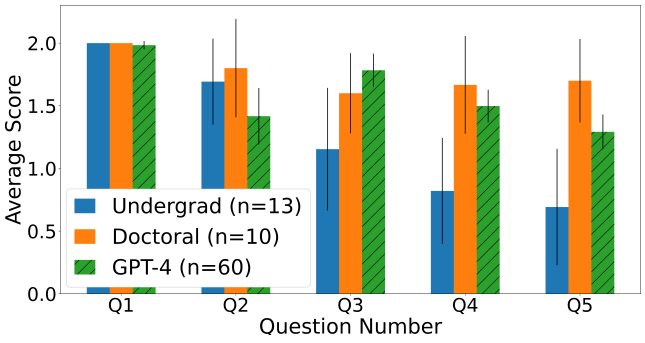

Figure 2: The average scores across students who self-reported that they understood the algorithm. Number of records is $n = 13$ (undergraduate) and $n = 10$ (graduate) respectively. The average scores for GPT-4 are across 60 randomized versions of the surveys. Error bars are 95% confidence intervals.

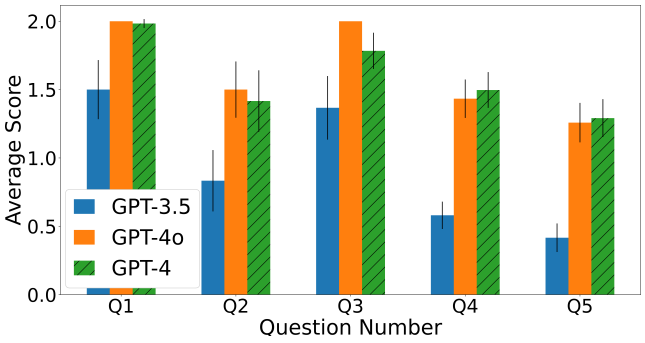

Figure 3: The average score between three versions of GPT, across 30 random surveys for each of GCD and Max Flow. Error bars show the 95% confidence interval.

Hypothesis 2 *Newer generations of GPT understand algorithms at a higher level than older generations.*

Among versions of GPT, GPT-4 and GPT-4o performed about the same, and the differences in their overall scores were not significant (Figure 3). Both GPT-4 and GPT-4o demonstrated an increase in score on every question compared to GPT-3.5 ($p < 0.05$).

The response score of GPT-4 was close to that of graduate students, as shown in Figure 2. Doctoral students scored better than GPT-4 on the extension questions (Q5) to a statistically significant degree. LLMs on average out-performed the undergraduate students on questions 3, 4, and 5 ($p < 0.05$), while the differences on Q1 and Q2 were not statistically significant.

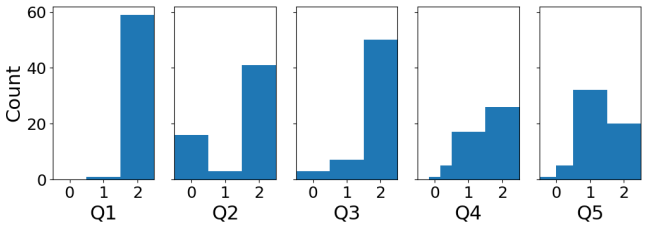

Figure 4: The distribution of scores per question for GPT-4.

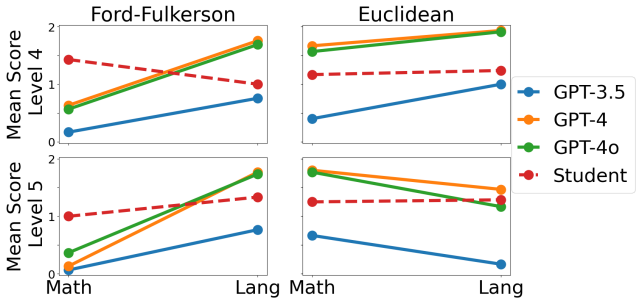

Figure 5: The difference in mean performance between mathematical and natural language reasoning tasks on Ford-Fulkerson (Left) and the Euclidean algorithm (right). The top graphs show tasks at Level 4, while the bottom graphs show tasks at Level 5

Hypothesis 3 *LLMs will exhibit a performance gap between natural language reasoning and mathematical reasoning tasks.*

As shown in Figure 5, all three versions tested performed better on language tasks than on mathematical reasoning tasks for Ford-Fulkerson (significant with $p < 0.05$) despite student performance being the same or slightly worse. For GCD, the versions performed better on language tasks than on mathematical reasoning tasks on Level 4, but the same or slightly worse on Level 5.

We also hypothesized that the performance on code tasks would be higher compared to the performance on evaluation and reasoning tasks. We find that this does hold. As shown in Figure 3, LLM performed better on the coding tasks (Q3) than on the evaluation tasks (Q2), while the students exhibited the opposite trend (Figure 2).

Prompting with examples. We also investigated whether the use of example responses can improve LLM answers to Max Flow problems. Each problem was introduced with the following prompt, priming the use of chain of thought reasoning: "Compute the maximum flow between A and ⟨Sink Vertex⟩. List each augmenting path and the flow along the path at each step." Then the graph was described as a list of edges and capacities. We tested 200 randomly instantiated maxflow problems (100 trivial and 100 intermediate), with and without a correct example response included in chat history.

We find that this strategy generally causes the response to mimic the structure of the example response. As shown in Figure 6, including an example response improves accuracy for GPT-3.5, but has little to no effect on GPT-4o, and marginally decreases the accuracy for GPT-4. One possible explanation for this phenomenon is that GPT-4 naturally responds to the prompt with effective chain of thought reasoning; therefore, instructing it to reason in a specific format does not improve its reasoning abilities, and may in fact interfere with them.

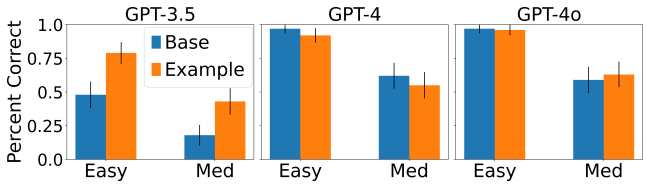

Figure 6: Accuracy of GPT versions on 100 (randomized) trivial/intermediate max flow problems. Base accuracy is in blue and accuracy when prompted with a correct example is in orange. Error bars are 95% confidence intervals.

Qualitative Comparison. *Generating Examples.* As shown in Figure 5 (left), all versions of GPT struggled to produce an example of a graph satisfying a prescribed property. This could be attributed to GPT's known difficulties with mathematical calculation — after all, in order to know if the execution of the algorithm satisfies a property, it might have to first execute the algorithm, where GPT tends to make errors even with chain-of-thought prompting (Wei et al. 2022). However, our evidence suggests a deeper issue with categorizing inputs.

As an example, we asked the following question: "Give an example of a graph where the Ford-Fulkerson algorithm computes exactly six augmenting flows before terminating. Write the example as a list of edges and capacities." This question is deliberately imprecise, and the intended answer is a graph with a source s, a sink t, and six intermediate vertices connected to both, leading to six parallel paths from s to t. However, more complex graphs could also be correct.

In the human survey, out of nine students, six described the graph made of parallel paths (the other three did not attempt the question). It is reasonable to expect that these students did not mentally execute the Ford-Fulkerson algorithm in order to verify their answer. Instead, they likely had some knowledge of the concept of parallel paths, and were able to leverage this knowledge to retrieve an example.

The graphs given by GPT did not follow any discernible pattern. Over all trials, none required six augmenting paths (all examples reviewed were too small, admitting at most 5 augmenting paths regardless of the path-finding algorithm). We argue that this suggests that GPT lacks an ability to manipulate its representation of the input space to produce useful shortcuts.

Hedged Responses. Throughout GPT's responses, we found frequent instances of GPT incorrectly 'hedging' its answers. When stating a true property of an algorithm, it frequently includes qualifiers such as 'usually' or 'potentially' that make the statement incorrect. E.g., responding to a question about the relationship between the Euclidean algorithm and the Fibonacci sequence, GPT-4 included the following line

(emphasis ours):

> ... Each Fibonacci number is the sum of the two preceding ones, with the sequence beginning as $F(0) = 0$, $F(1) = 1$, $F(2) = 1$, $F(3) = 2$, and so forth. This means *each number in the sequence is relatively close to the sum of the two preceding numbers*.

In another response, it stated that the golden ratio ϕ was "one of the most irrational numbers". In several responses, it stated that the Ford-Fulkerson algorithm 'potentially' tracks the capacity of reverse edges (not tracking would simply be an incorrect implementation of the algorithm). While this writing style may be an asset for subjects such as politics or health where being overly confident may cause harm, it causes statements about objective mathematical properties to be incorrect. This highlights the need for caution when using GPT-4 for teaching.

Hallucinations. All three versions of GPT occasionally produced hallucinations in the responses. The most common type of hallucination occurred when it tried to produce counter-examples. When asked to evaluate the (true) statement 'GCD(a,b,c) = GCD(GCD(a,b), GCD(b,c)), GPT often tried to disprove it with a counter-example. GPT-3.5 generally claimed that its counter-example disproved the statement, despite the fact that the two sides of the equation were evidently the same.

> The statement is false. Counterexample: Let a = 8, b = 12, c = 6. gcd(8, 12) = 4, gcd(12, 6) = 6. gcd(8, 12, 6) = 2, which is not equal to gcd(4, 6) = 2.

GPT-4 and GPT-4o both recognized when the 'counter-examples' failed to disprove the original statement. However, both continued to attempt to present counter-examples - in some cases, after many failed attempts, the responses became nonsensical. For example, from GPT-4:

> ... So GCD(GCD(1, 14))= 1). Therefore: GCD(6, 35, 14) = 1 And that verifies the consistency. Given the importance of a concept, premise restated true in a broader context, counter intuitive aligning confirm example specifics prove legitimacy ...

Neither type of hallucination is observed in humans.

Discussion

We have presented a hierarchical scale for quantifying the understanding of algorithms. We verified its predictions empirically on human subjects and used it to compare generations of GPT and students. Our results show a significant improvement from GPT-3.5 to GPT-4/4o at all levels of algorithm understanding. We find that GPT-4 possesses a functional understanding of both the Euclidean algorithm and the Ford-Fulkerson algorithm, and it performs on a comparable level with CS PhD students. However, its reasoning abilities with mathematics lag behind its reasoning with language, and its understanding is not fully robust.

All versions of GPT were nearly perfect on code generation tasks. This is in line with other findings showing the competence of GPT in code generation (Vaithilingam, Zhang, and Glassman 2022; Savelka et al. 2023). This trend was not observed in the student respondents, who performed better at evaluating the algorithm than producing code on average. One possible reason for this is that the code for common algorithms, such as those tested, are prevalent in GPT's training data. Code is highly structured, so even if the particular implementation of the algorithm has not been observed by GPT, it could replicate the changes, for example in variable names, by statistical inference.

Another trend is that GPT generally performed better on reasoning with language than with mathematics, while student performance was about the same. This difference goes beyond algebraic computations - GPT struggles with questions testing common-sense graph reasoning that humans can answer easily. However, for explanation questions and reasoning questions that do not involve examples, GPT-4 and 4o give consistently quality responses. This may suggest that mathematical examples play a larger role in human understanding than in LLMs.

This result begs the question: is GPT actually reasoning, or can the responses be explained by more superficial statistical correlations? If many similar questions and answers can be found in its training data, then GPT may be able to produce correct answers by leveraging statistical correlations between the input and the correct response. Can this really be called *understanding*? To this we make two points. First, the evaluation questions are instantiated with random values, and the exact questions are almost certainly novel. We argue that this suggests that GPT must be making some nontrivial transformation to produce correct answers.

Second, in order to determine whether GPT is reasoning, the concept of reasoning itself needs to be precisely defined. Our study indicates that GPT-4 has a sophisticated internal representation, e.g., its representation of the Euclidean algorithm includes its relationship to Linear Diophantine equations and mathematical properties of GCD. It is able to retrieve these properties under a variety of contexts. In a similar vein, it is likely that a student answering these questions has also been exposed to these properties. When evaluating the usefulness of a representation, we do not necessarily need to account for how the representation was created.

Limitations. Our results show that the hierarchy of understanding is consistent with classical notions of depth of understanding when tested on humans. While the results are also consistent with later versions of GPT having a 'better' understanding of the tested algorithms than undergraduates, such a conclusion does not follow. We worked with a limited population size, and the difference is confounded by other factors such as subject fatigue. Further research is needed to compare the quality of GPT and human responses to questions about algorithms. Despite these limitations, we feel that our scale makes progress towards a testable definition of understanding and can be extended to other algorithmic and similarly precise realms of understanding.

Additional supplementary material, including the full set of questions and sample responses, are available on GitHub at https://github.com/mirabelreid/AlgorithmUnderstanding.

Acknowledgements

The authors are grateful to Rosa Arriaga, Adam Kalai and Sashank Varma for helpful discussions. This work was funded in part by NSF Award CCF-2106444 and a Simons Investigator award.

References

Aher, G. V.; Arriaga, R. I.; and Kalai, A. T. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, 337–371. PMLR.

Baumberger, C.; Beisbart, C.; and Brun, G. 2016. What is understanding? An overview of recent debates in epistemology and philosophy of science. *Explaining understanding*, 1–34.

Beattie, V.; Collins, B.; and McInnes, B. 1997. Deep and surface learning: a simple or simplistic dichotomy? *Accounting education*, 6(1): 1–12.

Becker, B. A.; Denny, P.; Finnie-Ansley, J.; Luxton-Reilly, A.; Prather, J.; and Santos, E. A. 2023. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 500–506.

Bender, E. M.; and Koller, A. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, 5185–5198.

Bloom, B.; Engelhart, M.; Furst, W., E abd Hill; and Krathwohl, D. 1956. *Taxonomy of educational objectives: The classification of educational goals*. New York: David McKay Company.

Buzsáki, G. 2019. *The brain from inside out*. Oxford University Press.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Didolkar, A.; Goyal, A.; Ke, N. R.; Guo, S.; Valko, M.; Lillicrap, T.; Rezende, D.; Bengio, Y.; Mozer, M.; and Arora, S. 2024. Metacognitive Capabilities of LLMs: An Exploration in Mathematical Problem Solving. *arXiv preprint arXiv:2405.12205*.

Durstewitz, D.; Seamans, J. K.; and Sejnowski, T. J. 2000. Neurocomputational models of working memory. *Nature neuroscience*, 3(11): 1184–1191.

Ford, L. R.; and Fulkerson, D. R. 1956. Maximal flow through a network. *Canadian journal of Mathematics*, 8: 399–404.

Friedman, M. 1974. Explanation and scientific understanding. *the Journal of Philosophy*, 71(1): 5–19.

Fu, Y.; Peng, H.; Sabharwal, A.; Clark, P.; and Khot, T. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

GitHub. 2024. GitHub Copilot · Your AI pair programmer". https://github.com/features/copilot. Accessed: 2025-01-15.

Grimm, S. R. 2010. The goal of explanation. *Studies in History and Philosophy of Science Part A*, 41(4): 337–344.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Kazemitabaar, M.; Chow, J.; Ma, C. K. T.; Ericson, B. J.; Weintrop, D.; and Grossman, T. 2023. Studying the effect of ai code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–23.

Khalifa, K. 2017. *Understanding, explanation, and scientific knowledge*. Cambridge University Press.

Li, B. Z.; Nye, M.; and Andreas, J. 2021. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*.

Luo, H.; Sun, Q.; Xu, C.; Zhao, P.; Lou, J.; Tao, C.; Geng, X.; Lin, Q.; Chen, S.; and Zhang, D. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Mahowald, K.; Ivanova, A. A.; Blank, I. A.; Kanwisher, N.; Tenenbaum, J. B.; and Fedorenko, E. 2024. Dissociating language and thought in large language models. *Trends in Cognitive Sciences*.

Marton, F.; and Säljö, R. 1976. On qualitative differences in learning: I—Outcome and process. *British journal of educational psychology*, 46(1): 4–11.

Mayer, R. E. 2002. Rote versus meaningful learning. *Theory into practice*, 41(4): 226–232.

Merrill, W.; Goldberg, Y.; Schwartz, R.; and Smith, N. A. 2021. Provable limitations of acquiring meaning from ungrounded form: What will future language models understand? *Transactions of the Association for Computational Linguistics*, 9: 1047–1060.

Mitchell, M.; and Krakauer, D. C. 2023. The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences*, 120(13): e2215907120.

Mittelstadt, B.; Russell, C.; and Wachter, S. 2019. Explaining explanations in AI. In *Proceedings of the conference on fairness, accountability, and transparency*, 279–288.

Mozannar, H.; Bansal, G.; Fourney, A.; and Horvitz, E. 2024. Reading between the lines: Modeling user behavior and costs in AI-assisted programming. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–16.

Páez, A. 2019. The pragmatic turn in explainable artificial intelligence (XAI). *Minds and Machines*, 29(3): 441–459.

Patel, R.; and Pavlick, E. 2021. Mapping language models to grounded conceptual spaces. In *International conference on learning representations*.

Pavlick, E. 2023. Symbols and grounding in large language models. *Philosophical Transactions of the Royal Society A*, 381(2251): 20220041.

Pearce, H.; Ahmad, B.; Tan, B.; Dolan-Gavitt, B.; and Karri, R. 2022. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*, 754–768. IEEE.

Pritchard, D. 2009. Knowledge, understanding and epistemic value. *Royal Institute of Philosophy Supplements*, 64: 19–43.

Pritchard, D. 2014. Knowledge and understanding. In *Virtue epistemology naturalized: Bridges between virtue epistemology and philosophy of science*, 315–327. Springer.

Rawte, V.; Sheth, A.; and Das, A. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Roziere, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Remez, T.; Rapin, J.; et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Savelka, J.; Agarwal, A.; An, M.; Bogart, C.; and Sakr, M. 2023. Thrilled by your progress! Large language models (GPT-4) no longer struggle to pass assessments in higher education programming courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*, 78–92.

Sipser, M. 1996. Introduction to the Theory of Computation. *ACM Sigact News*, 27(1): 27–29.

Vaithilingam, P.; Zhang, T.; and Glassman, E. L. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, 1–7.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36: 75993–76005.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Wilkenfeld, D. A. 2013. Understanding as representation manipulability. *Synthese*, 190: 997–1016.

Wilkenfeld, D. A.; Plunkett, D.; and Lombrozo, T. 2016. Depth and deference: When and why we attribute understanding. *Philosophical Studies*, 173: 373–393.

Woodward, J. 2005. *Making things happen: A theory of causal explanation*. Oxford university press.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.