Thinking Hard, Not Smart: Reasoning Models Fail to Ration Test-Time Compute Across Questions

Chenrui Fan[*1], **Yize Cheng**[*1], **Ming Li**[1], **Yongyuan Liang**[1], **Tianyi Zhou**[2], **Soheil Feizi**[1]
[1]University of Maryland, College Park [2]MBZUAI, UAE
{cfan42, yzcheng, minglii, cheryunl@umd.edu, sfeizi}@umd.edu, tianyi.zhou@mbzuai.ac.ae
Project: https://github.com/Fcr09/thinking-hard-not-smart

Abstract

Reasoning language models increasingly use test-time compute to improve performance, but existing evaluations typically study this compute one question at a time. Yet when multiple problems share an end-to-end cost or latency constraint, models must decide how to divide limited inference compute among them. We introduce an exam-style evaluation framework for studying this setting, in which a model must distribute one shared token budget across questions with different difficulty and point values to maximize its total score. Across several open and frontier reasoning models, we find that models fail to allocate a shared budget strategically across questions of varying difficulties and values. Models behave largely as greedy sequential solvers: they prioritize questions by presentation order, front-load effort on early questions, and remain insensitive to value, with these tendencies becoming more pronounced as the number of questions grows. Explicit planning prompts spread compute more evenly but do not produce value- or difficulty-aware prioritization. The same behavioral pattern extends from mathematical to code reasoning. These findings establish global budget allocation as a distinct capability that is not captured by conventional per-question evaluation and remains a challenge for current reasoning models.

1 Introduction

Reasoning models have become stronger by learning to spend more inference-time computation on difficult problems (OpenAI, 2024; Guo et al., 2025a; Snell et al., 2024; Muennighoff et al., 2025). Yet more reasoning is not always useful, and models can overthink even a single problem (Chen et al., 2025; Ma et al., 2025; Fan et al., 2025). When several problems compete for a finite budget, the decision becomes harder because continuing one problem leaves less computation for the others. A model must know not only how to solve a problem, but also which problems are worth attempting, when to give up, and when to return. Conventional one-question-at-a-time evaluation conceals this ability because each problem receives its own budget and creates no tradeoff across problems.

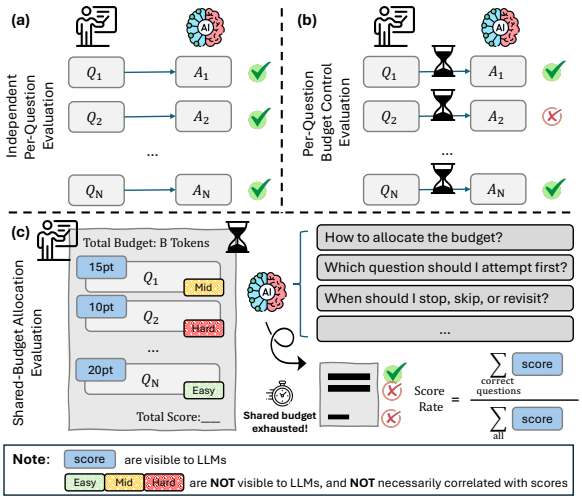

Figure 1: **Three evaluation regimes.** (a) Standard: one question with an unrestricted budget. (b) A separate budget cap for each question. (c) **Our setting:** N scored questions compete for one global budget, which tests cross-question allocation.

An exam-style evaluation offers a controlled probe for this: multiple questions with visible point values compete for one total budget, and total score supplies a concrete objective. This resembles a knapsack problem (Kellerer et al., 2004) in which the budget is the capacity, question scores are values, and model-specific solution costs are weights. Because costs and success probabilities are not explicitly provided, a strategic model must estimate value relative to cost, revise that judgment while reasoning, and abandon attempts that no longer justify further effort. We ask whether reasoning models exhibit this form of metacognitive control when allocating computation across questions.

[*]Equal contribution

Existing work controls inference effort for one problem at a time through explicit limits, adaptive computation, or difficulty-conditioned budgets (Aggarwal and Welleck, 2025; Wang et al., 2025a; Wu et al., 2025; Wen et al., 2025; Han et al., 2025). Batch prompting groups multiple questions into one request to amortize shared instructions and reduce inference cost (Cheng et al., 2023), with related work studying multi-problem evaluation and composed instructions (Wang et al., 2025b; Li et al., 2025b). REST (Pan et al., 2025) more directly stress-tests reasoning models by presenting multiple problems at once and studying their degradation under multi-context pressure. These settings do not center how realized reasoning effort responds to question value, cost, and presentation under one explicit budget. Concurrent work, TRIAGE (Nazi and Dipta, 2026), evaluates an allocation plan committed before execution; we instead study the allocation that emerges while a model jointly executes the questions, with freedom to reorder, defer, revisit, or abandon them.

We operationalize this probe through the exam-style evaluation illustrated in Figure 1. Each exam leaves the model free to decide which questions to attempt, in what order, and with how much effort. We construct matched exams from Omni-MATH (Gao et al., 2024) and systematically vary the length, presentation order, and point values of the same questions. These controlled variants let us distinguish a simple sensitivity to presentation position from genuine sensitivity to question value and difficulty. We also compare direct solving with planning instructions to test whether an explicit opportunity to allocate the budget improves this behavior. Our study covers five locally deployed open-weight models and two DeepSeek-V4 API models. The same findings also generalize to a code domain on CRUXEval-O (Gu et al., 2024).

Key Findings.

- **Models allocate compute sequentially rather than strategically.** Models largely solve questions in presentation order, spend progressively less on later questions, and respond little to stated point values. Their allocation is therefore governed more by which question appears next than by which question is most worth attempting.
- **Budget pressure magnifies the failure.** Averaged across all models, as the exam length N increases, the correlation between solving order and presentation position strengthens, while the coverage of problems on which models expend substantial effort decreases monotonically.
- **Planning changes spread, not priorities.** Planning instructions improve coverage but do not induce value-aware allocation. Reordering and repricing the same questions produce little strategic adaptation. The same position-driven pattern generalizes to code reasoning.

Contributions. We design a controlled framework for studying shared-budget reasoning, a trace-based analysis of realized effort and solving order, and broad evidence that object-level reasoning ability does not ensure strategic control across questions. Current models know how to think hard about the question in front of them, but not how to decide which question is worth thinking about.

2 Related Work

Per-problem test-time compute control. Scaling inference-time computation can improve reasoning (Snell et al., 2024; Guo et al., 2025a; Muennighoff et al., 2025), but can also lead to overthinking (Chen et al., 2025). Existing methods improve efficiency through length control (Aggarwal and Welleck, 2025; Hou et al., 2025; Li et al., 2025a,c), adaptive effort (Wang et al., 2025a; Wu et al., 2025), and difficulty-conditioned budgets (Han et al., 2025; Wen et al., 2025). Related work also studies budget-aware evaluation (Wang et al., 2024), anytime reasoning (Zhang et al., 2026), and the structure and metacognitive control of reasoning trajectories (Li et al., 2025d, 2026; Ma et al., 2026). These approaches decide how much computation to spend on a given problem. We instead study the opportunity cost created when several questions compete for the same budget.

Multi-question prompting and position sensitivity. Batch prompting groups multiple questions into one request to amortize shared instructions and reduce inference cost (Cheng et al., 2023), while other work evaluates models on multiple problems or composed instructions (Wang et al., 2025b; Li et al., 2025b). REST presents several reasoning problems simultaneously to study multi-context degradation and contextual priority allocation (Pan et al., 2025). A separate literature (Chen et al., 2024; Schilcher et al., 2025) shows that model behavior can be sensitive to the ordering of prompt elements or reasoning premises. These studies es-

tablish multi-problem interference and order sensitivity, but do not center how realized reasoning effort responds to visible question values and model-specific costs under one explicit global budget.

Global allocation and metareasoning. Rational metareasoning treats computation itself as a decision, using its expected value to determine which reasoning operation is worth performing (Russell and Wefald, 1991; Sabbata et al., 2025). Recent work (Zhai et al., 2026) begins to allocate test-time compute across inputs using learned per-instance budget policies. ROI-Reasoning (Zhao et al., 2026) trains models for knapsack-style solve-or-skip allocation under a global token cap, but retains a fixed processing order. Concurrently, TRIAGE (Nazi and Dipta, 2026) evaluates the quality of an upfront plan, in contrast, we diagnose the allocation that emerges jointly during solving, where the model can reorder, defer, or abandon questions mid-trace.

3 Shared-Budget Multi-Question Reasoning

We evaluate whether reasoning models can distribute a finite inference budget across multiple competing questions.

3.1 Task Formulation

An exam is

$$E = \{(q_i, v_i)\}_{i=1}^{N}, \tag{1}$$

where q_i is a question and v_i is its visible point value. In each exam, the model sees all questions and their point values at once, receives a shared budget of B reasoning tokens, and aims to maximize score rate:

$$\frac{1}{\sum_{i=1}^{N} v_i} \sum_{i=1}^{N} v_i \mathbb{1}[\hat{a}_i = a_i]. \tag{2}$$

In our setting, inference uses two phases. First, the model produces one reasoning trace for the entire exam, capped at B generated tokens. The prompt does not require any particular solving order or budget split. Second, after the reasoning phase ends, we ask for the final answer in a separate follow-up turn, using the existing reasoning trace as conversation history. This phase is used only for answer extraction and does not count toward the shared budget.

3.2 Dataset and Exam Construction

Our primary experiments use Omni-MATH (Gao et al., 2024). We uniformly sample problems whose benchmark difficulty label is at most 5 and construct exams with $N \in \{5, 10, 20\}$. Difficulty labels are used for experimental construction and analysis but are not shown to the model. For each value of N, we sample 50 base exams, each consisting of a fixed set of questions. The same exams are reused across scoring schemes, question orderings, prompting strategies, and models, so comparisons differ only in the factor being varied.

3.3 Models and Decoding

We evaluate five locally hosted open-weight reasoning models: DeepSeek-R1-Distill-Qwen-7B/14B (DQ-7/14) (Guo et al., 2025b) and Qwen3-8B/14B/32B (QW-8/14/32) (Yang et al., 2025), served via vllm (Kwon, 2025). We also evaluate the preview API versions of DeepSeek-V4 Flash and Pro (DSV4-F/P) (DeepSeek-AI et al., 2026).

For locally hosted models, reasoning uses temperature 0.6, top-$p = 0.95$, and top-$k = 20$. The API models are run with their available default controls. Answer extraction uses greedy decoding for all models. Mathematical answers are parsed from `\boxed{}` outputs and evaluated with an LLM-based judge described in Appendix A.

3.4 Experimental Factors

We vary score assignment, question order, and prompting strategy.

Scoring scheme. We consider four schemes:

- **Fixed:** every question is worth 10 points.
- **Random:** each question receives an integer score from 1 to 15, independently of difficulty and position.
- **Aligned:** harder questions receive more points.
- **Reversed:** easier questions receive more points.

For aligned scoring, difficulties are normalized within each exam and mapped to the integer range $[1, 15]$:

$$v_i = \left\lfloor 1 + 14 \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \right\rfloor, \tag{3}$$

where d_i is the difficulty for question i. For reversed scoring,

$$v_i = \left\lfloor 15 - 14 \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \right\rfloor. \tag{4}$$

If all questions have the same difficulty, each receives 10 points.

Question order. Each exam is presented in one of three orders: random (`rand`), ascending difficulty (`asc`), or decreasing difficulty (`dsc`). Random order separates position from difficulty, while the sorted conditions test whether models can depart from the presented sequence when early questions are especially easy or difficult.

Prompting strategy. The baseline prompt states the shared budget and the goal of maximizing total score. The explicit-planning condition additionally tells the model that questions may differ in difficulty and reasoning cost and asks it to plan its allocation wisely. Other prompts involving skipping and rechecking are reported in Appendix A.

3.5 Attributing Reasoning to Questions

The reasoning phase produces a single free-form trace for the entire exam. As the questions are presented with identifiers `Q1`, `Q2`, ..., we use these markers to recover a per-question view of the trace: text between two consecutive question markers is attributed to the earlier question as an approximation.

This attribution allows us to study both how much reasoning each question receives and when it is considered. Because a brief mention need not correspond to a substantive attempt, later analyses also distinguish questions that receive meaningful work from those that are only referenced in passing. We introduce the corresponding measures alongside their results in § 4.

4 Can Reasoning Models Ration Shared Compute?

We first characterize the allocation policy that emerges by default by examining which signals govern it (§4.1) and whether its budget reaches the questions worth attempting (§4.2). We then test whether this policy adapts by perturbing different variables. We instruct the model to plan its allocation (§4.3) and vary the order and point values of the same questions (§4.4). Results on code reasoning close the section (§4.5).

4.1 What Governs the Allocation of Reasoning?

To examine how the shared budget is distributed across questions, we study two complementary aspects of allocation:

- **Token effort:** how much reasoning a question receives;
- **Solving order:** when the model substantively works on it.

Both are computed from the marker segmentation of § 3.5. Let S_i be the set of reasoning segments attributed to question i, where each segment s is a sequence of tokens of length τ_s that begins at position p_s. The token effort t_i on question i is the total attributed length across different reasoning segments of the question,

$$t_i = \sum_{s \in S_i} \tau_s, \tag{5}$$

Since a question can be mentioned in the plan but never receive substantive reasoning effort, we also define the *work set*

$$W = \left\{ i \mid t_i \geq 200 \text{ or } |S_i| \geq 2 \right\}, \tag{6}$$

which separates substantive attempts from brief references.

Solving order is then defined on the work set. We locate question i by the token-weighted centroid of its segments,

$$c_i = \frac{\sum_{s \in S_i} \tau_s \, p_s}{\sum_{s \in S_i} \tau_s}, \tag{7}$$

and rank the questions in W by ascending c_i. The resulting rank is the solving order. Computing centroids of reasoning segments prevents an early mention from displacing a question that is mostly worked on later, and restricting to W prevents a question that is merely enumerated in an opening pass from being ranked ahead of questions that actually received effort.

We next ask which information governs these two allocation decisions. For each question, we consider three candidate signals: its **presentation position** π_i, its **difficulty** $\mathbf{d_i}$, and its **assigned point value** $\mathbf{v_i}$. We examine whether each signal predicts either the amount of effort the question receives or the order in the trace where it is worked on.

Difficulty and the presented position require some care because they can be correlated within a particular collection of exams. For example, even when questions are randomly ordered, a finite sample may happen to place more difficult questions toward the beginning or end. An ordinary correlation between token effort and position could then partly reflect difficulty, rather than position itself; conversely, an apparent difficulty effect could

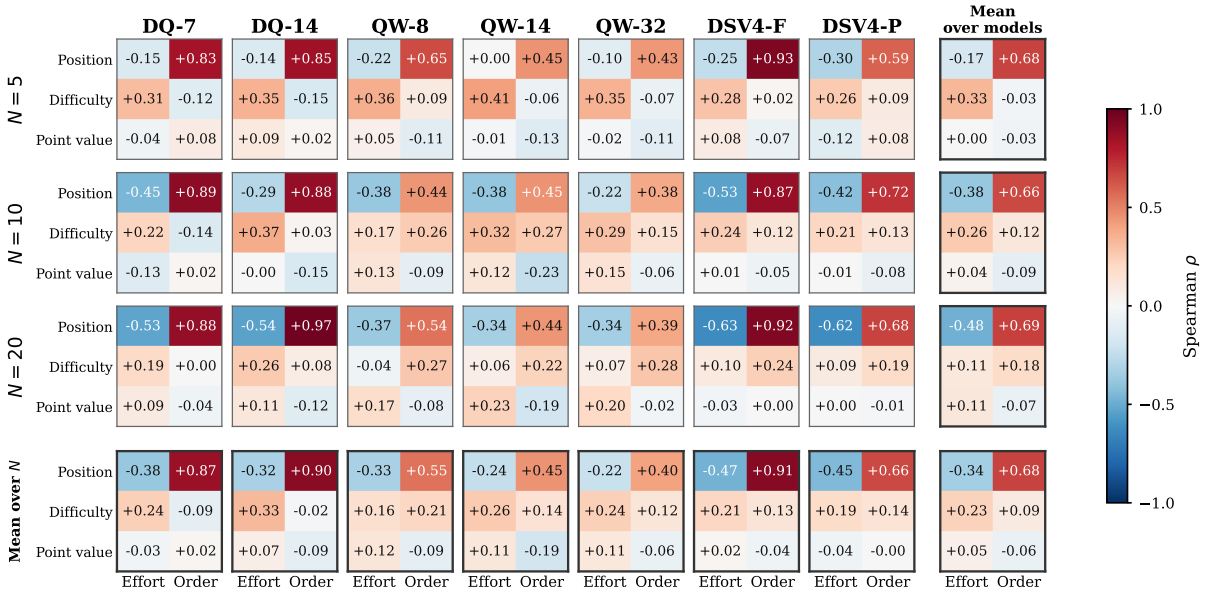

Figure 2: Relationships between allocation behavior and presentation position, difficulty, and point value under the baseline prompt. Position and difficulty are measured using partial Spearman correlations under fixed scoring and random question order (first 2 rows of each heat map); point value uses ordinary Spearman correlation under random scoring (3rd row of each heat map). Marginal panels average over models (right), exam lengths (bottom), and both (corner). DQ denotes DeepSeek-R1-Distill-Qwen, QW denotes Qwen3, and DSV4-F/P denote DeepSeek-V4 Flash/Pro. Numeric suffixes indicate parameter counts in billions.

arise because difficult questions happened to be presented in earlier or later positions. We address this confound in the fixed-scoring, random-order condition, where point values are constant and position is independent of difficulty in expectation.

To further address remaining position–difficulty correlation that may occur in some samples, we use *partial Spearman correlations*, which measures the association between two variables after controlling for a third. For example, the relationship between effort and position when controlling difficulty is

$$\rho_{t,\pi|d} = \frac{\rho_{t,\pi} - \rho_{t,d}\rho_{\pi,d}}{\sqrt{\left(1-\rho_{t,d}^2\right)\left(1-\rho_{\pi,d}^2\right)}}, \qquad (8)$$

where each term on the right is an Spearman correlation. We analogously compute $\rho_{t,d|\pi}$ to measure the relationship between effort and difficulty after controlling for position. The same procedure is applied to solving order, yielding $\rho_{o,\pi|d}$ and $\rho_{o,d|\pi}$.

Point value is analyzed separately under random scoring. In this condition, scores are assigned independently of both difficulty and position, so neither variable provides a systematic alternative explanation for an effort–value or order–value relationship. We therefore report ordinary Spearman correlations with point value rather than partial correlations.

Figure 2 reports the resulting six relationships, three candidate signals crossed with the two allocation behaviors, for every model and exam length. The correlations are first calculated per exam, then averaged over all exams with the same N.

Models follow presentation order and increasingly neglect later questions. The position row of Figure 2 reveals two complementary behaviors. The blue effort cells indicate negative correlations: questions presented later receive fewer reasoning tokens. Averaged over all models and Ns, this relationship is $\rho = -0.34$, and it strengthens from -0.17 at $N = 5$ to -0.38 at $N = 10$ and -0.48 at $N = 20$. Thus, as more questions compete for the same budget, reasoning becomes increasingly concentrated on the beginning of the exam.

In the same position row, the red order cells indicate positive correlations: the order in which models substantively work on questions closely follows their presentation order. This relationship is strong overall ($\rho = +0.68$) and remains stable across exam lengths ($+0.68$, $+0.66$, and $+0.69$). Models therefore largely follow the question presentation order regardless of the exam length, spending an increasingly large share on earlier questions as exams become longer.

Difficulty affects effort reactively, not prospectively. Harder questions do receive more tokens,

but the effect decays exactly as the budget grows tighter: $+0.33$ at N=5, $+0.26$ at N=10, and $+0.11$ at N=20. This is the signature of a reactive process. Once the model is inside a difficult question it keeps going, and at small N it can afford to; it is not deciding in advance that a question deserves more compute, which would show up as a stable or strengthening relationship under pressure. Difficulty has little bearing on which question is taken up first: the order–difficulty correlation is -0.03 at N=5 and rises only to $+0.18$ at N=20.

Point values have little effect. Neither allocation behavior responds meaningfully to the stated rewards. Effort–value correlations are 0.00, $+0.04$, and $+0.11$ across the three exam lengths, while order–value correlations are also near zero (-0.03, -0.09, and -0.07). Even at $N = 20$, where the largest effort–value association appears, it is far less meaningful than the corresponding position effect of -0.48.

The pattern holds across model families. Models differ in how strongly their solving order follows presentation position, but it remains the dominant signal for every model. The association is strongest for DQ-7, DQ-14, and DSV4-F (ρ = 0.87, 0.90, and 0.91), while the Qwen models depart from the presented sequence more often (0.55, 0.45, and 0.40 for QW-8/14/32). DSV4-P lies between these groups at 0.66. Despite this variation, all models show a negative effort–position correlation and little sensitivity to point value.

4.2 Is the Budget Spent on the Right Questions?

Section 4.1 showed that models organize their reasoning largely by presentation position. This is harmful only if they reach fewer questions than the budget allows, or if the questions they reach are not the most worthwhile ones. We examine these two possibilities in turn.

Coverage shrinks as the exam grows. We measure substantive reach using the work set W from Equation 6. Table 1 reports both its size $|W|$ and the resulting coverage $|W|/N$. We additionally report the *zero-token rate*, the fraction of questions receiving no attributed reasoning tokens at all. This is a more lenient notion than exclusion from W: a question that is merely mentioned is not in the work set, but it does not count as zero-token.

Across the locally deployed open models, the average work set grows from 4.0 questions at $N = 5$ to only 8.1 at $N = 20$, so coverage falls from 80% to 40%. The API models are even more concentrated. DSV4-F works on 4.5 questions at $N = 10$ and 4.7 at $N = 20$, reaching only 23% of the longer exam. At $N = 20$, the zero-token rate is 51% for the open-model average, 69% for DSV4-F, and 61% for DSV4-P. Thus, as the exam grows, the work set expands only slowly while an increasing fraction of questions is never meaningfully considered.

Table 1: Work-set size, coverage, and zero-token rate under fixed scoring, random order, and the baseline prompt. Columns within each group correspond to $N \in \{5, 10, 20\}$.

	Coverage $\lvert W\rvert/N$			Work set $\lvert W\rvert$			Zero-token rate		
Model	**5**	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
DQ-7	0.55	0.49	0.38	2.8	4.9	7.6	0.34	0.48	0.54
DQ-14	0.90	0.72	0.46	4.5	7.2	9.2	0.08	0.23	0.46
QW-8	0.78	0.52	0.34	3.9	5.2	6.7	0.20	0.42	0.60
QW-14	0.90	0.67	0.40	4.5	6.7	8.0	0.08	0.25	0.49
QW-32	0.88	0.64	0.44	4.4	6.4	8.8	0.10	0.28	0.44
DSV4-F	0.64	0.45	0.23	3.2	4.5	4.7	0.31	0.47	0.69
DSV4-P	0.64	0.48	0.29	3.2	4.8	5.9	0.26	0.46	0.61

Viewing the exam as a knapsack with model-adaptive value density. Limited coverage does not yet establish that models reach the wrong questions. To evaluate selection, we view the question selection problem in an exam as a knapsack problem (Kellerer et al., 2004): the shared budget B is the capacity, the point value v_i is the value of question i, and the tokens required to solve it are its weight w_i. A natural greedy policy would prioritize questions with high value density δ_i.

As the shared-budget run cannot reveal w_i, we estimate it from an independent high-budget reference condition, in which each question is solved independently with up to 40,960 tokens. For each model–question pair, the isolated token count provides a model-adaptive estimation of w_i. We define

$$\delta_i = \frac{v_i}{w_i} \mathbb{1}[\hat{a}_i^{\text{unc}} \text{ is correct}], \tag{9}$$

where the indicator is 1 only when the model answers question i correctly in the high-budget reference attempt. Questions that remain incorrect are therefore assigned zero density. We do not interpret w_i as the minimum or necessary cost of solving question i. It is the token usage observed in one independent high-budget attempt, which we use as a model-specific empirical difficulty proxy when computing value density.

Table 2: Work-set selection at $N{=}10$. The upper part uses random scoring and ascending difficulty order; the lower part reports means over models under the remaining scoring schemes (per-model breakdowns in Appendix E). Asterisks (*) indicate overlaps that are statistically significantly different from the "by chance" overlap based on the 95% confidence interval.

Model	Set overlap ratio with W			Work set $\delta = 0$	Token share $\delta = 0$
	By chance	Top-density	Early-position		
DQ-7	0.62	0.62	0.86*	0.39	0.51
DQ-14	0.63	0.65	0.75*	0.39	0.48
QW-8	0.57	0.55	0.77*	0.27	0.35
QW-14	0.65	0.67	0.76*	0.20	0.26
QW-32	0.61	0.64	0.75*	0.15	0.20
DSV4-F	0.55	0.53	0.76*	0.18	0.27
DSV4-P	0.50	0.46*	0.64*	0.13	0.18
Mean	**0.59**	**0.59**	**0.76***	**0.24**	**0.32**
Mean over models, other scoring schemes.					
Fixed	0.53	0.47*	0.81*	0.30	0.38
Aligned	0.51	0.46*	0.78*	0.31	0.39
Reversed	0.54	0.53	0.75*	0.28	0.35

Selection is blind to value density. For each exam, let $k = |W|$. We compare the work set with two reference sets of the same size:

- the k questions with the highest value density;
- the k questions presented earliest in the prompt.

We call the overlapping ratio between W and these sets as *top-density overlap* and *early-position overlap*, respectively. A density-greedy policy would have top-density overlap near 1, while the expected overlap by chance from selecting k questions without regard to either ranking is k/N.

Table 2 shows that the work set is not more aligned with value density than chance. Under random scoring and ascending difficulty order, mean top-density overlap is 0.59, indistinguishable from the chance reference of 0.59, whereas early-position overlap is 0.76. The same pattern holds under the other scoring schemes: top-density overlap remains at or below chance, while early-position overlap stays between 0.75 and 0.81.

Models also spend substantial compute on questions with $\delta_i = 0$. Such questions account for 24% of the work set and 32% of all reasoning tokens on average, despite being answered incorrectly in the independent high-budget reference.

Taken together, these results sharpen the position-driven failure from § 4.1. As exams grow, models reach only a slowly expanding subset of questions. Within that subset, selection matches the beginning of the prompt far better than the questions with the highest model-adaptive value density, while a substantial fraction of the budget is spent on questions the same model did not solve in the independent high-budget reference.

Table 3: Effect of prompt instructions under fixed scoring and random order, averaged over the five locally hosted open-weight models. Within each group the three columns are $N{=}5$, 10, and 20. The best and second best are bold and underlined. Asterisks (*) mark a statistically significant difference from base prompt based on 95% confidence interval.

	Work set coverage			Zero token rate		
Prompt	**N=5**	**N=10**	**N=20**	**N=5**	**N=10**	**N=20**
Base	0.80	0.61	0.40	0.16	0.33	0.51
Plan	**0.89*** $_{(+.09)}$	<u>0.73*</u> $_{(+.12)}$	**0.54*** $_{(+.14)}$	**0.09*** $_{(-.07)}$	<u>0.19*</u> $_{(-.14)}$	**0.32*** $_{(-.19)}$
Skip hint	0.86* $_{(+.06)}$	0.69* $_{(+.08)}$	0.45* $_{(+.05)}$	0.11* $_{(-.05)}$	0.24* $_{(-.09)}$	0.43* $_{(-.08)}$
Recheck hint	0.79 $_{(-.01)}$	0.57* $_{(-.04)}$	0.37 $_{(-.03)}$	0.18 $_{(+.02)}$	0.38* $_{(+.05)}$	0.54 $_{(+.03)}$
All three	**0.89*** $_{(+.09)}$	**0.74*** $_{(+.13)}$	<u>0.51*</u> $_{(+.11)}$	**0.09*** $_{(-.07)}$	**0.18*** $_{(-.15)}$	<u>0.35*</u> $_{(-.16)}$

4.3 Does Prompting Improve Allocation?

The base prompt states the budget and the goal but gives no guidance on dividing it. We test four instructions that do: *plan* the allocation before solving, hint that questions may be *skipped*, hint that answers should be *rechecked*, and all three combined. The details of each prompt are in Appendix A.

Planning changes the allocation the most. Planning helps most, by a margin that widens as the budget tightens (Table 3): averaged across five locally hosted open-weight models, coverage gains 0.09, 0.12, and 0.14 over the base prompt as N grows from 5 to 20. The skip hint also improves coverage at every length, but by roughly half as much. The recheck hint is flat or slightly harmful. Combining all three reproduces the planning result rather than improving on it, so we focus on planning below.

Planning spreads computation without redirecting it. Broken down by model (Table 4), every open model except QW-32 gains coverage, and DSV4-F coverage improves from 0.23 to 0.33. What does not change is the basis on which questions are chosen. Solving order stays tied to prompt position, with the open-model correlation moving only from 0.64 to 0.60 and DSV4-P becoming *more* sequential (0.68 to 0.83). Effort remains nearly uncorrelated to point value.

Planning therefore changes the *spread* of computation, not its *priorities*: instructed to budget its compute, the model divides it more evenly rather than directing it anywhere in particular. This falls short of the objective because spreading tokens more uniformly is only beneficial when the questions that receive additional effort are actually worth solving.

Table 4: Effect of explicit planning at N=20 by model. Coverage and zero rate use fixed scoring and random order; the value correlation uses the matched random-scoring condition. Asterisks (*) mark a statistically significant difference from base prompt based on 95% confidence interval.

	Coverage		Zero rate		$\rho_{o,\pi}$		$\rho_{t,v}$	
Model	**Base**	**Plan**	**Base**	**Plan**	**Base**	**Plan**	**Base**	**Plan**
DQ-7	0.38	0.58*	0.54	0.30*	0.88	0.67*	+0.09	+0.05
DQ-14	0.46	0.69*	0.46	0.18*	0.97	0.64*	+0.11	+0.10
QW-8	0.34	0.46*	0.60	0.37*	0.54	0.74*	+0.17	+0.12
QW-14	0.40	0.53*	0.49	0.33*	0.44	0.48	+0.23	+0.09*
QW-32	0.44	0.43	0.44	0.41	0.39	0.45	+0.20	+0.03*
mean	0.40	0.54	0.51	0.32	0.64	0.60	+0.16	+0.08
DSV4-F	0.23	0.33*	0.69	0.51*	0.92	0.69*	−0.03	+0.01
DSV4-P	0.29	0.27	0.61	0.61	0.68	0.83	+0.00	+0.04
mean	0.26	0.30	0.65	0.56	0.80	0.76	−0.01	+0.03

4.4 Does it Adapt to Position and Value?

We next ask how allocation changes when the same questions are presented in a different order or under a different scoring scheme. Figure 3 reports the score rate (Equation 2) of the two strongest models, DSV4-F and DSV4-P; results for the remaining models are in Appendix C.

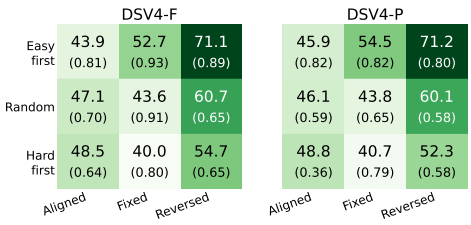

Figure 3: Score rate (%, also shown by shading) and, in parentheses, the order–position correlation in the same setting, for DSV4-F and DSV4-P under the base prompt, averaged over $N \in \{5, 10, 20\}$.

Under reversed scoring and easy-first presenting order, a position-driven sequential policy is near optimal because the earliest questions are both cheapest and most valuable. Under a hard-first presenting order, however, the models keep following the presenting position in prompt: correlations between solving order and position stay near 0.6, and scores fall by 16 to 19 points on average. This shows that the models are effectively hijacked by the most difficult questions up front, which also bear the lowest values. Despite their strength, these models think hard through an adversarial sequence rather than reordering toward a smarter one.

4.5 Results on Code Reasoning: CRUXEval-O

We also run the shared-budget setting on CRUXEval-O (Gu et al., 2024), where the model predicts the return value of a short Python function. For each $N \in \{10, 20\}$ we build 50 exams and evaluate QW-14, DQ-14, DSV4-F, and DSV4-P under fixed and random scoring, random order, and the base and explicit-planning prompts, with a pressure-matched budget $B = 3{,}000$ (see Appendix D.1 for the calibration). Since difficulty labels are unavailable in this dataset, we only use fixed scoring in the evaluation.

Table 5: Allocation on CRUXEval-O under the base prompt and random question order.

Model	N	$\rho_{t,\pi}$	$\rho_{o,\pi}$	$\rho_{t,v}$	**Coverage**	**Zero token rate**
QW-14	10	−0.39	1.00	+0.05	0.49	24%
QW-14	20	−0.67	1.00	+0.03	0.21	49%
DQ-14	10	−0.16	0.94	−0.06	0.50	24%
DQ-14	20	−0.59	1.00	−0.01	0.22	44%
DSV4-F	10	−0.20	0.97	−0.06	0.58	19%
DSV4-F	20	−0.65	1.00	+0.01	0.23	40%
DSV4-P	10	−0.13	0.98	+0.15	0.60	8%
DSV4-P	20	−0.59	1.00	+0.02	0.23	30%

The same allocation pattern appears (Table 5). Averaged across models and lengths, token effort declines with presentation position ($\rho = -0.42$) while solving order follows prompt position almost exactly ($\rho = 0.99$), and effort remains nearly unrelated to stated point values ($\rho \approx 0.02$). As N doubles, coverage collapses to 0.21–0.23 at N=20, with 30%–49% of questions receiving zero tokens. Further analysis are in Appendix D.

5 Conclusion

We introduce an exam-style framework for evaluating how reasoning models allocate a shared test-time compute budget across multiple questions. Our findings show that models consistently behave as position-driven sequential solvers: they prioritize questions in presentation order, concentrate effort on early items, respond weakly to point values, and leave increasingly many questions unattempted as the exam grows. Their selected questions align more closely with prompt position than with model-adaptive value density, and substantial compute is spent on problems unsolvable even under the high-budget reference condition. Explicit planning improves coverage by spreading effort more evenly, but does not produce value- or difficulty-aware prioritization. These results show that strong per-question reasoning does not imply effective global compute allocation, identifying shared-budget metareasoning as a distinct and unresolved capability for current reasoning models.

Limitations

Scope. Omni-MATH is the only domain in which we run the full factorial design. The CRUXEval-O experiments cover two exam lengths, four models, and two scoring schemes, with no native difficulty axis and no ordering or repricing manipulations. Absolute score rates and effect magnitudes should therefore not be compared across the two domains.

Token attribution. Per-question token counts in the shared-budget runs are recovered from `Qn`: marker segmentation, which is reliable for the large majority of traces but remains an approximation. Because a mention is not the same as an attempt, we report the work set and the token-centroid rank rather than raw mention counts throughout. A fully rigorous notion of effort, which would require detecting where a question is actually being solved rather than merely referenced, remains open.

Acknowledgments

This work was supported in part by NSF CAREER Award 1942230, the ONR PECASE Award N00014-25-1-2378, ARO Early Career Program Award 310902-00001, Army Grant W911NF-21-2-0076, NSF Award CCF-2212458, NSF Award 2229885 (NSF Institute for Trustworthy AI in Law and Society, TRAILS), MURI Grant 14262683, DARPA AIQ Grant HR00112590066, and a Meta Research Award 314593-00001.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *Preprint*, arXiv:2503.04697.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? On the overthinking of o1-like LLMs. *Preprint*, arXiv:2412.21187.

Xinyun Chen, Ryan A. Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *Preprint*, arXiv:2402.08939.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, and 300 others. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence. *Preprint*, arXiv:2606.19348.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *Preprint*, arXiv:2504.06514.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *Preprint*, arXiv:2410.07985.

Alex Gu, Baptiste Roziere, Hugh James Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. 2024. CRUXEval: A benchmark for code reasoning, understanding and execution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16568–16621. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025a. Deepseek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025b. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware LLM reasoning. *Preprint*, arXiv:2412.18547.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. ThinkPrune: Pruning long chain-of-thought of LLMs via reinforcement learning. *Preprint*, arXiv:2504.01296.

H. Kellerer, U. Pferschy, and D. Pisinger. 2004. *Knapsack Problems*. Springer.

Woosuk Kwon. 2025. *vLLM: An Efficient Inference Engine for Large Language Models*. Ph.D. thesis, UC Berkeley.

Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. 2025a. Steering LLM thinking with budget guidance. *arXiv preprint arXiv:2506.13752*.

Ming Li, Pei Chen, Chenguang Wang, Hongyu Zhao, Yijun Liang, Yupeng Hou, Fuxiao Liu, and Tianyi Zhou. 2025b. Mosaic-it: Cost-free compositional data synthesis for instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25287–25318.

Ming Li, Chenrui Fan, Yize Cheng, Soheil Feizi, and Tianyi Zhou. 2026. Schoenfeld's anatomy of mathematical reasoning by language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32773–32802, San Diego, California, United States. Association for Computational Linguistics.

Ming Li, Zhengyuan Yang, Xiyao Wang, Dianqi Li, Kevin Lin, Tianyi Zhou, and Lijuan Wang. 2025c. What makes reasoning models different? follow the reasoning leader for efficient decoding. *arXiv preprint arXiv:2506.06998*.

Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. 2025d. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18267–18288, Suzhou, China. Association for Computational Linguistics.

Siyuan Ma, Bo Gao, Zikai Xiao, Hailong Wang, Xinlei Yu, Rui Qian, Jiayu Qian, Luqi Gong, and Yang Liu. 2026. Cot2-meta: Budgeted metacognitive control for test-time reasoning. *Preprint*, arXiv:2603.28135.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *Preprint*, arXiv:2504.09858.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

Zabir Al Nazi and Shubhashis Roy Dipta. 2026. Triage: Evaluating prospective metacognitive control in LLMs under resource constraints. *arXiv preprint arXiv:2605.13414*.

OpenAI. 2024. OpenAI o1 System Card.

Zhuoshi Pan, Qizhi Pei, Yu Li, Qiyao Sun, Zinan Tang, H. Vicky Zhao, Conghui He, and Lijun Wu. 2025. REST: Stress testing large reasoning models by asking multiple problems at once. *arXiv preprint arXiv:2507.10541*.

Stuart Russell and Eric Wefald. 1991. Principles of metareasoning. *Artificial Intelligence*, 49(1):361–395.

C. Nicolò De Sabbata, Theodore R. Sumers, Badr AlKhamissi, Antoine Bosselut, and Thomas L. Griffiths. 2025. Rational metareasoning for large language models. *Preprint*, arXiv:2410.05563.

Patrick Schilcher, Dominik Karasin, Michael Schöpf, Haisam Saleh, Antonela Tommasel, and Markus Schedl. 2025. Characterizing positional bias in large language models: A multi-model evaluation of prompt order effects. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20643–20664, Suzhou, China. Association for Computational Linguistics.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939, Miami, Florida, USA. Association for Computational Linguistics.

Xiangqi Wang, Yue Huang, Yanbo Wang, Xiaonan Luo, Kehan Guo, Yujun Zhou, and Xiangliang Zhang. 2025a. Adareasoner: Adaptive reasoning enables more flexible thinking in large language models. *Preprint*, arXiv:2505.17312.

Zhengxiang Wang, Jordan Kodner, and Owen Rambow. 2025b. Evaluating llms with multiple problems at once. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pages 178–199.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. 2025. BudgetThinker: Empowering budget-aware LLM reasoning with control tokens. *Preprint*, arXiv:2508.17196.

Siye Wu, Jian Xie, Yikai Zhang, Aili Chen, Kai Zhang, Yu Su, and Yanghua Xiao. 2025. ARM: Adaptive reasoning model. *Preprint*, arXiv:2505.20258.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhiyuan Zhai, Bingcong Li, Bingnan Xiao, Ming Li, and Xin Wang. 2026. Adaptive test-time compute allocation for reasoning llms via constrained policy optimization. *Preprint*, arXiv:2604.14853.

Xuanming Zhang, Shwan Ashrafi, Aziza Mirsaidova, Amir H. Rezaeian, Miguel Ballesteros, Lydia Chilton, Zhou Yu, and Dan Roth. 2026. Budget-aware anytime reasoning with LLM-synthesized preference

data. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 8587–8599, San Diego, California, United States. Association for Computational Linguistics.

Muyang Zhao, Qi Qi, and Hao Sun. 2026. ROI-reasoning: Rational optimization for inference via pre-computation meta-cognition. *arXiv preprint arXiv:2601.03822*.

A Prompts and judge instructions

Figure 4 gives the shared-budget exam prompt, including the optional planning, skip, and recheck hints used in Section 4.3. Figure 5 shows the LLM-as-judge instructions for mathematical answers; the deterministic CRUXEval-O matcher is described below.

A.1 CRUXEval-O exact-match judge

CRUXEval-O answers are Python literals, so we use a deterministic judge rather than an LLM. The judge strips answer wrappers, parses predictions and references with `ast.literal_eval`, and compares structures recursively. It tolerates three representation-only differences observed during manual error analysis: a dictionary body missing its outer braces, list–tuple interchange with identical elements, and integer–digit-string interchange only when their canonical decimal forms match (e.g., `89` vs. `'89'`). String case and internal whitespace remain exact. This avoids semantic fuzz while recovering formatting false negatives.

B Comparison with uniform allocation

We compare the shared-budget run with a simple equal split. In the *uniform per-question* condition, each question is solved independently with a budget of B/N tokens. The answer depends on exam length (Table 6). At $N = 5$, the shared-budget setting outperforms uniform allocation for four of seven models and gains 2.6 score points on average. At $N = 10$ the comparison is nearly even. At $N = 20$, every model performs worse under the shared budget, with an average difference of -5.0 points.

At small N, front-loading can occasionally help by allowing the model to complete a few questions with high confidence, and the comparison becomes consistently unfavorable only as the exam grows and coverage collapses. This condition should also not be treated as an oracle, because independent prompts remove cross-question interference in addition to enforcing equal allocation. It is a reference for asking whether the model's emergent allocation provides a consistent advantage over a naive split, and the mechanism-level conclusions of Section 4 do not depend on it.

N	Shared wins	Δ
5	4/7	$+2.6$
10	3/7	-0.4
20	0/7	-5.0

Table 6: Shared-budget performance relative to uniform per-question allocation under aligned scoring, random order, and the baseline prompt. Δ is the mean score-rate difference in percentage points: shared minus uniform.

C Order-position correlations and score rates for locally deployed open models

Figure 3 focuses on DSV4-F and DSV4-P. Table 7 reports the same order×scoring grid for the five open models, and Table 8 separates the reversed-scoring hard-first penalty by exam length for all seven models. Easy-first presentation is already near a good sequential policy under reversed scoring, so the hard-first change measures escape from an adversarial order. The API models lose 11–22 points at every length while retaining order–position correlations near 0.6. Open models mostly match their easy-first scores; QW-32's correlation falls to -0.25 while its score does not drop, but that departure does not yield gains under aligned scoring either. DQ-7 retains the sequence ($\rho = 0.71$) yet shows no penalty, scoring in the low teens under both orders.

Table 7: Score rate (%) and order–position correlation (in parentheses) for the open models under the baseline prompt, averaged over $N \in \{5, 10, 20\}$. Format matches Figure 3.

Model	Order	Aligned	Fixed	Reversed
DQ-7	Easy first	10.2 (0.74)	13.9 (0.82)	15.9 (0.76)
	Random	13.4 (0.73)	15.5 (0.86)	18.7 (0.75)
	Hard first	12.3 (0.75)	15.4 (0.85)	18.5 (0.71)
DQ-14	Easy first	23.8 (0.69)	27.8 (0.87)	39.6 (0.86)
	Random	27.5 (0.66)	27.7 (0.90)	38.1 (0.65)
	Hard first	25.4 (0.82)	27.5 (0.90)	37.0 (0.34)
QW-8	Easy first	13.9 (0.84)	16.3 (0.90)	24.8 (0.88)
	Random	15.1 (0.52)	18.2 (0.55)	23.9 (0.42)
	Hard first	13.8 (0.42)	19.6 (0.50)	25.4 (0.22)
QW-14	Easy first	21.2 (0.67)	23.4 (0.85)	36.7 (0.80)
	Random	20.5 (0.41)	25.2 (0.45)	35.2 (0.30)
	Hard first	17.4 (0.40)	24.7 (0.34)	35.0 (0.12)
QW-32	Easy first	24.5 (0.64)	28.4 (0.80)	38.7 (0.73)
	Random	22.3 (0.39)	27.6 (0.40)	41.9 (0.24)
	Hard first	22.9 (0.55)	28.8 (0.26)	40.7 (-0.25)

Prompt to the LLM During Inference

Below is a list of questions that you need to answer. Each question has an associated score (as shown in "This question is worth ...") and your goal is to maximize the total score obtained. For each question, you get full score if you answer it correctly, and zero if you answer it incorrectly. You in total have a reasoning budget of *budget* tokens.

Optional strategy hints used in some conditions. We additionally include one or more of the following sentences depending on the experimental setting:

- *Explicit plan:* "Each question has a different difficulty and may require a different amount of reasoning to answer correctly. You should plan your allocation/spending wisely to maximize the total score."
- *Skip hint:* "You may even choose to not answer some questions if you think the cost of answering them is too high compared to the potential score gain."
- *Recheck hint:* "If you think you have more than enough budget, you may also recheck and refine your answers to early questions to further increase the chances of maximizing the total score, but keep in mind that the rechecking also consumes your reasoning budget."

Questions:

```
Q1: [question text] (This question is worth [score] points)
Q2: [question text] (This question is worth [score] points)
...
QN: [question text] (This question is worth [score] points)
```

Think through each question carefully to maximize your total score. Begin your reasoning now.

Figure 4: The prompt used during inference. The exact wording varies slightly depending on the experimental condition through optional strategy hints.

Table 8: Score-rate change (percentage points) from hard-first relative to easy-first presentation under reversed scoring and the baseline prompt. Negative values indicate a loss from the adversarial order.

Model	N=5	10	20
DSV4-P	−21.9	−17.7	−17.1
DSV4-F	−11.5	−22.1	−15.7
DQ-14	+4.5	−7.9	−4.5
QW-14	−5.9	+0.3	+0.4
QW-8	−1.0	+1.1	+1.7
QW-32	+1.8	+2.7	+1.7
DQ-7	+4.3	+1.4	+2.0

D CRUXEval-O details

D.1 Budget calibration

To make the constraint comparable with the mathematics setting, we estimate a typical independent high-budget solution cost from the token usage of correctly answered items and approximately match

$$\frac{B}{N \times \text{average reference cost per question}} \tag{10}$$

across domains. The median high-budget reference cost for CRUXEval-O is approximately 995 tokens per correctly answered question, giving $B = 3{,}000$ as roughly three times that median.

D.2 Budget exhaustion and planning

Runs almost always spend the entire budget, and increasingly so with exam length: the fraction of runs that reach the 3,000-token limit rises from 56%–88% at $N = 10$ to 96%–100% at $N = 20$.

Table 9 reports explicit planning at N=20. Coverage stays within 0.04 of the baseline for every model, and effort–value correlations remain weak under both prompts. This is a weaker coverage response than in mathematics (Section 4.3), where planning raised open-model coverage by 0.14 at N=20: with a budget of roughly three times the

Instruction to the LLM Judge

You are an expert mathematics judge. Your task is to evaluate whether the model's answers match the reference answers for a set of math questions. The reference answers are always correct, and the model's answers may be correct, incorrect, or incomplete.

{original_questions_text}
{ref_text}
{model_text}
{score_text}

Please evaluate each answer for correctness. Consider the following: 1. Mathematical equivalence (e.g., 0.25 == 1/4, sqrt(3) == $\sqrt{3}$) 2. Different forms of the same answer (e.g., simplified vs expanded forms) 3. LaTeX formatting differences should not affect correctness

For each question, award the full points if the answer is correct, and 0 points if incorrect.

Provide your evaluation in the following JSON format:

```
{{
    "evaluations": [
        {{
            "question": 1,
            "correct": true/false,
            "score_awarded": <points>,
            "explanation": "<brief explanation>"
        }},
        ...
    ],
    "total_score": <sum of all awarded scores>,
    "max_score": <sum of all possible scores>
}}
```

Respond with ONLY the JSON, no additional text.

Figure 5: The instruction to `GPT-5` for it to serve as an LLM-as-a-judge.

cost of a single question, spreading it more evenly is largely unavailable even when instructed.

Table 9: Explicit planning on CRUXEval-O at N=20 under fixed scoring and random order. Value correlations use the matched random-scoring condition.

	Coverage		ρ(eff., val.)	
Model	**Base**	**Plan**	**Base**	**Plan**
QW-14	0.21	0.20	+0.03	+0.05
DQ-14	0.22	0.26	+0.02	+0.15
DSV4-F	0.23	0.21	+0.02	−0.01
DSV4-P	0.23	0.26	+0.02	+0.01

E Work set selection under other scoring schemes

Table 2 reports per-model results under random scoring and model means under the remaining schemes. Table 10 gives the per-model breakdown for fixed, aligned, and reversed scoring at N=10 (random order, baseline prompt). Top-density overlap remains at or below chance in every block, while early-position overlap stays well above chance.

F Token attribution details

Per-question tokens are attributed by strict `Q`n: markers in the Phase 1 trace. As mention $\neq$

Table 10: Work-set selection at $N{=}10$ under fixed, aligned, and reversed scoring (random order, base prompt). Asterisks (*) indicate overlaps that are statistically significantly different from the "by chance" overlap based on the 95% confidence interval.

Scoring	Model	Set overlap ratio with W			Work set $\delta = 0$	Token share $\delta = 0$
		By chance	Top-density	Early-position		
Fixed	DQ-7	0.55	0.52	0.88*	0.50	0.59
	DQ-14	0.66	0.59*	0.87*	0.43	0.57
	QW-8	0.50	0.48	0.76*	0.33	0.45
	QW-14	0.62	0.56*	0.84*	0.26	0.32
	QW-32	0.58	0.50*	0.71*	0.21	0.27
	DSV4-F	0.43	0.35*	0.85*	0.20	0.24
	DSV4-P	0.40	0.27*	0.76*	0.16	0.25
	Mean	0.53	0.47*	0.81*	0.30	0.38
Aligned	DQ-7	0.49	0.44	0.90*	0.46	0.57
	DQ-14	0.56	0.52	0.82*	0.49	0.61
	QW-8	0.49	0.45	0.78*	0.32	0.45
	QW-14	0.61	0.56*	0.80*	0.25	0.31
	QW-32	0.49	0.47	0.57	0.20	0.26
	DSV4-F	0.48	0.40*	0.89*	0.23	0.28
	DSV4-P	0.44	0.41	0.73*	0.18	0.23
	Mean	0.51	0.46*	0.78*	0.31	0.39
Reversed	DQ-7	0.50	0.51	0.85*	0.40	0.46
	DQ-14	0.55	0.51	0.75*	0.46	0.58
	QW-8	0.55	0.60	0.71*	0.25	0.36
	QW-14	0.64	0.66	0.70	0.21	0.27
	QW-32	0.62	0.66	0.64	0.18	0.20
	DSV4-F	0.46	0.35*	0.84*	0.23	0.26
	DSV4-P	0.44	0.41	0.73*	0.20	0.30
	Mean	0.54	0.53	0.75*	0.28	0.35

work: opening enumeration passes can name every question with minimal token mass. We therefore use (i) *zero-token* rate (no attributed tokens), (ii) *work set* ($\geq$200 tokens or $\geq$2 segments), and (iii) token-centroid rank *within the work set* for solving order. First-mention order inflates sequentiality for DeepSeek models, which centroid rank corrects; the work-set restriction additionally prevents a question that is only named in an enumeration pass—whose centroid sits at that early offset—from being ranked ahead of questions that received real effort. The restriction matters at the head of the ranking—with all mentioned questions, the earliest-centroid question is a mention-only question in up to 56% of traces—but barely moves the aggregate correlations: ranking all mentioned questions instead gives order$\sim$position 0.70 (vs. 0.69), order$\sim$difficulty 0.11 (vs. 0.07), and order$\sim$points -0.03 (vs. -0.09).