	ZS	FS	CoT	*Reprompting*
BBH				
Date Understanding	63.6	46.4	**76.8**	76.4
Formal Fallacies	49.2	53.6	48.4	**56.8**
Movie Recommendation	59.2	72.4	25.6	**78.4**
Reasoning About Colored Objects	66.8	48.8	**76.0**	74.0
Ruin Names	53.2	66.8	60.8	**74.8**
Salient Translation Error Detection	43.2	53.2	32.8	**54.8**
Word Sorting	58.0	72.0	46.0	73.2
GSM8K	45.6	26.5	75.6	**79.5**
MATH				
Number Theory	24.1	13.5	26.5	**28.5**
Algebra	37.3	17.4	**46.8**	43.0
Average	50.0	47.1	51.5	**63.9**

Table 3: Performance of ChatGPT using *Reprompting* versus *ZS* (zero-shot), *FS* (few-shot), and *CoT* prompting methods on seven additional tasks from Big-Bench Hard (BBH) (Suzgun et al., 2022), GSM8K (Cobbe et al., 2021) and two subtasks from MATH (Hendrycks et al., 2021).

	CoT	Complex-CoT	*Reprompting*
Penguins in a Table	67.1	76.7	**85.6**
Date Understanding	**76.8**	76.4	76.4
GSM8K	75.6	72.6	**79.5**
Average	73.2	75.2	**80.5**

Table 4: Performance of ChatGPT using *Reprompting* versus *CoT* and Complex-CoT (Fu et al., 2022) prompting methods on Penguins in a Table, Date Understanding from Big-Bench Hard (BBH) (Suzgun et al., 2022), and GSM8K (Cobbe et al., 2021).

A Additional Evaluation Results

We further evaluate *Reprompting* on seven additional tasks from Big-Bench Hard (BBH) (Suzgun et al., 2022), GSM8K (Cobbe et al., 2021) and two subtasks from MATH (Hendrycks et al., 2021). On BBH, we intentionally selected the tasks on which CoT does not improve much or does not improve consistently over zero-shot prompting, such as Formal Fallacies, Movie Recommendation, Ruin Names, Salient Translation Error Detection, and Word Sorting. As shown in Table 3, *Reprompting* still outperforms zero-shot and few-shot prompting consistently and substantially by 14-17 points on average. Compared with CoT, *Reprompting* achieves better performance on all tasks except on Date Understanding, Reasoning About Colored Objects, and MATH Algebra, on which the score differences are very small (< 4 point). On average, *Reprompting* outperforms CoT by +12.4 point.

Interestingly, on tasks where CoT even underperforms zero-shot prompting, such as Movie Recommendation, Salient Translation Error Detection, and Word Sorting, *Reprompting* still improves over zero-shot prompting by large margins. This suggests that not all CoT recipes improve model performance, and some may even lead to degradation. This further emphasizes the need for algorithms like *Reprompting* for discovering and optimizing the CoT prompt to best exploit and compare LLMs.

B Comparison with Complex-CoT

We also compare *Reprompting* with Complex-CoT, a complexity-based CoT prompt selection method (Fu et al., 2022) on three popular benchmarks for commonsense and arithmetic reasoning. For Complex-CoT, we use the complex CoT prompt released in Fu et al. (2022) on ChatGPT. Note that Complex-CoT is built on top of human-written CoT solutions, while *Reprompting* does not rely on human-written CoT. As shown in Table 4, *Reprompting* achieves competitive or better

	$p_{rej} = 0$	$p_{rej} = 0$	No Recombination	Standard *Reprompting*
Logical Deduction	56.3	61.9	54.7	**66.3**
Object Counting	52.0	**97.2**	95.6	**97.2**
Temporal Sequences	74.8	74.4	90.4	**93.2**
Average	61.0	77.8	80.2	**85.6**

Table 5: Ablation study on rejection sampling (including no rejection ($p_{ref} = 0$) and always rejecting ($p_{ref} = 1$)) and recombination on Logical Deduction, Object Counting, and Temporal Sequences from Big-Bench Hard (BBH) (Suzgun et al., 2022).

performance than Complex-CoT on all tasks. On average, *Reprompting* outperforms Complex-CoT by +5 point.

C Ablation Study

We further conducted an ablation study on the rejection sampling and recombination process. Results in Table 5 show that, without rejection sampling, the test performance degrades substantially by 25 point on average. Always rejecting solutions that lead to incorrect answers also causes a degradation of 8 point. Additionally, not allowing multiple solutions to be recombined when sampling new solutions at the iterative sampling stage also hurts performance.

D Additional Illustrations

On sensitivity to initialization We have shown that *Reprompting* can be sensitive to initial zero-shot recipe generation. In each task we tested, armed with a suitable prompt InstructGPT could reach test set accuracy equalling or besting ChatGPT. However, such a prompt could not be discovered if the prompt recombination and evolution through *Reprompting* was started with initial prompts generated by InstructGPT itself. Fig. D.1 points to a likely explanation: ChatGPT can generate a wider range of useful recipes, and whether these initial recipes lead to the correct solution or not, InstructGPT can follow them and, through *Reprompting*, refine them and correct them. Thus, as we have shown in our experiments, with good initialization, LLMs that may appear inferior based on their zero-shot performance may end up performing just as well or better than LLMs whose zero-shot performance is more encouraging. It would be interesting to see if *Reprompting* can use other LLMs in initialization to perform even better, or if the humans can be put back into the loop to provide some initial recipes, or some generic instructions on how to generate them.

On transferability of discovered recipes The fact that LLM_1 (ChatGPT) can point LLM_2 (InstructGPT) in the right direction(s) for prompt discovery does not mean that the discovered prompts, having been optimized for training performance on LLM_2 will perform well when used to prompt LLM_1. In fact, Table 2 in the main text indicates that the discovered CoT recipes that work for one model may not necessarily work for other models. For example, in the case of *Temporal Sequences*, the best performance is achieved with a prompt trained with InstructGPT (after initialization with ChatGPT as LLM_1). But when using that prompt in testing with ChatGPT the performance is by 18% lower. Figure D.2 illustrates the solution strategy that emerged from training: The time intervals that need to be reasoned over are sorted, and among the sorted list, the missing interval was inserted as the possible interval when the person in question **could have** performed an activity. Then the answer is generated. InstructGPT follows this procedure with accuracy over 99%, but ChatGPT sometimes skips the generation of that crucial line (for this recipe) with the missing interval within the timeline. Therefore, among initial "ideas" from ChatGPT, some can be refined to work well for InstructGPT, and others can be refined to work well for ChatGPT itself, as the best performance of ChatGPT (using the CoT prompt optimized for itself) is only slightly lower than that of the ChatGPT+InstructGPT combination.

These results suggest that fair comparison between different LLMs may be difficult, as one needs to optimize the CoT prompt for each model, and that optimization is typically non-trivial.

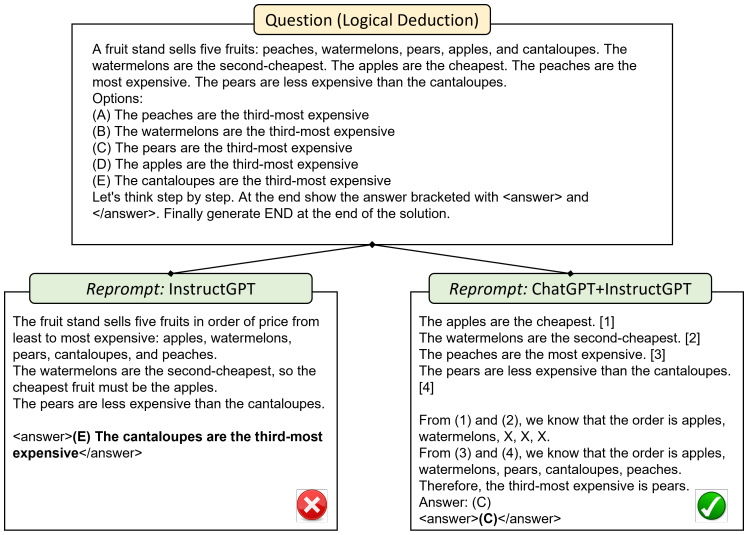


Figure D.1: Comparing the CoT recipes inferred through *Reprompting* using IntructGPT alone versus ChatGPT (for initialization) + InstructGPT (for sampling).

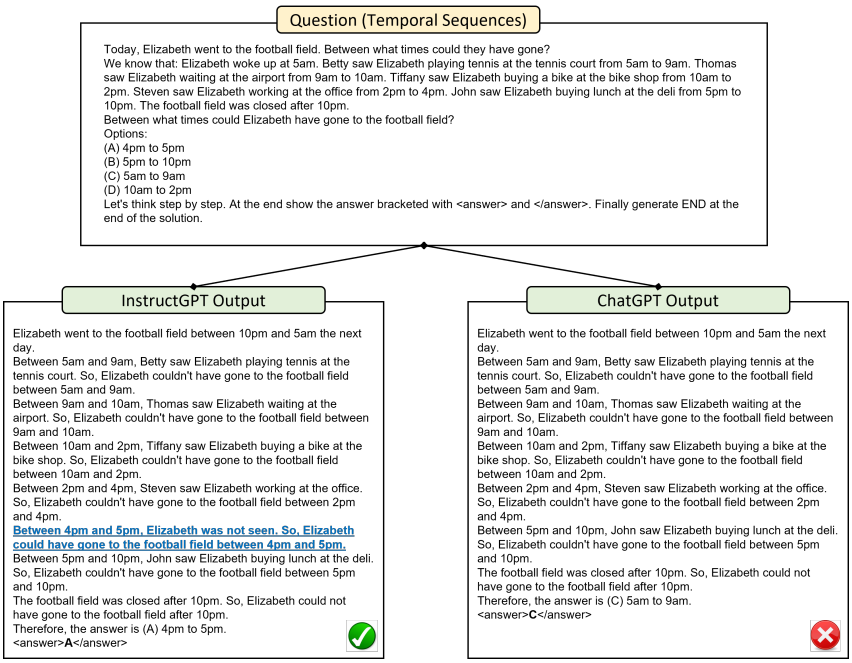
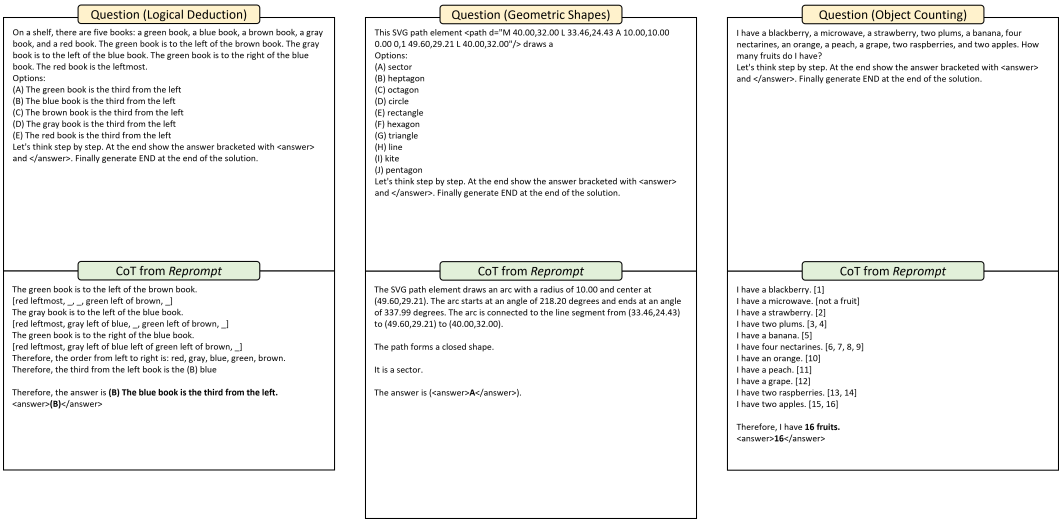
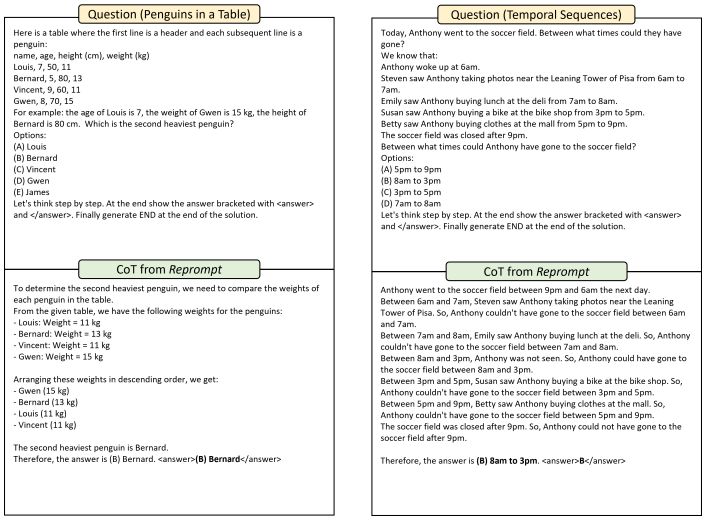


Figure D.2: An example on *Temporal Sequences* where ChatGPT underperforms InstructGPT using the same CoT prompt optimized for InstructGPT via *Reprompting* (using ChatGPT+InstructGPT). ChatGPT fails to correctly execute the recipe as it skips a key step (the blue underlined text from InstructGPT) to reach the final answer. (The illustration does not show the full prompt that precedes the puzzle x for brevity; it consists of 5 training examples with worked-out solutions that all follow the same strategy of solving these types of problems.)

How do the model-generated CoT recipes differ from human-written ones? In the main text, We evaluated the performance of the CoT recipes generated using *Reprompting* and contrasted it with human-written ones in Suzgun et al. (2022). As illustrated by the example recipes in Figure D.3, the model-generated CoT recipes share some similarities to human-written ones on some tasks (such as *Logical Deduction*), but differs on other tasks. For instance, on *Object Counting*, the CoT generated using *Reprompting* computes the total number of objects by incrementing the count one by one (e.g.



(a)



(b)

Figure D.3: Examples of the best-performing CoT recipes inferred via *Reprompting* on *Logical Deduction* (best score: 66.3), *Geometric Shapes* (best score: 72.8), *Object Counting* (best score: 99.6), *Penguins in a Table* (best score: 85.6), *Temporal Sequences* (best score: 99.2), and *Causal Judgement* (best score: 68.4).

adding 4 to the count 5 by "$[6, 7, 8, 9]$"), while in the human written recipe, it computes the addition through an arithmetic formula at the end.