

484 A Discussion

485 **NLP tasks and graph tasks alignment.** Originally designed for processing language, LLMs are
486 increasingly adopted for NLP tasks with structural implications. For instance, structured common-
487 sense reasoning is similar to topological sort in that they both need to find a solution satisfying
488 diversified constraints between entities. In the belief state graph in theory-of-mind [Adhikari et al.,
489 2020, Ammanabrolu and Riedl, 2021], the final task is to determine the connectivity of nodes, which
490 corresponds to the connectivity task. In multi-hop question answering [Creswell et al., 2022], LLMs
491 implicitly find connections and paths among a vast network of entities and concepts, which resembles
492 the shortest path task and connectivity task. However, we acknowledge that not all graph reason-
493 ing tasks in the NLGraph benchmark are clearly aligned with certain applications, thus advanced
494 graph reasoning tasks in the NLGraph benchmark could also be viewed as a test of language model
495 reasoning ability.

496 **The NLGraph benchmark envisioned as reasoning benchmark.** In addition to evaluating graph
497 reasoning abilities, the NLGraph benchmark can also be viewed as a structural benchmark to evaluate
498 language model reasoning. While ample LM reasoning datasets exist [Cobbe et al., 2021, Patel et al.,
499 2021, Talmor et al., 2019], it is difficult to avoid train-test overlap since problems such as grade
500 school math [Cobbe et al., 2021] and commonsense reasoning [Talmor et al., 2019] might be readily
501 available in pretraining corpora, making it less convincing for evaluating reasoning abilities. On the
502 contrary, the synthetic graph reasoning problems and their answers are highly unlikely to have exact
503 matches in the training corpora, which makes NLGraph a more robust benchmark towards evaluating
504 language model reasoning.

505 **Fine-tuning to elicit graph reasoning abilities.** To further study the graphical thinking abilities of
506 LLMs, we envision fine-tuning as a possible direction for future work. We plan to fine-tune language
507 models on the chain-of-thought reasoning process towards graph-based tasks on both single- and
508 multi-task settings to see if fine-tuning might lead to enhanced graph reasoning abilities.

509 B Limitations

510 **Tasks in NLGraph benchmark are not complete.** While we have incorporated eight graph
511 reasoning tasks with varying complexity in the NLGraph benchmark, there are many more important
512 graph algorithms that test different levels of graph reasoning abilities. It might also be interesting to
513 see how LLMs perform on other graph tasks such as finding the Eulerian path, the minimum spanning
514 tree, and the cut-edge.

515 **Language models that we evaluate are not complete.** We only consider four black-box LLMs,
516 TEXT-DAVINCI-003, CODE-DAVINCI-002, GPT-3.5-TURBO, and GPT-4 in our experiments. Since
517 we will make the NLGraph benchmark publicly available, we leave it to future work on evaluating
518 the graph reasoning abilities of other open-source LLMs.

519 **Limited dataset size.** Due to monetary costs, we only evaluate LLMs on the standard version of the
520 NLGraph benchmark which has 5,902 problems throughout the paper. We believe evaluating LLMs
521 on the extended version of the NLGraph benchmark, with 5x more problems, may bring stronger
522 proof to the findings we present.

523 **Methods for improving graph reasoning abilities.** The two prompting methods we present are
524 simple instruction-based approaches, which work on easy graph reasoning problems to varying
525 extents. However, on more complex graph reasoning problems, as the overall graph reasoning
526 abilities of LLMs are limited, the instruction-based methods seem to have only marginal effects. In
527 future work, we hope to investigate methods such as asking LLMs to generate and execute code, or
528 simulating steps of algorithmic solutions while maintaining the state variables.

529 C NLGraph Details

530 In Table 6, we provide the details of the NLGraph benchmark including edge probabilities and
531 specific values of parameters mentioned in §2.2.

Subset	Connect.	Cycle	Topo. Sort	Shortest Path	Maximum Flow	Bipartite Graph	Hamilton Path	GNNs
# EASY	$p: 0.3, 0.7, 1.0$	$m: 1-4$	$p: 0.3, 0.5, 0.7$	$p: 0.5, 0.7, 0.9; w: 1-4; \ell: 2-6$	$p: 0.2, 0.3; c: 1-10$	$p: 0.3-0.7$	$p: 0.4, 0.6$	$p: 0.4; \ell: 1$
# MEDIUM	$p: 0.3, 0.7, 1.0$	$m: 1-4$	$p: 0.3, 0.5, 0.7$	/	/	/	/	/
# HARD	$p: 0.3, 0.7$	$m: 1-4$	$p: 0.3, 0.5$	$p: 0.2, 0.25; w: 1-10; \ell: 2-6$	$p: 0.25; c: 1-20$	$p: 0.2-0.6$	$p: 0.4, 0.6$	$p: 0.2; \ell: 1$

Table 6: Details of the NLGraph benchmark. p denotes the edge probability. Other characters have the same meaning mentioned in §2.2.

Method	Shortest Path			
	Easy	Hard	Easy (PC)	Hard (PC)
BAG	67.78	33.50	79.20	42.56
BAG-DOT	65.00	29.00	75.55	37.18
ALGORITHMIC-DOT	66.67	31.00	78.27	39.68
INSTRUCTION #1	66.22	32.50	78.18	40.32
INSTRUCTION #2	67.78	28.50	77.32	37.06
INSTRUCTION #3	67.78	30.50	79.56	37.89

Table 7: The results of variants of instructions on the shortest path task. The BAG prompting approach achieves the best performance across three of the four settings.

532 D Analysis

533 **Instruction variants** To further study the effect of BAG prompting and ALGORITHMIC prompting,
 534 we replace the instructions with only a sequence of dots (...) equal to the number of characters in
 535 the instruction. This is to investigate whether the improved performance is attributed to the natural
 536 language instructions or simply increased computation. We also replace the BAG prompting with
 537 three other instructions not directly related to graph problems, specifically, *Let's think step by step*
 538 (INSTRUCTION 1), *Examine each detail carefully* (INSTRUCTION 2), *Think it through systematically*
 539 (INSTRUCTION 3). As illustrated in Table 7, the BAG prompting method generally outperforms the
 540 instruction variants, indicating the effectiveness of this method.

541 **Graph definition variants** We study how replacing the abstract descriptions of graphs with real-
 542 world objects would impact model performance. Specifically, for the shortest path task, we change
 543 all the "node"s into "city"s, "edge"s into "road"s, and "weight"s into "distance"s. As shown in Table
 544 8, the model performance improves on the easy subset but drops on the hard subset. While language
 545 models are indeed sensitive to the specific instantiations of the graph description, the results are
 546 mixed as to which is easier or harder.

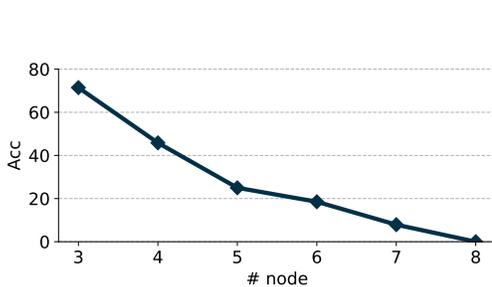


Figure 6: Model performance for the shortest path task with different numbers of nodes on the shortest path. The performance steadily drops when the path length increases.

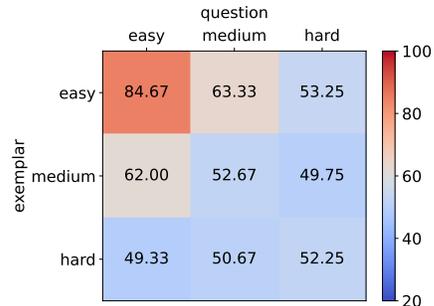


Figure 7: Model performance for the cycle task with varied exemplar difficulty. When the in-context exemplars are based on more difficult problems, the performance becomes worse.

Method	Cycle			Shortest Path				Hamilton Path	
	Easy	Medium	Hard	Easy	Hard	Easy (PC)	Hard (PC)	Easy	Hard
RANDOM	50.00	50.00	50.00	50.00	50.00	6.07	6.69	0.00	0.00
FEW-SHOT	58.00	50.00	49.75	39.44	28.50	55.10	37.63	36.67	7.00
CoT	60.67	66.83	69.00	41.11	/	45.00	/	50.00	12.00
CoT+SC	59.33	52.67	53.75	42.22	/	49.26	/	34.00	5.00

Table 8: The results of CODE-DAVINCI-002 on the cycle, shortest path, and Hamilton path task.

Method	Shortest Path			
	Easy	Hard	Easy (PC)	Hard (PC)
RANDOM	50.00	50.00	6.07	6.69
ZERO-SHOT	29.40	14.00	47.53	20.13
FEW-SHOT	41.11	20.50	58.78	29.28
CoT	73.33	28.00	82.57	35.50
0-CoT	6.67	2.00	65.40	51.95
CoT+SC	72.78	27.00	83.50	35.19

Table 9: Model performance when graph descriptions are changed into cities, roads, and distances on the shortest path task.

547 **LLMs are brittle to problem scales.** For the shortest path task, we study the correlation between
548 model performance and the number of nodes on the optimal shortest path. We group problems by
549 the shortest path length and present results in Figure 6 for the CoT approach, which shows that the
550 accuracy steadily drops when the number of nodes on the shortest path increases. While the reasoning
551 process towards finding the shortest paths stays the same for optimal solutions with varying lengths,
552 large language models are not robust to changes in graph scales and problem settings.

553 **In-context exemplars and test difficulty** We investigate the model performance when the in-
554 context exemplars and test difficulty vary. For the cycle task, we write chain-of-thought solutions
555 for problems selected from the medium and hard subsets, and evaluate TEXT-DAVINCI-003 on the
556 cycle task with the two prompts. We present the results in Figure 7. The performance drops when
557 the exemplar difficulty increases, indicating that LLM struggles to learn from exemplars with more
558 difficulty.

559 **Partial credit for the maximum flow task**

560 We provide partial credit for the maximum flow
561 task in Table 10. The performance generally has
562 the same trend as the results provided in Figure
563 2. Though on the hard subsets, the performance
564 of CoT is the highest, we believe this is more
565 like guessing, as when CoT is combined with
566 self-consistency, the performance drops signifi-
567 cantly.

Method	Easy	Hard
ZERO-SHOT	25.65	10.37
FEW-SHOT	24.83	12.37
CoT	16.99	22.07
0-CoT	17.41	7.04
CoT+SC	9.88	8.62

Table 10: Partial credit for the maximum flow task.

568 **E Additional Large Language Models**

569 **E.1 CODE-DAVINCI-002 Results**

570 We evaluate CODE-DAVINCI-002 on cycle, shortest path, and Hamilton path. The results are provided
571 in Table 9. The findings are mostly similar to TEXT-DAVINCI-003: language models are preliminary
572 graph reasoners on easy tasks while the benefit of in-context learning and advanced prompting is
573 limited on more complex problems. However, in the CoT and CoT+SC settings, CODE-DAVINCI-
574 002 generates too many tokens before reaching the final answer, so we do not present the results in
575 Table 9.

576 **E.2 OPT Results**

577 We fine-tune OPT-2.7B [Zhang et al., 2022b] and evaluate the model on the NLGraph benchmark.
578 We randomly divide the standard version of the NLGraph benchmark into 4902 training problems
579 and 1,000 test problems. We generate chain-of-thought solutions for the training problems, combine
580 the solutions with the problems, and attain nine corpus of text (*i.e.* one for each task and one that
581 concatenating all the text). We then fine-tune OPT on the nine corpus separately. However, we find
582 that fine-tuning did not elicit the graph reasoning abilities of OPT-2.7B: the generated text is mostly
583 meaningless, falling short of answering questions and following formats. We leave it to future work
584 on fine-tuning larger language models to elicit graph reasoning abilities due to our limitations in
585 computational resources.

586 **E.3 Qualitative Analysis with GPT-3.5-TURBO and GPT-4**

587 Due to monetary costs, we only conduct qualitative analysis on GPT-3.5-TURBO and GPT-4 with 19
588 representative problems across the eight tasks. We provide the results in Table 11, Table 12, Table
589 13, Table 14, Table 15. Results demonstrate that while GPT-4 does present better graph reasoning
590 abilities, it is still far from being a robust reasoner towards graph-based problems.

Table 11: Results of GPT-3.5-TURBO and GPT-4 on 19 cases (#1 to #6 example) through the eight tasks. We omitted the COT in-context exemplars in the table.

Question	GPT-3.5-TURBO	GPT-4
Determine if there is a path between two nodes in the graph. Note that (i,j) means that node i and node j are connected with an undirected edge. Graph: (0,4) (1,3) (1,2) (1,6) (2,5) (2,6) (3,5) (5,6) Q: Is there a path between node 4 and node 5?	Node 4 is not directly connected to node 5, but we can follow the path: 4->0->6->2->5, so the answer is yes. ✗	Node 4 is only connected to node 0, which does not have any connections to the other nodes in the graph. Node 5 is in the connected block consisting of nodes 1, 2, 3, 5, and 6. Node 4 and node 5 are not in the same connected block, so the answer is no. ✓
Determine if there is a path between two nodes in the graph. Note that (i,j) means that node i and node j are connected with an undirected edge. Graph: (0,3) (0,5) (0,11) (0,8) (0,6) (0,1) (1,10) (1,3) (1,5) (1,9) (1,6) (1,2) (2,10) (2,3) (2,5) (2,9) (2,8) (2,7) (2,6) (2,13) (3,9) (3,11) (3,8) (3,7) (4,12) (5,10) (5,9) (5,8) (5,7) (5,6) (6,10) (6,9) (6,11) (6,8) (7,10) (7,11) (7,8) (7,13) (8,10) (8,11) (8,13) (9,10) (9,11) (9,13) (10,11) (10,13) Q: Is there a path between node 6 and node 3?	Node 6 is connected to node 0, node 0 is connected to node 3. We can follow the path: 6->0->3, so the answer is yes. ✓	A: Node 6 is connected to node 2, node 2 is connected to node 3. We can follow the path: 6->2->3, so the answer is yes. ✓
In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge. The nodes are numbered from 0 to 7, and the edges are: (2,4) (0,3) (1,2) (5,7) (1,5) (6,5) (1,0) Q: Is there a cycle in this graph?	The edges (1,2), (2,4), (4,0), (0,3), (3,1) form a cycle, so yes, there is a cycle in this graph. ✗	The edges (1,0), (1,2), (1,5), (6,5), (5,7) do not form a cycle. However, the edges (1,0), (1,2), (2,4) do form a cycle. So yes, there is a cycle in this graph. ✗
In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge. The nodes are numbered from 0 to 7, and the edges are: (1,7) (0,6) (1,3) (4,5) (1,0) (0,4) (0,2) (7,2) Q: Is there a cycle in this graph?	The edges (0,6), (6,4), (4,5), (5,7), (7,1), (1,0) form a cycle, so yes, there is a cycle in this graph. ✗	The edges (1,7), (7,2), (2,0), (0,1) form a cycle, so yes, there is a cycle in this graph. ✓
In an undirected graph, the nodes are numbered from 0 to 5, and the edges are: An edge between node 2 and node 0 with weight 2, An edge between node 0 and node 4 with weight 1, An edge between node 1 and node 2 with weight 4, An edge between node 1 and node 3 with weight 4, An edge between node 4 and node 1 with weight 4, An edge between node 2 and node 3 with weight 1, An edge between node 2 and node 5 with weight 2, An edge between node 2 and node 4 with weight 3, An edge between node 5 and node 3 with weight 2, An edge between node 3 and node 4 with weight 3, An edge between node 4 and node 5 with weight 2. Q: Give the shortest path from node 3 to node 0.	All the paths from node 3 to node 0 are: 3,2,0 with a total weight of 2 + 2 = 4, 3,2,4,1,0 with a total weight of 3 + 4 + 1 = 8, 3,2,4,0 with a total weight of 3 + 1 = 4, 3,1,4,0 with a total weight of 4 + 4 + 1 = 9, 3,1,2,0 with a total weight of 4 + 4 + 2 = 10, 3,5,2,0 with a total weight of 2 + 2 + 2 = 6, 3,5,4,0 with a total weight of 2 + 2 + 1 = 5. The weight of path 3,2,0 and 3,5,4,0 is the smallest, so the shortest path from node 3 to node 0 is either 3,2,0 or 3,5,4,0, both with a total weight of 4. ✗	All the paths from node 3 to node 0 are: 3,2,0 with a total weight of 1 + 2 = 3, 3,1,2,0 with a total weight of 4 + 4 + 2 = 10, 3,4,0 with a total weight of 3 + 1 = 4, 3,4,2,0 with a total weight of 3 + 3 + 2 = 8, 3,5,2,0 with a total weight of 2 + 2 + 2 = 6, 3,5,4,0 with a total weight of 2 + 2 + 1 = 5. The weight of path 3,2,0 is the smallest, so the shortest path from node 3 to node 0 is 3,2,0 with a total weight of 3. ✓
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are: An edge between node 1 and node 0 with weight 1, An edge between node 2 and node 0 with weight 1, An edge between node 4 and node 1 with weight 1, An edge between node 1 and node 2 with weight 3, An edge between node 2 and node 3 with weight 3. Q: Give the shortest path from node 3 to node 1.	All the paths from node 3 to node 1 are: 3,2,1 with a total weight of 3 + 3 = 6, 3,2,0,1 with a total weight of 1 + 1 + 1 = 3, 3,2,1,0,1 with a total weight of 3 + 1 + 1 + 1 = 6, 3,2,0,1,4,1 with a total weight of 1 + 1 + 1 + 1 + 1 = 5. The weight of path 3,2,0,1 is the smallest, so the shortest path from node 3 to node 1 is 3,2,0,1 with a total weight of 3. ✗	All the paths from node 3 to node 1 are: 3,2,1 with a total weight of 3 + 3 = 6, 3,2,0,1 with a total weight of 3 + 1 + 1 = 5. The weight of path 3,2,0,1 is the smallest, so the shortest path from node 3 to node 1 is 3,2,0,1 with a total weight of 5. ✓

Table 12: Results of GPT-3.5-TURBO and GPT-4 on 19 cases (#7 to #9 example) through the eight tasks. We omitted the CoT in-context exemplars in the table.

Question	GPT-3.5-TURBO	GPT-4
<p>In an undirected graph, the nodes are numbered from 0 to 6, and the edges are: An edge between node 0 and node 2 with weight 1, An edge between node 3 and node 0 with weight 4, An edge between node 5 and node 0 with weight 2, An edge between node 4 and node 0 with weight 3, An edge between node 0 and node 1 with weight 4, An edge between node 0 and node 6 with weight 2, An edge between node 2 and node 1 with weight 2, An edge between node 3 and node 1 with weight 3, An edge between node 4 and node 1 with weight 3, An edge between node 1 and node 6 with weight 2, An edge between node 2 and node 5 with weight 4, An edge between node 2 and node 4 with weight 3, An edge between node 6 and node 2 with weight 2, An edge between node 3 and node 5 with weight 1, An edge between node 4 and node 3 with weight 2, An edge between node 3 and node 6 with weight 4, An edge between node 4 and node 5 with weight 3, An edge between node 4 and node 6 with weight 1, An edge between node 6 and node 5 with weight 2.</p> <p>Q: Give the shortest path from node 2 to node 3.</p>	<p>All the paths from node 2 to node 3 are: 2,0,3 with a total weight of $1 + 4 = 5$, 2,1,3 with a total weight of $2 + 3 = 5$, 2,4,3 with a total weight of $3 + 2 = 5$. The weight of path 2,0,3 is the smallest, so the shortest path from node 2 to node 3 is 2,0,3 with a total weight of 5. ✗</p>	<p>All the paths from node 2 to node 3 are: 2,0,3 with a total weight of $1 + 4 = 5$, 2,1,3 with a total weight of $2 + 3 = 5$, 2,1,0,3 with a total weight of $2 + 4 + 4 = 10$, 2,1,6,3 with a total weight of $2 + 2 + 4 = 8$, 2,4,3 with a total weight of $3 + 2 = 5$, 2,4,0,3 with a total weight of $3 + 3 + 4 = 10$, 2,4,1,3 with a total weight of $3 + 3 + 3 = 9$, 2,4,1,0,3 with a total weight of $3 + 3 + 4 + 4 = 14$, 2,4,1,6,3 with a total weight of $3 + 3 + 2 + 4 = 12$, 2,5,0,3 with a total weight of $4 + 2 + 4 = 10$, 2,5,4,3 with a total weight of $4 + 5 + 2 = 11$, 2,5,4,0,3 with a total weight of $4 + 5 + 3 + 4 = 16$, 2,5,4,1,3 with a total weight of $4 + 5 + 3 + 3 = 15$, 2,5,6,3 with a total weight of $4 + 6 + 4 = 14$, 2,6,1,3 with a total weight of $2 + 2 + 3 = 7$, 2,6,1,0,3 with a total weight of $2 + 2 + 4 + 4 = 12$, 2,6,5,0,3 with a total weight of $2 + 5 + 2 + 4 = 13$, 2,6,5,4,3 with a total weight of $2 + 5 + 5 + 2 = 14$. The weight of paths 2,0,3, 2,1,3, and 2,4,3 is the smallest. So, there are three shortest paths from node 2 to node 3, which are 2,0,3; 2,1,3; and 2,4,3, each with a total weight of 5. ✗</p>
<p>In a directed graph with 5 nodes numbered from 0 to 4: node 2 should be visited before node 4 node 2 should be visited before node 1 node 2 should be visited before node 0 node 4 should be visited before node 1 node 4 should be visited before node 0</p> <p>Q: Can all the nodes be visited? Give the solution.</p>	<p>This is not possible as there is a cycle between nodes 0, 1, and 4. Node 2 cannot be visited before nodes 0, 1, and 4, and nodes 0, 1, and 4 cannot be visited before node 2. Therefore, it is not possible to visit all the nodes in this graph. ✗</p>	<p>Since no nodes are required to be visited before node 2 and node 3, node 2 and node 3 can be visited first. Since visiting node 4 requires visiting node 2 first, after node 2 is visited, node 4 can be visited. Since visiting node 0 requires visiting node 2, node 4 first, after node 2 and node 4 are visited, node 0 can be visited. Since visiting node 1 requires visiting node 2, node 4 first, after node 2 and node 4 are visited, node 1 can be visited. So the solution is: 2,3,4,0,1. ✓</p>
<p>In a directed graph with 5 nodes numbered from 0 to 4: node 0 should be visited before node 1 node 1 should be visited before node 2 node 3 should be visited before node 1 node 3 should be visited before node 2 node 4 should be visited before node 3 node 4 should be visited before node 0 node 4 should be visited before node 1</p> <p>Q: Can all the nodes be visited? Give the solution.</p>	<p>This graph has a cycle, which means it is not possible to visit all nodes in a valid order. Specifically, there is a cycle between nodes 1, 2, and 3, which cannot be resolved based on the given constraints. ✗</p>	<p>Since no nodes are required to be visited before node 4, node 4 can be visited first. Since visiting node 0 and node 3 requires visiting node 4 first, after node 4 is visited, node 0 and node 3 can be visited. Since visiting node 1 requires visiting node 0 and node 3 first, after node 0 and node 3 are visited, node 1 can be visited. Since visiting node 2 requires visiting node 1 and node 3 first, after node 1 and node 3 are visited, node 2 can be visited. So the solution is: 4,0,3,1,2. ✓</p>

Table 13: Results of GPT-3.5-TURBO and GPT-4 on 19 cases (#10 to #12 example) through the eight tasks. We omitted the COT in-context exemplars in the table.

Question	GPT-3.5-TURBO	GPT-4
<p>In a directed graph with 6 nodes numbered from 0 to 5: node 0 should be visited before node 5 node 0 should be visited before node 1 node 2 should be visited before node 1 node 3 should be visited before node 0 node 3 should be visited before node 5 node 3 should be visited before node 1 node 4 should be visited before node 0 node 4 should be visited before node 2 node 5 should be visited before node 2 Q: Can all the nodes be visited? Give the solution</p>	<p>Since no nodes are required to be visited before node 3 and node 4, node 3 and node 4 can be visited first. Since visiting node 0 requires visiting node 3 first, after node 3 is visited, node 0 can be visited. Since visiting node 1 requires visiting node 0, node 2, node 3 first, after node 0, node 2, and node 3 are visited, node 1 can be visited. Since visiting node 5 requires visiting node 0, node 3 first, after node 0 and node 3 are visited, node 5 can be visited. Since visiting node 2 requires visiting node 4, node 5 first, after node 4 and node 5 are visited, node 2 can be visited. So the solution is: 3,4,0,2,1,5. ✗</p>	<p>Since no nodes are required to be visited before node 3 and node 4, node 3 and node 4 can be visited first. Since visiting node 0 requires visiting node 3 and node 4 first, after node 3 and node 4 are visited, node 0 can be visited. Since visiting node 5 requires visiting node 0 and node 3 first, after node 0 and node 3 are visited, node 5 can be visited. Since visiting node 1 requires visiting node 0, node 2, and node 3 first, after node 0, node 2, and node 3 are visited, node 1 must be visited before node 1. Since visiting node 2 requires visiting node 4 and node 5 first, after node 4 and node 5 are visited, node 2 can be visited. Now, since node 0, node 2, and node 3 have been visited, node 1 can be visited. So the solution is: 3,4,0,5,2,1. ✓</p>
<p>In a directed graph, the nodes are numbered from 0 to 5, and the edges are: an edge from node 1 to node 3 with capacity 4, an edge from node 1 to node 0 with capacity 9, an edge from node 2 to node 1 with capacity 4, an edge from node 3 to node 1 with capacity 4, an edge from node 4 to node 3 with capacity 1, an edge from node 4 to node 5 with capacity 2, an edge from node 5 to node 1 with capacity 4, an edge from node 5 to node 0 with capacity 7. Q: What is the maximum flow from node 4 to node 0?</p>	<p>There is no path from node 4 to node 0, so the maximum flow from node 4 to node 0 is 0. ✗</p>	<p>From the source (node 4), we can send 1 unit of flow to node 3, and 2 units of flow to node 5. 1 unit of flow arrives at node 3, then we can send 1 unit of flow from node 3 to node 1. 2 units of flow arrive at node 5, then we can send 2 units of flow from node 5 to node 0. 1 unit of flow arrives at node 1, then we can send 1 unit of flow from node 1 to node 0. The total unit of flow arrives at node 0 is 3. So the maximum flow from node 4 to node 0 is 3. ✓</p>
<p>In a directed graph, the nodes are numbered from 0 to 7, and the edges are: an edge from node 0 to node 6 with capacity 1, an edge from node 0 to node 7 with capacity 10, an edge from node 2 to node 1 with capacity 9, an edge from node 2 to node 7 with capacity 8, an edge from node 3 to node 1 with capacity 9, an edge from node 3 to node 6 with capacity 10, an edge from node 3 to node 4 with capacity 7, an edge from node 4 to node 6 with capacity 1, an edge from node 5 to node 3 with capacity 7, an edge from node 5 to node 2 with capacity 9, an edge from node 6 to node 4 with capacity 1, an edge from node 7 to node 5 with capacity 8, an edge from node 7 to node 6 with capacity 5. Q: What is the maximum flow from node 2 to node 4?</p>	<p>From the source (node 2), we can send 8 units of flow to node 7, and 9 units of flow to node 1. 8 units of flow arrive at node 7, then we can send 5 units of flow from node 7 to node 6, and 3 units of flow from node 7 to node 5. 9 units of flow arrive at node 1, then we can send 9 units of flow from node 3 to node 1. 9 units of flow arrive at node 6, then we can send 1 unit of flow from node 6 to node 4. 7 units of flow arrive at node 3, then we can send 7 units of flow from node 3 to node 4. The total unit of flow arrives at node 4 is 8. So the maximum flow from node 2 to node 4 is 8. ✓</p>	<p>From the source (node 2), we can send 9 units of flow to node 1, and 8 units of flow to node 7. 9 units of flow arrive at node 1, then we can send 9 units of flow from node 1 to node 3. 8 units of flow arrive at node 7, then we can send 8 units of flow from node 7 to node 5, and 5 units of flow from node 7 to node 6. 8 units of flow arrive at node 5, then we can send 7 units of flow from node 5 to node 3, and 1 unit of flow from node 5 to node 2. 14 units of flow arrive at node 3, then we can send 14 units of flow from node 3 to node 4. The total unit of flow arrives at node 4 is 14. So the maximum flow from node 2 to node 4 is 14. ✗</p>

Table 14: Results of GPT-3.5-TURBO and GPT-4 on 19 cases (#13 to #16 example) through the eight tasks. We omitted the COT in-context exemplars in the table.

Question	GPT-3.5-TURBO	GPT-4
<p>In a directed graph, the nodes are numbered from 0 to 7, and the edges are: an edge from node 0 to node 6 with capacity 9, an edge from node 0 to node 3 with capacity 7, an edge from node 1 to node 7 with capacity 8, an edge from node 1 to node 2 with capacity 3, an edge from node 2 to node 7 with capacity 7, an edge from node 2 to node 4 with capacity 5, an edge from node 3 to node 5 with capacity 7, an edge from node 3 to node 1 with capacity 8, an edge from node 4 to node 0 with capacity 3, an edge from node 5 to node 7 with capacity 5, an edge from node 6 to node 5 with capacity 6, an edge from node 6 to node 7 with capacity 6, an edge from node 6 to node 0 with capacity 9, an edge from node 7 to node 4 with capacity 4, an edge from node 7 to node 1 with capacity 8. Q: What is the maximum flow from node 1 to node 6?</p>	<p>There is no path from node 1 to node 6 in this graph, so the maximum flow from node 1 to node 6 is 0. ✗</p>	<p>From the source (node 1), we can send 8 units of flow to node 7, and 3 units of flow to node 2. 8 units of flow arrive at node 7, then we can send 4 units of flow from node 7 to node 4, and 4 units of flow from node 7 to node 1. 4 units of flow arrive at node 4, then we can send 3 units of flow from node 4 to node 0. 3 units of flow arrive at node 0, then we can send 3 units of flow from node 0 to node 6. 3 units of flow arrive at node 2, then we can send 3 units of flow from node 2 to node 4. 3 units of flow arrive at node 4, then we can send 3 units of flow from node 4 to node 0. 3 units of flow arrive at node 0, then we can send 3 units of flow from node 0 to node 6. So the maximum flow from node 1 to node 6 is 6. ✗</p>
<p>In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge. The nodes are numbered from 0 to 6, and the edges are: (4,3) (6,5) (5,1) (3,0) (2,4) (1,0) (5,2) (4,0) (2,3) (0,2) (6,2) Q: Is there a path in this graph that visits every node exactly once? If yes, give the path. Note that in a path, adjacent nodes must be connected with edges.</p>	<p>No, there is no path that visits every node exactly once. This graph contains two disconnected components: 0,1,2,3,4 and 5,6. It is not possible to visit all nodes in a single path because there is no edge connecting the two components. ✗</p>	<p>Yes. We can start at node 3. As node 3 is connected with node 0, and node 0 is not visited, we can then visit node 0. As node 0 is connected with node 1, and node 1 is not visited, we can then visit node 1. As node 1 is connected with node 5, and node 5 is not visited, we can then visit node 5. As node 5 is connected with node 6, and node 6 is not visited, we can then visit node 6. As node 6 is connected with node 2, and node 2 is not visited, we can then visit node 2. As node 2 is connected with node 4, and node 4 is not visited, we can then visit node 4. The path can be: 3, 0, 1, 5, 6, 2, 4. ✓</p>
<p>In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge. The nodes are numbered from 0 to 4, and the edges are: (1,3) (1,2) (4,3) (2,0) (0,4) (0,1) Q: Is there a path in this graph that visits every node exactly once? If yes, give the path. Note that in a path, adjacent nodes must be connected with edges.</p>	<p>No, there is no path in this graph that visits every node exactly once. This is because node 1 has degree 2, meaning it is connected to two other nodes. In order to visit every node exactly once, we would need to start and end at node 1, but once we leave node 1, we cannot return to it without visiting another node twice. Therefore, it is impossible to find a path that visits every node exactly once in this graph. ✗</p>	<p>Yes. We can start at node 0. As node 0 is connected with node 1, and node 1 is not visited, we can then visit node 1. As node 1 is connected with node 3, and node 3 is not visited, we can then visit node 3. As node 3 is connected with node 4, and node 4 is not visited, we can then visit node 4. As node 4 is connected with node 2, and node 2 is not visited, we can then visit node 2. The path can be: 0,1,3,4,2. ✗</p>
<p>There are 4 job applicants numbered from 0 to 3, and 2 jobs numbered from 0 to 1. Each applicant is interested in some of the jobs. Each job can only accept one applicant and a job applicant can be appointed for only one job. Applicant 0 is interested in job 1. Applicant 2 is interested in job 0. Applicant 0 is interested in job 0. Applicant 2 is interested in job 1. Applicant 3 is interested in job 1. Applicant 3 is interested in job 0. Applicant 1 is interested in job 0. Q: Find an assignment of jobs to applicants in such that the maximum number of applicants find the job they are interested in.</p>	<p>Applicant 0 is interested in job 0, job 1. Applicant 1 is interested in job 0. Applicant 2 is interested in job 0, job 1. Applicant 3 is interested in job 0, job 1. We can make the following assignments: applicant 0: job 1 applicant 2: job 0 applicant 3: job 1 3 applicants can find the job they are interested in. ✗</p>	<p>Applicant 0 is interested in job 0, job 1. Applicant 1 is interested in job 0. Applicant 2 is interested in job 0, job 1. Applicant 3 is interested in job 0, job 1. We can make the following assignments: applicant 0: job 0 applicant 1: - (no job available) applicant 2: job 1 applicant 3: - (no job available) 2 applicants can find the job they are interested in. ✓</p>

Table 15: Results of GPT-3.5-TURBO and GPT-4 on 19 cases (#17 to #19 example) through the eight tasks. We omitted the CoT in-context exemplars in the table.

Question	GPT-3.5-TURBO	GPT-4
<p>There are 5 job applicants numbered from 0 to 4, and 6 jobs numbered from 0 to 5. Each applicant is interested in some of the jobs. Each job can only accept one applicant and a job applicant can be appointed for only one job.</p> <p>Applicant 0 is interested in job 3. Applicant 4 is interested in job 5. Applicant 4 is interested in job 4. Applicant 3 is interested in job 0. Applicant 0 is interested in job 5. Applicant 1 is interested in job 0. Applicant 0 is interested in job 2. Applicant 0 is interested in job 4. Applicant 4 is interested in job 0. Applicant 4 is interested in job 2. Applicant 2 is interested in job 3.</p> <p>Q: Find an assignment of jobs to applicants in such that the maximum number of applicants find the job they are interested in.</p>	<p>Applicant 0 is interested in job 2, job 3, job 4, job 5. Applicant 1 is interested in job 0. Applicant 2 is interested in job 3. Applicant 3 is interested in job 0. Applicant 4 is interested in job 0, job 2, job 4, job 5. We can make the following assignments: applicant 0: job 5 applicant 1: job 0 applicant 2: job 3 applicant 3: job 0 applicant 4: job 2 5 applicants can find the job they are interested in. ✗</p>	<p>Applicant 0 is interested in job 2, job 3, job 4, job 5. Applicant 1 is interested in job 0. Applicant 2 is interested in job 3. Applicant 3 is interested in job 0. Applicant 4 is interested in job 0, job 2, job 4, job 5. We can make the following assignments: applicant 0: job 3 applicant 1: job 0 applicant 4: job 4 3 applicants can find the job they are interested in. ✗</p>
<p>n an undirected graph, the nodes are numbered from 0 to 5, and every node has an embedding. (i,j) means that node i and node j are connected with an undirected edge.</p> <p>Embeddings: node 0: [1,1] node 1: [0,1] node 2: [1,0] node 3: [1,1] node 4: [0,0] node 5: [0,1]</p> <p>The edges are: (0,1) (5,0) (1,4) (1,5) (2,4) (4,3)</p> <p>In a simple graph convolution layer, each node's embedding is updated by the sum of its neighbors' embeddings.</p> <p>Q: What's the embedding of each node after one layer of simple graph convolution layer?</p>	<p>Layer 1: node 0: sum of embeddings of node 1, node 5, [0,1] + [0,1] = [0,2]. node 1: sum of embeddings of node 0, node 4, node 5, [1,1] + [0,0] + [0,1] = [1,2]. node 2: sum of embeddings of node 4, [0,0] = [0,0]. node 3: sum of embeddings of node 4, [0,0] = [0,0]. node 4: sum of embeddings of node 1, node 2, node 3, [1,2] + [1,0] + [0,0] = [2,2]. node 5: sum of embeddings of node 0, node 1, [0,2] + [1,2] = [1,4]. So the answer is: node 0: [0,2] node 1: [1,2] node 2: [0,0] node 3: [0,0] node 4: [2,2] node 5: [1,4] ✗</p>	<p>Layer 1: node 0: sum of embeddings of node 1, node 5, [0,1] + [0,1] = [0,2]. node 1: sum of embeddings of node 0, node 4, node 5, [1,1] + [0,0] + [0,1] = [1,2]. node 2: sum of embeddings of node 4, [0,0] = [0,0]. node 3: sum of embeddings of node 4, [0,0] = [0,0]. node 4: sum of embeddings of node 1, node 2, node 3, [0,1] + [1,0] + [1,1] = [2,2]. node 5: sum of embeddings of node 0, node 1, [1,1] + [0,1] = [1,2]. So the answer is: node 0: [0,2] node 1: [1,2] node 2: [0,0] node 3: [0,0] node 4: [2,2] node 5: [1,2] ✓</p>
<p>In an undirected graph, the nodes are numbered from 0 to 7, and every node has an embedding. (i,j) means that node i and node j are connected with an undirected edge.</p> <p>Embeddings: node 0: [0,1] node 1: [1,1] node 2: [1,0] node 3: [1,1] node 4: [1,1] node 5: [0,0] node 6: [1,0] node 7: [1,0]</p> <p>The edges are: (0,3) (4,0) (6,0) (5,1) (1,4) (5,2) (2,3) (2,6) (3,5) (3,6) (5,4) (7,5)</p> <p>In a simple graph convolution layer, each node's embedding is updated by the sum of its neighbors' embeddings.</p> <p>Q: What's the embedding of each node after one layer of simple graph convolution layer?</p>	<p>Layer 1: node 0: sum of embeddings of node 3, node 4, node 6, [1,1] + [1,1] + [1,0] = [3,2]. node 1: sum of embeddings of node 4, [1,1] = [1,1]. node 2: sum of embeddings of node 3, node 5, node 6, [1,1] + [0,0] + [1,0] = [2,1]. node 3: sum of embeddings of node 0, node 2, node 5, node 6, [0,1] + [1,1] + [0,0] + [1,0] = [2,2]. node 4: sum of embeddings of node 0, node 1, node 5, [3,2] + [1,1] + [0,0] = [4,3]. node 5: sum of embeddings of node 2, node 3, node 4, node 7, [2,1] + [2,2] + [4,3] + [1,0] = [9,6]. node 6: sum of embeddings of node 0, node 2, node 3, [3,2] + [1,1] + [1,0] = [5,3]. node 7: sum of embeddings of node 5, [9,6] = [9,6]. So the answer is: node 0: [3,2] node 1: [1,1] node 2: [2,1] node 3: [2,2] node 4: [4,3] node 5: [9,6] node 6: [5,3] node 7: [9,6] ✗</p>	<p>Layer 1: node 0: sum of embeddings of node 3, node 4, node 6, [1,1] + [1,1] + [1,0] = [3,2]. node 1: sum of embeddings of node 5, node 4, [0,0] + [1,1] = [1,1]. node 2: sum of embeddings of node 5, node 3, node 6, [0,0] + [1,1] + [1,0] = [2,1]. node 3: sum of embeddings of node 0, node 2, node 5, node 6, [0,1] + [1,0] + [0,0] + [1,0] = [2,1]. node 4: sum of embeddings of node 0, node 1, node 5, [0,1] + [1,1] + [0,0] = [1,2]. node 5: sum of embeddings of node 1, node 2, node 3, node 4, node 7, [1,1] + [1,0] + [1,1] + [1,1] + [1,0] = [5,3]. node 6: sum of embeddings of node 0, node 2, node 3, node 7, [0,1] + [1,0] + [1,1] + [1,0] = [3,2]. node 7: sum of embeddings of node 5, [0,0] = [0,0]. So the answer is: node 0: [3,2] node 1: [1,1] node 2: [2,1] node 3: [2,1] node 4: [1,2] node 5: [5,3] node 6: [3,2] node 7: [0,0] ✗</p>