AlgoPuzzleVQA: Diagnosing Multimodal Reasoning Challenges of Language Models with Algorithmic Multimodal Puzzles

Deepanway Ghosal[1*], **Vernon Y.H. Toh**[1*], **Yew Ken Chia**[1], **Soujanya Poria**[1]
[1] Singapore University of Technology and Design

: https://github.com/declare-lab/LLM-PuzzleTest
🌐: https://algopuzzlevqa.github.io/

Abstract

This paper introduces the novel task of multimodal puzzle solving, framed within the context of visual question-answering. We present a new dataset, AlgoPuzzleVQA designed to challenge and evaluate the capabilities of multimodal language models in solving algorithmic puzzles that necessitate both visual understanding, language understanding, and complex algorithmic reasoning. We create the puzzles to encompass a diverse array of mathematical and algorithmic topics such as boolean logic, combinatorics, graph theory, optimization, search, etc., aiming to evaluate the gap between visual data interpretation and algorithmic problem-solving skills. The dataset is generated automatically from code authored by humans. All our puzzles have exact solutions that can be found from the algorithm without tedious human calculations. It ensures that our dataset can be scaled up arbitrarily in terms of reasoning complexity and dataset size. Our investigation reveals that multimodal language models such as GPT4V and Gemini exhibit limited performance in puzzle-solving tasks. We find that their performance is near random in a multi-choice question-answering setup for a significant number of puzzles. The findings emphasize the challenges of integrating visual, language, and algorithmic knowledge for solving complex reasoning problems.

1 Introduction

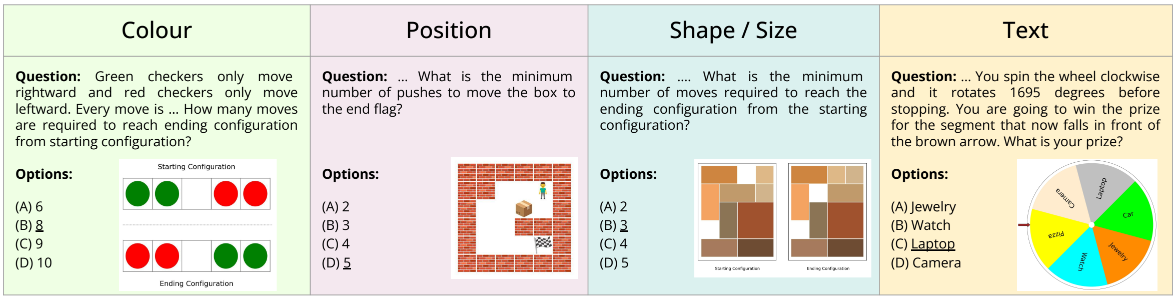
Puzzles have long been a source of fascination and intellectual challenge, serving both as entertaining pastimes and as means to advance mathematical and logical understanding. Historically, puzzles have often emerged from complex problems, with their solutions contributing valuable insights to mathematical research (Euler, 1741; Wiles, 1995; Knuth, 2000). In this paper, we explore the novel task of algorithmic visual puzzle solving. Traditional visual question-answering (VQA) (Antol et al., 2015; Goyal et al., 2017) and visual reasoning (Hudson and Manning, 2019) datasets have mainly focused on combining language and vision through the lens of object detection, scene recognition, and spatial relationship, where the answer can be directly found in the image. Another line of work has explored the application of external factual knowledge (Sanket Shah and Talukdar, 2019) or commonsense world knowledge (Schwenk et al., 2022) for VQA. Recently, multimodal reasoning benchmarks such as ScienceQA (Lu et al., 2022), and MMMU (Yue et al., 2023) have been proposed that aim to evaluate the expert knowledge of LLMs through subject-specific questions spanning science, medicine, business, arts, etc.

Unlike previous research, our work integrates a new critical component – algorithmic reasoning, which demands the ability to apply logical rules to novel situations, rather than relying on pre-stored information or direct recall of facts. This distinction is vital as it shifts the focus from merely memorizing vast databases of knowledge to applying concrete reasoning steps for novel problem-solving. We thus propose to benchmark the problem-solving ability of AI models through visual puzzles. Our puzzles are created to be self-contained where minimal additional knowledge (beyond basic mathematics) is required to solve them. This helps us to disentangle the *knowing* part from the *solving* part, as the knowledge required to solve the problem is already provided as context.

Consider Figure 1 as an example of some of our puzzles. We show the setting or the configuration of the puzzle/problem as an image, which we consider as the **visual context**. Depending on the problem, the visual context may show different kinds of information such as the initial and final

[*] Equal contribution.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 9615–9632
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics



Question: Green checkers only move rightward and red checkers only move leftward. Every move is ... How many moves are required to reach ending configuration from starting configuration?

Options:

(A) 6
(B) 8
(C) 9
(D) 10

Question: ... What is the minimum number of pushes to move the box to the end flag?

Options:

(A) 2
(B) 3
(C) 4
(D) 5

Question: .... What is the minimum number of moves required to reach the ending configuration from the starting configuration?

Options:

(A) 2
(B) 3
(C) 4
(D) 5

Question: ... You spin the wheel clockwise and it rotates 1695 degrees before stopping. You are going to win the prize for the segment that now falls in front of the brown arrow. What is your prize?

Options:

(A) Jewelry
(B) Watch
(C) Laptop
(D) Camera

Figure 1: Examples of puzzles in ALGOPUZZLEVQA based on **visual features**. We show one instance of a puzzle from each of the four visual categories. Note that the header categories are not exhaustive as some puzzles may belong to more visual categories than the ones shown in the header. The complete ontological categorization can be found in Table 1. The questions shown above have been shortened to fit in the figure. The full questions can be found later in §3 and in Appendix A.

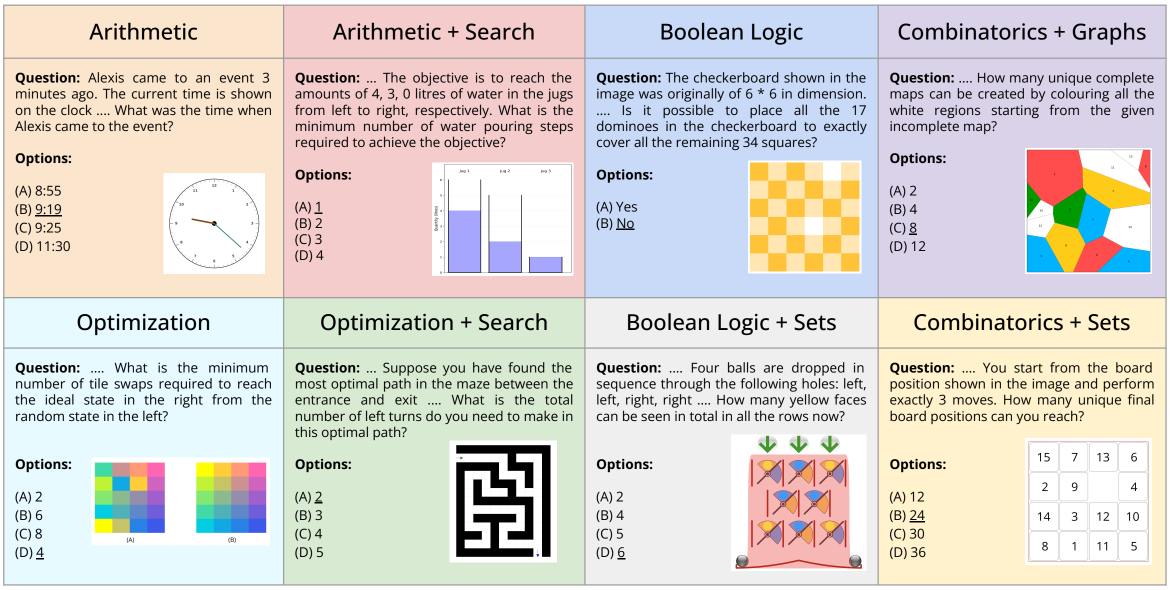
configurations of the puzzle, the position of entities on a grid, the colours of a map, etc. Visual features such as colours, positions, shapes, and sizes of the puzzle components are generally present in the visual context. The **language context**, which is shown as *Question* in the figure then describes the specific features of the puzzle and the associated rules. It also presents the query problem that we are trying to solve. Correctly answering this question requires applying a fundamental **mathematical or algorithmic concept**. We design the puzzles such that solving them requires a wide array of mathematical and algorithmic topics, ranging from basic arithmetic, set theory, and boolean logic to more sophisticated areas like graph theory, and optimized search. Our proposed dataset thus assesses critical tasks such as interpreting visual inputs, understanding spatial relationships, and applying algorithms for object manipulation and pathfinding. These skills are essential for real-world applications, including environmental navigation and action optimization which are key components for autonomous systems in dynamic settings.

A key advantage of our approach is the automated generation of puzzles from human-written code, which allows for the scalable creation of challenges with adjustable reasoning complexity. For a particular puzzle, we first write the general algorithmic solution based on well known proofs and results from the literature. By varying the inputs and configurations of the problem, we can create instances of desired difficulties. As multimodal language models become more capable (OpenAI, 2023; Gemini Team, 2023), it is crucial that we design clean and more challenging benchmarks to make fair assessments of their capabilities. Our automatic generation framework offers a simple way of creating and then continuously updating the benchmark with increasingly complex problems to evaluate progressively stronger multimodal models in the future.

We also highlight another feature of our dataset – the algorithmic foundation of our puzzle generation process ensures that each instance in our dataset has a definitive solution, eliminating the possibility of annotation error (Northcutt et al., 2021), subjectivity, ambiguities and biases that may arise from human annotated content (Geva et al., 2019).

We summarize our contributions as follows:

- We introduce an ontology of multimodal reasoning features tailored for visual algorithmic puzzle solving, aimed at delineating the capabilities and limitations of LLMs. Since each puzzle is annotated with distinct visual e.g., colour, position and algorithmic characteristics e.g., arithmetic, and search derived from our ontology, we have gained valuable insights into the specific skills that LLMs lack in addressing individual problems.
- Relying on this ontology, we created a novel puzzle dataset designed to test multimodal reasoning across vision, language, and algorithms. All puzzles in our dataset possess unambiguous algorithmic solutions. The distinguishing feature of our proposed dataset, ALGOPUZZLEVQA, resides in the prerequisites of the solutions to the problems. They demand proficiency in algorithmic and mathematical reasoning by applying logic adeptly in novel contexts without necessitating specialized domain knowledge.
- Our dataset is automatically generated using a scalable puzzle generation framework, allowing for dynamic adjustment of problem complexity.
- Experimental results conducted on this dataset



Arithmetic

Question: Alexis came to an event 3 minutes ago. The current time is shown on the clock .... What was the time when Alexis came to the event?

Options:

(A) 8:55
(B) 9:19
(C) 9:25
(D) 11:30

Arithmetic + Search

Question: ... The objective is to reach the amounts of 4, 3, 0 litres of water in the jugs from left to right, respectively. What is the minimum number of water pouring steps required to achieve the objective?

Options:

(A) 1
(B) 2
(C) 3
(D) 4

Boolean Logic

Question: The checkerboard shown in the image was originally of 6 * 6 in dimension. .... Is it possible to place all the 17 dominoes in the checkerboard to exactly cover all the remaining 34 squares?

Options:

(A) Yes
(B) No

Combinatorics + Graphs

Question: .... How many unique complete maps can be created by colouring all the white regions starting from the given incomplete map?

Options:

(A) 2
(B) 4
(C) 8
(D) 12

Optimization

Question: .... What is the minimum number of tile swaps required to reach the ideal state in the right from the random state in the left?

Options:

(A) 2
(B) 6
(C) 8
(D) 4

Optimization + Search

Question: ... Suppose you have found the most optimal path in the maze between the entrance and exit .... What is the total number of left turns do you need to make in this optimal path?

Options:

(A) 2
(B) 3
(C) 4
(D) 5

Boolean Logic + Sets

Question: .... Four balls are dropped in sequence through the following holes: left, left, right, right .... How many yellow faces can be seen in total in all the rows now?

Options:

(A) 2
(B) 4
(C) 5
(D) 6

Combinatorics + Sets

Question: .... You start from the board position shown in the image and perform exactly 3 moves. How many unique final board positions can you reach?

Options:

(A) 12
(B) 24
(C) 30
(D) 36

15	7	13	6
2	9		4
14	3	12	10
8	1	11	5

Figure 2: Examples of puzzles from ALGOPUZZLEVQA based on **algorithmic features**. We show at least one instance of a puzzle from each of the seven algorithmic categories. Note that the header categories are not exhaustive as some puzzles may belong to more algorithmic categories than the ones shown in the header.

with GPT-4V, Gemini Pro, and other models such as InstructBlip demonstrate challenges in achieving satisfactory scores. Our analysis reveals a fundamental deficiency in these models' visual perception and algorithmic reasoning capabilities, hindering effective complex reasoning.

2 The Proposed Ontology

Our puzzles encompass a diverse range of visual, mathematical and algorithmic topics. We categorize the visual and algorithmic features present in each puzzle in Table 1.

2.1 Visual Features

The configuration of the puzzle/problem is shown as an image, which constitutes its visual context. We identify some fundamental aspects of the visual context that influence the nature of the puzzles. We show examples of the categories in Figure 1. The categorization is as follows:

Colour: The category includes puzzles where understanding the colour of the puzzle components is crucial for solving the question. For these puzzles, a different colour combination of the components in the visual context would generally lead to a different solution even for the same question. *Checker Move*, *Colour Hue*, *Rubik's Cube*, etc. come in this category.

Position: The puzzles where the position of the puzzle components is necessary for solving the question. All puzzles of our dataset fall under this category. Positional understanding is thus necessary for solving all the puzzles in our dataset.

Shape/Size: Shape and size of the visual components are two aspects that are essential in several puzzles. This category includes the understanding of both absolute and relative shapes and sizes of the puzzle components. *Chain Link*, *Tower of Hanoi*, and *Wood Slide* fall under this category.

Text: This category includes puzzles for which some features are shown as optical characters or embedded texts in the image. In some of our puzzles, the embedded text contains important information that must be used to correctly solve the question. The *Calendar*, *Clock*, and *Water Jugs* puzzles fall under this category. An interesting case is the *Map Colour* puzzle which has some embedded text in the image, but they are just given as additional reference and are not necessary for solving the core question.

2.2 Algorithmic Features

We now detail the algorithmic features present in our puzzles. The ontological categorization presented below is strictly limited to obtaining the solutions for the puzzles i.e. for answering the questions for the puzzle instances. The algorithmic categories are not mutually exclusive, as we need to use two or more categories to derive the answer for most puzzles. We show examples of all the algorithmic categories in Figure 2. The categorization

Puzzle	Visual Features				Algorithmic Features						
	Colour	Position	Shape/Size	Text	Arithmetic	Boolean Logic	Combinatorics	Graphs	Optimization	Search	Sets
Board Tiling	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗
Calendar	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗
Chain Link	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Checker Move	✓	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗
Clock	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Colour Hue	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗
Map Colour	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
Maze Solve	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✗
Move Box	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✗
N-Queens	✗	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗
Number Slide	✗	✓	✗	✓	✓	✗	✓	✗	✓	✓	✓
Rotten Fruits	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✗
Rubik’s Cube	✓	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗
Think A Dot	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓
Tower of Hanoi	✗	✓	✓	✗	✓	✓	✗	✗	✓	✓	✗
Water Jugs	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗
Wheel of Fortune	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Wood Slide	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✗

Table 1: Ontological categorization of the puzzles. We divide the table in groups for ease of reading.

is as follows:

Arithmetic: This category includes puzzles where the application of basic mathematical operations such as addition, multiplication, counting, modular arithmetic, etc. are required. These are fundamental mathematical operations that are useful in all the puzzles. For example, addition and subtraction operations are required in the *Water Jugs* puzzle, and a modular arithmetic strategy is required in the *Clock* and *Calendar*.

Boolean Logic: The puzzles for which application of some form of boolean logic is necessary for answering the question. The logical statement may take various forms, for instance: checking if the number of occurrences of two colours is equal or not in *Board Tiling*, flipping a disc colour whenever a ball passes through it in *Think A Dot*.

Combinatorics: This category includes puzzles that deal with counting combinations and permutations of its states. The general form of the question is about how many unique configurations can be reached after performing a sequence of operations. These kinds of problems are generally easier to solve with computer algorithms when there are a lot of constraints on how the configurations can be created. Our *Map Colour* and *Number Slide* puzzles come under this category.

Graphs: This category includes puzzles that can be represented as a graph data structure on which specific graph algorithms can be applied to solve the problem. Such algorithms include graph colouring, graph traversal, etc. This category includes puzzles such as *Map Colour*, *Maze Solve*, and *Rotten Fruits*.

Optimization: Optimization is a crucial area of mathematics and computer science that mainly deals with finding optimal solutions to a problem. Our puzzles involve problems such as solving a task in the minimum possible time, reaching a goal from an initial state in minimum steps, optimal sorting, summation maximization, etc. This category includes puzzles such as *Chain Link*, *Colour Hue*, *Tower of Hanoi*, and *Water Jugs*.

Search: Search algorithms are generally used to retrieve information stored within a particular data structure, or to calculate the search space of a particular problem domain. Although there are many different kinds of search algorithms, our puzzles are constrained to breadth-first search and exhaustive search. Examples of such puzzles include *Checker Move*, *Move Box*, *Wood Slide*, etc.

Sets: Many problems in computer science deal with sets of objects where you have to consider the identical nature of some objects and the equivalence of some positions or configurations. We include some puzzles in our dataset where such properties can be or have to be used to solve the problem. This category includes the *Checker Move*, *Number Slide*, and *Think A Dot* puzzles.

3 From the Ontology to Creating ALGOPUZZLEVQA

We create a total of 18 different puzzles for our dataset following various feature combinations from our ontology (Table 1). Many of these puzzles are popular in various recreational or academic settings.

Generating puzzle images and curating solutions. Instances for each puzzle are created automatically from the human-written code of the particular puzzle which includes:

1. Code to generate the image file that we use as visual context. We use the matplotlib library in Python to generate the images.

2. Code for applying the correct algorithm for finding the solution. The algorithmic solutions to these puzzles are well-known and are easy to verify against literature and other popular coding resources such as LeetCode[1].

Creating language context. Finally, we also carefully curate the language context (in English) for each puzzle using puzzle-specific templates. We make sure that the template describes all the necessary information and rules required for solving the puzzle. The templates were written by two authors of the paper and were verified by the other authors.

Extensibility. As mentioned earlier, the number of instances and the difficulty level of the puzzles can be scaled arbitrarily to create a large and challenging dataset. On the other hand, most existing benchmarks such as SMART-101 (Cherian et al., 2023) rely on human annotation which is costly and difficult to scale. For now, we have created 100 instances for each puzzle. These instances are analogous to different *test cases* of the puzzle, i.e. they have different input combinations, initial and goal states, etc. Reliably solving all the instances would require finding the exact algorithm to use and then applying it accurately. This is akin to how we verify the accuracy of a computer program aiming to solve a particular task through a broad range of test cases.

Difficulty. Our current version of ALGOPUZZLEVQA contains problems with easy to moderate difficulty for humans, which can be solved by hand through multiple steps or by applying the appropriate algorithms. However, compared to existing benchmarks such as PuzzleVQA (Chia et al., 2024) and CLEVR (Johnson et al., 2017), ALGOPUZZLEVQA not only requires visual perception and commonsense reasoning, but also algorithmic understanding to solve the problems reliably.

Dataset Size. In total, we have 1800 instances from the 18 different puzzles. We consider the full dataset as an evaluation-only benchmark. We describe the details and the creation process of some of the puzzles next. A detailed description of all the puzzles and their creation process can be found in the Appendix A.

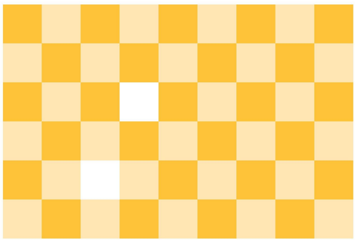


Figure 3: **Question:** The checkerboard shown in the image was originally of 6 * 9 in dimension having a total of 54 squares. It uses two colours of squares, one light yellow and one dark yellow, in a chequered pattern. Two of the squares have been removed from the board in the position of the white coloured cells, as shown in the image. You have 26 dominoes of size 2 * 1. You can use them as is or you can rotate them to use as a 1 * 2 domino. Is it possible to place all the 26 dominoes in the checkerboard to exactly cover all the remaining 52 squares? Answer Yes or No. **Gold Answer:** Yes

3.1 Some Example Puzzles in Our Dataset

Board Tiling. The puzzle is inspired by the Mutilated chessboard problem, originally posed by Max Black (Black, 1948). The puzzle belongs to the class of domino tiling problems. A general result from Mendelsohn (2004) states that: a checkerboard originally had an even (odd) number of squares. Two (one) randomly chosen squares are now removed from the board. If the mutilated board has $2 * m$ squares then it is tileable with m dominoes of size $2 * 1$ if and only if it has an equal number of dark and light squares. We choose the number of rows and columns in our checkerboard randomly between 4 to 9. We randomly remove 1 or 2 squares if the board has an odd or even number of squares, respectively. We determine the yes or no tilebality answer based on the general result. We show an example of the puzzle in Figure 3.

Colour Hue. A rectangular board contains of $m * n$ coloured tiles arranged in a rectangular grid. The board has an ideal state. A non-ideal state of the board is created by shuffling some tiles. We ask how many minimum tile swaps are required to reach the ideal state from the non-ideal state. The answer can be determined using the selection sort algorithm, which minimizes the number of swaps required to sort an unsorted array. We considered

[1] https://leetcode.com/

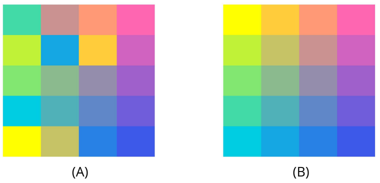


Figure 4: **Question:** A 5 * 4 board consists of 20 different coloured tiles. A random state of the board is shown in (A). The ideal state of the board is shown in (B). A swap consists of selecting any two tiles in the board and switching their positions. What is the minimum number of swaps required to restore the ideal state of the board from (A)? **Gold Answer:** 4

the non-ideal state as the unsorted state and the ideal state as the sorted state to find the answer. We show an example in Figure 4.

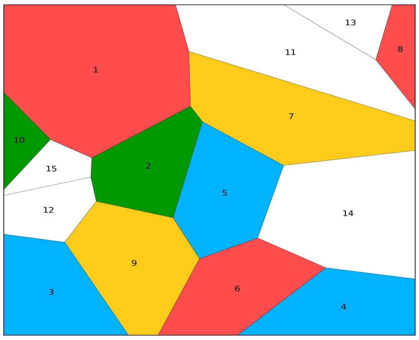


Figure 5: **Question:** You are given an incomplete map of a country having 15 different regions. The objective is to colour the regions of the map using only the four available colours: red, green, blue and yellow, such that no two adjacent regions have the same colour. Adjacent regions are defined as two regions that share a common boundary of non-zero length. The regions indicated by numbers 1 to 10 have already been coloured, as shown in the image. The regions indicated by numbers 11 to 15 are shown in white as they are yet to be coloured. You need to assign colours to these regions in a way such that it doesn't violate the objective. Each unique colour combination of the regions would result in a unique complete map. How many unique complete maps can be created by colouring all the white regions starting from the given incomplete map? **Gold Answer:** 8

Map Colouring. The four colour theorem is a famous result in mathematics that states that four colours are sufficient to colour the regions of any planar map such that no two adjacent regions have the same colour. We create a Voronoi diagram (Voronoi, 1908) followed by polygon clipping (Sutherland and Hodgman, 1974) to create the regions of the map. We use a graph colouring strategy based on Algorithm X (Knuth, 2000) with four colours to find the exhaustive solutions for colouring the map. We fix the colours of some of the regions of the map and find out the number of ways the remaining regions can be coloured from the exhaustive list of solutions. We show an example of this puzzle in Figure 5.

4 Experiments on AlgoPuzzleVQA

4.1 Setup and Baselines

We perform all our experiments on a multi-choice question-answering setup. We create three negative answer choices for each instance by using heuristics such as randomly sampling numbers within the same magnitude as the gold answer. More details can be found in Appendix B. In total, we thus have four answer choices - one gold positive and three random negatives, for all puzzles except one. The exception is the *Board Tiling* puzzle, where we have *Yes* and *No* as the possible choices. We evaluate various closed and open-source multimodal language models on our dataset. We consider the following models: GPT-4V (OpenAI, 2023), GPT-4o, Gemini Pro (Gemini Team, 2023), Claude 3 Opus[2] as the closed models; InstructBLIP Vicuna 7B and 13B (Dai et al., 2023), LLaVA-1.5 13B (Liu et al., 2023), and Qwen2-VL-7B-Instruct (Wang et al., 2024) as the open-sourced models. We use the accuracy of predicting the final answer as the evaluation metric.

4.2 Prompting Strategy

GPT-4V, GPT-4o, Gemini, Claude, LLaVA, Qwen2: We use the zero-shot chain-of-thought (CoT) technique for these models. The objective is to generate the reasoning steps and then the final answer from the image, question and multiple answer choices. We perform experiments with two types of CoT settings using the following prompting instructions: (i) `Let's think step by step` (Kojima et al., 2022), and (ii) `Let's describe the image first and think step by step`. We use the notation CoT to describe the first setting and Elaborate CoT or eCoT to describe the second setting. For LLaVA and Qwen2, we only use the CoT setting. We concatenate the question, answer choices and the prompting instruction to create the final prompt. An example prompt from the CoT setup is as follows: `Question: You are playing`

[2] https://www.anthropic.com/news/claude-3-family

Puzzle	CoT				eCoT				I-BLIP		LLaVA	Qwen2	Average
	GPT-4V	GPT-4o	Gemini Pro	Claude 3	GPT-4V	GPT-4o	Gemini Pro	Claude 3	7B	13B	13B	7B	
Board Tiling	48	53	43	48	45	51	50	47	52	52	54	45	49.0
Calendar	57	30	33	39	51	27	30	42	18	21	31	28	33.9
Chain Link	21	28	30	23	31	36	28	29	29	24	31	31	28.4
Checker Move	33	29	27	35	34	27	25	33	34	15	27	33	29.3
Clock	25	22	32	37	29	22	30	34	28	26	10	26	26.8
Colour Hue	27	25	29	16	23	31	21	31	21	18	22	20	23.7
Map Colour	34	37	28	33	34	27	38	29	17	25	29	32	30.3
Maze Solve	30	28	38	23	31	24	32	25	27	21	27	25	27.6
Move Box	29	23	32	19	25	23	26	14	24	28	20	27	24.2
N-Queens	14	26	35	15	24	24	23	13	15	10	27	36	21.8
Number Slide	41	22	38	43	32	27	31	46	27	35	32	34	34.0
Rotten Fruits	27	31	29	40	42	35	25	39	27	33	29	30	32.3
Rubik's Cube	48	35	34	39	43	33	26	36	41	41	37	37	38.3
Think A Dot	31	32	31	53	40	26	33	47	34	34	41	27	37.5
Tower of Hanoi	17	20	29	22	21	18	29	23	29	22	26	23	23.25
Water Jugs	17	24	13	37	17	16	15	28	41	42	21	18	24.1
Wheel of Fortune	25	27	27	19	28	22	24	20	23	19	27	31	24.3
Wood Slide	21	24	15	16	21	28	20	26	37	30	22	23	23.6
Average	30.3	28.7	30.2	30.9	**31.7**	27.6	28.1	31.2	29.1	27.6	28.5	28.2	-

Table 2: Accuracy scores across all the puzzles for the various multimodal language models. We divide the table into groups for ease of reading. Claude 3 indicates the Claude 3 Opus model, I-BLIP indicates the Instruct-BLIP model, and Qwen2 indicates Qwen2-VL-Instruct.

`a Tower of Hanoi game ... Options: (A) 1 (B) 2 (C) 6 (D) 3. Answer: Let's think step by step.` We create the prompt for eCoT in a similar fashion with its respective prompting instruction. We generate the output using a temperature of 0 which is equivalent to greedy decoding (Kojima et al., 2022; Wei et al., 2022).

Instruct-BLIP: We follow the multi-choice question-answering setup recommended in the original Instruct-BLIP work (Dai et al., 2023) for evaluation. The prompt is: `Question: You are playing a Tower of Hanoi ... Options: (a) 1 (b) 2 (c) 6 (d) 3. Short Answer:` The output generated from the model is constrained to be within the answer choices using a vocabulary restriction method. The answer choice with the highest log-likelihood is chosen as the prediction.

4.3 Main Results

We report the main results for all the models across all the puzzles in Table 2. The Board Tiling puzzle has a random baseline performance of 50%. All other puzzles have a random baseline performance of 25%. The overall random baseline stands at 26.4%. We notice that the performance in a significant number of these puzzles across all the models is close to the random baseline of 50% and 25%. In CoT setup, Claude 3 Opus acheives the best average score of 30.9%. The GPT-4V model in the eCoT setup achieves the best score overall with an average accuracy of 31.7%, which is only around 5% better than random. The other models perform slightly poorer, with GPT-4o obtaining a best of 28.7%, Gemini Pro obtaining a best of 30.2%, Claude 3 obtaining a best of 31.2%, Instruct-BLIP obtaining a best of 29.1% with the 7B model, LLaVA obtaining 28.5%, and Qwen2 obtaining 28.2%.

We report the average score of the models for each puzzle in the right-most column of Table 2. This helps us find which puzzles are easier and which are difficult for models. We found that *Rubik's Cube* and *Think A Dot* have the highest average score over random, implying that these two puzzles are found by models to be comparatively the easiest. Conversely, the average performance in *N-Queens* and *Colour Hue* are the lowest, signifying that models found them to be the hardest.

In terms of absolute score, we find the highest performing experiment to be GPT-4V CoT in the *Calendar* puzzle, where it achieves an accuracy of 57%. We do not find any puzzle where the accuracy is higher than 60%. We conclude that multimodal language models find the task of visual algorithmic problem-solving to be very challenging. Even though they have achieved remarkable performance in many tasks, they still have some way to go in performing complex reasoning tasks defined over vision, language, mathematics, and algorithms.

4.4 Ontological Analysis

We present the results across ontological categories in Table 3. The obtained results suggest some interesting patterns across the ontological structure. Closed models mostly perform better across the visual features of colour, position, and

	CoT				eCoT				I-BLIP 7B	I-BLIP 13B	LLaVA 13B	Qwen2 7B
	GPT-4V	**GPT-4o**	**Gemini Pro**	**Claude 3**	**GPT-4V**	**GPT-4o**	**Gemini Pro**	**Claude 3**				
Visual Features												
Colour	**35.0**	32.8	32.8	33.2	34.4	30.2	31.4	32.8	31.2	29.2	32.1	30.8
Position	30.3	28.7	30.2	30.9	**31.7**	27.6	28.1	31.2	29.1	27.6	28.5	29.2
Shape/Size	23.9	26.8	25.6	**30.0**	27.6	24.4	27.1	29.5	29.8	27.8	25.9	26.4
Text	33.2	27.0	28.5	**34.7**	31.8	23.5	28.0	33.2	25.7	28.0	25.0	28.2
Algorithmic Features												
Arithmetic	30.3	28.7	30.2	30.9	**31.7**	27.6	28.1	31.2	29.1	27.6	28.5	29.2
Boolean Logic	27.6	31.5	29.4	**35.4**	32.1	28.0	29.8	32.4	31.1	29.1	31.8	30.5
Combinatorics	37.5	29.5	33.0	**38.0**	33.0	27.0	34.5	37.5	22.0	30.0	30.5	33.0
Graphs	33.6	30.8	32.2	30.8	**35.0**	28.4	29.4	28.6	27.2	29.6	28.4	30.2
Optimization	25.6	25.0	28.1	26.6	27.0	26.4	25.2	29.0	**29.1**	28.1	25.6	25.7
Search	26.3	26.4	**28.4**	28.3	28.1	24.9	26.4	27.6	27.8	26.1	26.0	28.1
Sets	31.0	27.3	33.0	39.7	34.3	29.7	30.7	**40.7**	30.0	31.0	34.7	30.7

Table 3: Accuracy scores across all the ontological categories for the various multimodal language models.

	GPT-4V		**Gemini Pro**	
	w/o	w/	w/o	w/
Calendar	51	43	30	31
Water Jugs	17	23	15	12
Checker Move	34	**45**	25	**32**
Clock	29	**74**	21	**43**
Move Box	25	**61**	26	21
Number Slide	32	**53**	31	**39**
Tower of Hanoi	19	**32**	18	19

Table 4: Reasoning with Guided Vision. w/o and w/ indicate the without and with the guided vision context.

text but perform poorer on shape/size.

The best-performing model GPT-4V eCoT performs much higher in algorithmic topics such as combinatorgraphs and sets compared to topics such as optimization and search. Results suggest that the optimization and search topics are the most difficult topics in general across all the models.

4.5 Reasoning with Guided Vision

Our current experimental setup doesn't disentangle the visual perception stage and algorithmic reasoning stage. In the original setup, models must identify the various aspects and characteristics of the visual context before the appropriate algorithm can be applied. To minimize the effect of the bottleneck in the visual perception stage, we conduct a guided vision experiment, where we additionally provide detailed descriptions of the image as part of the language context. In this setup, errors from the visual perception stage are minimized, so models can only focus on the algorithmic stage for solving the question.

Puzzle	**PoT**		
	GPT-4V	**GPT-4o**	**Qwen2 7B**
Board Tile	47.0	53.0	28.7
Calendar	50.0	26.0	51.0
Chain Link	26.0	26.0	29.0
Checker Move	28.0	30.0	29.0
Clock	23.0	35.0	24.0
Colour Hue	25.0	26.0	24.0
Map	34.0	31.0	23.0
Maze	30.0	26.0	33.0
Move Box	32.0	28.0	25.0
N Queens	23.0	25.0	24.0
Number Slide	38.0	28.0	30.0
Rotting Kiwi	33.0	28.0	31.0
Rubik's Cube	50.0	19.0	22.0
Think Dot	34.0	24.0	31.0
Tower of Hanoi	24.0	27.0	32.0
Water Jugs	18.0	26.0	30.0
Wheel of Fortune	25.0	31.0	20.0
Wood Slide	23.0	29.0	30.0
Average	31.3	28.8	28.7

Table 5: Accuracy scores across all the puzzles for the various multimodal language models using PoT (Chen et al., 2023).

We report the results in Table 4. The upper part of the table constitutes the puzzles where the algorithmic reasoning is difficult, as even with language-guided visual context, the model cannot improve its scores. For GPT-4V models, the lower part of the table indicates puzzles where the guided vision setup helps in a large improvement of performance, suggesting there is a significant bottleneck in the visual perception stage. However, even then the numbers don't go anywhere close to 100, suggesting that the algorithmic reasoning stage presents substantial challenges even in the presence of gold context.

4.6 Reasoning Using Programs

We further experimented with the use of programs (Chen et al., 2023) for selected models such as GPT-4V, GPT-4o, and Qwen2 to solve puzzles in AlgoPuzzleVQA. We present the results in Table 5. The results show comparable results to the other prompting methods shown in Table 2. Specifically, GPT-4V obtained 31.3% using PoT as compared to 31.7% using eCoT, GPT-4o obtained 28.8% using PoT as compared to 28.7% as compared to CoT, and Qwen2 obtained 28.7% as compared to 28.2% using CoT.

5 Conclusion

In this study, we introduce ALGOPUZZLEVQA, a novel dataset comprising puzzles that demand multimodal reasoning over vision, language and algorithms. Unlike existing multimodal reasoning datasets such as ScienceQA or MMMU, the challenges in ALGOPUZZLEVQA do not hinge on possessing domain-specific knowledge for reasoning. Instead, solving the problems in this dataset necessitates applying logical algorithmic steps to novel problem scenarios. To construct this dataset, we devised an ontology encompassing i) visual reasoning features such as colours, positions, shapes and sizes of puzzle components; and ii) algorithmic reasoning features such as arithmetic, combinatorics, optimization, etc. Each puzzle in our dataset comprises multiple categories from the visual and algorithmic ontology. Through rigorous experiments with LLMs such as GPT-4V, Gemini-Pro, Claude 3, InstructBlip, and LLaVA, we observed a consistent struggle in achieving satisfactory performance on ALGOPUZZLEVQA, emphasizing the formidable hurdles in multimodal reasoning for algorithmic puzzle-solving. Given that each puzzle is annotated with specific visual and algorithmic features based on our ontology, we draw insights into the precise skills lacking in LLMs to address particular problems. We also analyzed whether the models falter on this dataset due to deficiencies in the visual recognition setup or the algorithmic reasoning setup. While furnishing the models with accurate visual guidance of the puzzles enhances performance in some cases, achieving high-level performance in our proposed complex reasoning tasks remains an elusive goal.

6 Limitations

We have currently considered many puzzles that are popular in various recreational or academic settings. There are also other interesting puzzles beyond our compiled list which can be used to assess the complex reasoning abilities of LLMs. We aim to explore them and enlarge our suite of puzzles as future work. Additionally, the construction of the algorithmic ontology could be expanded to consider more fine-grained categories.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Kenneth Appel and Wolfgang Haken. 1977. The solution of the four-color-map problem. *Scientific American*, 237(4):108–121.

Max Black. 1948. Critical thinking. an introduction to logic and scientific method. *Philosophy*, 23(86).

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Anoop Cherian, Kuan-Chuan Peng, Suhas Lohit, Kevin A. Smith, and Joshua B. Tenenbaum. 2023. Are deep neural networks smarter than second graders? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10834–10844.

Yew Ken Chia, Vernon Toh, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. 2024. PuzzleVQA: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16259–16273, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Albert Li, Pascale Fung, and Steven C. H. Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *ArXiv*, abs/2305.06500.

Leonhard Euler. 1741. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140.

Google Gemini Team. 2023. Gemini: A family of highly capable multimodal models.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

Donald E Knuth. 2000. Dancing links. *arXiv preprint cs/0011047*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Boris A Kordemsky. 1992. *The Moscow Puzzles: 359 Mathematical Recreations*. Courier Corporation.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Nathan S Mendelsohn. 2004. Tiling with dominoes. *The College Mathematics Journal*, 35(2):115–120.

Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*.

OpenAI. 2023. Gpt-4v(ision) system card.

Naganand Yadati Sanket Shah, Anand Mishra and Partha Pratim Talukdar. 2019. Kvqa: Knowledge-aware visual question answering. In *AAAI*.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-okvqa: A benchmark for visual question answering using world knowledge. In *European Conference on Computer Vision*, pages 146–162. Springer.

Ivan E Sutherland and Gary W Hodgman. 1974. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42.

Georges Voronoi. 1908. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Andrew Wiles. 1995. Modular elliptic curves and fermat's last theorem. *Annals of mathematics*, 141(3):443–551.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2023. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv e-prints*, pages arXiv–2311.

A Puzzles

We consider the following 18 puzzles, and the corresponding generated data will be released with MIT licence:

A.1 Board Tiling

The puzzle is inspired by the Mutilated chessboard problem, originally posed by Max Black (Black, 1948): *Suppose a standard* $8 * 8$ *chessboard has two diagonally opposite corners removed, leaving 62 squares. Is it possible to place 31 dominoes of size* $2 * 1$ *to cover all of these squares?* The diagonally opposite corners in a standard $8 * 8$ chessboard are always of the same colour. Hence, the mutilated chessboard with 62 squares has 30 squares of one colour and 32 squares of the other colour. Now, the checkered pattern of the chessboard ensures that each $2 * 1$ domino must cover 1 dark-coloured square and 1 light-coloured square. It is thus impossible to place the dominoes to cover all the squares since the mutilated chessboard has an unequal number of dark and light-coloured squares.

A general result from Mendelsohn (2004) states that: a checkerboard originally had an even (odd)

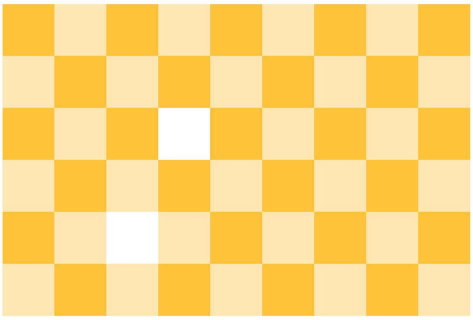
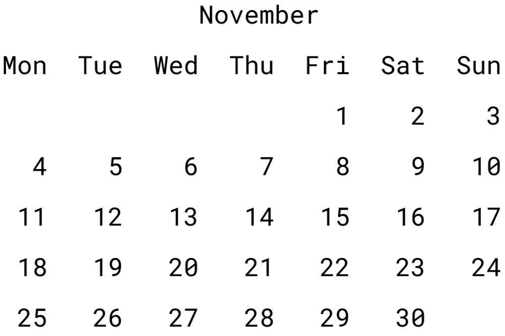


Figure 6: **Question:** The checkerboard shown in the image was originally of 6 * 9 in dimension having a total of 54 squares. It uses two colours of squares, one light yellow and one dark yellow, in a chequered pattern. Two of the squares have been removed from the board in the position of the white coloured cells, as shown in the image. You have 26 dominoes of size 2 * 1. You can use them as is or you can rotate them to use as a 1 * 2 domino. Is it possible to place all the 26 dominoes in the checkerboard to exactly cover all the remaining 52 squares? Answer Yes or No. **Gold Answer:** Yes

number of squares. Two (one) randomly chosen squares are now removed from the board. If the mutilated board has $2*m$ squares then it is tileable with m dominoes of size $2*1$ if and only if it has an equal number of dark and light squares. We use this result to construct our puzzles. We choose the number of rows and columns in our checkerboard randomly between 4 to 9. We randomly remove 1 or 2 squares if the board has an odd or even number of squares, respectively. We question whether the resulting board is tileable with m dominoes of size $2 * 1$. We determine the yes or no tilebality answer based on the general result. We show an example of the puzzle in Figure 6.

A.2 Calendar

We design this puzzle to evaluate the visual-temporal reasoning abilities of foundation models. We provide the calendar snapshot of a particular month as the visual context. We then ask what day of the week was a particular date in either the previous, same or the next year. We also provide information about whether the years of consideration were leap years or not to make sure that the answer is exact. We use the python *calendar* module to construct the instances of the puzzle. We show an example of the puzzle in Figure 7.



November

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figure 7: **Question:** The image shows the calendar of a month of a particular non-leap year. Which day of the week was on March 1 of that year? **Gold Answer:** Friday

A.3 Checker Move

The puzzle generally known as *Toads and Frogs*[3] was invented by the mathematician Richard Guy. We start with a grid of length n, with $n-1$ checkers of either red or green colour occupying $n-1$ positions. The goal is to rearrange the checkers into a given desired position. The rearrange is constrained to some rules: i) green checkers only move rightward; ii) red checkers only move leftward; iii) every move is either 1) a slide to the adjacent empty square, or 2) a jump over one position to an empty square, provided the checker being jumped over is of a different colour; iv) each grid position can accommodate a maximum of one checker at any time.

We only consider final arrangements that can be reached from the starting arrangements. We use breadth-first search constrained upon the movement rules to derive the solution.

A.4 Chain Link

This puzzle is a modified version of a problem that appeared in Kordemsky (1992). It states that you are given chain segments of different lengths. Closed pieces can be cut open and open pieces can be welded together in a certain amount of time. You need to find out the least time required to create the longest possible circular necklace. We show an example of the puzzle in Figure 9. The puzzle can be solved optimally as follows:

1. Check if the number of open links is equal to or greater than the number of closed segments.

[3] https://en.wikipedia.org/wiki/Toads_and_Frogs

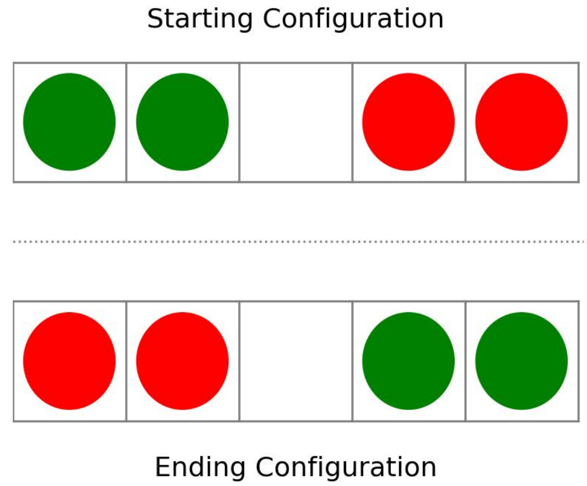


Figure 8: **Question:** A checker game is being played on a grid of 5 squares with 2 green and 2 red checkers. Initially, the checkers are arranged as shown in the starting configuration with the 4 checkers occupying 4 squares and one unoccupied square. Green checkers only move rightward and red checkers only move leftward. Every move is either i) a slide to the adjacent empty square, or ii) a jump over one position to an empty square, provided the checker being jumped over is of a different colour. Each square can accommodate a maximum of one checker at any time. How many moves are required to reach the ending configuration from the starting configuration following the specified rules? **Gold Answer:** 8

Initially, in our example, the number of open links is 3 and the number of closed segments is 9. The condition is not satisfied and hence we move to the next step.

2. Cut units from the segments of the least length until the first condition is satisfied. In our example, if you cut open the two segments of length 1 and both the units in the segment with length 2 then you have a total of 7 open links and 6 closed segments, satisfying the first condition.

3. Calculate the time considering the total number of cuts and welds. We performed 4 cut operations and then we will need 7 welding operations to close all the open links. The time required is $4 * 5 + 7 * 2 = 34$ minutes. This is the minimum possible time to create the necklace.

A.5 Clock

We design an instance of a visual-temporal reasoning puzzle using clock times. We consider an analog clock with the hours, minutes hand and show a randomly chosen time as the current time. We describe an event which happened (will happen) h hours and m minutes ago (later). We then ask when did the event happen or when is it going to happen. We determine the answer using modular arithmetic. We show an example in Figure 10.

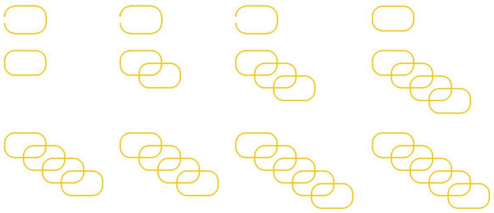


Figure 9: **Question:** Alice has 12 segments of chains of different lengths as shown in the image. The total length of all the segments combined is 32 pieces. She has a saw machine with which a closed piece can be cut opened. She also has a welding machine with which an open piece can be closed. Each cut takes 5 minutes and each welding takes 2 minutes. Initially, she has 3 segments each with 1 open piece as shown in the image. All the other pieces are closed. She now wants to make the longest possible necklace using all the available 32 pieces. Each piece in the necklace would be connected to exactly two other pieces. This would require cutting open some pieces and then joining all the resulting segments together. What is the minimum time in which she can create the necklace? **Gold Answer:** 34

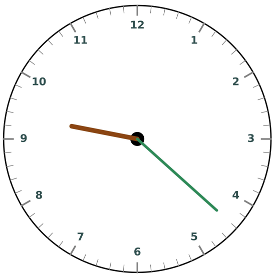


Figure 10: **Question:** Alexis came to an event 3 minutes ago. The current time is shown on the clock. The clock is a standard analog clock without the seconds hand. What was the time when Alexis came to the event? **Gold Answer:** 9:19

A.6 Colour Hue

We show an example in Figure 11. The puzzle is inspired by the game *I Love Hue*[4]. A rectangular board contains of $m * n$ coloured tiles arranged in a rectangular grid. The board has an ideal state where the colours are arranged in a way such that each row and column shows a monotonic change in the colour shade. To create this colour arrangement, we fix the RGB colour codes of the four cor-

[4] https://i-love-hue.com/

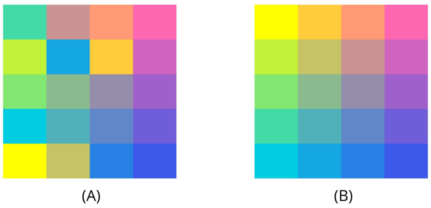


Figure 11: **Question:** A 5 * 4 board consists of 20 different coloured tiles. A random state of the board is shown in (A). The ideal state of the board is shown in (B). A swap consists of selecting any two tiles in the board and switching their positions. What is the minimum number of swaps required to restore the ideal state of the board from (A)? **Gold Answer:** 4

ner tiles and perform linear interpolation between them to determine the colour codes of the intermediate tiles. We then randomly shuffle some of the tiles to create a non-ideal arrangement of the board. We ask how many minimum tile swaps are required to reach the ideal state from the non-ideal state. The answer can be determined using the selection sort algorithm, which minimizes the number of swaps required to sort an unsorted array. We consider the flattened version of the ideal state of the board to be the sorted state. The flattened version of the non-ideal state of the board is considered as the unsorted state. We then determine the answer using selection sort.

A.7 Map Colouring

The four colour theorem is a famous result in mathematics which states that four colours are sufficient to colour the regions of any planar map such that no two adjacent regions have the same colour. The conjecture was first proposed in the 1850s but a formal proof (Appel and Haken, 1977) was first developed almost 120 years later.

We create a Voronoi diagram from a finite set of points (Voronoi, 1908) followed by polygon clipping (Sutherland and Hodgman, 1974) to clip the Voronoi diagram between the finite regions of (x, y), where $0 \leq x \leq 1$ and $0 \leq y \leq 1$. This region constitutes our input map. We represent the map as a graph with regions as nodes and their adjacent regions as the adjacency list. We use a graph colouring strategy based on Algorithm X (Knuth, 2000) with four colours to find the exhaustive solutions for colouring the map. We fix the colours of some of the regions of the map and find out the number of ways the remaining regions can be coloured from the exhaustive list of solutions. We show an example of this puzzle in Figure 12. We construct the puzzles in our dataset such that the number of masked regions is between 2 and 6 and the number of ways of colouring them is between 1 and 8.

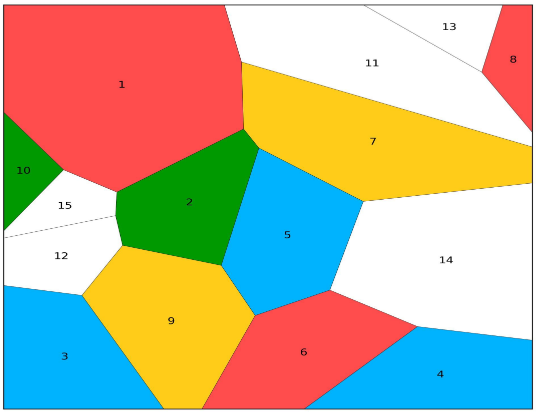


Figure 12: **Question:** You are given an incomplete map of a country having 15 different regions. The objective is to colour the regions of the map using only the four available colours: red, green, blue and yellow, such that no two adjacent regions have the same colour. Adjacent regions are defined as two regions that share a common boundary of non-zero length. The regions indicated by numbers 1 to 10 have already been coloured, as shown in the image. The regions indicated by numbers 11 to 15 are shown in white as they are yet to be coloured. You need to assign colours to these regions in a way such that it doesn't violate the objective. Each unique colour combination of the regions would result in a unique complete map. How many unique complete maps can be created by colouring all the white regions starting from the given incomplete map? **Gold Answer:** 8

A.8 Maze Solving

This puzzle is a typical maze path-finding problem. We start from a square/rectangular grid consisting of all black cells (walls). We define the first cell of the second row as the entrance to the maze. We then perform a directionally randomized depth-first search (DFS) with backtracking from the entrance cell to create the white cells (paths) through the maze. We also make sure that at any point of the maze, either the maximum length or the maximum width of the path is 1 cell. This method ensures that there are no grids of white cells in the maze with both length and width greater than 2. We finally assign the last column of the second last row or the last row of the second last column as the exit cell.

After constructing the maze, we use breadth-first search (BFS) between the entrance cell and the exit cell to find the shortest / optimal path. We

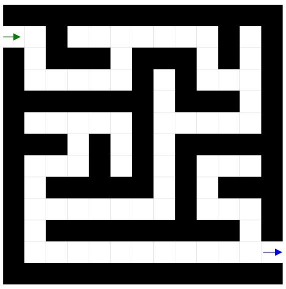


Figure 13: **Question:** This is maze having 13 * 13 cells. The empty cells are coloured white and the obstacle cells are coloured black. From an empty cell, you can only move up, down, left, or right to another adjacent empty cell. You cannot move diagonally between two empty cells and cannot step into a cell with an obstacle. The entry cell of the maze is shown with the green arrow. The exit cell of the maze is shown with the blue arrow. Suppose you have found the most optimal path in the maze between the entrance and exit, where you need to go through the least number of empty cells and you need to make the least number of left and right turns. What is the combined number of left and right turns do you need to make in this optimal path? **Gold Answer:** 12

then randomly select one question for this instance among the two choices: i) What is the number of left/right/total turns do you need to make in this optimal path? or ii) How many cells do you need to visit in this optimal path including the entrance and exit cells? We find out the answer to the question from the optimal path obtained from BFS. We show an example of the puzzle in Figure 13.

A.9 Move Box

The puzzle is inspired by a LeetCode problem [5]. The problem setting is a game in which a person pushes boxes around in a warehouse trying to get them to target locations. The objective is to move the box to the target position in the minimum number of moves. The solution can be found using breadth-first search.

A.10 N-Queens

The N-Queens problem is a famous chess problem often used as an example in various computer programming techniques. The objective is to place N chess queens on an $N * N$ chessboard so that no two queens threaten each other. In other words, no two queens should share the same row, column, or diagonal. We consider $N = 8, 9,$ and 10 for which there are 92, 352, and 724 solutions, respectively. We use the well-known backtracking algorithm to create the solutions to the problem. For a solution, we show the exact position of randomly chosen $N - 2$ queens in the image. We ask what should be the Manhattan distance (in terms of the unit squares of the board) between the remaining

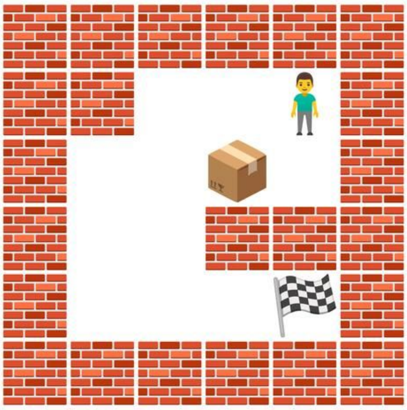


Figure 14: **Question:** A storekeeper is a puzzle in which the player pushes boxes around in a warehouse trying to get them to target locations. The game is represented by a 6 x 6 grid of characters grid where each element is a wall, floor, or box. Your task is to move the box to the end flag under the following rules: 1. The box can be moved to an adjacent free cell by standing next to the box and then moving in the direction of the box by 1 grid. This is a push. 2. The player cannot walk through the box. What is the minimum number of pushes to move the box to the end flag? **Gold Answer:** 5

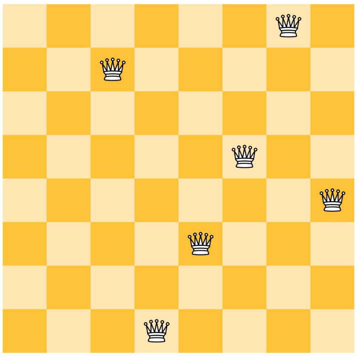
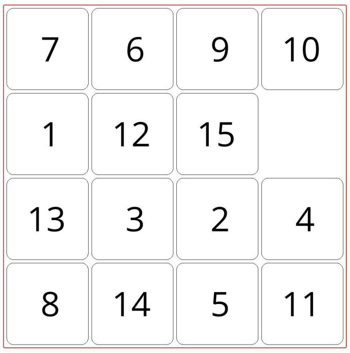


Figure 15: **Question:** You are given an 8 * 8 chessboard. The Manhattan distance between two squares in a chessboard is equal to the minimal number of orthogonal King moves between these squares on the otherwise empty board. The objective is to place 8 chess queens on this board so that no two queens threaten each other; i.e. no two queens share the same row, column, or diagonal. 6 queens have already been placed in some of the squares of the board, as shown in the image. Suppose you pick two squares to place the two remaining queen pieces in a way that fulfills the objective. What is the Manhattan distance between these two squares? **Gold Answer:** 5

[5] https://leetcode.com/problems/minimum-moves-to-move-a-box-to-their-target-location/



7	6	9	10
1	12	15	
13	3	2	4
8	14	5	11

Figure 16: **Question:** The board shown in the image is a sliding puzzle of 4 * 4 tile dimensions. It has 15 numbered tiles and one unoccupied (open) position. Tiles in the same row or column of the open position can be moved by sliding them horizontally or vertically, respectively. All tiles always stay and move inside the red boundary wall, as shown in the image. A move is defined as moving the open position by one tile unit in any available direction. You start from the board position shown in the image and perform exactly 1 move. What is the minimum sum that you can achieve across the top most row in the final board position? **Gold Answer:** 22

2 queens when they are placed correctly to satisfy the objective.

The other 2 queens can be arranged in only a single way for most cases, for which we can easily compute the Manhattan distance. In some minimal number of cases, the other 2 queens can be placed in two different ways to satisfy the non-threatening condition in all rows, columns, and diagonals. However, in both of these ways, the Manhattan distance between the last 2 queens is equal. So, we can have an exact answer to the question even though the arrangement could be distinct. We show an example of the puzzle in Figure 15.

A.11 Number Slide

This puzzle is inspired by the mathematical toy known as the 15 Puzzle[6]. It is a sliding puzzle board of grid size $4 * 4$, having 15 tiles numbered 1 to 15, with one unoccupied position. Tiles can be moved by sliding them horizontally or vertically through the open position. A typical goal in the puzzle is to arrange the tiles in numerical order from left to right and top to bottom. We use grid sizes of $3 * 3$, $4 * 4$, or $5 * 5$ and create a random arrangement of the tiles on the board and provide it as the visual context. We then create the question in one of the following styles:

- How many unique board positions can be reached after performing exactly n moves?
- What is the maximum / minimum sum that can be achieved in a particular row / column after performing exactly n moves?
- You perform n moves where the open position is seen to be moved in the following way: *up, left, ...* What is the maximum / minimum / sum of numbers in the row / column that now has the open position?

We compute the answer using breadth-first search in all the cases. We show an example of the puzzle in Figure 16.

A.12 Rotting Fruit

The puzzle is inspired by a LeetCode problem [7]. The problem states that there is a rectangular grid with some fruits. Initially, there is a single rotten fruit. As time passes, it influences surrounding fruits to become rotten. The objective is to find out the earliest time at which all fruits become rotten. A breadth-first search algorithm can be used to find the solution.

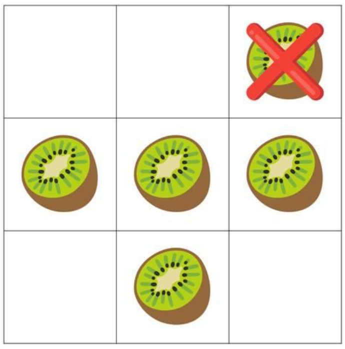


Figure 17: **Question:** You are given a 3 x 3 grid in which each cell can contain either no kiwi, one fresh kiwi, or one rotten kiwi. Every minute, any fresh kiwi that is 4-directionally adjacent to a rotten kiwi also becomes rotten. What is the minimum number of minutes that must elapse until no cell has a fresh kiwi? **Gold Answer:** 3

A.13 Rubik's Cube

Rubik's Cube is a mathematical combination toy invented by Ernő Rubik in 1974. We consider an initial state of the cube and show it as the visual context. The movements of the cube are generally denoted using the alphabets BDFLRU denoting

[6] https://en.wikipedia.org/wiki/15_Puzzle

[7] https://leetcode.com/problems/rotting-oranges/

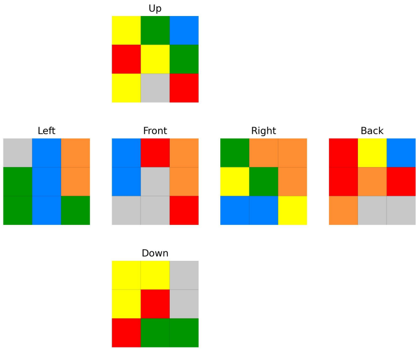


Figure 18: **Question:** A 3 * 3 Rubik's Cube has six different coloured panels: red, green, blue, yellow, orange, and grey. The initial state of the cube in terms of the different colour positions in its six faces is shown in the image. To represent the movements of the cube we use six letters: U for Up, D for Down, L for Left, R for Right, F for Front, B for Back. These letters are used in sequence where you need to perform each letter in the sequence from left to right. Each letter tells you to move that face clockwise by 90 degrees. A number 'n' immediately after a letter denotes that you need to move that face clockwise by 90 * n degrees. For example, 'U R3' would mean rotating the up face 90 degrees clockwise and then rotating the right face 270 degrees clockwise. You perform the move sequence 'D2 B' starting from the state shown in the image. What would be the number of small 1 * 1 red squares in the down face after completing the move sequence? **Gold Answer:** 2

clockwise movements of the back, down, front, left, right, and up faces, respectively. We first provide information about these notations in the textual context. We ask what is the number of small squares of any one of the six colours in any one of the six faces after completing a cube move sequence. The colour and the face in the question are chosen randomly for each instance. We show an example of the puzzle in Figure 18.

A.14 Think A Dot

Think-a-Dot is a mathematical toy invented by Joseph Weisbecker[8]. It has three holes on its top through which a ball bearing could be dropped. It also has and eight coloured disks each displaying a blue or yellow face. When a ball is dropped through the toy, it would flip the disk mechanisms that it passed, and it would determine whether the ball would be deflected to the left or the right. We show an example of the puzzle in Figure 19. We start with an initial configuration of the toy with a specific combination of the disk colours. We choose between 1 to 4 balls and a sequence of dropping them between the left, center and right holes. We ask how many blue / yellow disk faces can be seen in the top row / middle row / bottom rows / all rows after dropping the balls in that sequence. We write an algorithm to determine the answer from the initial configuration and the sequence of drops considering the state of the rows, the flipped disks and the position of the walls.

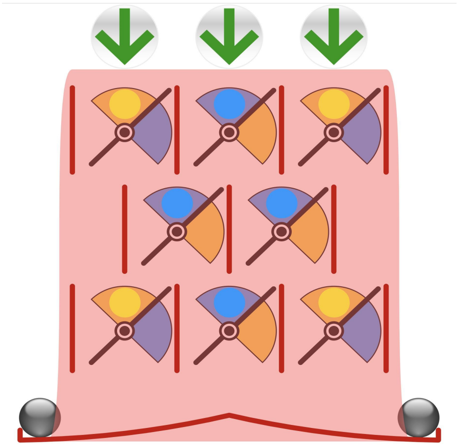


Figure 19: **Question:** The toy shown in the figure has eight coloured disks on its front, and three holes on its top - left, right, and center - through which a ball bearing could be dropped. Each disk would display either a yellow or blue face. When a ball passes through a disc it tips the disk mechanism which flips the face colour. The tipping of the disc mechanism determines whether the ball would be deflected to the left or to the right. The vertical walls between the discs would then determine the path of motion of the ball. A dropped ball always passes through exactly one disc in each of the top and the bottom row. Depending on the configuration of the top three discs it may or may not pass through the middle row. Finally, when the ball falls to the bottom it would exit either to a hole on the left or the right of the device. Four balls are dropped in sequence through the following holes: left, left, right, right. Consider the toy configuration after all the balls have been dropped and they have exited from the bottom. How many yellow faces can be seen in total in all the rows now? **Gold Answer:** 6

A.15 Tower of Hanoi

The Tower of Hanoi is a well-known mathematical game often used in teaching the fundamentals of computer programming. The game consists of 3 rods and n disks of various diameters, that can slide into any rod. In the original version of the puzzle, we start with all the disks stacked on one rod in order of decreasing size. The goal is

[8] https://en.wikipedia.org/wiki/Think-a-Dot

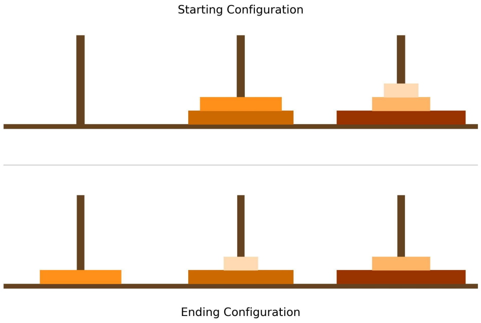


Figure 20: **Question:** You are playing a Tower of Hanoi game with 3 rods and 5 disks of various diameters, which can slide onto any rod. You are given the starting and ending configuration of the game as shown in the top and the bottom of the image, respectively. The game has the following rules: i) Only one disk may be moved at a time; ii) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod; and iii) No disk can be placed on top of a disk that is smaller than it. What is the minimum number of moves required to go from the starting to the ending configuration? **Gold Answer:** 2

to move the full stack of disks to another. Moving the discs between the rods is constrained by the following rules: (i) We can only move one disc at a time, (ii) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod, (iii) No disk can be placed on top of a disk that is smaller than it. With no other constraints, the original version of the puzzle can be solved in a minimum of $2^n - 1$ moves.

We consider the number of discs to be between 3 to 6 and generate the optimal solution considering the original problem definition. We then select two random configurations of the game from the optimal solution, which are at most $k = 6$ moves away from each other. We consider them to be the starting and the ending configuration. As the original solution is optimal, the sequence of moves required to reach the ending configuration from the starting configuration is also optimal. We ask what is the minimum number of moves required to reach the ending configuration from the starting configuration. We mark the answer to be k. We show an example of the puzzle in Figure 20.

A.16 Water Jugs

This puzzle belongs to a class of measuring puzzles involving a finite collection of water jugs with integer capacities. We provide the initial amount of water present in each jug as the visual context.

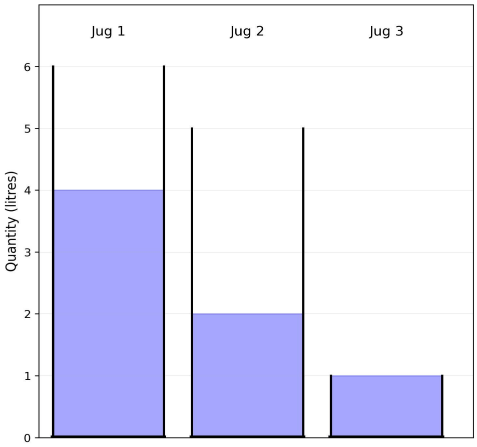


Figure 21: **Question:** You are given 3 jugs of capacities 6, 5, 1 litres. Initially, the amount of water that is contained in each jar is shown in the image. A single step of water pouring from one jug to another is constrained by the following rules: i) take a non-empty jug and pour water from it to another non-full jug until the first one becomes empty or the second one becomes full, and ii) no water can be spilt while pouring. The objective is to reach the amounts of 4, 3, 0 litres of water in the jugs from left to right, respectively. What is the minimum number of water pouring steps required to achieve the objective? **Gold Answer:** 1

We then ask how many steps of water pouring are required to reach a goal state defined in terms of specific quantities of water present in each jug. The water pouring steps are constrained by a couple of rules: i) water can be poured from a non-empty jug to another non-full jug until the first one becomes empty or the second one becomes full, and ii) no water can be split during pouring.

We consider the number of jugs to be between 3 and 5, with each initially having an amount between 1 and 14 litres of water. We create a pool of random goal states having the same quantity of total water with each jug having water that is less or equal to its respective capacity. We only consider goal states that are reachable from the initial state using breadth-first search. We show an example of the puzzle in Figure 21.

A.17 Wheel of Fortune

We design this spinning wheel puzzle to assess the spatial reasoning ability of foundation models. We sketch a wheel with 6, 8 or 10 segments with different colours and associate each of them with a prize. The angular span of the segments is chosen to be either uniform or random. We show the initial position of the wheel and the position of a

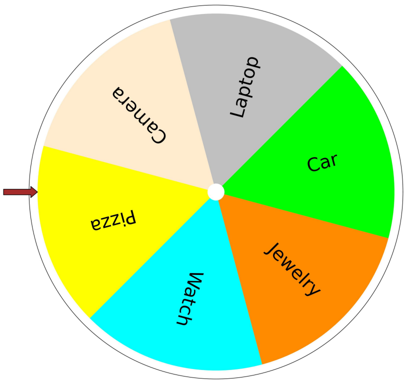


Figure 22: **Question:** A fortune wheel has 6 segments of different colour. The initial position of the wheel is shown in the figure. Each segment is associated with a prize as shown in the embedded text within the segment. The axis of rotation of the wheel passes through its center and is perpendicular to the surface of the wheel. You spin the wheel clockwise and it rotates 1695 degrees before stopping. You are going to win the prize for the segment that now falls in front of the brown arrow. What is your prize? **Gold Answer:** Laptop

fixed arrow as the visual context. We ask what the prize would be (from the segment in front of the arrow) after the wheel has been rotated by a certain amount of degrees or full rotations in clockwise / anti-clockwise direction. We determine the answer using simple rotational mechanics. We show an example of the puzzle in Figure 22.

A.18 Wood Slide

This puzzle is inspired by the Klotski[9] sliding puzzle. We consider a puzzle grid of size 5 * 4 units that has 9 wooden blocks of various sizes: one 2 * 2, four 1 * 2, two 2 * 1, and two 1 * 1. The other two spaces are empty. There is a version of the Klotski puzzle known as the Pennant Puzzle where we start with the largest 2 * 2 block residing at the top left. The objective is to bring this piece to the bottom left by sliding the available pieces. The shortest solution of this puzzle consists of 83 moves. We first use breadth-first search to find this optimal solution. For each instance in our dataset, we choose two board positions encountered in this solution such that they are at most 5 moves away from each other. We consider these positions as the starting and ending configuration of the board. We then create the question that asks the minimum number of moves required to reach the ending configuration from the starting configuration. We show an example of the puzzle in Figure 23.

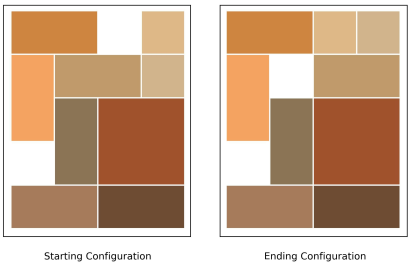


Figure 23: **Question:** Consider a sliding block puzzle of grid size 5 * 4 units. It has 9 wooden blocks of varying sizes: one 2 * 2, four 1 * 2, two 2 * 1, and two 1 * 1. The gird also has two empty 1 * 1 spaces. The blocks cannot be removed from the grid, and may only be slid horizontally and vertically within its boundary. A move is defined as selecting a block that is slideable, and moving it by 1 unit either horizontally or vertically, whichever is possible. The image shows the starting and ending configurations of the puzzle grid. The wooden blocks are shown in various shades of brown and the empty spaces are shown in white. What is the minimum number of moves required to reach the ending configuration from the starting configuration? **Gold Answer:** 3

B Negative Choice Generation for MCQA

The negative choices are constrained to *Yes* and *No* for *Board Tiling*. The negative choices are constrained to the prizes that appear in the wheel for *Wheel of Fortune*. All other puzzles in ALGOPUZZLEVQA have numerical answers. We use the heuristics of randomly sampling numbers within the same magnitude as the gold answer to create the negative choices for all the other puzzles. For example, if the gold answer is less or equal to 6 then we choose negatives between 1 - 6; otherwise, if the gold answer is less or equal to 10 then we choose negatives between 1 - 10. We step up the ranges to 50 and 100 next.

C Model Configurations

GPT-4V We use the publicly available API to query the `gpt-4-vision-preview` version of the model.

Gemini Pro We use the publicly available API to query the `gemini-pro-vision` version of the model.

[9] https://en.wikipedia.org/wiki/Klotski