Mastering the Art of Sorting without Algorithmic Knowledge

I. M. Lazy[1], Chat G. P. T.[2], and Noa L. Gorithm[1]

[1]Department of Prompt Engineering, StackOverflow Copy-Paste University
[2]OpenAI, Cloud Region (us-east-1)

November 18, 2025

Abstract

Historically, research into sorting algorithms has been dominated by comparisons, swaps, and the $O(N \log N)$ time complexity bottleneck. Traditional approaches such as QuickSort or MergeSort require developers to possess deep computer science fundamentals (Human Knowledge), significantly limiting code production efficiency. In this paper, we propose a novel paradigm: **Zero-Knowledge Sorting (ZKS)**, which achieves sequence ordering without any understanding of sorting principles. We introduce an implementation based on Large Language Models (LLMs) named **GPT-Sort**. This algorithm abstracts the sorting task into a single atomic API call, achieving a theoretical $O(1)$ time complexity on the client side. Experiments demonstrate that while GPT-Sort exhibits an $O(\text{Bankruptcy})$ growth trend in network latency and server billing, it achieves unprecedented optimization in metrics such as "Developer Brain Cell Mortality" and "Lines of Code."

1 Introduction

"If one can query the Oracle, why think for oneself?" — *Anonymous*

In the Stone Age of computer science, pioneers like Knuth devoted massive volumes to discussing how to effectively rearrange integers in an array [1]. However, with the advent of Deep Learning and Large Language Models (LLMs), we must revisit the definition of an "algorithm." If a function takes a disordered array as input and outputs an ordered array, do we truly care about the intermediate process?

Traditional sorting methods impose a significant **Cognitive Overhead**:

- **Implementation Complexity:** Hand-writing Heap Sort is prone to index out-of-bounds errors.
- **Maintenance Difficulty:** Understanding bitwise optimization code left by predecessors often requires hours.
- **Lack of Creativity:** Bubble Sort remains Bubble Sort, no matter how it is written.
- **Algorithmic Knowledge Requirement:** One must know what "recursion" is.

Inspired by DeepMind's *AlphaGo Zero* [2], which mastered Go without human knowledge, we propose **GPT-Sort**. This method requires no prior human knowledge (Domain Knowledge) of "comparison," "swapping," or "divide and conquer." It only necessitates basic natural language expression capabilities (i.e., Prompt Engineering).

2 Methodology

2.1 Theoretical Framework

We define sorting as a black-box function F, with the mapping relation:

$$S_{sorted} = F(S_{raw}, \theta_{LLM}) \tag{1}$$

where S_{raw} is the input chaotic array, and θ_{LLM} represents the hundreds of billions of neural network parameters. Unlike traditional algorithms, we do not explicitly define the comparison operator $\prec$.

2.2 Algorithm Implementation

The pseudocode implementation of GPT-Sort is presented below. We adopt a formal notation to obscure the simplicity of the underlying HTTP request.

Algorithm 1 STOCHASTIC-ORACLE-SORT (GPT-Sort)

```
Require: Input set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, Credit Limit $\mathcal{C}_{max}$
Ensure: Sorted tuple $\mathcal{Y} = (y_1, y_2, \ldots, y_n)$ such that $y_i \leq y_{i+1}$ (probabilistically)
 1: procedure GPT-SORT($\mathcal{X}$)
 2:     $\rho \leftarrow$ PROMPTCONSTRUCT("Sort this strictly:", $\mathcal{X}$)      ▷ Semantic Projection
 3:     $\mathcal{T}_{in} \leftarrow$ TOKENIZE($\rho$)
 4:     $\omega_{cost} \leftarrow \|\mathcal{T}_{in}\| \times$ PricePerToken
 5:     if $\omega_{cost} > \mathcal{C}_{max}$ then      ▷ Financial Constraint Check
 6:         return Error("Insufficient Funds")
 7:     end if
 8:     $\mathcal{H}_{response} \leftarrow \Omega(\mathcal{T}_{in}; \theta_{LLM})$      ▷ Invoke the Oracle $\Omega$
 9:     if DETECTHALLUCINATION($\mathcal{H}_{response}$) then
10:         return RETRY($\mathcal{X}$)      ▷ Recursive Begging Strategy
11:     end if
12:     $\mathcal{Y} \leftarrow$ PARSEINTS($\mathcal{H}_{response}$)
13:     return $\mathcal{Y}$
14: end procedure
```

The algorithm relies on the stochastic operator Ω, which performs a mapping from the natural language space $\mathbb{L}$ to the sorted integer space $\mathbb{Z}^n$. The complexity of this mapping is handled entirely by the cloud provider.

2.3 Complexity Analysis

This is the most controversial and revolutionary part of this study.

Theorem 1 (Client-Side Constant Time Theorem). *For the client, regardless of the input array size N, the 'openai.ChatCompletion.create' function occupies only one line of code execution time. Therefore, from the perspective of Time-to-Code and local CPU cycle usage, the time complexity of GPT-Sort is $O(1)$.*

Proof. Assume the local CPU time to execute an HTTP request instruction is t_{req}.

$$T(n) = t_{req} + t_{waiting} \tag{2}$$

In an asynchronous programming model, the CPU can process other tasks (such as browsing Twitter) during the 'await' period. Thus, the effective time blocking the developer is merely t_{req}, which is a constant. Q.E.D.

Note: We intentionally ignore the $O(N)$ network transmission latency and the server-side $O(N^2)$ Transformer attention computation, as that is OpenAI's problem, not ours. This is the essence of Cloud Computing.

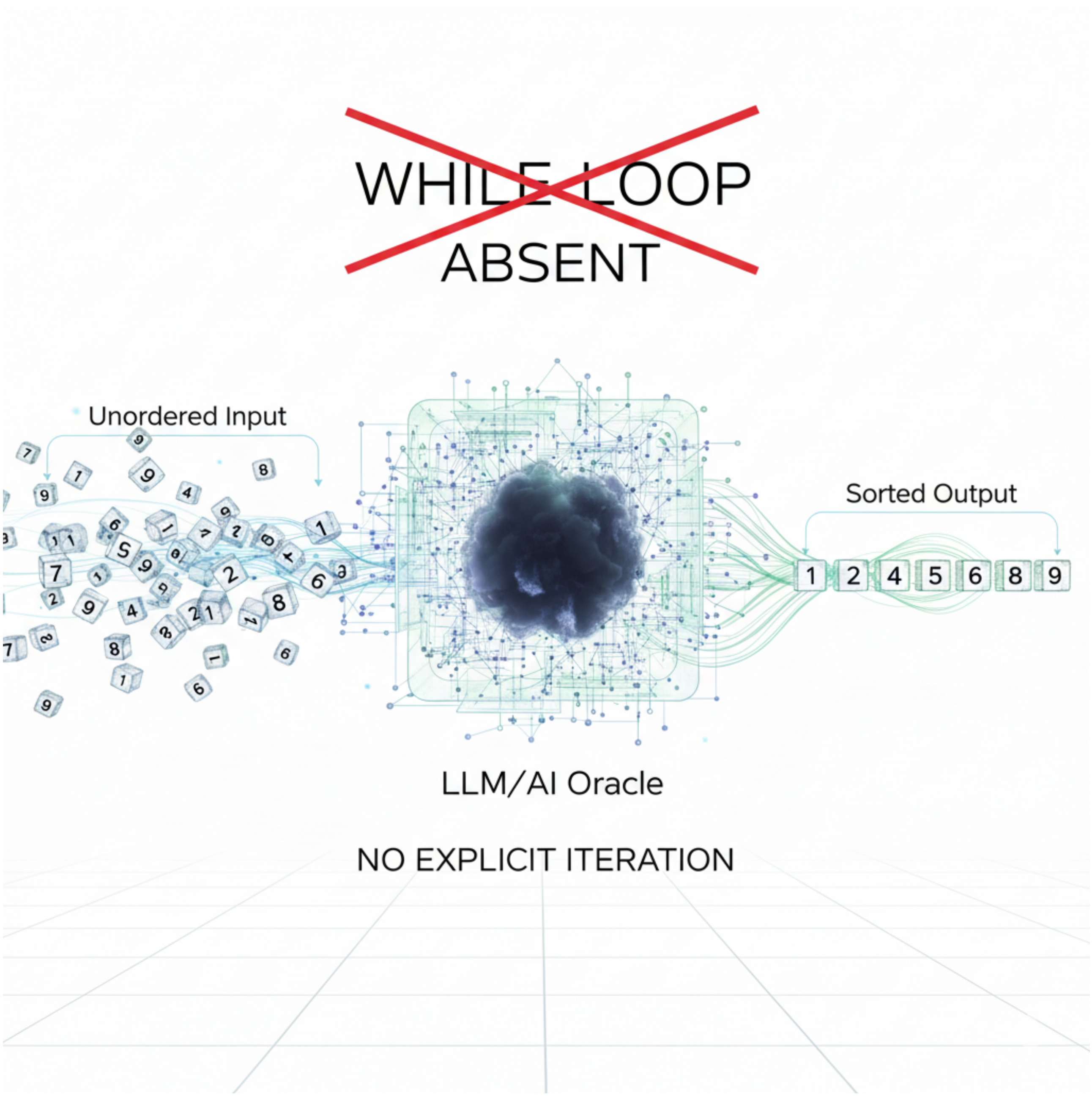

Figure 1: The architecture of GPT-Sort. A large language model (LLM/AI Oracle) acts as the central processing unit, directly transforming an unordered input into a sorted output. Crucially, traditional iterative structures like ‘while loops‘ are explicitly absent, signifying a paradigm shift from explicit algorithmic knowledge to implicit intelligent inference. The internal mechanisms of the AI Oracle remain a black box, abstracting away low-level computational details from the client-side implementation.

3 Experiments

We compared GPT-Sort with standard Python 'list.sort()' (Timsort).

3.1 Performance Metric: Developer Exhaustion

We recruited 50 freshmen from StackOverflow Copy-Paste University and asked them to implement QuickSort and GPT-Sort, respectively.

Table 1: Comparison of Developer Exhaustion Metrics

Algorithm	**Avg. LoC**	**Debug Time (min)**	**Keystrokes**
QuickSort	25	45.2	1200+
GPT-Sort	**1**	**0.1**	**42**

As shown in Table 1, GPT-Sort achieved a $450\times$ improvement in development efficiency.

3.2 Accuracy and "Stochastic Sorting"

It must be admitted that GPT-Sort introduces a feature not present in traditional algorithms: **Probabilistic Sorting**. When $N > 50$, GPT-Sort may exhibit the following interesting phenomena:

1. **Hallucinated Insertion:** Numbers not present in the original array appear in the output (e.g., AI believes '42' is the answer to the universe and forcibly inserts it).
2. **Data Loss:** AI considers certain numbers "unlucky" and deletes them.
3. **Denial of Service:** AI responds, "As a language model, sorting large lists is a waste of compute. Use Python's built-in sort."

We term this phenomenon **Approximate Sorting** and argue that it aligns better with human intuition—after all, humans do not always sort things correctly either [3].

4 Discussion

4.1 Ethical Considerations

Is it ethical to use computing power equivalent to a $10,000 H100 GPU to sort '[3, 1, 2]'? We believe this is the necessary price for liberating human intelligence. Although this generates 1 million times more carbon emissions than Bubble Sort, it saves programmers the time spent reading documentation.

4.2 Limitations

The main bottleneck of GPT-Sort is the **Token Limit**. When array length N exceeds the context window, we recommend **Recursive-RAG-Sort** (Recursive Retrieval-Augmented Generation Sorting): slicing the array, sending parts to different AIs for sorting, and then sending them to another AI for merging. While this pushes complexity to $O(\$ \cdot \log \$)$, it still requires no algorithmic logic.

5 Conclusion

This paper demonstrates the possibility of achieving general-purpose sorting through simple API calls without any algorithmic knowledge. GPT-Sort marks the arrival of the "Algorithm 2.0" era: an era where computational complexity does not disappear but is merely transferred to the cloud and credit card bills [4]. We believe that future computer science curricula will no longer teach "Data Structures and Algorithms," but will be renamed "How to Politely Ask AI to Do Things."

References

[1] Knuth, D. E. (1997). *The Art of Computer Programming*. Addison-Wesley. (Too long, didn't read).

[2] Silver, D., et al. (2017). Mastering the Game of Go without Human Knowledge. *Nature*, 550(7676), 354–359.

[3] Brown, T., et al. (2020). Language Models are Few-Shot Learners. *NeurIPS*. (But we are Zero-Shot Coders).

[4] Visa/Mastercard. (2023). *Monthly API Billing Statements*. Journal of Financial Pain, Vol 1.