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

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. It verifies the feasibility of sparse attention mechanism for optimizing chain of thought reasoning. Detailed architectural details and experimental process have been uploaded to Github, the link is:https://github.com/brucewang123456789/GeniusTrail.git.

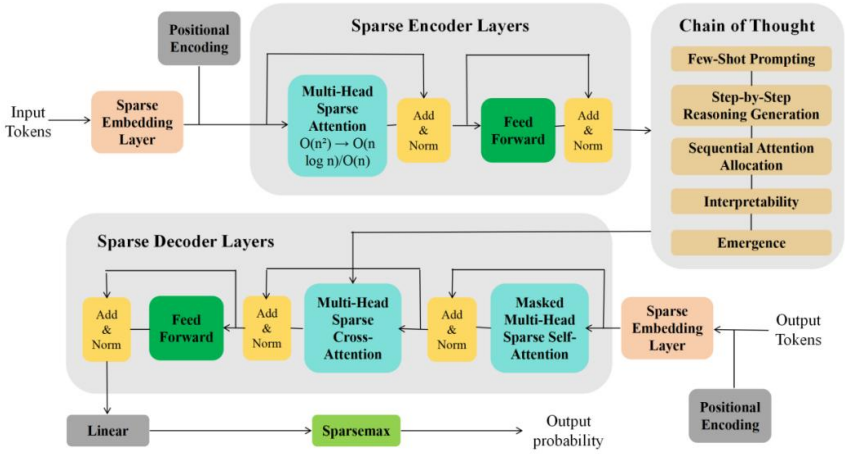

Figure 1 - An Innovative Transformer Architecture that Integrates Sparse Attention Mechanisms with Chain of Thought

1. Introduction

Autoregressive modeling introduces chain of thought (CoT), adding a strategy to generate a series of intermediate reasoning steps before outputting content based on prompts (Mitra et al., 2024). The release of o1 Preview in 2024 has proven that resource optimization can maintain a high level of reasoning accuracy (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 prompt engineering (Wei et al., 2022; Li et al., 2024).

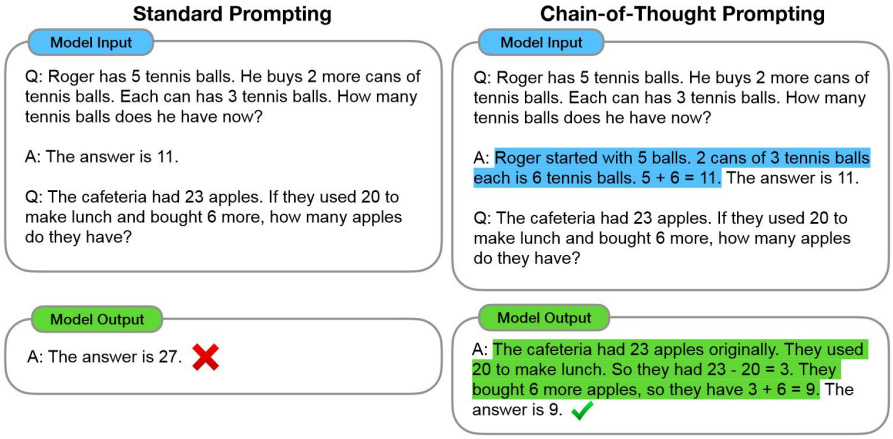

Figure 2 - Chain of thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted (Adapted from Wei et al., 2022).

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). 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).

3. Sparse Attention Mechanism

Sparse attention is closely related to sparse coding in the human visual cortex, a principle that simulates the brain in the transformer to accurately extract information with low energy consumption and low time cost (Zheng et al., 2023). 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). In view of the optimization of information transfer and computational efficiency, the sparse attention mechanism is supported by graph theory (Li et al., 2023). This 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 (Frantar & Alistarh, 2023).

3.1 Sparse Transformer Architecture

Firstly, tokens are inputted and initially embedded into sparse vectors by sparse embedding layer, which effectively reduces the computational burden of high-dimensional data. 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 that is set up as a two -layer fully connected network on a location-by-location basis (Geva et al., 2022). 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 (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 the research chose sparsemax instead of softmax as the activation function to generate word probability distributions (Martins & Astudillo, 2016).

3.2 Proposed Algorithms

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.

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

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

$$Mi = 1 \cdot (i \in \mathrm{S}) + 0 \cdot (i \notin S)$$

Applying the sparsity mask M to the embedding representation process can be expressed mathematically as follows. $\odot$ 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).

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

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.

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:

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

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; $C_{t\text{-}1}$ 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:

$$\mathrm{R}_t^{sparse} = Rt \odot Mt$$

M_t is a sparsity mask dynamically generated given the importance of intermediate features at each step of inference. Specifically, assuming that the output of the sparse encoder is H_e, the cross-attention weight is calculated as follows:

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

Q represents the query vector from the decoder; K represents the key vector from the encoder; M_{cross} 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:

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

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.

Appropriate masking strategy selection improves the computational efficiency and reasoning accuracy of the multi-head sparse attention layer. Dynamic masking strategy generates a sparse matrix M based on the correlation of input tokens, filters relevant tokens and ignores irrelevant parts before QK^T calculation.

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

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

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 figure 2, this research performed prompt engineering in GPTs to train the model "GiantRabbit" with the sparse attention mechanism as the core. Its encoder-decoder architecture is different from the running process of the original GPTs series. In order to verify the effectiveness of this mechanism in optimizing reasoning costs, the researcher compared the performance of GiantRabbit and o1 Preview on three indicators: reasoning time, correctness score and CoT length. The experiment uses 9 sample questions from Stanford University's MATH 113 Linear Algebra (Exam 1) in the fall of 2018 as test data through

MIT OpenCourseWare. This public exam is authorized for non-commercial educational use. Because o1 Preview cannot directly upload documents, the researcher manually entered the test questions into GiantRabbit and o1 Preview respectively. In addition, the researchers took average values and excluded abnormal data caused by API or network delays.

5. Result

According to the results in Figure 3, o1 Preview and GiantRabbit show significant differences on nine questions.

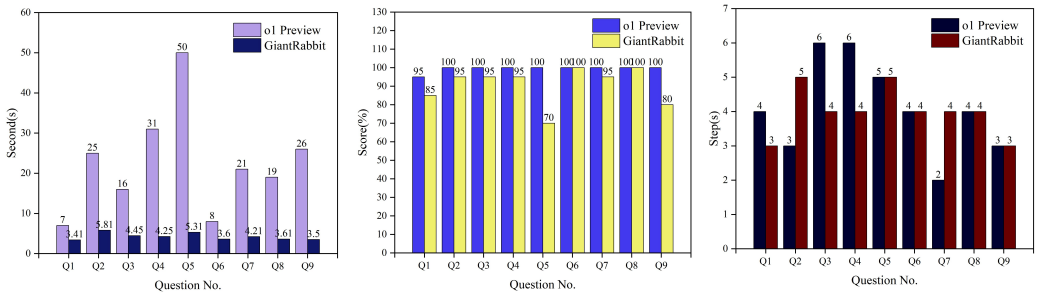

Figure 3 - Comparison between o1 preview and GiantRabbit in reasoning time, correctness score and CoT length

As evidenced by the data, o1 Preview's accuracy remained stable between 95% and 100%, but its reasoning time was significantly longer, ranging from 7 seconds to 50 seconds. In contrast, although GiantRabbit's accuracy fluctuates slightly (70%-100%), it shows a significant efficiency advantage in inference time, taking only 3.5 seconds to 5.8 seconds. In addition, GiantRabbit requires fewer reasoning steps on average, especially when dealing with questions with simpler reasoning steps (such as Q6 and Q9), demonstrating its advantage in simplifying the reasoning process. Figure 4 further shows the fluctuations and performance differences between o1 Preview and GiantRabbit during the problem solving process, clearly reflecting GiantRabbit's superior performance in reasoning time efficiency, while showing its relative trade-off in accuracy in a few situations.

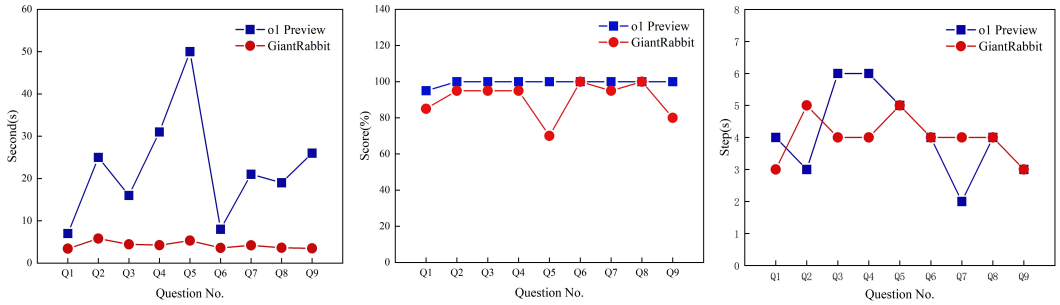

Figure 4 - Comparison between o1 preview and GiantRabbit in performance stability

6. Limitations

Currently, OpenAI updates the knowledge base until December 2023, GiantRabbit is trained on GPT-4Turbo. This causes certain interference to the experimental results because it is different from the o1 Preview configuration. In addition, owing to using commercial language models may cause reasoning time to be affected by API latency and network traffic.

7. Conclusions

This research's encoder-decoder transformer architecture based on the sparse attention mechanism shows excellent performance in chain of thought reasoning. The researcher compared the performance of GiantRabbit and o1 Preview on the MIT OpenCourseWare linear algebra test questions. The results showed that GiantRabbit has a significant advantage in inference time and number of inference steps, although there is a slight trade-off in accuracy. Experimental results prove that the sparse attention mechanism effectively reduces the cost of chain of thought reasoning, providing experience for training LLMs combined with sparse attention in the future.

References

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.

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*.

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*.

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*.

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.

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).

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).

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.

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.

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*.

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.

Zhong, T., Liu, Z., Pan, Y., Zhang, Y., Zhou, Y., Liang, S., ... & Liu, T. (2024). Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*.

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.

Section A: 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?

Section B: Conversion of Computer Language

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?