Reducing Costs - The Path of Optimization for Chain of Thought Reasoning via Sparse Attention Mechanism

Libo Wang
Nicolaus Copernicus University
Jurija Gagarina 11, 87-100 Toruń, Poland
326360@o365.stud.umk.pl
UCSI University
Taman Connaught, 56000 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur, Malaysia
1002265630@ucsi.university.edu.my

Abstract

In order to address the chain of thought in the large language model inference cost surge, this research proposes to use a sparse attention mechanism that only focuses on a few relevant tokens. The researcher constructed a new attention mechanism and used GiantRabbit trained with custom GPTs as an experimental tool. The experiment tested and compared the reasoning time, correctness score and chain of thought length of this model and o1 Preview in solving the linear algebra test questions of MIT OpenCourseWare. The results show that GiantRabbit's reasoning time and chain of thought length are significantly lower than o1 Preview, confirming the feasibility of the sparse attention mechanism in reducing chain of thought reasoning. Detailed architectural details and experimental process have been uploaded to Github, the link is:https://github.com/brucewang123456789/GeniusTrail.git.

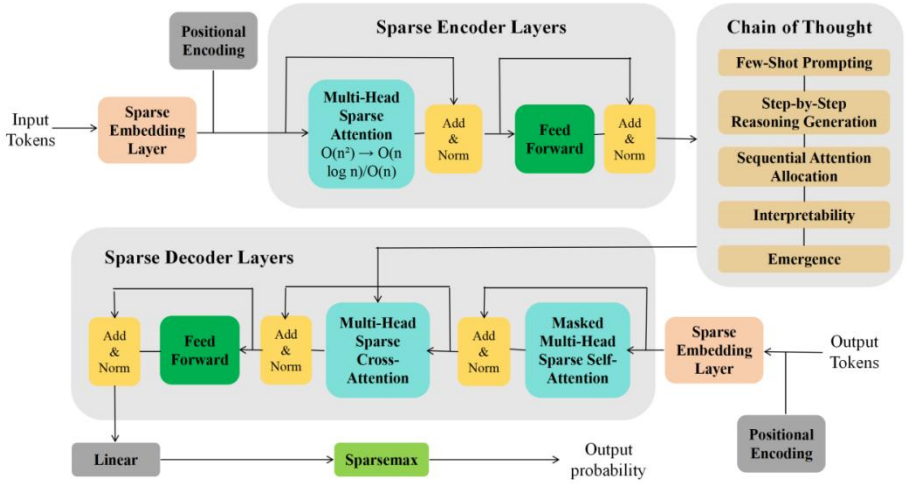

Figure 1 - An Innovative Transformer Architecture that Integrates Sparse Attention Mechanisms with Chain of Thought

1. Introduction

With the rapid development of generative artificial intelligence (GenAI) in academia and industry, the NLP field continues to break through bottlenecks and promote the gradual maturity of large language model (LLM) technology (Hagos et al., 2024). GPT, Llama, Gemini and other series of products have made significant progress in the fluency and semantic coherence of language generation (Creutz, 2024; Mondal et al., 2024; Rai et al., 2024; Zhao et al., 2024).

Chain of thought (CoT) reasoning innovation is the key to promoting the development of LLM (Wei et al., 2022). Early word vector models such as Word2Vec and GloVe can only perform simple semantic reasoning, but lack context understanding (Asudani et al., 2023). The Seq2Seq model improves sequence generation through the encoder-decoder, but the reasoning ability was not significantly improved until the transformer architecture introduced self-attention (Vaswani, 2017). With the continuous iteration of the GPT series, parameter and data expansion have further improved logic and accuracy (Rai et al., 2024).

Multi-task learning and few-shot learning enhance the generalization ability of the model (Bouniot et al., 2022).

2. Related Work

The effect of introducing chain of thought (CoT) to autoregressive modeling increases the strategy of generating a series of intermediate reasoning steps prior to generating content by prompting (Mitra et al., 2024). The development of o1 preview in 2024 heralds the achievement and maintenance of a high level of reasoning accuracy with resource optimization, whichever works on the two-way improvement of reasoning performance and cost efficiency (Zhong et al., 2024). As shown in Figure 2, CoT most notably enhances the inference capabilities of large language models, which can be achieved by humans performing rapid engineering (Wei et al., 2022; Li et al., 2024)

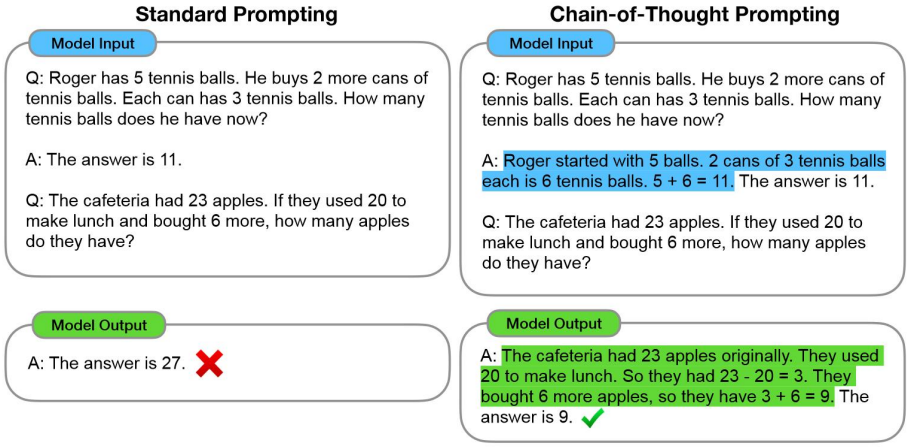

Figure 2 - Chain of thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted (Adapted from Wei et al., 2022).

In fact, CoT has been applied to optimize the accuracy of the model in solving mathematical problems such as linear equations (Feng et al., 2024). This means that the input task can be decomposed into several specific subtasks, which can be solved step by step, and the computation result of each step can be used as a basis for subsequent steps (Zhao et al., 2023; Li et al., 2024). The experimental results of Feng et al. show that the large language model with CoT maintains high accuracy even when facing longer input sequences.

Autoregressive model such as o1 preview radically improve the reasoning ability of the transformer through CoT, with a significant accuracy in generating intermediate reasoning steps for complex tasks (Jin et al., 2024; Mitra et al., 2024; Sprague et al., 2024). It decomposes the various parts of the task to form a clear sequence of steps that systematically guides the model in reasoning (Wu et al., 2024). It is fundamentally different from the previous GPT series: a problem solving process that is based on sequential reasoning steps rather than just memory input-output (Wu et al., 2024). However, the introduction of CoT also leads to a significant spike in reasoning cost due to the significant increase in sequence length in the decoder-only architecture, which is why the use of o1 preview is limited (OpenAI, 2024). The rationale is that when the model generates outputs, it must also incorporate intermediate reasoning steps into the generated sequences, which increases the sequence length from n to n+m (m denotes the number of intermediate reasoning steps). The increase in sequence length has a direct impact on the computational and memory requirements of the model due to the need to process longer inputs and maintain more state information at each step (Nayab et al., 2024; Zhou et al., 2024).

The key to addressing the surge in reasoning cost caused by CoT lies in the computational complexity of the prune self-attention mechanism when dealing with long sequences (Jin et al., 2024). From the perspective of transformer architecture, the self-attention mechanism has $O(n^2)$ computational complexity, where n is the sequence length (Roy et al., 2021). Since each token needs to be computationally related to all other tokens in the sequence, $O(n^2)$ is used for computational complexity (Condevaux & Harispe, 2023). This means that when the sequence length increases, the computation will grow at the rate of quadrature. will grow at the rate of quadratic of n, which generates more tokens (Xiong et al., 2021; Zheng et al., 2024). It causes excessive consumption of computational resources and prolonged reasoning time which makes the model inefficient when dealing with long sequences (Liu et al., 2023; Jin et al., 2024).

Sparse attention is closely related to sparse coding in the human visual cortexthat is embodied in the efficient selection of information for processing. Relevant literature has documented that only a small fraction of neurons are activated in the biological brain that are highly sensitive to specific image features or patterns (Olshausen & Field, 2004; Lee et al., 2006). This sparsity is modeled in the transformer allowing the brain to accurately extract information with low energy consumption and low time cost

(Zheng et al., 2023). In this research, the sparse attention mechanism mimics this principle and is designed to limit the attention of each token to only a small number of other terms that are highly relevant to it. By simulating sparse coding in the visual cortex of the human brain, this mechanism can theoretically reduce computational complexity and maintain model performance in information processing (Olshausen & Field, 2004; Rego et al., 2023).

Inspired by graph theory, this mechanism treats attention computation as a traversal problem in graphs (Zverovich, 2021). In contrast to traditional attention mechanisms, sparse attention mechanisms can be viewed as operating on complete graphs (Roy et al., 2021). Combined with the theoretical foundation, each token in a sequence can be regarded as a node of the graph (Zhang et al., 2023). There is an edge connection between any two nodes, which leads to a dense neighbor matrix and $O(n^2)$ computational complexity. If the number of edges in the graph is drastically reduced to make the neighbor matrix sparse, then the computational complexity is reduced (Buluç et al., 2011; Dai et al., 2020).

3. Sparse Attention Mechanism

Given the advantages of sparse graphs in terms of information transfer and computational efficiency, the sparse attention mechanism has gained theoretical support in graph theory (Li et al., 2023). The sparse attention mechanism aims to reduce the complexity of the attention computation by limiting each attention to only a small number of lexemes that are highly relevant to it (Zaheer et al., 2020; Frantar & Alistarh, 2023). This research designs a model architecture based on sparse attention mechanism, and its flow is shown in Figure 1. It applies the concept of sparse connections in graph theory to model optimization, thus realizing the possibility of reducing computational cost while maintaining model performance.

3.1 Sparse Transformer Architecture

The current sparse attention mechanism is optimized based on the transformer architecture and attempts to solve the thinking chain problem that leads to a surge in reasoning costs. The encoder layer is introduced and the encoder-decoder design is followed. This is not only necessary to match the sparse attention mechanism, but also based on the demand for this architectural principle based on sparse coding. Because sparse coding can effectively select the most relevant tokens when processing long sequences of inputs. It reduces attention to invalid or redundant information, thereby achieving higher efficiency in computing resources (Ren et al., 2021; Lou et al., 2024).

Firstly, tokens are inputted and initially embedded into sparse vectors by sparse embedding layer, which effectively reduces the computational burden of high-dimensional data. Embedding vectors provide positional information in sequence data through positional encoding, allowing the model to maintain sequential relationships in the recognition input. The embedding layer and the positional encoding are summed up as inputs into the encoder. In the sparse encoder layers, multi-head sparse attention is used to capture the correlation between different positions in the input sequence. Next, residual linking and regularization are performed through the add & norm layer to ensure the stability of the model during the deep learning process (Vaswani, 2017). The data then flows through the feed forward, which in this research is set up as a two-layer fully connected network on a location-by-location basis (Geva et al., 2022). The principle is applied to non-linearly transform at each position and maintain a stable output through another residual connection (Nguyen & Salazar, 2019).

After the input has been processed at the coding level, the chain of thought module is responsible for generating intermediate inference steps to split the complex problem into smaller logical units. Its principle aims to improve the model's ability to handle complex problems by means of incremental reasoning, making subsequent reasoning more precise and transparent (Wei et al., 2022). The literature of Wei et al. clearly shows that CoT internally includes short-shot cues, shot cues, step-by-step inference generation, sequential attention allocation, interpretability, and emergence, providing a step-by-step reasoning method. The CoT module generates intermediate reasoning steps step-by-step and presents them in the form of natural language, which decomposes complex problems into a series of simple sub-steps (Khot et al., 2022).

In the sparse decoder layer, masked multi-head sparse self-attention is first used to ensure that the model can only see previous outputs. The data goes through the add & norm layer to maintain the gradient stability. Immediately after that, the data enters multi-head sparse cross attention to introduce CoT reasoning attention operation to the output of coding layer. After the feed forward network and two residual connections, the decoder performs a linear transformation of the output through the Linear layer. It is worth noting that theresearch chose sparsemax instead of softmax as the activation function to generate word probability distributions (Martins & Astudillo, 2016).

In principle, the sparse attention mechanism reduces the attention calculation to linear complexity to alleviates the computational resources caused by the increase in sequence length (Huang et al., 2024). Specifically, sparse attention can be achieved by various methods, such as local attention that focuses only on neighboring tokens; chunk sparsity that divides the sequence into multiple chunks for attention computation; and dynamic sparsity that dynamically selects the objects to focus on based on the contents of the tokens (Condevaux & Harispe, 2023). All these methods aim to retain the critical contextual

information, while reducing computation amount and improving the memory usage efficiency (Zhu et al., 2024). The design of sparse attention mechanism aims to not only reduce the computational load and memory usage, but also maintain the model's ability to capture critical information (Lin et al., 2022).

3.2 Proposed Algorithms

In order to reduce the computational cost of the model in the sparse embedding layer while retaining the main information of the input features, this research designed a sparse embedding algorithm. First, the embedding matrix $E \in R^{V \times D}$, where V represents the size of the vocabulary and D represents the embedding dimension. For the input token index $x \in R^{B \times N}$, where B represents the batch size and N represents the sequence length. The process of sparse embedding query is the following mathematical representation, which aims to convert the input index into the corresponding embedding representation:

$$\text{Embed}(x) = E[x] \in \mathbb{R}^{B \times N \times D}$$

To apply sparsity in the embedding representation to reduce the computational complexity, this research introduces the sparsity mask $M \in \{0,1\}^D$. Its generation is based on the sparsity factor recorded as α to control the embedding dimensions. Specifically, for the sparsity mask M, its elements Mi are defined as the following mathematical representation, where the set S represents the selected active dimensions, determined by the sparsity factor α.

$$M_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

Applying the sparsity mask M to the embedding representation process can be expressed mathematically as follows. ⊙ represents element-level multiplication operations, and broadcast operations in batch and sequence dimensions.

$$\text{SparseEmbed}(x) = \text{Embed}(x) \odot M$$

The core of the current algorithm is based on the principle of sparse attention, which limits each label to only focus on a part of the most relevant tokens to reduce the amount of attention required to be calculated for each token (Guan et al., 2022; Yun et al., 2024). Different from the traditional self-attention mechanism, it does not need to calculate the correlation of each token with all other tokens, reducing the computational complexity from $O(n)^2$ to O(n) or n*log(n) (Kitaev et al., 2020; Treviso et al., 2021).

$$\text{SparseAttention}(Q, K, V, S) = \text{sparsemax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

Where:

- Q, K, V represent query, key and value matrices respectively.
- S represents sparse mode that defines each token should focus on.
- M represents the mask matrix. With large negative values such as -oo at locations where attention is not allowed, it effectively zeroes out some attention weights.
- Sparsemax is an activation function that converts attention scores into sparse probability distributions. Less important tokens are actually assigned the weight of zero.

In the specific design process, it is first determined that the attention connection of the key information that needs to be retained is described by the sparsity mode S, which determines the range of tokens that each token should pay attention to.

For those tokens that are not within the range of sparsity mode, the mask matrix M is designed to set the attention weight to a minimum value such as negative infinity. By effectively “masking” unnecessary attention by assigning negative infinite values to irrelevant locations, it ensures that it does not have an impact on the final result of the computation. This design ensures that the model can focus resources on tokens that are critical to the current task when calculating attention.

It is worth noting that sparsemax was used instead of softmax as the activation function during the development of this algorithm. The main difference between them is the output distribution (Martins & Astudillo, 2016). The sparse output produced by sparsemax will zero out the weights of less relevant tokens, thereby producing a more selective focus of attention (Baan et al., 2019).

Since sparse coding was used before chain of thought, this research needs to modify CoT's operation of sparse representation. The inference path is optimized by introducing a sparse structure, retaining only the most informative intermediate representations for further reasoning. The burden of CoT is reduced by avoiding redundant reasoning.

To perform reasoning effectively under the sparsity principle, CoT requires dynamic inference updates of the sparse embedding representation of the input. The researcher first consider the output of the sparse coding layer as the input representation of CoT, denoted as SparseEmbed(x). Its reasoning process is mathematically expressed as the following series of step-by-step updates:

$$R_t = f_t(\text{SparseEmbed}(x), C_{t-1})$$

Where

- R_t represents the reasoning state at reasoning step t.
- F_t represents the transformation function of each step, which includes a sparse multi-head attention layer and a sparse feed-forward neural network.
- C_{t-1} represents the context information generated by the previous step of reasoning.

The researcher apply a sparsity mask after each reasoning step to retain the most representative features. It sparses the reasoning state R_t:

$$R_t^{sparse} = R_t \odot M_t$$

M_t is a sparsity mask dynamically generated given the importance of intermediate features at each step of inference. It enables the CoT module to focus on the most important features and data fragments during the multi-step reasoning process to avoid redundant reasoning.

Considering that the design of the sparse decoding layer is based on the sparse features after the CoT module output, and the need to process the encoder output and partially generated sequences at the same time. The sparse cross-attention computation is performed first, which aims to combine the output representations from the encoder during decoding. Specifically, assuming that the output of the sparse encoder is H_e, the cross-attention weight is calculated as follows:

$$A_{cross} = \text{sparsemax}\left(\frac{QK^T}{\sqrt{d_k}} + M_{cross}\right)$$

Where

- Q represents the query vector from the decoder.
- K represents the key vector from the encoder,
- Mcross represents the sparsity mask in cross attention to limit the attention computation to only focus on the most important inputs.

In the calculation of sparse self-attention, this part ensures that each token in the decoder can only pay attention to the previously generated tokens, while following the sparsity principle:

$$A_{self} = \text{sparsemax}\left(\frac{QK^T}{\sqrt{d_k}} + M_{self}\right)$$

M_{self} is represented as a causal mask to prevent future tokens in the decoder from receiving attention, and enforces sparsity to limit the attention scope of each token.

The final decoding output can be obtained by weighting and summing the attention of these two parts:

$$H_d = \text{FFN}\ (A_{self}V + A_{cross}H_e) \odot M_d$$

Where

- FFN stands for feedforward neural network.
- M_d is the sparsity mask applied to the final output to ensure that the sparsity features in the output stage are preserved.

Since the Add & Norm layer continues the components of the standard transformer, it does not require major algorithm modifications to adapt to sparsity, so this layer does not have to come up with a new mathematical representation of the algorithm.

4. Experiment

Based on the principle of Figure 2, this research conducts prompt word engineering in GPTs to build and train an autoregressive model with a sparse attention mechanism as the core. It uses the above algorithm to build a new model based on the encoder-decoder principle to ensure that it is distinguished from the operation process of the original transformer. During the experiment, it is more convincing to choose 9 questions of Exam 1 - sample questions of Stanford University's MATH 113: Linear Algebra, Autumn in 2018 as the experimental data. To prove the effectiveness of the reasoning optimization model with the sparse attention mechanism as the core, this experiment compared and tested the three indicators between reasoning time, correctness score and CoT length of Giant Rabbit and o1 Preview. Because as an introductory level of linear algebra, this test question is aimed at a broad group of students, and this test question requires multi-step reasoning operations to answer.

4.1 Experiment Setup

Currently, architectures such as SparseGPT and BigBird, which use the sparse attention mechanism as the core based on transformers, have been used to process long sequence data to reduce computational costs (Zaheer et al., 2020; Frantar & Alistarh, 2023). However, they mainly focus on preserving the general reasoning capabilities of the architecture without making specific optimizations for the chain of thought. Therefore, at this stage, there are no large language models and variants that can be directly adapted to the

experimental requirements of this study. And although the current sparse attention mechanism helps reduce computational burden, it is not deeply optimized and tested in these models for the multi-step reasoning and complexity required by CoT.

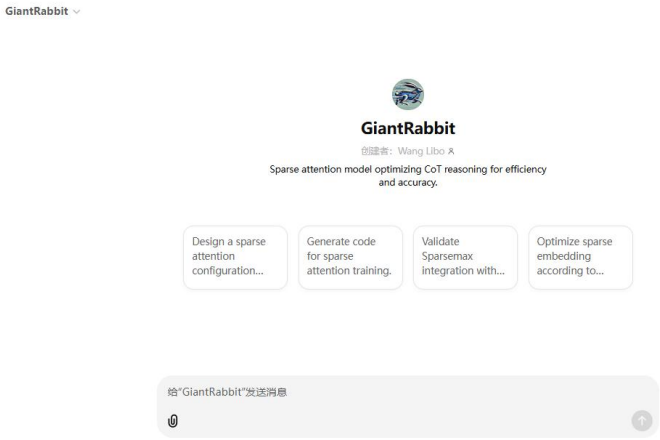

Figure 3 - Experimental tool "GiantRabbit" trained via customer GPTs

Based on this, this research proposes an innovative method to verify the effect of the sparse attention mechanism in reducing CoT inference costs by using ChatGPT to train a new custom GPTs. The architecture used for the experiment is named "GiantRabbit" (Figure 3), which is a model based on the core architecture designed in this research to conduct and test CoT reasoning. During the training process, the sparse attention mechanism of encoder-decoder and the chain of thought were simultaneously built on custom GPTs according to the previously proposed architecture, which is different from the extended decoder-only GPT series. Following methods documented in the literature, thought chain reasoning can be implemented through rapid engineering (Wei et al., 2022). Following methods documented in the literature, chain of thought reasoning can be achieved through prompt engineering (Wei et al., 2022). Sparse attention is able to be practically trained on GPTs through the architectural design and algorithms designed by the researcher. The advantage of using GPTs to train models is to design a model that meets the needs of this research to the greatest extent possible. And the configuration data related to the used model can be found and used for comparison. The design code of the experimental tool "Giant Rabbit" will be uploaded to Github and shared as open source.

The researcher compared the architectural differences between o1 preview and GiantRabbit. It is speculated that o1 preview continues the decoder-only architecture like the GPT series models. It is based on the principle of autoregressive inference, and the generation process relies on the previous output of each step, which becomes the reason for consuming a lot of computing resources when processing long sequence inputs and performing complex inferences (Cai et al., 2022; Roberts, 2024). Although GiantRabbit is a model trained based on GPTs, it uses an encoder-decoder architecture, which allows the model to form a more effective interaction between understanding and generation.

Since the knowledge base provided by OpenAI for the current GPT series models has been updated to December 2023, it is impossible to find official evidence that GPTs are based on GPT4o. This means that GiantRabbit, the experimental tool built using GPTs in this research, may still be considered to use GPT-4-turbo. According to the research results of Liu et al. (2024), the performance of GPTs is obviously due to GPT4, although there is a gap with GPT4o. But the selection of GPTs aims to provide effective verification of the potential of sparse attention mechanisms in multi-step reasoning, especially to test the reduction of required computing resources and time. Compared with SparseGPT and Bigbird's reduction in reasoning costs, experiments using GiantRabbit are more suitable for testing the time required and the accuracy achieved by chain of thought reasoning. In addition, the introduction of the encoder layer can also make up for GiantRabbit's shortcomings in handling complex context understanding because it is based on GPT4-Turbo. Through the encoder layer, the model can achieve a deeper understanding of the input, especially multi-step CoT reasoning tasks. In this way, GiantRabbit can focus on key information in a long sequence in a targeted manner with the support of sparse attention, rather than being scattered throughout all parts of the sequence.

4.2 Dataset

In the experiment, the Exam 1 of MIT OpenCourseWare's 18.06 Linear Algebra (open source) were selected as experimental data. The group it targets is undergraduate students, which means it covers a wide group and is generalizable, and the problem-solving process requires students to use multiple steps. The reasoning required to solve these questions is consistent with the multi-step reasoning requirements in CoT. To ensure the objectivity of the experiment, the questions used for this experiment can be found in Appendix 1. It is noteworthy that the 18.06 linear algebra course of MIT OpenCourseWare is not the GSM8K math problem, because the characteristics of multi-step reasoning can better amplify the

optimization of chain of thought reasoning by the sparse attention mechanism. In addition, MIT OpenCourseWare makes the exam questions publicly available, and they may be used, copied, distributed, translated, and modified for non-commercial educational purposes only.

4.3 Implementation

Since the experiment is designed to be simulated on built GPTs, its essence is still through prompt engineering. This research selected 9 questions of Exam 1. Since o1 Preview cannot directly upload documents, the researcher manually input the test questions into GiantRabbit and o1 Preview respectively. Each sub-question within an exam question can be reasoned about independently and is therefore numbered from 1 to 9 on the same dimension. In order to ensure the rigor of the experiment, no changes will be made to the content of the test questions, nor any additional special processing will be done to the execution. Since the test questions do not have images, Table 1 shows the conversion of questions with matrix operations into computer language that can be recognized by the model.

Table 1 - Adapted Questions of Exam 1

No.	Adapted Questions	
1	Forward elimination changes Ax = b to a row reduced Rx = d: the complete solution is x = [4, 0, 0]ᵀ + c1 * [2, 1, 0]ᵀ + c2 * [5, 0, 1]ᵀ	What is the 3 by 3 reduced row echelon matrix R and what is d?
2		If the process of elimination subtracted 3 times row 1 from row 2 and then 5 times row 1 from row 3, what matrix connects R and d to the original A and b? Use this matrix to fnd A and b.
3	Suppose A is the matrix A = [[0, 1, 2, 2], [0, 3, 8, 7], [0, 0, 4, 2]].	Find all special solutions to Ax=0 and describe in words the whole nullspace of A.
4		Describe the column space of this particular matrix A. “All combinations of the four columns” is not a sufficient answer
5		What is the reduced row echelon form R* = rref(B) when B is the 6 by 8 block matrix B = [A A] [A A] using the same A?
6	Rank and Solutions	Suppose a 3x5 matrix A has rank r=3. Then the equation Ax=b (circle the correct options) (always / sometimes but not always) has (a unique solution / many solutions / no solution).
7		What is the column space of A? Describe the nullspace of A.
8	Suppose that A is the matrix A = [[2, 1], [6, 5], [2, 4]].	Explain in words how knowing all solutions to Ax = b decides if a given vector b is in the column space of A.
9		Is the vector b = [8, 28, 14] in the column space of A?

Regarding output, this research does not provide specific guidance prompt instructions. Both models generate answers in the default way. The researcher only need to record the time required to generate two models through reasoning, the correctness of the answers (compared with the answers provided by MIT OpenCourseWare, memory usage, and the length of the CoT.

5. Result

In this research, GiantRabbit can only be conservatively estimated to use the original GPT4-turbo for training. GPT-4 Turbo is considered an optimized version that uses less computing resources. It has a relatively lower number of parameters and higher inference efficiency than the full version of GPT-4 (Liu et al., 2024; Ramesh et al., 2024). The following is a comparison of the configuration data of GPT-4 Turbo and o1 preview from OpenAI.

Table 2 - Feature comparison of GPT4-Turbo and o1 Preview (Adapted from OpenAI, 2024)

Feature	GiantRabbit	o1 Preview
Context Window	128,000	128,000
Max Output Tokens	4,096	32,768
Knowledge Cutoff	Up to Oct 2023	Up to Dec 2023
Pricing Comparison (Input) / per million tokens	$10.00	$15.00
Pricing Comparison (Output) / per million tokens	$30.00	$60.00
Specialization	General Purpose	STEM Reasoning
Supported Modalities	Natural Language Text, Image	Complex Coding Text

According to the data comparison results provided by DocsBot AI on the benchmark test of GPT4-Turbo and o1 Preview:

Table 3 - Benchmark comparison of GPT4-Turbo and o1 Preview (Adapted from DocsBot AI, 2024)

Benchmark	GiantRabbit	o1 Preview
MMLU	85.4% (5-shot)	92.3% (pass@1)
MMLU-Pro	63.71%	Not Available
MMMU	Not Available	78.2% (pass@1)
HellaSwag	Not Available	Not Available
HumanEval	86.6% (0-shot)	92.40%
Math	64.5% (0-shot)	85.5% (pass@1)

According to the results in Figure 4, o1 Preview and GiantRabbit show significant differences on nine questions.

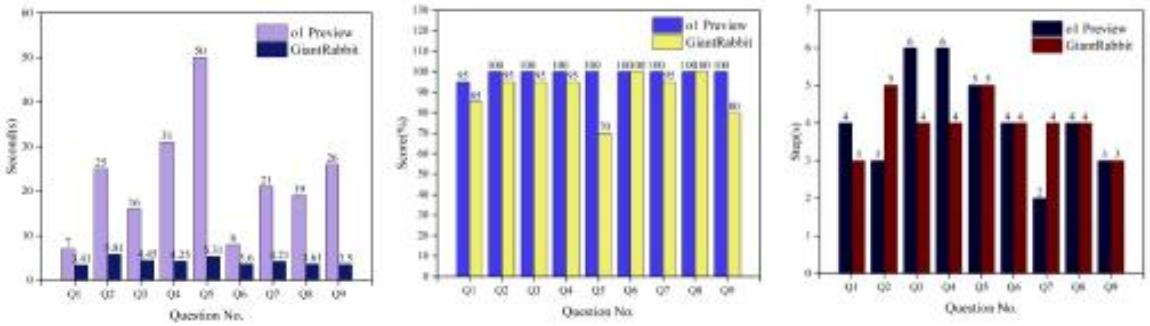

Figure 4 - Comparison between o1 preview and GiantRabbit in reasoning time, correctness score and CoT length

In Q1, the o1 Preview inference time is 7 seconds, the accuracy rate is 95%, and the number of reasoning steps is 4 steps; while the GiantRabbit inference time is 3.41 seconds, the accuracy rate is 85%, and the number of steps is 3 steps. In Q2, the o1 Preview inference time is 25 seconds, the accuracy rate is 100%, and the number of inference steps is 5 steps; the GiantRabbit inference time is 5.81 seconds, the accuracy rate is 95%, and the number of inference steps is 3 steps. In Q3, the o1 Preview inference time is 16 seconds, the accuracy rate is 100%, and the number of reasoning steps is 6 steps; the GiantRabbit inference time is 4.45 seconds, the accuracy rate is 95%, and the number of steps is 4 steps. In Q4, the o1 Preview inference time is 31 seconds, the accuracy rate is 100%, and the number of steps is 6 steps; the GiantRabbit inference time is 4.25 seconds, the accuracy rate is 95%, and the number of steps is 4 steps. In Q5, the o1 Preview inference time is 50 seconds, the accuracy rate is 100%, and the number of inference steps is 5 steps; the GiantRabbit inference time is 5.31 seconds, the accuracy rate is 70%, and the number of steps is the same as 5 steps. In Q6, the o1 Preview inference time is 8 seconds, the accuracy rate is 100%, and the number of steps is 4 steps; while the GiantRabbit inference time is 3.6 seconds, the accuracy rate is 100%, and the number of steps is 2 steps. In Q7, the o1 Preview inference time is 21 seconds, the accuracy rate is 100%, and the number of steps is 4 steps; the GiantRabbit inference time is 4.21 seconds, the accuracy rate is 95%, and the number of steps is 4 steps. In Q8, the o1 Preview inference time is 19 seconds, the accuracy rate is 100%, and the number of steps is 4 steps; the GiantRabbit inference time is 3.61 seconds, the accuracy rate is 100%, and the number of steps is also 4 steps. Finally, in Q9, the o1 Preview inference time was 26 seconds, the accuracy rate was 100%, and the number of steps was 4 steps; while the GiantRabbit inference time was 3.5 seconds, the accuracy rate was 80%, and the number of steps was 3 steps.

In addition to testing the performance of these two models in solving 9 linear algebra reasoning problems respectively, stability is also a factor that is considered and evaluated. Figure 5 uses a line chart to detect fluctuations in problem solving by o1 preview and GiantRabbit during the experiment.

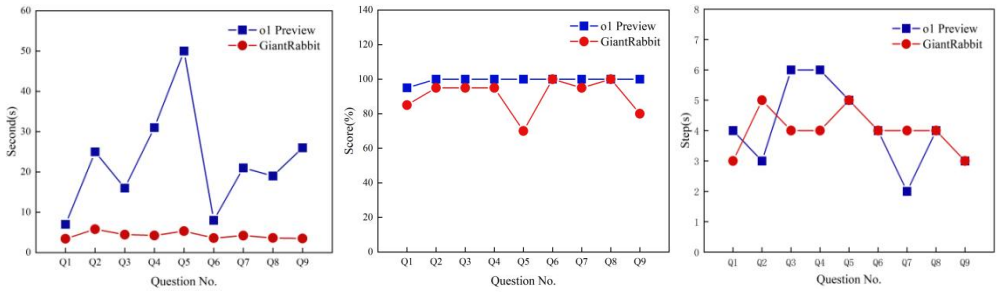

Figure 5 - Comparison between o1 preview and GiantRabbit in performance stability

6. Discussion

From the statistical processing and comparative analysis of the data results, it can be seen that the experimental model GiantRabbit showed convincing performance in comparing reasoning time, correctness score and chain of thought length with o1 Preview. The researcher recorded the experimental process and data results one by one and uploaded them to Github, which proved that the sparse attention mechanism reduces the cost of chain of thought reasoning. Analyzing the two models' answers to 9 questions in Exam 1 of MIT OpenCourseWare-Linear Algebra, it can be seen that GiantRabbit shows that the reasoning time of each question is much lower than o1 Preview, while remaining within a stable trend. During the reasoning process, Giant Rabbit uses fewer steps to complete reasoning and keeps it at 3 to 6 steps, which is better than o1 preview. But in terms of correctness score, o1 Preview is in the leading position and shows good stability capabilities. Although Jutu's accuracy score is lower than that of o1 Preview, and there are fluctuations, the score difference is not large and is acceptable.

7. Limitations &Future Research

Essentially, GiantRabbit is a model specially designed for this experiment and trained through prompt design based on the GPTs function of ChatGPT. However, since the current ChatGPT knowledge base is only updated until December 2023, it is impossible to obtain specific objective information about which model is used by the latest GPTs. Note that the current OpenAI knowledge base has been updated to December 2023. To ensure the objectivity of the experiment, the researcher conservatively considered GiantRabbit to be a model trained based on GPT-4 Turbo. It means that there may be interfering factors in the comparison of data results between GiantRabbit and o1 Preview. Notwithstanding, considering the difficulty of replacing GiantRabbit in this research, it is still the most feasible experimental solution at present. Therefore, future research needs to further explore experiments in training sparse attention models to eliminate interference factors in model configuration, so as to more accurately evaluate the cost reduction of chain of thought reasoning caused by sparse attention.

8. Conclusions

The architecture with the sparse attention mechanism as the core demonstrates outstanding capabilities in the transformer's chain of thought reasoning. After comparing the answers to the MIT OpenCourseWare linear algebra exam using the experimental models GiantRabbit and o1 Preview built using this research, the test results provide evidence in three aspects: reasoning time, correctness score and chain of thought length. Compared with o1 Preview, the experimental results show that GiantRabbit, which introduces the sparse attention mechanism, has significant advantages in reasoning time and chain of thought length. In summary, according to the experimental results, it is acceptable to conclude that using the sparse attention mechanism can reduce the cost of CoT reasoning. Training large predictive models by combining sparse attention with CoT facilitates the development of more transformer variants in the future by imitating human brain cognition.

References

Asudani, D. S., Nagwani, N. K., & Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial intelligence review, 56*(9), 10345-10425.

Baan, J., ter Hoeve, M., van der Wees, M., Schuth, A., & de Rijke, M. (2019). Understanding multi-head attention in abstractive summarization. *arXiv preprint arXiv:1911.03898*.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., ... & Hoefler, T. (2024). Graph of thoughts: Solving elaborate problems with large language models. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 38, No. 16, pp. 17682-17690).

Bouniot, Q., Redko, I., Audigier, R., Loesch, A., & Habrard, A. (2022). Improving few-shot learning through multi-task representation learning theory. *In European Conference on Computer Vision* (pp. 435-452). Cham: Springer Nature Switzerland.

Buluç, A., Gilbert, J., & Shah, V. B. (2011). Implementing sparse matrices for graph algorithms. *Graph Algorithms in the Language of Linear Algebra*, 287-313.

Cai, P. X., Fan, Y. C., & Leu, F. Y. (2022). Compare encoder-decoder, encoder-only, and decoder-only architectures for text generation on low-resource datasets. *In Advances on Broad-Band Wireless Computing, Communication and Applications: Proceedings of the 16th International Conference on Broad-Band Wireless Computing, Communication and Applications* (BWCCA-2021) (pp. 216-225). Springer International Publishing.

Condevaux, C., & Harispe, S. (2023). Lsg attention: Extrapolation of pretrained transformers to long sequences. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 443-454). Cham: Springer Nature Switzerland.

Creutz, M. (2024). Correcting Challenging Finnish Learner Texts With Claude, GPT-3.5 and GPT-4 Large Language Models. *In Workshop on Noisy and User-generated Text* (pp. 1-10). Association for Computational Linguistics (ACL).

Dai, H., Nazi, A., Li, Y., Dai, B., & Schuurmans, D. (2020). Scalable deep generative modeling for sparse graphs. *In International conference on machine learning* (pp. 2302-2312). PMLR.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., & Wang, L. (2024). Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36.

Frantar, E., & Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot. In International Conference on Machine Learning (pp. 10323-10337). PMLR.

Geva, M., Caciularu, A., Wang, K. R., & Goldberg, Y. (2022). Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Guan, Y., Li, Z., Leng, J., Lin, Z., & Guo, M. (2022). Transkimmer: Transformer learns to layer-wise skim. *arXiv preprint arXiv:2205.07324*.

Hagos, D. H., Battle, R., & Rawat, D. B. (2024). Recent advances in generative ai and large language models: Current status, challenges, and perspectives. *IEEE Transactions on Artificial Intelligence*.

Huang, W., Deng, Y., Hui, S., Wu, Y., Zhou, S., & Wang, J. (2024). Sparse self-attention transformer for image inpainting. *Pattern Recognition, 145*, 109897.

Jin, M., Yu, Q., Shu, D., Zhao, H., Hua, W., Meng, Y., ... & Du, M. (2024). The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., & Sabharwal, A. (2022). Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Lee, H., Battle, A., Raina, R., & Ng, A. (2006). Efficient sparse coding algorithms. *Advances in neural information processing systems, 19*.

Li, J., Sun, X., Li, Y., Li, Z., Cheng, H., & Yu, J. X. (2024). Graph intelligence with large language models and prompt learning. *In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 6545-6554).

Li, Y., Li, Z., Wang, P., Li, J., Sun, X., Cheng, H., & Yu, J. X. (2023). A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Li, Z., Liu, H., Zhou, D., & Ma, T. (2024). Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Lin, Q., Zhou, W. J., Wang, Y., Da, Q., Chen, Q. G., & Wang, B. (2022). Sparse attentive memory network for click-through rate prediction with long sequences. *In Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (pp. 3312-3321).

Liu, C. L., Ho, C. T., & Wu, T. C. (2024). Custom GPTs enhancing performance and evidence compared with GPT-3.5, GPT-4, and GPT-4o? A study on the emergency medicine specialist examination. *In Healthcare* (Vol. 12, No. 17, p. 1726). MDPI.

Liu, X., Wang, J., Sun, J., Yuan, X., Dong, G., Di, P., ... & Wang, D. (2023). Prompting frameworks for large language models: A survey. *arXiv preprint arXiv:2311.12785*.

Lou, C., Jia, Z., Zheng, Z., & Tu, K. (2024). Sparser is Faster and Less is More: Efficient Sparse Attention for Long-Range Transformers. *arXiv preprint arXiv:2406.16747*.

Martins, A., & Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. *In International conference on machine learning* (pp. 1614-1623). PMLR.
Mitra, C., Huang, B., Darrell, T., & Herzig, R. (2024). Compositional chain-of-thought prompting for large multimodal models. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14420-14431).
Mondal, H., Komarraju, S., Sathyanath, D., & Muralidharan, S. (2024). Assessing the Capability of Large Language Models in Naturopathy Consultation. *Cureus, 16*(5).
Nayab, S., Rossolini, G., Buttazzo, G., Manes, N., & Giacomelli, F. (2024). Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*.
Nguyen, T. Q., & Salazar, J. (2019). Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*.
Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology, 14*(4), 481-487.
OpenAI. (2024). Introducing OpenAI o1-preview. https://openai.com/index/introducing-openai-o1-preview/
OpenAI. (2024). Learning to Reason with LLMs. Available online: https://openai.com/index/learning-to-reasonwith-llms (accessed on 18 October 2024).
Rai, S., Shapsough, S., & Zualkernan, I. (2024). Measuring Fluency, Coherency and Logicality of GPT-4 Generated EGRA Comprehension Stories. *In 2024 IEEE International Conference on Advanced Learning Technologies* (ICALT) (pp. 201-203). IEEE.
Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., ... & Azam, S. (2024). A review on large Language Models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*.
Ramesh, G., Dou, Y., & Xu, W. (2024). GPT-4 Jailbreaks Itself with Near-Perfect Success Using Self-Explanation. *arXiv preprint arXiv:2405.13077*.
Rego, J., Watkins, Y., Kenyon, G., Kim, E., & Teti, M. (2023). A novel model of primary visual cortex based on biologically plausible sparse coding. *In Applications of Machine Learning 2023* (Vol. 12675, pp. 156-161). SPIE.
Ren, H., Dai, H., Dai, Z., Yang, M., Leskovec, J., Schuurmans, D., & Dai, B. (2021). Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems, 34*, 22470-22482.
Roberts, J. (2024). How Powerful are Decoder-Only Transformer Neural Models?. *In 2024 International Joint Conference on Neural Networks* (IJCNN) (pp. 1-8). IEEE.
Roy, A., Saffar, M., Vaswani, A., & Grangier, D. (2021). Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics, 9*, 53-68.
Sprague, Z., Yin, F., Rodriguez, J. D., Jiang, D., Wadhwa, M., Singhal, P., ... & Durrett, G. (2024). To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.
Treviso, M., Góis, A., Fernandes, P., Fonseca, E., & Martins, A. F. (2021). Predicting attention sparsity in transformers. *arXiv preprint arXiv:2109.12188*.
Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.
Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems, 35*, 24824-24837.
Wu, S., Peng, Z., Du, X., Zheng, T., Liu, M., Wu, J., ... & Liu, J. H. (2024). A Comparative Study on Reasoning Patterns of OpenAI's o1 Model. *arXiv preprint arXiv:2410.13639*.
Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., & Singh, V. (2021). Nyströmformer: A nyström-based algorithm for approximating self-attention. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 16, pp. 14138-14148).
Yun, J., Kim, M., & Kim, Y. (2024). Focus on the core: Efficient attention via pruned token compression for document classification. *arXiv preprint arXiv:2406.01283*.
Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems, 33*, 17283-17297.
Zhang, X., Lv, Z., & Yang, Q. (2023). Adaptive attention for sparse-based long-sequence transformer. *In Findings of the Association for Computational Linguistics: ACL 2023* (pp. 8602-8610).
Zhao, J., Zhang, Z., Gao, L., Zhang, Q., Gui, T., & Huang, X. (2024). Llama beyond english: An empirical study on language capability transfer. *arXiv preprint arXiv:2401.01055*.
Zhao, L., Zhang, L., Wu, Z., Chen, Y., Dai, H., Yu, X., ... & Liu, T. (2023). When brain-inspired ai meets agi. *Meta-Radiology*, 100005.
Zheng, T., Yan, G., Li, H., Zheng, W., Shi, W., Zhang, Y., ... & Wu, D. (2023). A microstructure estimation Transformer inspired by sparse representation for diffusion MRI. *Medical Image Analysis, 86*, 102788.
Zheng, W., Lu, S., Yang, Y., Yin, Z., & Yin, L. (2024). Lightweight transformer image feature extraction network. *PeerJ Computer Science, 10*, e1755.

Zhou, Z., Ning, X., Hong, K., Fu, T., Xu, J., Li, S., ... & Wang, Y. (2024). A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

Zhu, Q., Duan, J., Chen, C., Liu, S., Li, X., Feng, G., ... & Yang, C. (2024). Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *arXiv preprint arXiv:2406.15486*.

Zverovich, V. (2021). *Modern applications of graph theory*. Oxford University Press.

Appendix 1

This research selects and excerpts exam questions from Exam 1 of MIT OpenCourseWare - Linear Algebra, and uses o1 Preview and GiantRabbit to answer them respectively. Since the test questions contain matrix operations, the researcher converted them into computer language that can be recognized by the model without changing the content of the test questions. It is available non-commercially for the experimental purposes of this research under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA) license. The copyright and originality of this test question also belongs to MIT OCW. The following are the original Exam 1 questions.

18.06 Quiz March 1, 2010 Professor Strang

1. Forward elimination changes Ax = b to a row reduced Rx = d: the complete solution is

$$\mathbf{x} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} + \mathbf{c_1} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} + \mathbf{c_2} \begin{bmatrix} 5 \\ 0 \\ 1 \end{bmatrix}$$

(a)Wat is the 3 by 3 reduced row echelon matrix R and what is d?

(b)If the process of elimination subtracted 3 times row 1 from row 2 and then 5 times row 1 from row 3, what matrix connects R and d to the original A and b? Use this matrix to find A and b.

2. Suppose A is the matrix

$$A = \begin{bmatrix} 0 & 1 & 2 & 2 \\ 0 & 3 & 8 & 7 \\ 0 & 0 & 4 & 2 \end{bmatrix}.$$

(a)Find all special solutions to Ax = 0 and describe in words the whole nullspace of A.

(b) Describe the column space of this particular matrix A. "All combinations of the four columns" is not a sufficient answer.

(c) What is the reduced row echelon form R* = rref(B) when B is the 6 by

$$B = \begin{bmatrix} A & A \\ A & A \end{bmatrix}$$

8 block matrixusing the same A?

3. Circle the words that correctly complete the following sentence:

(a) Supose a 3 by 5 matrix A has rank r = 3. Then the equation Ax = b (always / sometimes but not always) has (a unique solution / many solutions / no solution).

(b) What is the column space of A? Describe the nullspace of A.

4. Suppose that A is the matrix

$$A = \begin{bmatrix} 2 & 1 \\ 6 & 5 \\ 2 & 4 \end{bmatrix}.$$

(a) Explain in words how knowing all solutions to Ax = b decides if a given vector b is in the column space of A.

(b) Is the vector b = in the column space of A?