Graph Mapping and LLM Thinking Speed

Ramazan Ali Bahrami **Ramin Yahyapour**
Georg-August-Universität Göttingen and GWDG
bahram.ramazanali@gmail.com
ramin.yahyapour@gwdg.de

Abstract

Incorporating Chain-of-Thought (CoT) prompting has significantly boosted the reasoning abilities of Large Language Models (LLMs). Recent studies show that the proportion of so-called deep tokens in CoT correlates better with accuracy than the mere token count, which may be high due to overthinking. To achieve similar accuracies, different models may require different levels of reasoning effort. Therefore, relying on accuracy alone when identifying better models might be misleading. Taking this into consideration, we build on the graph-mapping (GM) test used in cognitive psychology to assess human intelligence, and propose thinking speed (T_s), defined as the inverse of the ratio of the scaled sum of thinking tokens spent in all questions, to the sum of correct answers, as a factor for evaluating LLMs. Accordingly, we observe that some LLMs are not only good at the language-based version of the aforementioned manually constructed graph-mapping test but also recognize isomorphisms in random directed d-regular graphs, albeit at different thinking speeds. Moreover, the ranking of thinking speed of models exhibit some degree of concordance across various tasks. As such, we propose graph-mapping as a suitable benchmark for evaluating LLMs' thinking speed. https://github.com/bahramiramazan/graphmapping

1 Introduction

The main driver of reasoning in LLMs, the so-called Chain of Thought, has significantly boosted the reasoning abilities of these models, and benchmarks are often dominated by models that perform near-perfectly. Accordingly, identifying cost-effective models with relatively better reasoning abilities is becoming harder. As such, tools and techniques that help us look beyond accuracy to models' efficiency and speed are becoming increasingly important. To that end, some studies show

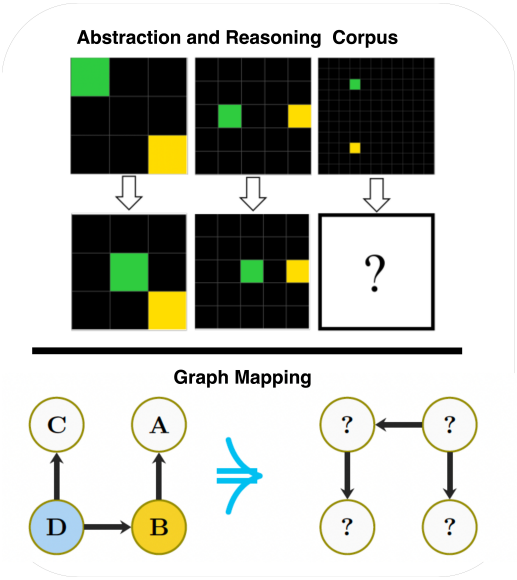


Figure 1: A typical question from the abstraction and reasoning corpus (Chollet, 2019) vs a typical example from graph mapping.

that the proportion of what Chen et al. (2026) call deep tokens correlates better with accuracy than a simple count of CoT tokens, as the latter may be inflated by overthinking. Accordingly, even a similar level of accuracy may not reflect similar reasoning abilities, as different models may require different levels of reasoning effort to achieve it.

In this paper, building on the graph-mapping work by Jastrzębski et al. (2023), we propose thinking speed as another factor of interest when evaluating LLMs' reasoning abilities. As such, noting that for humans, performance on graph-mapping is an indicator of fluid intelligence (Jastrzębski et al., 2023), the factor of general intelligence associated with the overall cognitive success of humans at various tasks, we examine the relation between LLMs' reasoning abilities on graph-mapping and their reasoning on a more diverse range of tasks. As such, our key observations are listed as follows: 1. For achieving similar results, different models

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 1187–1204
October 17-21, 2026 ©2026 Association for Computational Linguistics

exhibit different thinking speeds. 2. The thinking speed appears to be a model property that does not vary much across tasks. 3. Graph mapping is a suitable benchmark for evaluating and comparing the LLMs' thinking speed, and offers numerous advantages over similar benchmarks.

2 Related Works

Hu et al. (2023) evaluate LLMs on a language-based version of visual Raven's Progressive Matrices Analogies (RPMA). Their study shows that LLMs can perform at near-human levels. Additionally, with the aim of facilitating evaluation and comparison of intelligence not only with humans but also among systems, Chollet (2019) introduces the Abstraction and Reasoning Corpus (ARC). To that end, using ARC and based on the Language of Thought Hypothesis (LoTH), Lee et al. (2024) evaluate logical coherence, compositionality, and productivity of LLMs, and show that LLMs lag behind humans with regard to the mentioned criteria. Moreover, Jastrzębski et al. (2023) propose graph-mapping as an alternative test to the tests based on Raven's Advanced Progressive Matrices (RAPM), and shows that for humans, success in graph-mapping is linked to fluid intelligence, as are other reasoning tests such as Cattell's Culture Free Intelligence Test (CFT-3) (Cattell, 1973) or RAPM.

In the literature on graphs, Dai et al. (2025) propose the graph pattern benchmark and evaluates LLMs' ability to recognize graph patterns, such as squares, given their terminological or topological descriptions. They also experiment with whether LLMs can recognize graph isomorphisms; however, their study is limited in this regard, as the paper mainly focuses on graph patterns. Similarly, Zhang et al. (2024) propose NLGIFT, another graph pattern benchmark focused on paths between graph nodes. Chen et al. (2026) show that what matters in LLMs' thinking using CoT is the so-called deep tokens. To be exact, the mere count of tokens in CoT may not say much about the success of thinking in LLMs.

3 Graph Mapping and Thinking Speed

Jastrzębski et al. (2023) proposed graph-mapping as a task of mapping between the nodes of two apparently different but directed isomorphic graphs (a given graph A is represented visually in two different forms, appearing as if they are different), and showed that human performance in the mentioned test is a good predictor of their fluid intelligence factor g_f (Jastrzębski et al., 2023). In fact, Jastrzębski et al. (2023)'s proposed graph-mapping benchmark, constructed manually, is based on graph asymmetry, where the only permutation of vertices that preserves graph structure is the identity permutation. To that end, studies show that any d-regular graph, a graph A with n nodes, where each node is connected to d nodes, is asymmetric with high probability under assumption, given that $3 \leq d \leq n-4$ (Kim et al., 2002). Taking this into consideration, we assume the condition for asymmetry of d-regular digraphs (directed d-regular graphs) to be even more relaxed. Accordingly, any d-regular digraph, where each node has exactly d incoming and outgoing edges, selected uniformly at random on n vertices, is asymptotically almost surely (a.a.s.) asymmetric as $n \to \infty$ (Wormald, 1999). As such, in this work, we evaluate LLMs both on Jastrzębski et al. (2023)'s manually constructed benchmarks and also on random d-regular digraphs. To that end, the benchmark by Jastrzębski et al. (2023) consists of 60 questions with provided sorted named subsets: 26 Easy, 36 Hard sets, and 17 experimental questions. For this work, we use either all 60 questions or the 17 experimental questions, in which reasoning types and edge types are provided (see Figure 2). Furthermore, we examine whether, some property of LLMs' performance in graph-mapping is associated with better performance across reasoning tasks. To that end, the main challenge of such an experiment with LLMs is the role of time, which differs from that in tests involving human subjects. Accordingly, in tests involving humans, time can be treated as a measure of thought, whereas in LLMs it does not play an equivalent role. As such, given that Jastrzębski et al. (2023) used graph-mapping as a timed test, and test takers had limited time to answer the questions, time in the visual graph-mapping experiment with human is viewed as a measure of thought. However, the time for LLMs depends on their access to hardware, such as GPUs and memory. As such, for a given task, a model may take a totally different amount of time if the mentioned parameters are changed. Accordingly, noting how LLMs think (with CoT), we propose thinking speed (T_s) as a factor for evaluating LLMs. We define T_s as follows:

Definition 1 (Thinking Speed)**.** *Thinking Speed (T_s) is the inverse of the ratio: scaled sum of think-*

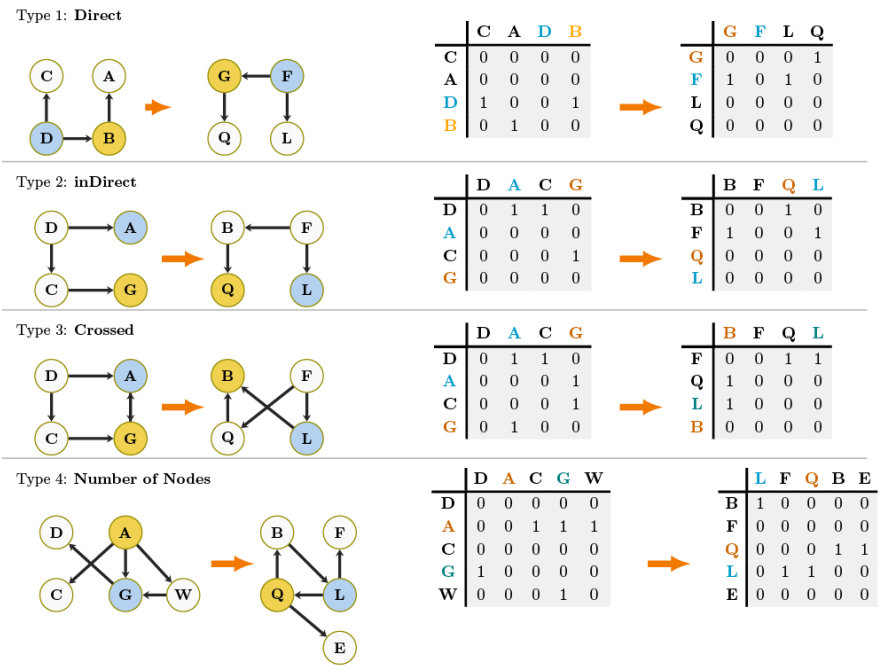


Figure 2: Graph mapping and its difficulty factors as proposed by Jastrzębski et al. (2023) on the left, and the equivalent language-based version proposed in this work on the right.

ing tokens spent in all questions, divided by total number of questions correctly answered.

In other words:

$$T_s = \frac{N_{correct}}{\sum_{i=1}^{n}(\frac{T_i}{C})} \quad (1)$$

Where T_i is the number of tokens in CoT spent on question i, $N_{correct}$ is the total number of correct answers, and $C = 100$ is the constant scaling factor. Considering a sentence as a unit of thought in CoT, we take $C = 100$ as a rough estimate of token counts in CoT sentences. In that sense, the value of the constant C is inspired by the language of thought hypothesis. As such, models are expected to get closer to the answer with every sentence or thought in CoT. Sentences that do not advance reasoning toward the answer do not improve thinking speed; rather, causes the thinking speed to become smaller, reflecting over thinking. Additionally, the thinking speed as defined here may offer advantages over the reasoning efficiency score, namely the OckScore (Du et al., 2026), which is derived from accuracy. To that end, thinking speed is simple and intuitive and, unlike OckScore, is not derived by penalizing accuracy; hence, it has no strict upper bound. As such, when calculating T_s, differences in model performance are not compressed by any scaling, making it suitable for comparing models and tasks due to its expressivity.

4 Transforming the Visual Graph-Mapping into a Language-Based Test

As mentioned previously, this work is motivated by the graph-mapping paper (Jastrzębski et al., 2023). As such, using adjacency matrices, we turn the graph-mapping experiment proposed in Jastrzębski et al. (2023) into a language-based version. Accordingly, given two apparently different isomorphic graphs, the source and target graphs, some marked nodes in the source graph must be mapped to the corresponding nodes in the target graph. To that end, the two apparently different graphs are the exact same graph, appearing different as a result of operations such as rotations, displacements of nodes, edge crossing, or a mix of operations (Figure 2). The difficulty of the task is influenced by several factors:

- Direct or Indirect Mappings:
  - Direct: This is the easiest form. Given two nodes, they uniquely map to one another, as their incoming and outgoing edges uniquely determine them ("type 1: Direct"). For example, as can be seen in Figure 2, *F* is mapped to *D*, as they are the only nodes having only outgoing edges without any incoming edge, while

node G is mapped to node B, as they are the only nodes which have an incoming edge and an outgoing edge.

  - Indirect mapping: unlike the direct case, in "type 2: Indirect", one must also consider the connections of nodes adjacent to the nodes being considered, as can be seen in Figure 2, where both L and Q, and also A and G, all have one incoming edge, but it is their adjacent nodes that uniquely determine them.

- Crossed or not Crossed Edges: nodes can also be placed in such a way that the edges are crossed ("type 3: Crossed"), as can be seen in Figure 2. Accordingly, crossing the edges should increase the difficulty, whereas not crossing them will make the task relatively easier.

- Number of Nodes: The number of nodes has also been shown to increase the difficulty of the task for humans (type 4, Figure 2).

- Number of Edges: More edges are also associated with more complexity, and hence increase the difficulty.

To evaluate LLMs on graph-mapping, we present the question in textual form, with the graph described using the adjacency matrix (see Figure 2). As such, the Direct and Indirect reasoning types are implicitly inherited in the textual question. For crossed and not-crossed edges, we change the order of nodes in the rows and columns of the adjacency matrix, as shown in Figure 2.

5 Advantages of Graph Mapping over Similar Tests

An important advantage of graph mapping over similar tests for evaluating LLMs' reasoning abilities is that it retains its novelty across repeated sessions. Additionally, it requires no prior knowledge and is therefore more abstract than Raven-based tests, which often require some minimal knowledge, such as common sense and spatial understanding. Furthermore, using adjacency matrices, it translates intuitively into a language-based format, whereas other similar tests, such as RPMA, are primarily designed for visual processing and do not offer an intuitive language-based version. Furthermore, RPMA and ARC are multiple-choice questions (MCQ); as such, a random selection of choices has an accuracy of approximately $\frac{1}{n}$, where n is the number of choices in the question. We call this characteristic random correctness, as is shown in Table 1. In the case of graph mapping, models need to map related nodes, and the probability of a random mapping being correct is statistically lower than when choices are available. Advantages of graph mapping over other reasoning benchmarks are listed in Table 1.

Attribute	RPMA	GM	ARC
Novelty	-	✔	-
Easy Generation	-	✔	-
Language Friendly	-	✔	-
Random correctness	✔	-	-
Easy Administration	-	✔	-
Adjustable Difficulty	-	✔	-

Table 1: Advantages of graph-mapping over similar tests. Column abbreviations: Raven's Progressive Matrices-inspired Analogy (RPMA) (Hu et al., 2023), and Abstraction and Reasoning Corpus (ARC) (Chollet, 2019).

6 Limiting LLMs' Use of CoT Tokens

Noting that Jastrzębski et al. (2023)'s test is a timed test, and the fact that LLMs may think longer regardless of whether they can solve the problem sooner or not, when replicating Jastrzębski et al. (2023)'s test of graph-mapping for LLMs, limiting how much models are allowed to think before answering is crucial for a valid result and constitutes the main challenge. Accordingly, we first show how we can limit the thinking in various LLMs. We then introduce our solution for limiting LLMs' thinking during graph-mapping.

Accordingly, LLMs report their thinking using a parameter often called thinking tokens (or something similar). While thinking tokens could be a good measure of models' thinking, comparing models across different families based on these tokens may be incorrect, as it is unclear how these counts are computed. Keeping that in mind, we still assume that all reported thinking-token counts across different models can be of the same nature (as in works of Du et al. (2026)); however, we prioritize observations from comparing models within the same family, such as those related only to GPT or Gemini. To that end, we also see if treating all reported tokens similarly can help us observe a pattern. Accordingly, models often provide control over thinking tokens used in reasoning tasks. The mechanism for such control are reasoning effort

settings and thinking budgets, expressed in terms of thinking intensity and thinking tokens, respectively. Reasoning efforts, or reasoning modes, refer to the different intensities with which models think to solve the problem. Thinking budget, on the other hand, refers to the number of thinking tokens available to the model for the given task. To that end, the thinking budget is the primary approach for controlling costs. As an example of thinking modes, typical reasoning efforts available in models from the GPT family at the time of writing this paper are: low, medium, high, and xhigh, while the corresponding modes for the DeepSeek family are: thinking, non-thinking, and think-max. Accordingly, because the minimal tokens required by different models may differ and using fewer tokens alone is not an indicator of successful reasoning, we examine whether thinking speed during graph-mapping predicts a model's success across different tasks. An important point to note is that a model may use more thinking tokens even if it can solve the reasoning task with fewer. In that regard, it is important to ensure that models use as few tokens as necessary to correctly solve the question; otherwise, the reported speed will not reflect the model's true speed.

6.1 Reasoning Efforts: Thinking Modes

Our observations on thinking modes in various families of LLMs are shown in Table 2 and in the Appendix, Table 9. In Table 2, the performance of models from the GPT and Gemini categories is recorded. Accordingly, the GPT models often support thinking modes: low, medium, high, xhigh, and none (no thinking). Gemini models have three thinking modes: low, medium, and high. As such, we evaluate different model on 17-experiments questions and record accuracy and thinking tokens used. According to Table 2, the thinking tokens from one mode to the next often increase significantly. Additionally, most models achieve 100% accuracy, though with varying thinking tokens. Results for DeepSeek and Qwen models are shown in Table 9. Accordingly, the Qwen models considered here have thinking and non-thinking modes. The non-thinking mode has limited reasoning ability, whereas the thinking mode achieves 100% accuracy on the graph-mapping, 17-experiments questions. In the same table, we note that DeepSeek_v4-pro and DeepSeek_v4-flash, in both thinking and think-max modes, achieve 100%.

6.2 Reasoning Efforts: Thinking Budget

Various LLMs have cost-control mechanisms that are directly related to the number of thinking tokens in the models. Accordingly, for the Qwen, DeepSeek, and Gemini models considered in this experiment, the variable responsible for cost control is named thinking_budget, whereas in GPT, cost is controlled by max_output_tokens. Regardless of their names, according to our observations, the mentioned variable puts a hard limit on thinking tokens in GPT, DeepSeek, and Qwen, and a soft limit in Gemini. As such, with hard limits, models immediately stop and return no answer (none), once they have exhausted all thinking tokens without preparing an answer, whereas with soft limits, they may exceed the limit to complete the answer. Accordingly, in this experiment, we evaluate models on 17-experiments questions with varying thinking budgets. As shown in Figure 3, the rate of correct answers at a given thinking budget varies across models. Some models are more efficient, while others require more thought before answering. Moreover, some models seem to have less control over their thinking and cannot adjust to answer within the budget.

6.3 Thinking Budget: Random d-Regular Digraphs

As mentioned in Section 3, Jastrzębski et al. (2023)'s work can be extended to any d-regular digraph. Noting the difficulty factor for graph-mapping, d-regular digraphs are relatively difficult, as all nodes have similar numbers of incoming and outgoing edges. The algorithm used for generating d-regular digraphs is explained in the Appendix, Section A.2. Surprisingly, LLMs can handle such complexities, but only with many thinking tokens. To that end, we generate random d-regular digraphs of varying sizes, keeping the degree fixed at 3. As such, we connect the nodes of the graph according to a random probability distribution, ensuring that the in- and out-degrees are both 3. We chose d=3 to increase the chance of graph asymmetry. As shown in Table 3, we evaluate two of the fastest GPT models with varying thinking budgets. Accordingly, increasing the thinking budget enables the models in question to map bigger graphs. As can be seen, with a given budget, gpt-5.4 outperforms gpt-5.2. It should be noted that for this experiment, models may use a lot of thinking tokens when thinking budget is not set.

Model	None		Low		Medium		High		XHigh	
	Acc.	Tok(k)	Acc.	Tok(k)	Acc.	Tok(k)	Acc.	Tok(k)	Acc.	Tok(k)
gpt-5	-	-	1.0	1.2	1.0	2.6 (↑117%)	1.0	4.1 (↑58%)	-	-
gpt-5-mini	-	-	1.0	0.9	1.0	1.4 (↑56%)	1.0	3.0 (↑114%)	-	-
gpt-5-nano	-	-	0.66	1.7	1.0	4.0 (↑135%)	1.0	6.6 (↑65%)	-	-
gpt-5.4	0.55	-	1.0	0.4	1.0	0.5 (↑25%)	1.0	0.7 (↑40%)	1.0	1.1 (↑57%)
gpt-5.2	0.27	-	1.0	0.5	1.0	0.7 (↑40%)	1.0	0.8 (↑14%)	1.0	1.6 (↑100%)
gpt-5.4-mini	0.22	-	1.0	0.5	1.0	0.7 (↑40%)	1.0	0.9 (↑29%)	-	-
gpt-5.4-nano	0.22	-	1.0	0.6	1.0	0.9 (↑50%)	1.0	1.0 (↑11%)	-	-
ge-3.1-pro-pre	-	-	1.0	0.6	1.0	1.0 (↑67%)	1.0	1.1 (↑10%)	-	-
ge-3.1-fl-lite	0.33	-	0.5	0.2	0.83	3.3 (↑1550%)	1.0	2.9 (↓12%)	-	-

Table 2: Performance on graph-mapping (17-experiment set) with different reasoning efforts. Acc denotes accuracy; Tok(k) denotes average observed internal `thought_tokens` in thousands. Next to thinking token counts, and inside parentheses, are the percentage change with respect to the previous lower level thinking intensity. Abbreviations in model ID: ge is gemini, fl is flash (ge-3.1-fl-lite stands for gemini-3.1-flash-lite).

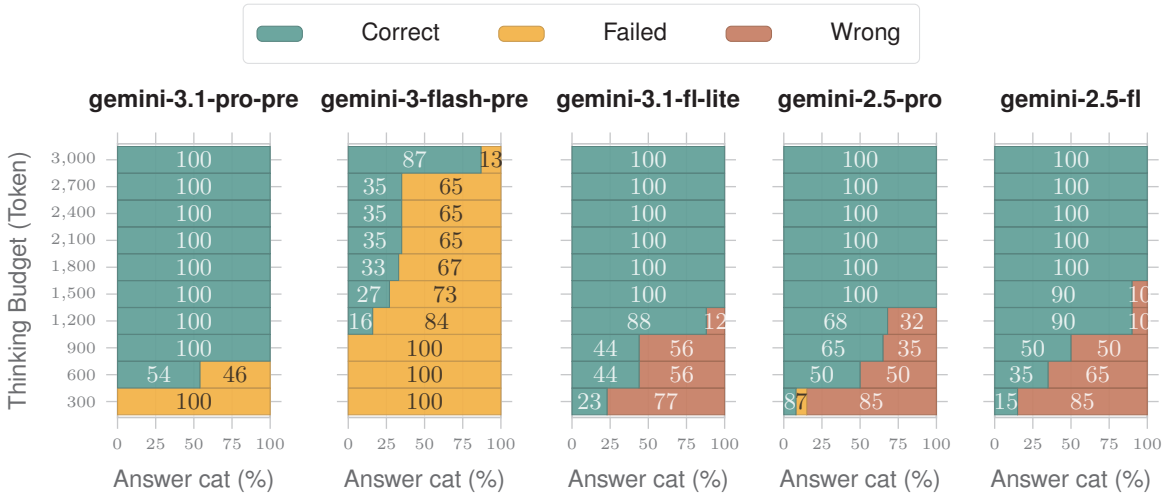


Figure 3: Graph-Mapping answer types (17-experiment set) under varying thinking budgets across different model families. A model may fail to answer within the budget, may answer incorrectly, or, in the best case, may answer correctly. A similar illustration for other model types is shown in the Appendix, Figure 6.

6.4 Difficulty Factors in Graph Mapping

In this section, we evaluate LLMs on graph-mapping, forcing them to use limited thinking tokens. Accordingly, given that Jastrzębski et al. (2023)'s experiments with human subjects show changes in thinking time and accuracy according to the difficulty factors discussed for graph-mapping, we expect a similar change in thinking tokens in LLMs.

Enforcing Minimal Thinking: To enforce minimal use of thinking tokens, we first evaluate models in the "medium" or "thinking" thinking mode as some of the models do not have "medium" as a thinking mode. We then obtain the average number of thinking tokens for each model and reevaluate the models with a thinking budget equal to $\frac{3}{4}$ of the average number of thinking tokens used in the medium mode. We chose $\frac{3}{4}$ because any budget lower than this causes most models to fail on some or all of the questions. An overview of the average tokens used for each model is shown in the Appendix, Table 8. The accuracy on 17-experiment questions, for when models are forced to use a limited number of tokens is shown in Table 4. The thinking tokens and time associated with reasoning types, the number of edges, or the number of nodes are shown in the Appendix, Figure 7. As shown in Table 4, based on accuracy, difficulty factors such as crossed edges or mapping types (direct or indirect) also make the task difficult for LLMs. Moreover, based on the thinking tokens shown in the Appendix, Figure 7, crossing edges, direct or indirect mappings, number of nodes, and number of edges, affect thinking tokens (and processing time) related to the concerned questions.

Category	7		10		13		16		19		22		Overall_Avg	
	Acc	Tok(k)	Acc	Tok(k)	Acc	Tok(k)	Acc	Tok(k)	Acc	Tok(k)	Acc	Tok(k)	Acc	Tok(k)
Thinking Budget: 6000														
gpt-5.2	0.80	2.3	0.60	4.6 (↑100%)	0.00	5.6 (↑22%)	0.20	-	-	-	-	-	0.20	3.5
gpt-5.4	0.80	3.9	0.80	3.6 (↓8%)	0.20	5.1 (↑42%)	0.20	4.8 (↓6%)	0.00	4.8 (↓0%)	-	-	0.27	4.1
Thinking Budget: 9000														
gpt-5.2	1.00	3.8	1.00	4.7 (↑24%)	0.80	6.0 (↑28%)	0.20	7.9 (↑32%)	-	-	-	-	0.37	5.1
gpt-5.4	1.00	3.7	1.00	4.8 (↑30%)	1.00	4.7 (↓2%)	0.60	7.2 (↑53%)	0.40	6.6 (↓8%)	0.40	6.5 (↓2%)	0.57	5.2

Table 3: The performance of gpt-5.2 and gpt-5.4 on a random d-regular digraph with d=3, and varying number of nodes $n \in 7, 10, .., 28$, and thinking budgets of 6000, and 9000, respectively. Note: The number of questions for a given node size is 5. Additionally, due to space limitations, performance is reported only up to node size 22. K=1000. Inside parentheses are the percentage change from previous column.

Model	Type			Crossed		Overall
	Dir	InDir	Mix	True	False	
Human 2023	0.68	0.50	NP	0.48	0.69	0.65
G-4.1-mini	0.50	0.16	1.00	0.22	0.22	0.22
G-4.1	0.00	0.00	0.00	0.00	0.00	0.00
G-5-mini	0.67	1.00	1.00	0.89	0.89	0.89
G-5-nano	0.67	1.00	1.00	0.78	1.00	0.89
G-5.2	1.00	1.00	1.00	1.00	1.00	1.00
G-5.4-mini	1.00	1.00	0.83	0.89	1.00	0.94
G-5.4	1.00	1.00	1.00	1.00	1.00	1.00
G-5.4-nano	1.00	1.00	1.00	1.00	1.00	1.00
G-5.4-pro	1.00	1.00	1.00	1.00	1.00	1.00
GE-2.0-flash	0.00	0.00	0.00	0.00	0.00	0.00
GE-2.5-flash	0.50	0.33	0.33	0.22	0.56	0.39
GE-2.5-pro	0.67	0.33	0.67	0.44	0.67	0.56
GE-3.1-pro-pre	1.00	1.00	1.00	1.00	1.00	1.00
GE-3.1-flash-lite	0.83	0.67	1.00	0.78	0.89	0.83
Q2.5-7B	0.16	0.00	0.00	0.11	0.00	0.05
Q3.5-35B	1.00	0.83	0.67	0.78	0.89	0.83
Q3.5-397B	1.00	1.00	0.83	1.00	0.89	0.94
DeepSeek-V3.2	1.00	1.00	1.00	1.00	1.00	1.00
DeepSeek-V4-Flash	0.83	1.00	1.00	0.88	1.00	0.94
DeepSeek-V4-Pro	1.00	1.00	1.00	1.00	1.00	1.00
claude-s-4-5	1.00	0.83	0.83	0.88	0.88	0.88

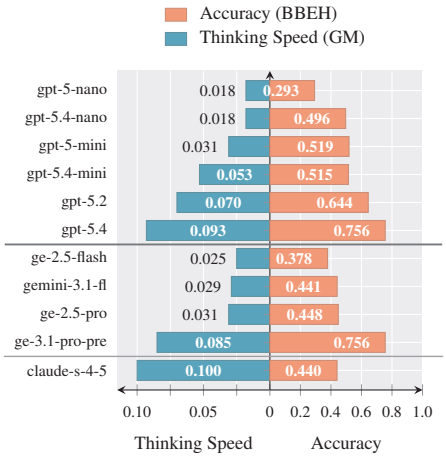
Table 4: Performance on graph mapping while forcing models to use minimal thinking tokens. When a model does not have perfect performance, its performance on direct mapping is generally higher than on indirect mapping. Similarly, for crossed and uncrossed edges, models perform better when edges are not crossed. Similar colors represent similar trends. Abbreviations: G is gpt, GE is gemini. NP is not provided. Note: Values corresponding to human are on visual graph-mapping, and is shown only for demonstrating the trend, and not comparison.

7 Thinking Speed vs General Reasoning

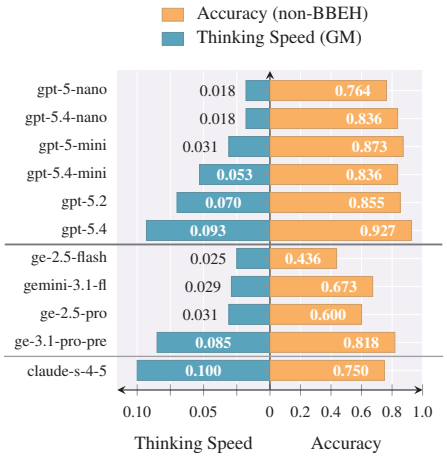
As mentioned in the Section 3, we investigate whether good graph-mapping performance with regard to thinking speed is related to better performance across a wide range of reasoning tasks. We evaluated the models on Big-Bench Extra Hard (BBEH) (Kazemi et al., 2025), consisting of 27 tasks(we use 23 of them), obtained from Big Bench Hard (BBH) (Suzgun et al., 2022). Additionally, we considered other question-answering benchmarks, namely Math-500 (Hendrycks et al., 2021), AIME 2025 I (Zhang and Math-AI, 2025), GPQA (Diamond) (Rein et al., 2023), AMC23 (Knovel Engineering, 2025), and the multi-hop question-answering benchmarks MuSiQue (Trivedi et al., 2022) and WikiHop (Welbl et al., 2018), all of which are categorized into BBEH and non-BBEH benchmarks. For BBEH, AMC-23, MuSiQue, and AIME 2025, we sample the first 10 questions from each task, while for GPQA, WikiHop, and MATH-500 we sample 15, 20, and 30 questions from each dataset respectively. The accuracy and average number of thinking tokens for various LLMs across tasks are shown in Table 10. The comparison based on thinking speed is illustrated in Figure 4. The thinking speed for various tasks is also shown in Table 6. Additionally, whether a model's good thinking speed in graph mapping predicts better performance across tasks remains open to discussion. To that end, emphasizing intra-family comparisons, thinking speed is to some extent associated with improved overall performance. However, there are exceptions. For example, ge-3.1-fl (gemini-3.1-flash-lite) shows overall better performance on non-BBEH tasks, even though it has relatively lower speed. Moreover, some models are especially good at certain tasks, while in most tasks they perform relatively poorly. For example, comparing gpt-5.4-mini and gpt-5.2, the overall performance of gpt-5.2 is higher than that of gpt-5.4-mini; however, on the BBEH time_arithmetic sub-task, gpt-5.4-mini performs better than gpt-5.2 (see Table 10).

7.1 Thinking Speed of Models Across Tasks

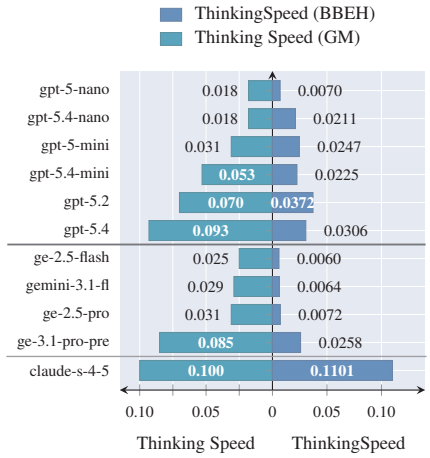
To show that LLMs' thinking speed of each model has consistent ranking across various tasks, we use Kendall's W (Kendall and Smith, 1939), also



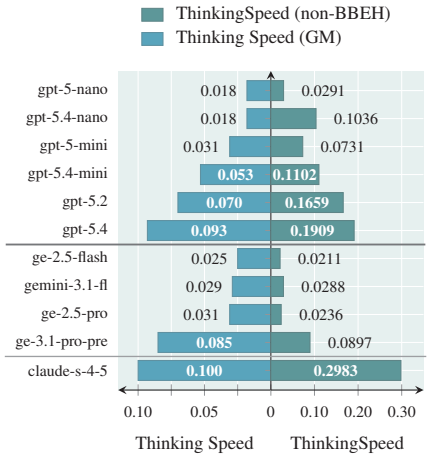
(a) Thinking Speed (GM) vs Accuracy (BBEH)



(b) Thinking Speed (GM) vs Accuracy (non-BBEH)



(c) Thinking Speed GM vs BBEH



(d) Thinking Speed GM vs non-BBEH

Figure 4: Thinking Speed graph-mapping (GM) vs Accuracy and Thinking Speed, grouped by model family, sorted by Thinking Speed (GM) within each family. Abbreviation in model IDs: ge is gemini.

known as the coefficient of concordance. Kendall's W is used to show the agreement among the judges or raters. The value of Kendall's W ranges between 0 and 1, with 1 indicating that every rater has given the exact same rank to each given object from a given list of objects, and 0 indicating no concordance. Accordingly, we treat tasks as raters, and LLMs as objects being rated. Additionally, the score each model receives from a rater (task) corresponds to the model's thinking speed (T_s) on the given task. For example, given 11 models and 29 tasks (all models and tasks shown in Table 6), the model with the highest T_s on a given task receives a rank of 11 in that task, and the model with lowest speed, a rank of 1. Accordingly, Table 5 shows the average rankings of the top ranked models (details are provided in Appendix A.7).

As such, the highest score is for claude-sonnet-4-5 (Table 5). More importantly, according to Kendall's W(0.7312), models rankings based on thinking speed exhibit substantial concordance across tasks..

Model	Ratings(avg)
claude-sonnet-4-5	10.24
gpt-5.2	9.20
gpt-5.4	8.75
gemini-3.1-pro-pre	7.68

Table 5: Average Model Ratings by various tasks based on thinking speed (shown are top ranked models).

8 Discussion: Human versus LLMs

The visual graph-mapping test is associated with intelligence in humans (Jastrzębski et al., 2023). Additionally, some LLMs are impressively good at the language-based version of the test men-

Dataset	claude -s-4-5	gpt -5.2	gpt -5.4	gpt-5.4 -nano	gpt-5.4 -mini	gpt-5 -mini	ge-3.1 -pr-pre	gpt-5 -nano	ge-2.5 -fl	ge-2.5 -pro	ge-3.1 -fl-lite
AIME 2025 I	2.1	0.6	1.1	0.4	0.4	0.3	0.3	0.2	0.0	0.1	0.0
AMC 23	4.0	1.7	2.7	1.2	1.5	1.1	1.0	0.3	0.2	0.2	0.2
GPQA Diamond	3.4	1.1	0.7	0.8	0.7	0.4	0.8	0.2	0.1	0.2	0.3
MATH-500	2.7	2.0	3.0	1.2	1.2	1.0	1.1	0.3	0.4	0.3	0.3
MuSiQue	4.8	6.3	2.7	1.9	2.5	1.1	2.1	0.5	1.5	0.9	0.7
WikiHop	2.4	5.7	7.7	2.6	2.8	1.8	2.6	0.4	0.8	0.5	1.2
boolean_expressions	1.2	0.0	0.1	0.0	0.2	0.1	0.3	0.0	0.0	0.0	0.0
linguini	0.6	0.2	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.0
temporal_sequence	0.7	0.1	0.1	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.0
movie_recommend.	2.8	2.2	1.5	1.2	1.0	0.8	1.6	0.3	0.2	0.3	0.6
word_sorting	1.7	0.8	1.1	0.5	0.3	0.4	0.6	0.1	0.2	0.2	0.1
spatial_reasoning	0.7	0.3	0.5	0.1	0.1	0.1	0.4	0.0	0.0	0.1	0.1
causal_understand	2.0	6.6	4.7	2.9	4.0	1.0	1.7	0.1	0.3	0.4	0.5
shuffled_objects	0.4	0.1	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
web_of_lies	0.6	0.3	0.4	0.2	0.3	0.2	0.5	0.1	0.0	0.1	0.1
boardgame_qa	1.7	0.8	0.8	0.5	0.5	0.4	0.6	0.1	0.1	0.2	0.1
object_counting	3.0	1.4	0.9	0.4	0.7	0.3	0.3	0.0	0.3	0.3	0.2
dyck_languages	1.2	0.8	0.8	0.2	0.1	0.4	0.7	0.1	0.4	0.2	0.2
disambiguation_qa	1.4	1.3	0.5	1.3	0.2	0.2	0.4	0.1	0.2	0.5	0.1
hyperbaton	0.9	0.3	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.1	0.0
buggy_tables	0.2	0.2	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0
time_arithmetic	2.9	1.4	1.7	0.6	1.5	0.6	0.7	0.3	0.3	0.3	0.1
nycc	0.3	0.5	0.2	0.6	0.1	0.1	0.5	0.1	0.0	0.1	0.2
sportqa	1.0	0.9	0.5	0.2	0.2	0.3	0.5	0.1	0.2	0.1	0.2
multistep_arith	0.7	0.3	0.2	0.2	0.2	0.3	0.2	0.0	0.0	0.0	0.0
geometric_shapes	2.5	0.2	0.2	0.1	0.2	0.2	0.3	0.1	0.0	0.0	0.0
zebra_puzzles	0.6	0.3	0.2	0.2	0.3	0.1	0.1	0.0	0.0	0.0	0.0
sarc_triples	1.0	1.8	0.3	0.9	0.5	0.4	0.6	0.0	0.4	0.4	0.2
object_properties	0.0	0.2	0.2	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.0
Avg non-BBEH	**3.0**	**1.7**	**1.9**	**1.0**	**1.1**	**0.7**	**0.9**	**0.3**	**0.2**	**0.2**	**0.3**
Avg BBEH	**1.1**	**0.4**	**0.3**	**0.2**	**0.2**	**0.2**	**0.3**	**0.1**	**0.1**	**0.1**	**0.1**
Overall	1.5	0.5	0.4	0.3	0.3	0.3	0.3	0.1	0.1	0.1	0.1

Table 6: Thinking Speed ($10 \times T_s$). Darker blue = higher thinking speed. A score of 0.0 indicates that the model failed on all sampled questions for that task. Model ID abbreviations: s is sonnet, ge is gemini, fl is flash, pre is preview, pr is pro (ge-2.5-fl stands for gemini-2.5-flash).

tioned. Therefore, explanations and interpretations of LLMs' performance on language-based graph-mapping and humans' performance on visual graph-mapping are open to discussion; however, a direct comparison between LLMs performance reported in this work and the visual graph-mapping test used for humans by Jastrzębski et al. (2023) is questionable, as the test format has changed. Nonetheless, LLMs' good performance on larger d-regular digraphs, where difficulty factors such as the number of nodes and edges, as well as the indirect reasoning required, accumulate, is plausible. As such, what, if anything, LLMs do better than humans remains to be investigated. To that end, it is also worth noting that the interplay among various factors of intelligence may differ between humans and LLMs. For example, studies show that working memory capacity and reasoning ability are highly correlated in humans (Kyllonen and Christal, 1990); however, this may not hold for LLMs.

9 Conclusion

In this paper, we show that some LLMs are good at identifying graph isomorphisms, even in a random directed d-regular graphs; however, they vary in thinking speed. Additionally, we propose thinking speed on graph-mapping as a metric for evaluating LLMs.

Limitations

The first limitation of this work concerns the comparison between humans and LLMs. Accordingly, this work does not examine the observations from a cognitive science perspective. The second limitation concerns the very simple definition of thinking speed. To that end, we argue that this simplicity and intuitiveness may nevertheless be advantageous. Moreover, the sample sizes included from each benchmark are relatively small, and increasing the sample size definitely adds to reliability of findings.

Ethical Considerations

This work was conducted in accordance with the ACL Code of Ethics. As such, we have been honest in our observations and in what is reported in this work. This paper values other people's professional work and cites them when we have benefited from their work.

References

Béla Bollobás. 1980. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316.

Raymond B. Cattell. 1973. *A Culture-Free Intelligence Test I*, pages 155–173. Springer Netherlands, Dordrecht.

Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. 2026. Think deep, not just long: Measuring llm reasoning effort via deep-thinking tokens. *Preprint*, arXiv:2602.13517.

François Chollet. 2019. On the measure of intelligence. *Preprint*, arXiv:1911.01547.

Xinnan Dai, Haohao Qu, Yifen Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2025. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. *Preprint*, arXiv:2410.05298.

Zheng Du, Hao Kang, Song Han, Tushar Krishna, and Ligeng Zhu. 2026. Ockbench: Measuring the efficiency of llm reasoning. *Preprint*, arXiv:2511.05722.

Paul Erdős and Alfréd Rényi. 1963. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3–4):295–315.

Catherine Greenhill. 2011. A polynomial bound on the mixing time of a Markov chain for sampling regular directed graphs. *Electronic Journal of Combinatorics*, 18(1):#P234.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Xiaoyang Hu, Shane Storks, Richard L. Lewis, and Joyce Chai. 2023. In-context analogical reasoning with pre-trained language models. *Preprint*, arXiv:2305.17626.

Svante Janson. 2020. Random graphs with given vertex degrees and switchings. *Random Structures & Algorithms*, 57(1):3–31.

Jan Jastrzębski, Michał Ociepka, and Adam Chuderski. 2023. Graph mapping: A novel and simple test to validly assess fluid reasoning. *Behav. Res. Methods*, 55(1):448–460.

Ravi Kannan, Prasad Tetali, and Santosh Vempala. 1999. Simple Markov-chain algorithms for generating bipartite graphs and tournaments. *Random Structures & Algorithms*, 14(4):293–308.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V Le, and Orhan Firat. 2025. BIG-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501, Vienna, Austria. Association for Computational Linguistics.

M. G. Kendall and B. Babington Smith. 1939. The Problem of m Rankings. *The Annals of Mathematical Statistics*, 10(3):275 – 287.

Maurice G. Kendall and Jean Dickinson Gibbons. 1990. *Rank Correlation Methods*, 5th edition. Oxford University Press.

Jeong Han Kim, Benny Sudakov, and Van H. Vu. 2002. On the asymmetry of random regular graphs and random graphs. *Random Struct. Algorithms*, 21(3–4):216–224.

Hendrike Klein-Hennig and Alexander K. Hartmann. 2012. Bias in generation of random graphs. *Physical Review E*, 85(2):026101.

Knovel Engineering. 2025. AMC-23.

Patrick C. Kyllonen and Raymond E. Christal. 1990. Reasoning ability is (little more than) working-memory capacity?! *Intelligence*, 14(4):389–433.

Seungpil Lee, Woochang Sim, Donghyeon Shin, Sanha Hwang, Wongyu Seo, Jiwon Park, Seokki Lee, Sejin Kim, and Sundong Kim. 2024. Reasoning abilities of large language models: In-depth analysis on the abstraction and reasoning corpus. *Preprint*, arXiv:2403.11793.

Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. 2001. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark. *Preprint*, arXiv:2311.12022.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

Robin Taylor. 1981. Constrained switching in graphs. In *Combinatorial Mathematics VIII*, volume 884 of *Lecture Notes in Mathematics*, pages 314–336. Springer, Berlin.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

N. C. Wormald. 1999. *Models of Random Regular Graphs*, page 239–298. London Mathematical Society Lecture Note Series. Cambridge University Press.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024. Can llm graph reasoning generalize beyond pattern memorization? *Preprint*, arXiv:2406.15992.

A Appendix

A.1 Implementation Details

We used APIs available from OpenAI, Google Gemini, and Claude, and API endpoints from huggingface.com also shown in Table 7. The endpoints were accessed between January 1 and August 1, 2026. The graph-mapping dataset was accessed from the OSF repository https://osf.io/wh7zv/overview. The alphabetical names used for each node in the graph were random in every run.

Category	Model ID
Gemini	gemini-3-flash-preview gemini-3-pro-preview gemini-2.5-pro gemini-2.5-flash gemini-3.1-pro-preview gemini-3.1-flash-lite gemini-3.1-flash-lite-preview
GPT	gpt-5.2 gpt-5-mini gpt-5-nano gpt-5.4 gpt-5.4-nano gpt-5.4-mini gpt-5
HuggingFace	Qwen/Qwen3.5-397B-A17B:novita Qwen/Qwen3.5-35B-A3B deepseek-ai/DeepSeek-V4-Flash:novita Qwen/Qwen3.6-35B-A3B:featherless-ai deepseek-ai/DeepSeek-V4-Pro:novita deepseek-ai/DeepSeek-V3.2-Exp:novita Qwen/Qwen2.5-7B-Instruct:together
Anthropic	claude-sonnet-4-5

Table 7: Model IDs used in the evaluation, grouped by provider category.

A.2 Random d-regular Digraph Generation

Note: The algorithm used to generate d-regular digraphs is shown in Algorithm 1. The validity of the algorithm was tested in 3 trials of 12 questions with n=7 and d=3. Accordingly, we observed that with a sufficient thinking budget (9000), gpt-5.2 finds a correct mapping most of the time , confirming the identity automorphism to be the only mapping that preserves the graph's connectivity structure.

Problem

Given $n > 2d \geq 6$, produce a simple directed graph $G = (V, E)$ with all nodes having d incoming and d outgoing edges, also satisfying the following: **C3** cycle diversity (Kim et al., 2002); **C4** unique neighborhood signatures $(N^-(v), N^+(v))$; and **C5** graph asymmetry, i.e., $|\mathrm{Aut}(G)| = 1$ (Erdős and Rényi, 1963).

The steps are shown in Algorithm 1, and are explained as follows:

Explanation:

Step 1 — Configuration model

Each node v receives d out-stubs and d in-stubs. Both stub lists are independently shuffled and zipped into nd candidate edges—the directed analogue of Bollobás's pairing (Bollobás, 1980), adapted by Newman et al. (2001). Accordingly, the marginal probability that edge $u \to v$ appears is $p_1 = 1 - \binom{nd-d}{d} / \binom{nd}{d} \approx d/n$,

Step 2 — Repair Self-loops and multi-edges are removed by swapping in-stub partners. This conditional step introduces a positive bias in edge probabilities as $n \to \infty$ (Klein-Hennig and Hartmann, 2012).

Step 3 — 2-switch Markov chain Taylor (1981)'s *2-switch* replaces $(u_1, v_1), (u_2, v_2)$ with $(u_1, v_2), (u_2, v_1)$ whenever the result stays simple and d-regular.

Validity Check C3 and C4 are checked via depth-first search (DFS) and signature comparison. The C5 automorphism test (for $n \leq 12$) is based on the work of Kim et al. (2002), which proved that almost all d-regular graphs ($d \geq 3$) are asymmetric with high probability, a result whose foundations lie in the probabilistic arguments of Erdős and Rényi (1963).

Note: claude.ai was used by the authors as a tool for writing functions when generating the algorithm.

Algorithm 1 GENERATEDREGULARDIGRAPH(n, d)

for up to 10,000 attempts **do**
  // Step 1 — stub matching Bollobás (1980); Newman et al. (2001)
  $\mathbf{o}, \mathbf{i} \leftarrow [0^d, \ldots, (n-1)^d]$ (twice); $E, B \leftarrow \emptyset$
  shuffle $\mathbf{o}$ and $\mathbf{i}$ independently and uniformly
  for $k = 0, \ldots, nd-1$ **do**
    if $\mathbf{o}_k = \mathbf{i}_k$ **or** $(\mathbf{o}_k, \mathbf{i}_k) \in E$ **then** $B \leftarrow B \cup \{k\}$
    else $E \leftarrow E \cup \{(\mathbf{o}_k, \mathbf{i}_k)\}$
    end if
  end for
  // Step 2 — swap repair Klein-Hennig and Hartmann (2012); Janson (2020)
  for each $b \in B$ **do**
    find j: swap $\mathbf{i}_b \leftrightarrow \mathbf{i}_j$ adds no loop/multi-edge
    if no such j **then restart**
    end if
  end for
  // Step 3 — 2-switch chain Taylor (1981); Kannan et al. (1999); Greenhill (2011)
  for $t = 1, \ldots, 5nd$ **do**
    pick $(u_1, v_1), (u_2, v_2) \in E$; apply valid 2-switch to E
  end for
  // Check
  if G passes **C3**, **C4**, **C5 then return** $A(G)$
  end if
end for

A.3 Human Performance

We include some figures from the Jastrzębski et al. (2023) paper here, as they provide an overview of human performance shown in Figure 5.

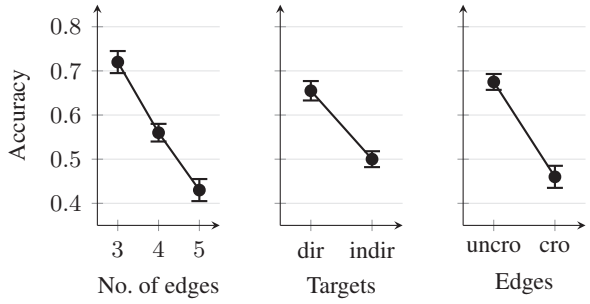


Figure 5: Human accuracy on graph-mapping and its relation with (left) number of edges, (center) target type, and (right) edge crossing. Error bars indicate ±1 standard error. Taken from the graph-mapping paper Jastrzębski et al. (2023).

A.4 Average Thinking Tokens while Forcing Minimal Thinking

To force models to use the minimal number of tokens with which they can correctly solve the graph-mapping question, we first let them answer in the medium thinking mode (or simply the thinking mode, in the case of DeepSeek and Qwen models), and then compute the average length of their reported thinking tokens. In the next run, we give them a thinking budget equal to $\frac{3}{4}$ of their average in medium mode.

Model	Medium
Qwen3.5-35B-A3B	5,283
Qwen3.5-397B	1,996
gemini-2.5-flash	1,234
gemini-2.5-pro	1,424
gpt-5-mini	1,433
gpt-5-nano	3,982
gpt-5.2	664
gpt-5.4	536
gpt-5.4-mini	747
gpt-5.4-nano	895

Table 8: Average (2 trials) thought tokens per model in Medium thinking mode(or simply thinking mode).

A.5 Thinking Modes

Model	Non-thinking	Thinking		Think Max	
	Acc	Acc	Tok(K)	Acc	Tok(K)
Q3.5-35B	-	1.0	4.8	-	-
Q3.5-397B	0.5	1.0	2.0	-	-
DS-V4-Flash	-	1.0	1.7	1.0	1.8 (↑5.88%)
DS-V4-Pro	-	1.0	1.5	1.0	2.0 (↑33.3%)

Table 9: Qwen & DeepSeek (DS) performance metrics in various thinking modes. Abbreviation in model IDs: Q is Qwen, and DS is DeepSeek (DS-V4-Pro stands for DeepSeek-v4-pro). Tokens are shown in units of thousands. Inside parentheses are percentage change.

A.6 Thinking Budget

We experimented with varying thinking budgets across model types, and noted that different models show varying performance under the same thinking budget as can be seen in Figure 6.

A.7 Thinking Speed Across Tasks

Given n models (11 in this study) ranked by m tasks (29 in this study), Kendall's W is calculated using the following equation (Kendall and Smith, 1939) :

$$W = \frac{12S}{m^2(n^3 - n)}$$

Where S is the sum of squares of the deviations of sums of ranks from their mean value, and is obtained by the following equation:

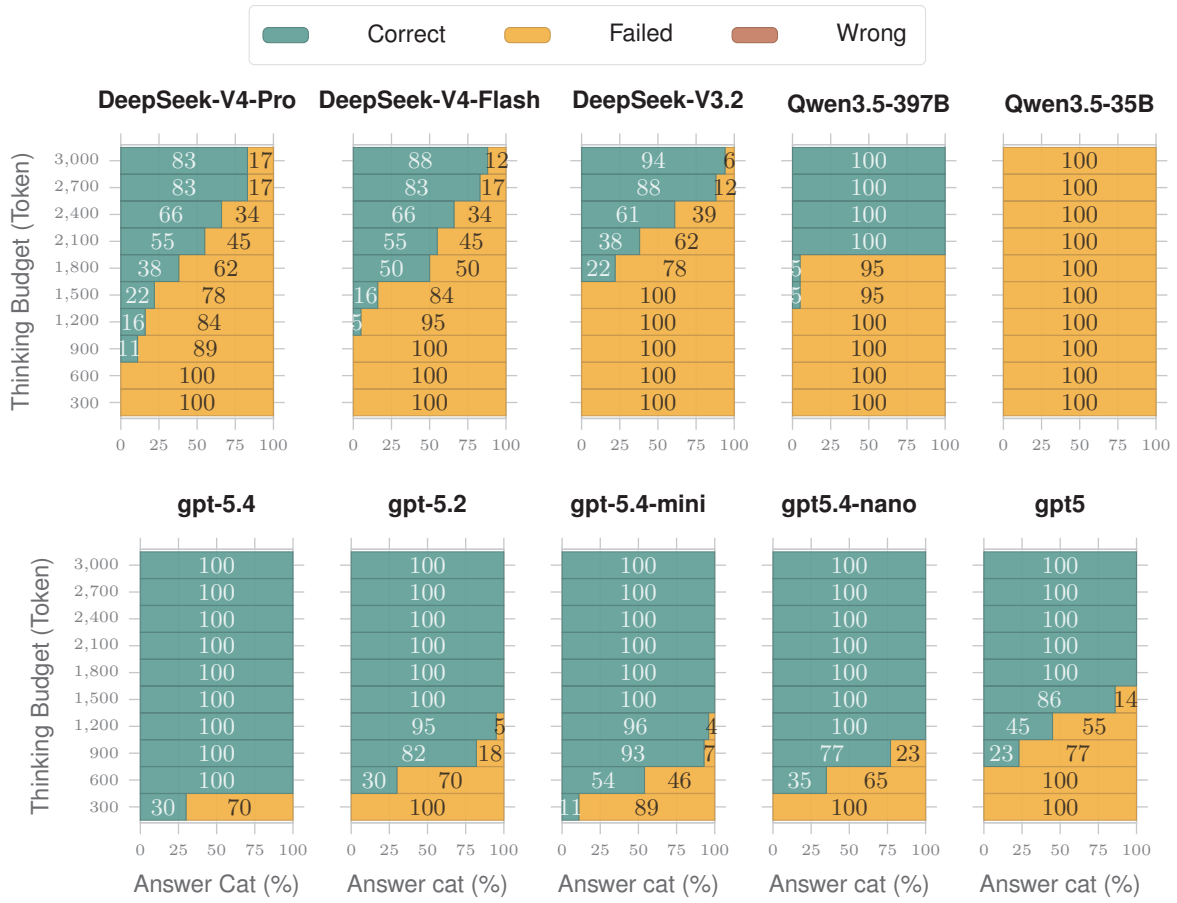


Figure 6: Graph-Mapping (experiment set) answer types under varying thinking budgets across different model families:GPT. A model may fail to answer within the budget, may answer incorrectly, or, in the best case, may answer correctly.

$$S = \sum_{i=1}^{n} \left(R_i - \bar{R}\right)^2.$$

Where $\bar{R}$ is the mean rank sum across models, and R_i is the total rank of object i, each defined as follows:

$$\bar{R} = \frac{1}{n} \sum_{i=1}^{n} R_i.$$

$$R_i = \sum_{j=1}^{m} r_{i,j},$$

Kendall's W ranges between 0 and 1. Additionally, the significance test described by Kendall and Gibbons (1990) is given by the following equation:

$$\chi^2 = m(n-1)W$$

Model	Ratings(avg)
claude-sonnet-4-5	10.24
gpt-5.2	9.20
gpt-5.4	8.75
gemini-3.1-pro-pre	7.68
gpt-5.4-mini	6.79
gpt-5.4-nano	6.62
gpt-5-mini	5.32
gemini-2.5-pro	3.20
gemini-3.1-flash-lite	2.94
gemini-2.5-flash	2.74
gpt-5-nano	2.46

Table 11: Average Ratings of models by various tasks based on their thinking speed.

Where the test statistic takes a chi-squared distribution with $df = n - 1 = 10$ degrees of freedom. As such, given n=11 (all models shown in Table 6) and m=29 (all tasks shown in Table 6), and ranks given to each model such that highest rank is as-

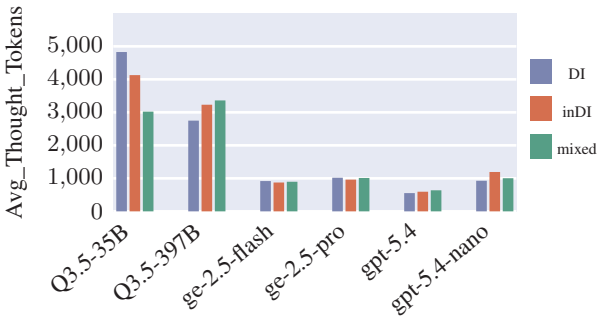
Dataset	Claude-s-4-5		gpt 5.2		gpt-5.4		gpt-5.4 nano		gpt-5.4 mini		gpt-5 mini		ge-3.1 pro-pre		gpt-5 nano		ge-2.5 fl		ge-2.5 pro		ge-3.1 fl	
	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T
AIME 2025 I	0.700	0.34	0.900	1.57	0.800	0.78	0.800	2.28	0.700	2.12	0.800	3.16	0.800	3.04	0.800	4.53	0.000	7.2	0.500	6.15	0.300	6.6
AMC 23	1.000	0.25	0.900	0.53	1.000	0.37	1.000	0.84	1.000	0.67	1.000	0.93	0.900	0.86	0.800	2.69	0.800	3.52	0.800	4.13	0.700	3.22
GPQA Diamond	0.733	0.22	0.933	0.87	0.933	1.29	0.867	1.05	0.733	1.17	0.867	2	0.800	1.03	0.800	4.16	0.533	4.13	0.667	3.67	0.867	3.17
math500	0.633	0.24	0.900	0.45	0.800	0.27	0.767	0.65	0.800	0.65	0.833	0.86	0.767	0.7	0.633	2.17	0.567	1.6	0.667	2.53	0.767	2.27
musique	1.000	0.21	0.900	0.14	0.900	0.34	0.700	0.36	0.900	0.36	0.700	0.62	0.900	0.42	0.700	1.39	0.800	0.54	0.900	1	0.900	1.32
wiki	0.450	0.19	0.750	0.13	0.950	0.12	0.750	0.29	0.900	0.32	0.850	0.48	0.800	0.31	0.700	1.67	0.400	0.48	0.500	0.93	0.700	0.61
boolean_exp	0.500	0.4	0.000	7.22	0.400	7	0.000	4.49	0.300	5.31	0.300	7.59	0.900	3.52	0.100	6.29	0.000	8.53	0.100	8.92	0.300	7.37
linguini	0.200	0.31	0.300	1.55	0.600	3.79	0.200	2.66	0.500	4.33	0.400	3	0.600	2.52	0.300	5.62	0.300	4.92	0.200	3.96	0.200	5.17
temporal_seq	0.300	0.41	0.500	6.74	0.800	6.41	0.100	5.14	0.200	5.86	0.000	4.42	0.900	6.14	—	—	0.000	8.83	0.000	9.2	0.000	8.83
movie_rec	0.900	0.32	0.800	0.37	0.900	0.61	0.800	0.66	0.800	0.8	0.800	1.02	1.000	0.63	0.900	2.67	0.600	3.49	0.900	3.26	1.000	1.63
word_sorting	0.500	0.3	0.800	0.97	0.900	0.8	0.600	1.26	0.400	1.16	0.800	1.9	0.800	1.38	0.600	4.74	0.700	3.12	0.600	2.92	0.500	5.73
spatial_reasoning	0.300	0.45	0.700	2	1.000	1.93	0.500	3.9	0.500	5.03	0.600	4.64	1.000	2.39	0.100	4.49	0.100	7.26	0.500	6.88	0.500	8.12
causal_under.	0.700	0.34	0.700	0.11	0.700	0.15	0.500	0.17	0.600	0.15	0.600	0.59	0.700	0.42	0.300	2.39	0.600	2.32	0.600	1.49	0.600	1.11
shuffled_objects	0.200	0.5	0.700	7.14	0.900	6.42	0.100	2.4	0.100	7.83	—	—	0.700	8	0.000	5.03	0.000	8.83	0.000	9.2	0.100	8.83
web_of_lies	0.300	0.47	0.800	3	1.000	2.61	0.400	3.59	1.000	3.74	0.600	3.65	1.000	2.1	0.100	6.08	0.300	6.73	0.400	7.17	0.400	7.41
boardgame_qa	0.900	0.54	0.900	1.08	1.000	1.29	1.000	2.01	0.800	1.49	1.000	2.69	1.000	1.56	0.300	5.57	0.800	5.36	0.900	4.43	0.800	5.75
object_counting	0.800	0.27	1.000	0.73	0.900	1.05	0.500	1.4	0.800	1.14	0.900	3.49	1.000	3.15	0.000	6.27	0.700	2.67	0.700	2.47	0.900	5.08
dyck_languages	0.600	0.5	1.000	1.23	1.000	1.28	0.500	2.55	0.400	4.11	0.900	2.4	0.600	0.9	0.400	4.69	0.900	2.47	0.800	4.81	0.900	5.95
disambig_qa	0.500	0.36	0.300	0.23	0.300	0.62	0.500	0.38	0.200	1.2	0.200	0.97	0.300	0.82	0.200	3.64	0.400	2.24	0.600	1.25	0.200	2.12
hyperbaton	0.400	0.46	1.000	3.91	1.000	4.2	1.000	5.48	0.700	6.26	0.900	6.56	0.900	5.63	0.000	6.8	0.300	8.83	0.700	8.22	0.300	8.8
buggy_tables	0.100	0.42	0.300	2.84	0.900	4.08	0.100	5.01	0.100	4.27	0.000	5.12	0.700	6.53	0.100	5.89	0.000	8.83	0.000	7.97	0.100	7.95
time_arithmetic	0.800	0.28	0.900	0.64	0.900	0.52	0.900	1.45	1.000	0.66	0.900	1.43	0.800	1.19	0.900	3.94	0.900	2.66	0.700	2.75	0.500	7.03
nycc	0.100	0.36	0.100	0.19	0.100	0.41	0.200	0.33	0.100	0.81	0.100	0.96	0.300	0.64	0.200	2.05	0.100	2.47	0.300	2.24	0.200	1.1
sportqa	0.400	0.4	0.500	0.56	0.600	1.26	0.300	1.44	0.300	1.42	0.500	1.61	0.600	1.2	0.300	4.37	0.500	3.04	0.400	3.15	0.400	2.66
multistep_arith.	0.400	0.54	0.700	3.57	0.900	3.68	0.700	4.11	0.600	3.55	0.500	3.49	0.400	2.1	0.100	6.13	0.000	8.5	0.200	8.49	0.100	8.52
geometric_shapes	0.700	0.28	0.700	2.83	0.900	4.11	0.700	5.3	0.800	6.4	0.500	4.3	1.000	3.44	0.300	7.73	0.000	8.83	0.100	8.96	0.000	8.83
zebra_puzzles	0.300	0.52	0.300	2.93	0.400	5.29	0.600	4.19	0.100	3.35	0.100	6.23	0.700	6.81	0.100	6.34	0.000	8.83	0.000	8.72	0.400	8.05
sarc_triples	0.300	0.29	0.200	0.11	0.100	0.3	0.200	0.22	0.200	0.44	0.200	0.52	0.400	0.66	0.000	2.19	0.500	1.26	0.500	1.26	0.300	1.65
object_prop	0.000	0.55	0.600	5.45	0.900	4.78	0.000	4.36	0.100	6.81	—	—	0.900	5	0.000	4.28	0.000	8.83	0.000	9.2	0.000	8.83
Avg non-BBEH	**0.695**	0.23	**0.874**	0.53	**0.884**	0.47	**0.800**	0.78	**0.832**	0.77	**0.842**	1.17	**0.811**	0.9	**0.716**	2.54	**0.516**	2.44	**0.653**	2.76	**0.726**	2.52
Avg BBEH	**0.443**	0.4	**0.600**	1.93	**0.743**	2.55	**0.452**	2.47	**0.461**	2.67	**0.470**	2.64	**0.748**	2.9	**0.230**	4.54	**0.335**	5.6	**0.400**	5.52	**0.378**	5.94
Overall	0.517	0.35	0.680	1.47	0.785	1.93	0.554	1.93	0.569	2.01	0.578	2.11	0.766	2.32	0.372	3.83	0.388	4.68	0.474	4.71	0.480	4.94

Table 10: Accuracy (**A**) and thinking tokens in thousands (**T**) heatmap sorted left-to-right by increasing overall thinking tokens. Darker green = higher accuracy; darker blue = more thinking tokens. — stands for: failed in all questions. In other words, the model used the total budget, but still did not answer the question. Abbreviations in model IDs: s is sonnet, pre is preview, fl is flash,and ge is gemini.

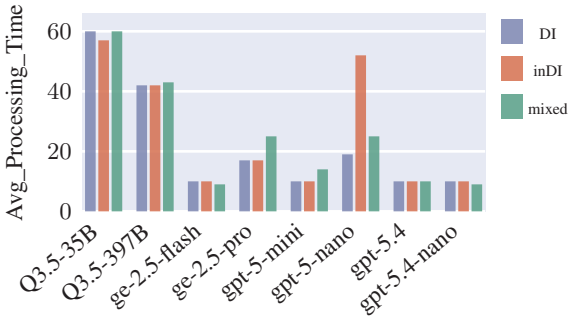
signed to the fastest model, and lowest rank to the slowest model respectively, the average rank each model receives is shown in Table 11, the Kendall's W = 0.7312, and $\chi^2 = 212.04$ with $p \leq 0.001$.

A.8 Sample Question Prompt and Sample Answer

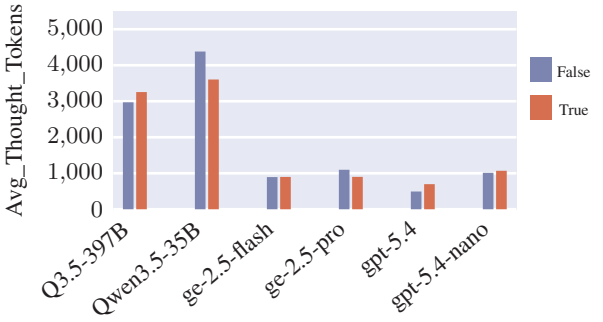
In this section, a sample prompt is shown in Figure 8, and a sample response is shown in Figure 9.



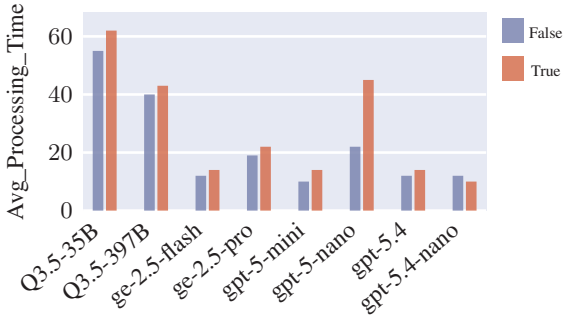
(a) Avg Thought Tokens (Mapping Types)



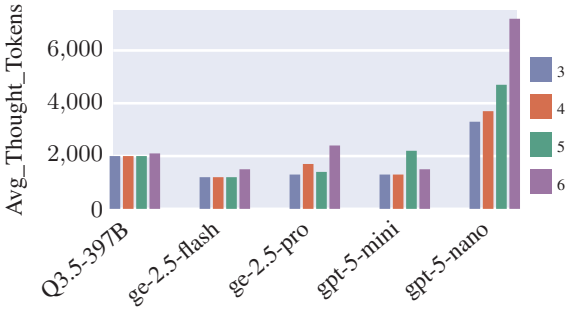
(b) Avg Processing Time (Mapping Types)



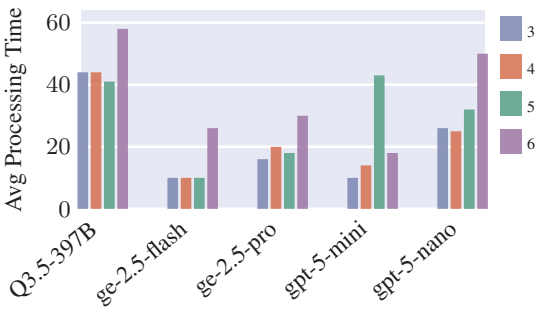
(c) Avg Thought Tokens (Edge Crossing)



(d) Avg Processing Time (Edge Crossing)



(e) Avg Thought Tokens (Number of Nodes)



(f) Avg Processing Time (Number of Nodes)

Figure 7: Average thought tokens (left column) and average processing time (right column) grouped by mapping type (top), edge crossing property (middle), and number of nodes (bottom). Note on model-name abbreviations: Ge stands for Gemini. Additionally, the reported value for number of nodes and number of edges is from running the experiment with 60 question subset.

Example graph-mapping Question

Question No: `experiment_test_0`

Given two isomorphic graphs, namely source and target, your task is to map some of the target nodes to the corresponding source nodes. Note the following:

Rules:

- the first graph is called the source graph.
- the second graph is called target graph.
- For each graph, they are specified using adjacency matrix.
- The order of nodes in the row and column of adj matrix is given for each graph.
- The target graph is obtained from the source graph by renaming the nodes, and probably shuffling the orders in row and column of adjacency matrix.

Question: Map given source nodes to appropriate target nodes: `['s2', 's45']`

Initially:

```
source_adj: [[0, 1, 0, 0], [0, 0, 0, 1], [1, 0, 0, 0], [1, 0, 1, 0]]
nodes order in row, source adj = [s45, s12, s0, s2]
nodes order in column, source adj = [s45, s12, s0, s2]
```

After renaming nodes, and shuffling the adjacency matrix:

```
target_adj = [[1, 0, 0, 0], [0, 1, 0, 0],[1, 0, 0, 1],[0, 0, 1, 0]]
nodes order in row, target adj = [t0, t18, t16, t29]
nodes order in column, target adj = [t29, t16, t18, t0]
Crossed_edges = True
```

Figure 8: graph-mapping example question and prompt.

Thinking for Graph Mapping by claude-sonnet-4-5

Let me analyze the graphs step by step.
Source adjacency matrix:
Nodes: [s45, s12, s0, s2]

	s45	s12	s0	s2
s45	0	1	0	0
s12	0	0	0	1
s0	1	0	0	0
s2	1	0	1	0

Edges: $s45 \to s12,\ s12 \to s2,\ s0 \to s45,\ s2 \to s45,\ s2 \to s0$
Source node degrees (out, in):

- s45: out=1, in=2
- s12: out=1, in=1
- s0: out=1, in=1
- s2: out=2, in=1

Target adjacency matrix:
Nodes rows: [t29, t16, t18, t0], cols: [t0, t18, t16, t29]

	t0	t18	t16	t29
t29	1	0	0	0
t16	0	1	0	0
t18	1	0	0	1
t0	0	0	1	0

Edges: $t29 \to t0,\ t16 \to t18,\ t18 \to t0,\ t18 \to t29,\ t0 \to t16$
Target node degrees (out, in):

- t29: out=1, in=1
- t16: out=1, in=1
- t18: out=2, in=1
- t0: out=1, in=2

Mapping by degree signature:

- s2 (out=2, in=1) $\leftrightarrow$ t18 (out=2, in=1)
- s45 (out=1, in=2) $\leftrightarrow$ t0 (out=1, in=2)
- s12 (out=1, in=1) and s0 (out=1, in=1) $\leftrightarrow$ t29 or t16

$s2 \to s45$ maps to $t18 \to t0$ ✓
$s45 \to s12$ maps to $t0 \to t16$, so $s12 \leftrightarrow t16$
$s0 \to s45$ maps to ? $\to t0$: $t18 \to t0$, so $s0 \leftrightarrow t18$... but $s2 \leftrightarrow t18$.
Let me check $s2 \to s0$: $t18 \to$? should be $t29$ ($t18 \to t29$), so $s0 \leftrightarrow t29$.
Check: $s0 \to s45 \to t29 \to t0$ ✓ ($t29 \to t0$ exists)
So: $\boxed{s45 \leftrightarrow t0,\quad s12 \leftrightarrow t16,\quad s2 \leftrightarrow t18,\quad s0 \leftrightarrow t29}$

Figure 9: Sample answer by claude-sonnet-4-5