

DoubtLess: Training-Free Contrastive Decoding for Efficient Reasoning

Minghao Luo

The Chinese University of Hong Kong
luominghao@link.cuhk.edu.hk

Liang Chen*

EPFL
l.chen@epfl.ch

Abstract

Reasoning models achieve strong performance through long chains of thought (CoT), but often spend more tokens than needed to solve a problem. Existing ways to shorten the CoT require additional training for each model, add a fixed steering vector to the model’s hidden states, or use decoding heuristics such as stopping early or banning words like *wait*, which can reduce accuracy. We propose DoubtLess, a training-free decoding method in which a much smaller reasoning model steers the large one towards brevity. The small model reads the large model’s CoT so far twice, once prompted to be brief (the *brief thinker*) and once prompted to be verbose (the *verbose thinker*), and the difference between its two logit vectors, scaled by a strength, is added to the large model’s logits at every step of the CoT. On seven benchmarks of competition mathematics, graduate-level science and competitive programming, DoubtLess shortens the CoT of Qwen3-32B by 31% with unchanged accuracy, outperforming five inference-time baselines. We also show that DoubtLess reduces thinking tokens per turn by 20% in a self-debugging coding agent with unchanged task performance. It also generalizes to two other model families with unchanged accuracy. When combined with training-based compression, it further shortens the CoT of a model already trained for brevity. Ablations and behavioral analysis show that DoubtLess reduces repeated checking, with most token savings coming after the correct answer first appears in the CoT. Our code is available at <https://github.com/leoluolol/DoubtLess>.

1 Introduction

Reasoning models such as Qwen3 and DeepSeek-R1 write a chain of thought (CoT) between the tokens `<think>` and `</think>` before they answer.

*Corresponding author.

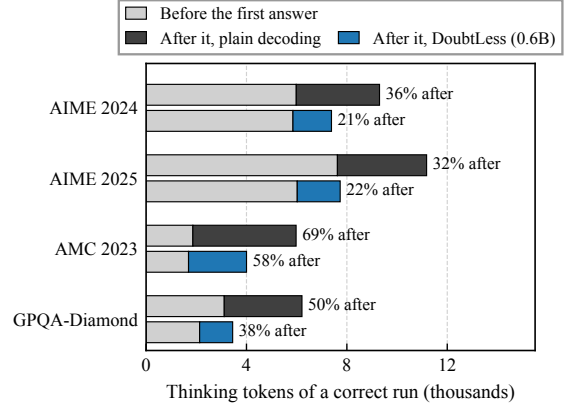


Figure 1: Qwen3-32B: mean thinking tokens of a correct run before and after the correct answer first appears in the CoT, under plain decoding (the model as released) and under DoubtLess. Of the tokens DoubtLess removes, 65–89% come after the answer (Table 11).

On a hard problem the CoT holds most of the output, and a large part of it re-derives and re-verifies a result the model has already reached (Chen et al., 2024; Sui et al., 2025). Under plain decoding, that is, sampling from the model as released, a correct run of Qwen3-32B writes 32–69% of its CoT after the correct answer first appears (Figure 1). Not all of this checking is redundant. A check made while the model works, or a first check of its answer, can catch an error, but repeating a check that has already been made adds tokens without adding information. A method that shortens the CoT should remove the redundant checks and keep the necessary ones.

Existing methods for shortening the CoT fall into two groups. Training-based methods add a length penalty or a token limit to reinforcement learning (Aggarwal and Welleck, 2025; Hou et al., 2025; Luo et al., 2025) or fine-tune on short traces (Chen et al., 2024). Both change the weights, so every new model needs another round of training. Inference-time methods ask for brevity in the

prompt (Renze and Guven, 2024; Han et al., 2025; Xu et al., 2025), block words such as *wait* that open another round of checking (Wang et al., 2025), and the CoT once an answer looks ready or a token budget is spent (Yang et al., 2025b; Song et al., 2026; Muennighoff et al., 2025), or add a fixed steering vector to the model’s hidden states (Azizi et al., 2026; Chen et al., 2025; Lin et al., 2025). Prompting depends on how closely the model follows the instruction. Keyword bans suppress the same words across contexts, activation steering adds the same vector to the hidden states at every step, and early stopping ends the CoT once its criterion is met. These controls can shorten useful reasoning as well as redundant checks.

In this paper we propose DoubtLess, a training-free decoding method that steers a reasoning model with a much smaller reasoning model that shares its tokenizer (Figure 2). At every step of the CoT, the small model reads the text the large model has written so far twice: once after an instruction to go straight to the answer, as the *brief thinker*, and once after an instruction to double-check every step, as the *verbose thinker*. The difference between the two logit vectors, multiplied by a strength α , is added to the large model’s logits. The small model only suggests a direction, towards what the brief thinker would write next and away from what the verbose thinker would. The large model still decides which tokens are possible, and the CoT can end only where the large model itself would consider ending it (Section 2). Inspired by DExperts (Liu et al., 2021), DoubtLess steers the large model through the vocabulary it shares with a small model, but needs no trained expert, since both sides of the contrast come from one off-the-shelf small model under two instructions.

Two properties set DoubtLess apart from prompting, keyword bans, activation steering and early stopping. First, the contrast is dynamic. It is recomputed from the current context at every step, so a word such as *But* is pushed down after an answer and left almost untouched before it, whereas a keyword ban blocks it everywhere and activation steering applies one fixed vector at every step. Second, the brief and verbose thinkers disagree most after the correct answer first appears in the CoT, where the brief thinker would wrap up and the verbose one would check again (Section 5.1). Most of the removed tokens come from this later checking.

We compare DoubtLess with five inference-time baselines on seven benchmarks of competition

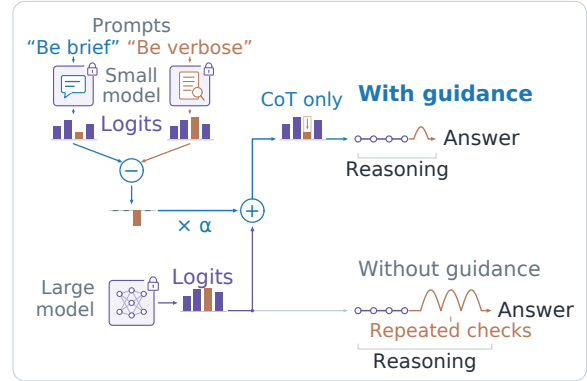


Figure 2: DoubtLess. A small model that shares the tokenizer reads the large model’s CoT so far twice, once prompted to be brief (brief thinker) and once prompted to be verbose (verbose thinker); the difference between the two logit vectors, scaled by α , is added to the large model’s logits while it writes the CoT.

mathematics, graduate-level science and competitive programming, with the model’s recommended sampling settings and three seeds per problem. With Qwen3-32B steered by Qwen3-0.6B, DoubtLess removes 31% of the thinking tokens with unchanged accuracy, more than any baseline. It keeps accuracy where the baselines lose it. On AIME 2025, for example, blocking the words that open another round of checking costs 14 points, while DoubtLess costs none. DoubtLess also extends to an agent, a model trained for brevity and two other model families. It reduces thinking per turn by 20% in a self-debugging coding agent and further shortens the CoT of a model trained for brevity with LACONIC (Liu et al., 2026). On AIME 2025 and AMC 2023, it removes 25% and 40% of the thinking of EXAONE-4.0 and Granite-4.2, each steered by a small model of its own family, where a concise prompt removes 8% and 14%. Finally, our ablations show that the strength sets how much is removed, that a negative strength lengthens the CoT instead, and that the shortening depends on which tokens the small model pushes down.

Our main contributions are as follows:

- DoubtLess, a training-free decoding method in which an off-the-shelf small model, run under a brief and a verbose instruction, supplies a logit direction that shortens a reasoning model’s CoT.
- An evaluation in which DoubtLess removes more thinking than five inference-time baselines on seven benchmarks with unchanged accuracy, and shortens the CoT of a self-debugging coding agent and of two other model families.

- An analysis showing that the two thinkers disagree mainly after the answer first appears in the CoT, which is where most of the removed tokens lie.

2 Method

2.1 Preliminaries

Decoding a reasoning model. Given a prompt x , a language model M with vocabulary $\mathcal{V}$ generates a response $y = (y_1, \dots, y_n)$ one token at a time,

$$p_M(y \mid x) = \prod_{t=1}^n p_M(y_t \mid x, y_{<t}). \quad (1)$$

At each step it returns a logit vector $z_t \in \mathbb{R}^{|\mathcal{V}|}$, and its next-token distribution at temperature T is

$$p_M^T(\cdot \mid x, y_{<t}) = \text{softmax}(z_t/T), \quad (2)$$

with $p_M = p_M^1$. A reasoning model writes its CoT between `<think>` and `</think>`, then its answer.

Sampling. Sampling draws y_t from Equation (2), usually truncated to the most probable tokens. The top- p set $\mathcal{V}_p(q)$ of a distribution q over $\mathcal{V}$ is the smallest set of tokens such that (Holtzman et al., 2020)

$$\sum_{v \in \mathcal{V}_p(q)} q(v) \geq p, \quad (3)$$

and the top- k set holds the k most probable tokens. Truncating to a set S sets every other logit to $-\infty$ (Liu et al., 2021),

$$[z]_S(v) = \begin{cases} z(v) & \text{if } v \in S, \\ -\infty & \text{otherwise,} \end{cases} \quad (4)$$

so that the softmax renormalizes over S . A sampler applies temperature, top- k and top- p in turn. For thinking, Qwen3 recommends 0.6, 20 and 0.95.

2.2 DoubtLess

Let D be a much smaller reasoning model that shares the tokenizer of M (Figure 2), and write z_{base} for the logits of M at the current step.

Definition. At every decoding step, D reads the same prefix as M (the problem, the instruction to reason step by step, and the tokens generated so far) under two versions of the instruction that differ in one trailing sentence. As the *brief thinker* it is prompted to go straight to the answer without re-deriving or re-verifying anything. As the *verbose thinker* it is prompted to double-check every step

Before the first answer

while the model works towards it

Now, we need divisors of 56 greater than 16. Looking at the list: From the divisors: 28 and 56 are greater than 16. Wait, 14 is 14, which is less than 16, so the next ones are 28 and 56. [...] So the possible bases are 21 and 49. Both are greater than 9, so they satisfy the condition. Therefore, the sum is $21 + 49 = 70$.

After the first answer

once 70 has been stated

Wait, but let me check if there are any other divisors. Wait, 56's divisors are only up to 56. But perhaps negative divisors? [...] But let me double-check my logic here. [...] But let me check if there are other divisors. For example, maybe 56 divided by something else? Wait, divisors are only those numbers. Let me check 56's divisors again: [...]

Figure 3: Before the answer, the direction leaves the CoT nearly untouched. After it, the words that open another check are pushed down. The CoT that plain decoding writes for AIME 2025 problem 1, read by Qwen3-0.6B under both instructions. **Red**: down-weighted; **green**: up-weighted (threshold in Appendix I); [...] marks omitted text.

and to consider alternatives (the two sentences are given in Appendix B). With z_{brief} and z_{verbose} for the two resulting logit vectors, the next token is sampled from

$$z = z_{\text{base}} + \alpha (z_{\text{brief}} - z_{\text{verbose}}), \quad (5)$$

restricted to the top- p set of $\text{softmax}(z_{\text{base}})$, the large model's distribution at temperature one (Equations (3) and (4)). We call $z_{\text{brief}} - z_{\text{verbose}}$ the *direction* of the small model: the way its next-token expectations move when its instruction changes from thorough to brief.

What the direction does. Sampling from Equation (5) at temperature T draws from the large model's own distribution at that temperature, p_M^T (Equation (2)), reweighted by the ratio of the brief thinker's probability for the next token to the verbose thinker's:

$$p(v) \propto p_M^T(v \mid x, y_{<t}) \left[\frac{p_D(v \mid x, y_{<t}, \text{brief})}{p_D(v \mid x, y_{<t}, \text{verbose})} \right]^{\alpha/T} \quad (6)$$

(a difference of logits is a difference of log-probabilities up to a constant shared by all tokens, and the temperature divides the whole of z ; Appendix C). A token that the brief thinker expects more than the verbose thinker becomes more likely, one that the verbose thinker expects more becomes less likely, and where the two thinkers agree nothing changes.

Table 1: DoubtLess removes the most thinking tokens overall. Qwen3-32B, three seeds per problem. Plain decoding: accuracy (%) and mean thinking tokens; other rows: change against plain decoding, in points and in % (intervals in Table 7). Overall: the 528 problems of the seven benchmarks together. Bold and underlined: largest and second-largest removal among the six methods. Agent: pass rate and thinking tokens per turn on HumanEval+.

		Mathematics					Science	Code	Agent	Overall
		AIME 2024	AIME 2025	HMMT 2025	AMC 2023	Olympiad- Bench	GPQA- Diamond	LiveCode- Bench	HumanEval+ agent	All seven
Plain decoding	Acc. Tokens	81.1 12,152	72.2 15,606	53.3 16,817	96.7 6,130	70.7 12,246	68.7 7,354	66.3 11,511	88.4 2,718	70.8 10,254
Concise prompt	Acc. change	+2.2	+3.3	-6.7	-0.8	-2.0	-1.0	-0.7	+1.4	-1.0
	Token change	-30%	<u>-28%</u>	-13%	<u>-31%</u>	<u>-25%</u>	<u>-40%</u>	-8%	-13%	<u>-25%</u>
NoWait	Acc. change	-3.3	-14.4	-6.7	-2.5	-1.0	-0.3	-4.0	-0.8	-2.7
	Token change	-21%	-23%	<u>-19%</u>	-24%	-17%	-21%	<u>-18%</u>	<u>-29%</u>	-20%
DEER	Acc. change	+1.1	+2.2	0.0	+0.8	+0.7	0.0	-4.7	+4.5	-0.5
	Token change	-8%	-10%	-6%	-20%	-12%	-2%	-17%	-41%	-10%
ThinkBrake	Acc. change	-1.1	0.0	-1.1	+0.8	-3.3	-2.0	-4.7	-6.9	-2.3
	Token change	-4%	-3%	-4%	-3%	-4%	-37%	-44%	-27%	-21%
ASC	Acc. change	+1.1	+1.1	-2.2	+1.7	-1.7	+1.2	-0.3	-1.4	+0.2
	Token change	-12%	-8%	+3%	-11%	-4%	-4%	-4%	-5%	-5%
DoubtLess (0.6B)	Acc. change	+2.2	0.0	+3.3	+0.8	-1.7	-2.4	-0.3	+2.0	-0.9
	Token change	<u>-27%</u>	-29%	-31%	-34%	-27%	-46%	-16%	-20%	-31%

When it acts. The two thinkers read the same text and differ only in the instruction, so they disagree only where the instruction matters (one CoT in Figure 3; the analysis in Section 5.1). While the model is still working towards an answer, both expect the derivation to continue, and the CoT is left almost unchanged: in the upper box of Figure 3 only two tokens are shaded, and the *Wait* that opens a check inside the derivation is not one of them. Once an answer has been stated, the verbose thinker expects another round of checking and the brief thinker expects the CoT to end, so the words that start a new check, such as *But*, *Let* and *maybe* in the lower box, are down-weighted by a factor of five or more. Unlike NoWait, DoubtLess therefore bans no word. It makes another round of checking less likely to begin once an answer is in place, and what the CoT loses is those rounds. Whether the next sentence starts another check or continues the derivation is largely a matter of form, which a small model can judge even on a problem it could not solve.

Where the contrast applies. The contrast acts only on the CoT. The answer after `</think>` is decoded by the large model alone. Within the CoT, the large model keeps its own logits for `</think>` and the end-of-sequence token and its own top- p candidates, so the small model can only reorder tokens the large model was already willing to write (Algorithm 1, Appendices A and C).

3 Experimental setup

Models. The large model M is Qwen3-32B (Yang et al., 2025a) in bfloat16 and the small model D is Qwen3-0.6B. We write DoubtLess (0.6B) for this pair. Section 4.2 repeats the comparison with plain decoding and the concise prompt on two other families: EXAONE-4.0-32B steered by EXAONE-4.0-1.2B (LG AI Research, 2025) and Granite-4.2-30B steered by Granite-4.2-3B (IBM Granite Team, 2026). Section 6 also uses Qwen3-1.7B and Qwen3-4B as the small model.

Benchmarks. We use seven benchmarks that are the standard testing ground of reasoning models: four mathematics competitions, AIME 2024 and AIME 2025 (30 problems each; Mathematical Association of America, 2025), HMMT February 2025 (30; HMMT, 2025) and AMC 2023 (40; Mathematical Association of America, 2023); graduate-level science questions from GPQA-Diamond (198; Rein et al., 2024); olympiad problems from OlympiadBench (100, drawn round-robin across subjects; He et al., 2024); and programming problems from LiveCodeBench (a fixed subset of 100 from release 6, drawn round-robin across the three difficulty levels; Jain et al., 2025). Every benchmark is scored with its published answer key, and LiveCodeBench by running the program against the test cases (Appendix E).

Baselines. We compare DoubtLess with plain decoding and with five training-free methods of the kinds in Section 1: a prompt, a ban on words, two rules that end the CoT early and a steering vector. **Concise prompt** asks the large model itself to be brief: the brief thinker’s sentence, which tells the model to go straight to the answer and not to re-derive or re-verify anything, is appended to the large model’s own instruction. How much it removes depends on how closely the large model follows the instruction. **NoWait** (Wang et al., 2025) blocks 17 words that open another round of checking (*wait*, *alternatively*, *hmm*, *but*, *however* and others) throughout the generation, by adding a bias of -100 to the logits of their single-token forms (89 token ids). **DEER** (Yang et al., 2025b) ends the CoT early. At each of the first ten occurrences of *Wait* it induces a trial answer, ends the CoT if the geometric mean of the trial’s token probabilities exceeds 0.95, and otherwise discards the trial and lets the CoT continue. It also ends the CoT once it reaches 0.8 of the budget. A DEER run is a prefix of the plain-decoding run followed by `</think>` and a freshly sampled answer, so we evaluate it on the stored plain-decoding runs with the same engine, sampler and seeds (Appendix D). **ThinkBrake** (Song et al., 2026) ends the CoT at a sentence boundary once `</think>` comes within a logit margin of 0.1 of the most likely token, read after a single newline as in the authors’ code, and is evaluated like DEER (Appendix D). **ASC** (Azizi et al., 2026) adds a fixed steering vector to the large model’s hidden states after one layer, at every decoding step. We use the training-free version in the authors’ released code, in which the vector is the mean difference between the hidden states at the last token of a verbose and of a concise CoT, here Qwen3-32B’s own CoTs on 100 MATH-500 problems (Hendrycks et al., 2021; Lightman et al., 2024) under its plain instruction and under the brief thinker’s sentence. It is added after layer 57 of 64, the layer the authors use for the 64-layer QwQ-32B, with the strength chosen on AIME 2026 (Mathematical Association of America, 2026) as for DoubtLess (Appendix D). We also decode the random-target control of Section 6.1.

Implementation details. All methods sample with Qwen3’s recommended thinking settings, temperature 0.6, top- p 0.95 and top- k 20, under a thinking budget of 32,768 tokens (24,576 on GPQA-Diamond). A run that reaches the budget without

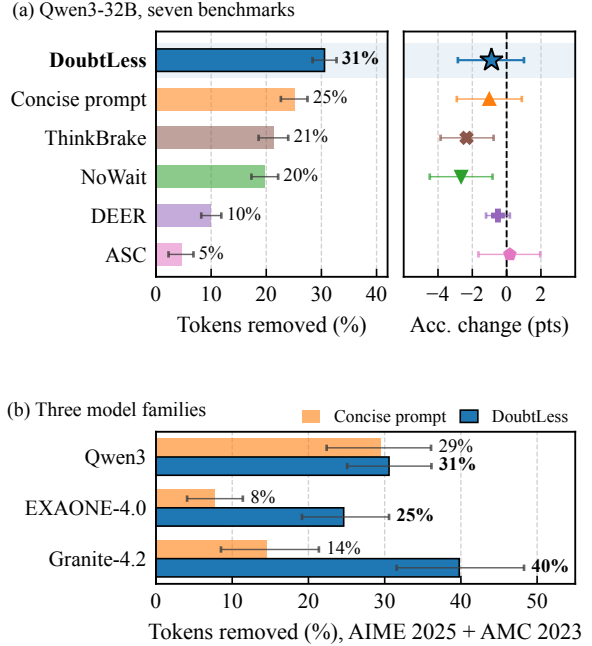


Figure 4: DoubtLess removes the most thinking tokens with unchanged accuracy. (a) Percentage of thinking tokens removed and accuracy change against plain decoding, for Qwen3-32B on the seven benchmarks. (b) When evaluated on other model families, DoubtLess removes far more than the concise prompt (AIME 2025 and AMC 2023, each large model steered by a small model of its own family). Three seeds per problem, 95% intervals (Tables 7 and 8).

ending its CoT counts as wrong. Every problem is decoded with three seeds. We call one decoding of one problem under one seed a *run*, and accuracy is the mean over runs. DoubtLess uses $\alpha=5$ (the best trade-off on AIME 2026; Appendix F), one value for every model pair and every benchmark. Section 6.2 varies it. All methods run in the same vLLM engine, with DoubtLess and the control computed inside it (Appendix D).

Evaluation. We report accuracy and the change in the number of thinking tokens, the tokens of the CoT, against plain decoding. The change is taken over all problems, solved or not, with a 95% bootstrap interval over problems. Appendix G gives the formula and the change over the problems both methods solve, the measure of earlier work. Since a run that reaches the budget counts as wrong, we also report the share of such runs.

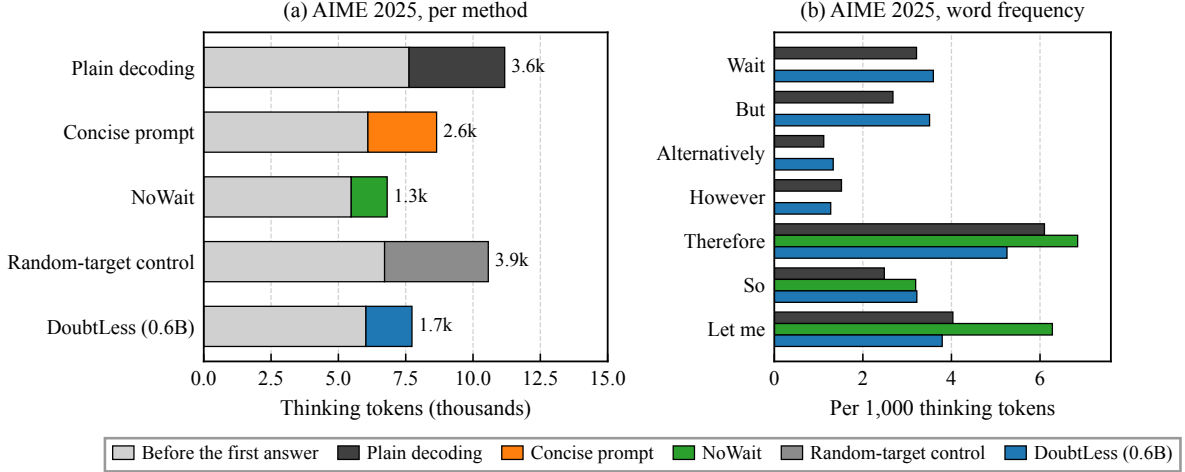


Figure 5: DoubtLess removes mostly the thinking after the first answer, and the words of hesitation stay. AIME 2025. (a) Mean thinking tokens of a correct run before (light) and after (dark) the first occurrence of the correct answer, per method; number: tokens after it. (b) Selected words per thousand thinking tokens.

4 Results

4.1 Main results

DoubtLess shortens the CoT more than any baseline. Across the seven benchmarks, DoubtLess removes 31% of the thinking tokens with unchanged accuracy (Table 1, overall column; intervals in Table 7). This is the largest reduction among the six methods (Figure 4a). The concise prompt comes closest, but DoubtLess removes more than it on six of the seven benchmarks. The other baselines remove less, and ThinkBrake and NoWait lose more accuracy than DoubtLess.

The reduction holds on harder problems.

DoubtLess shortens the CoT significantly on every benchmark (Table 7). The baselines are uneven. On mathematics, the concise prompt removes less on harder benchmarks, from 31% on AMC 2023 to 13% on HMMT 2025, the easiest and the hardest for plain decoding, while DoubtLess still removes 31% on HMMT 2025. ThinkBrake cuts the CoT on science and code but barely on mathematics.

DoubtLess keeps accuracy on every benchmark.

In contrast, NoWait, DEER and ThinkBrake each lose accuracy significantly on at least one benchmark (Table 7): NoWait on AIME 2025 and LiveCodeBench, DEER on LiveCodeBench, and ThinkBrake on OlympiadBench and LiveCodeBench. DoubtLess also lowers the share of runs that reach the budget, from 2.4% under plain decoding to 0.4%.

4.2 Other model families

DoubtLess carries over to EXAONE-4.0 and Granite-4.2, each steered by a small model of its own family, and shortens their CoT without losing accuracy (Figure 4b, Table 8 in Appendix H). The concise prompt, however, varies from model to model. It comes close to DoubtLess on Qwen3-32B, but on the other two DoubtLess removes about three times as much. This depends on whether the model complies. A prompt can only ask the large model to be brief, and models differ in how far they follow such a request, whereas DoubtLess asks the large model for nothing and changes its next-token distribution directly.

4.3 Coding agent

We extend DoubtLess to a self-debugging coding agent on HumanEval+ (Chen et al., 2021; Liu et al., 2023), which writes a program, runs the tests and gets one repair turn on failure. DoubtLess is applied inside the CoT of every turn (Appendix J). Over the 164 problems, the agent thinks 20% less per turn and improves its pass rate by 2.0 points over plain decoding (Table 1, Agent column). DEER removes 41% and improves the pass rate by 4.5 points. It ends the CoT once 80% of the turn’s thinking budget is spent, which leaves room for a program in every turn, whereas under plain decoding 16–20% of the turns use the whole budget for thinking and write no program (Appendix J). NoWait and ThinkBrake remove more but lower the pass rate, ThinkBrake by 6.9 points.

Plain decoding

3,414 thinking tokens, 2,369 after the first answer

[...] **Therefore, the sum is $21 + 49 = 70$.**

Wait, but let me **check if there are any other divisors**. [...]

But let me **check if there are other divisors**. [...]

Let me **check 56's divisors again**: [...]

Is there a possibility that there are **more divisors** if we consider other factorizations? [...]

But let me **check if there's a possibility of other divisors** [...]

Therefore, answer is **70**. [...]

DoubtLess (0.6B)

1,461 thinking tokens, 523 after the first answer

[...] **Their sum is $21 + 49 = 70$.**

Wait, is that all? Let me **verify if there are any other divisors**. [...]

But let me **check with actual numbers** to make sure. [...]

But let me just make sure there isn't a **negative divisor** or something? [...]

Therefore, the answer is **70**. [...]

Figure 6: After the answer, DoubtLess checks briefly and stops, while plain decoding gets stuck repeating the same check. AIME 2025 problem 1, first seed: the CoT under each method from the first statement of the answer (bold) to its end. Checks are in red, and [...] marks omitted text; full CoTs in Appendix L.

5 Analysis and discussion

5.1 What DoubtLess removes

Most removed tokens follow the first appearance of the answer. We split each run at the point where the correct answer first appears in the CoT (Appendix I). DoubtLess cuts the thinking after this point by half and the thinking before it by a fifth (Figure 5a), so most of the removed tokens lie after it on each of the six benchmarks on which the answer can be located (Table 11). In Figure 6, both methods reach the answer after about a thousand tokens, and plain decoding then returns to the same check again and again. Useful checks remain. On AIME 2025 problem 11, an algebraic check restores a wrongly excluded root, which is then included in the final answer (Appendix I).

The two thinkers disagree after the answer. Read by the small model under both instructions, correct plain-decoding runs on AIME 2025 show a mean divergence between the two next-token distributions seven times larger after the answer than before it (Figure 7a), and the same holds under human-annotated answer boundaries (Appendix I). After the answer the direction pushes down the words that open another round, such as *But*, *How-*

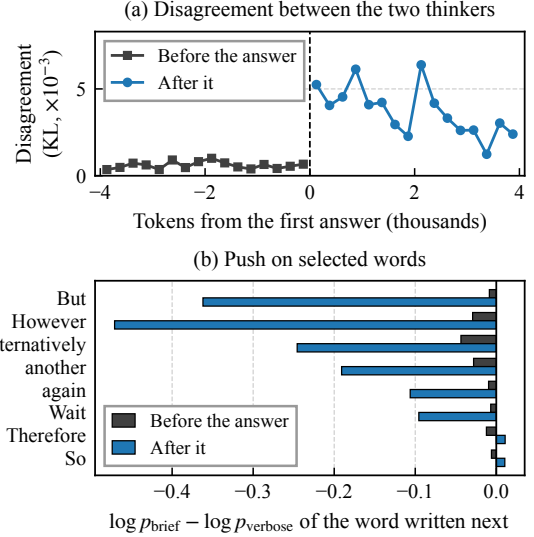


Figure 7: The two thinkers disagree after the answer. Correct plain-decoding runs on AIME 2025, read by the small model under both instructions. (a) Divergence of the two next-token distributions, aligned at the first automatic match to the correct answer. (b) Brief minus verbose log-probability of selected words, before and after the answer.

ever and *Alternatively*, which it leaves almost untouched before the answer (Figure 7b).

The words of hesitation stay. *Wait*, *But* and *Alternatively* occur at least as often per thousand thinking tokens under DoubtLess as under plain decoding, whereas NoWait suppresses them (Figure 5b). DoubtLess removes rounds of checking, and the checks that remain still begin with these words (Figure 6).

5.2 DoubtLess complements training-based compression

Training-based methods shorten the CoT by changing the weights, while DoubtLess acts only at decoding time, so the two can be combined. We test this on a Qwen3-32B checkpoint fine-tuned with LACONIC (Liu et al., 2026), which adds an adaptive length penalty to the task reward during reinforcement learning, and steer it with Qwen3-0.6B at $\alpha=5$. Across mathematics, science and coding-agent benchmarks, DoubtLess further reduces thinking tokens by 29–52% with unchanged task performance (Table 2). DoubtLess thus complements training-based compression and applies to an already compressed model without further training.

Table 2: DoubtLess further compresses a training-based compression model, Qwen3-32B fine-tuned with the adaptive length penalty of LACONIC (LACONIC-Qwen3-32B-3000). LACONIC: accuracy (%) and mean thinking tokens of this model; +DoubtLess: change in points (95% intervals) and in %. Three seeds per problem. Agent: pass rate and thinking tokens per turn.

		Mathematics	Science	Agent
		Olympiad-Bench	GPQA-Diamond	HumanEval+agent
LACONIC	Acc.	64.7	65.5	91.9
	Tokens	5,718	3,483	1,725
+DoubtLess (0.6B)	Acc. change	−1.3	−1.7	−1.2
		[−4.7, +1.7]	[−4.9, +1.5]	[−4.3, +1.6]
	Token change	−42%	−52%	−29%

5.3 Inference cost

We measure end-to-end inference latency on a single B200 GPU, decoding one problem at a time on 14 problems, two from each of the seven benchmarks. DoubtLess takes 119.4 seconds per problem on average, compared with 156.2 seconds for plain decoding and 137.5 seconds for the concise prompt, reducing latency by 24% and 13%, respectively. The small model’s two forward passes of 0.6B parameters per step add about 4% to the FLOPs of the 32B model.

6 Ablation study

We vary the direction and the strength of the contrast and the size of the small model.

6.1 Effect of the contrast direction

The *random-target* control tests whether the shortening comes from which tokens the small model pushes down or from perturbing the logits at all. At every step it keeps the adjustment values $\alpha(z_{\text{brief}} - z_{\text{verbose}})$ but assigns them to randomly chosen tokens of the top- p set, so the size and scope of the change stay the same and only its direction is lost. With random targets, the thinking length stays within 8% of plain decoding on every benchmark and changes by 1% overall, against 31% for DoubtLess (Table 7, Figure 10). The shortening therefore comes from the direction the small model supplies.

6.2 Effect of the strength

On AIME 2025, DoubtLess removes more as the strength grows, with unchanged accuracy up to the default $\alpha=5$, and at $\alpha=10$ it loses 7.8 points (Figure 8). A negative strength, $\alpha=-5$, lengthens the CoT by 26% instead, so the strength also sets the direction of the change.

6.3 Effect of the small-model size

We show that a larger small model at the same α acts as a stronger contrast. With Qwen3-1.7B or Qwen3-4B in place of Qwen3-0.6B, DoubtLess removes close to 60% of the thinking tokens on AIME 2025 and, like $\alpha=10$ with the 0.6B model, goes past the point where accuracy holds (Table 5). With Qwen3-32B itself as the small model (at $\alpha=1$), DoubtLess removes 26%, against 29% for the 0.6B model, so the smallest and cheapest model of the family, our default, gives up nothing.

7 Related work

Overthinking. Reasoning models spend a large share of their tokens re-deriving and re-verifying results they have already reached, even on simple problems (Chen et al., 2024; Sui et al., 2025), and their token spend tracks the difficulty of a problem only loosely (Pu et al., 2025). Section 5.1 shows that the tokens DoubtLess removes come mainly from the checking after the answer.

Shortening the CoT at inference time. Methods that leave the model unchanged end the CoT once a budget is spent or the answer appears settled (Muenighoff et al., 2025; Yang et al., 2025b; Song et al., 2026; Fu et al., 2024; Pu et al., 2025), skip it (Ma et al., 2025), block the words that open another round of checking (Wang et al., 2025), or ask for brevity in the prompt (Renze and Guven, 2024; Han et al., 2025; Xu et al., 2025), which works only as far as the model follows the request (Han et al., 2025; Aggarwal and Welleck, 2025). Activation steering adds a fixed vector to the hidden states, taken from concise and verbose CoTs (Azizi et al., 2026; Huang et al., 2025) or from execution and reflection steps (Chen et al., 2025), with a strength that can follow a signal read from the model (Lin et al., 2025; Li et al., 2026b). Budget Guidance (Li et al., 2026a) reweights the next-token distribution with a trained length predictor. DoubtLess also reweights the next-token distribution, but its direction comes from an off-the-shelf small model, is recomputed from the context at every step and leaves the large model’s own logit for `</think>` untouched.

Training-based compression. Training-based methods add a length term or a token limit to reinforcement learning (Aggarwal and Welleck, 2025; Hou et al., 2025; Luo et al., 2025; Liu et al., 2026) or fine-tune on short traces (Chen et al., 2024).

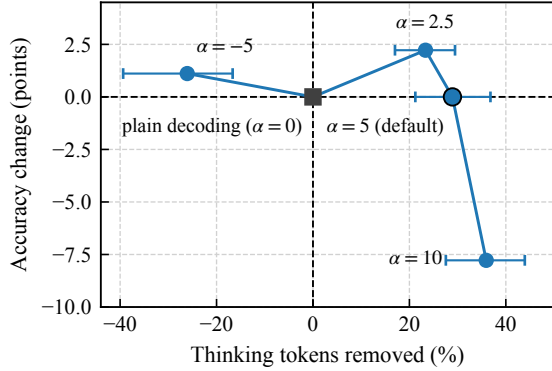


Figure 8: Larger strengths remove more thinking tokens, and accuracy holds up to the default $\alpha=5$. AIME 2025, three seeds per problem, 95% intervals (Table 4).

Each model needs its own training run, and the trade-off between length and accuracy is fixed once training ends. DoubtLess leaves the model unchanged, sets the trade-off with one strength at decoding time and can be added to a model trained this way (Section 5.2).

Contrastive decoding. Contrastive decoding scores the next token by the difference between an expert and a weaker amateur model (Li et al., 2023; O’Brien and Lewis, 2023), and DExperts and proxy-tuning steer a large model with small models that share only its vocabulary (Liu et al., 2021, 2024). ThinkLogit (Zhang et al., 2026) and ProxyThinker (Xiao et al., 2025) add the difference between a small reasoning model and its base to make a large model reason more, Thinking by Subtraction (Tang et al., 2026) contrasts a model with a masked copy of its own context, and PromptCD (Bi et al., 2026) and Dong et al. (2026) contrast two prompts on the steered model itself. DoubtLess draws the contrast from an off-the-shelf small model under two instructions and uses it to shorten the CoT. Small models also work beside large ones as drafters in speculative decoding (Leviathan et al., 2023), as token suggesters in Nudging (Fei et al., 2025) and as sentence completers in FoReal decoding (Li et al., 2025).

8 Conclusion

DoubtLess adds the difference between a small reasoning model’s brief and verbose thinkers to a large reasoning model’s logits while it writes its CoT. On seven benchmarks it removes 31% of the thinking tokens of Qwen3-32B with unchanged accuracy, more than five inference-time baselines.

It also shortens the CoT of a coding agent, and on two other model families it removes about three times as much as a concise prompt. Combined with training-based compression, it further shortens the CoT of a model already trained for brevity. Our analyses support the explanation that DoubtLess reduces redundant checking while retaining useful verification.

References

- Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling (COLM)*. ArXiv:2503.04697.
- Seyedarmin Azizi, Erfan Baghaei Potraghloo, Souvik Kundu, and Massoud Pedram. 2026. [Activation steering for chain-of-thought compression](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 36676–36687.
- Baolong Bi, Yuyao Ge, Shenghua Liu, Yuchen He, Siqian Tong, Lizhe Chen, Lingrui Mei, Zehao Li, Yiwei Wang, Yujun Cai, Ming-Hsuan Yang, and Xueqi Cheng. 2026. PromptCD: Test-time behavior enhancement via polarity-prompt contrastive decoding. *arXiv preprint arXiv:2602.20696*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. [SEAL: Steerable reasoning calibration of large language models for free](#). In *Second Conference on Language Modeling (COLM)*.
- 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. 2024. Do NOT think that much for $2+3=?$ on the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*.
- Yijiang River Dong, Tiancheng Hu, Zheng Hui, and Nigel Collier. 2026. Steer model beyond assistant: Controlling system prompt strength via contrastive decoding. *arXiv preprint arXiv:2601.06403*.
- Yu Fei, Yasaman Razeghi, and Sameer Singh. 2025. Nudging: Inference-time alignment of LLMs via guided decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*. ArXiv:2410.09300.
- Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang.

2024. Efficiently scaling LLM reasoning with Certaindex. *arXiv preprint arXiv:2412.20993*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. **Token-budget-aware LLM reasoning**. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. **OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems**. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. ArXiv:2103.03874.
- HMMT. 2025. **HMMT February 2025**. <https://www.hmmt.org/www/archive/282>.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *Proceedings of ICLR*.
- 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. *arXiv preprint arXiv:2504.01296*.
- Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. 2025. **Mitigating overthinking in large reasoning models via manifold steering**. In *Advances in Neural Information Processing Systems*.
- IBM Granite Team. 2026. **Granite 4.2 language models**. <https://huggingface.co/ibm-granite/granite-4.2-30b>.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. **Live-CodeBench: Holistic and contamination free evaluation of large language models for code**. In *International Conference on Learning Representations*, pages 58791–58831.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. **Efficient memory management for large language model serving with PagedAttention**. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, New York, NY, USA. ACM.
- Hynek Kydlíček. 2025. **Math-Verify: Math verification library**. <https://github.com/huggingface/Math-Verify>.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.
- LG AI Research. 2025. **EXAONE 4.0: Unified large language models integrating non-reasoning and reasoning modes**. *arXiv preprint arXiv:2507.11407*.
- Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. 2026a. **Steering LLM thinking with budget guidance**. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 37437–37451.
- Ming Li, Zhengyuan Yang, Xiyao Wang, Dianqi Li, Kevin Lin, Tianyi Zhou, and Lijuan Wang. 2025. What makes reasoning models different? Follow the reasoning leader for efficient decoding. *arXiv preprint arXiv:2506.06998*.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Yulin Li, Tengyao Tu, Li Ding, Junjie Wang, Hui-Ling Zhen, Yixin Chen, Yong Li, and Zhuotao Tian. 2026b. **Efficient reasoning with balanced thinking**. In *International Conference on Learning Representations*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*. ArXiv:2305.20050.
- Zhengkai Lin, Zhihang Fu, Ze Chen, Chao Chen, Liang Xie, Wenxiao Wang, Deng Cai, Zheng Wang, and Jieping Ye. 2025. **Controlling thinking speed in reasoning models**. In *Advances in Neural Information Processing Systems*.
- Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. In *First Conference on Language Modeling (COLM)*.
- Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Chang Liu, Yiran Zhao, Lawrence Liu, Yaoqi Ye, Csaba Szepesvári, and Lin F. Yang. 2026. **LACONIC: Length-aware constrained reinforcement learning for LLM**. *arXiv preprint arXiv:2602.14468*.

- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-Pruner: Length-harmonizing fine-tuning for O1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.
- Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.
- Mathematical Association of America. 2023. *American Mathematics Competitions (AMC 12)*. <https://maa.org/student-programs/amc/>. AMC 12A and 12B, 2023.
- Mathematical Association of America. 2025. *American Invitational Mathematics Examination (AIME)*. <https://maa.org/maa-invitational-competitions/>.
- Mathematical Association of America. 2026. *American Invitational Mathematics Examination (AIME) 2026*. <https://maa.org/maa-invitational-competitions/>.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- Sean O’Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*.
- Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. 2025. ThoughtTerminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. *arXiv preprint arXiv:2504.13367*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. *GPQA: A graduate-level Google-proof Q&A benchmark*. In *First Conference on Language Modeling*.
- Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2nd International Conference on Foundation and Large Language Models (FLLM)*. ArXiv:2401.05618.
- Sangjun Song, Minjae Oh, Seungkyu Lee, Sungmin Jo, and Yohan Jo. 2026. *ThinkBrake: Efficient reasoning via log-probability margin guided decoding*. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 21765–21790.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*. ArXiv:2503.16419.
- Lexiang Tang, Weihao Gao, Bingchen Zhao, Lu Ma, Qiao Jin, Bang Yang, and Yuexian Zou. 2026. Thinking by subtraction: Confidence-driven contrastive decoding for LLM reasoning. *arXiv preprint arXiv:2602.18232*.
- Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025. Wait, we don’t need to “wait”! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*.
- Zilin Xiao, Jaywon Koo, Siru Ouyang, Jefferson Hernandez, Yu Meng, and Vicente Ordóñez. 2025. ProxyThinker: Test-time guidance through small visual reasoners. *arXiv preprint arXiv:2505.24872*.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.
- An Yang, Anfeng Li, Baosong Yang, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.
- Yunxiang Zhang, Muhammad Khalifa, Lechen Zhang, Xin Liu, Ayoung Lee, Xinliang Frederick Zhang, Farima Fatahi Bayat, and Lu Wang. 2026. Logit arithmetic elicits long reasoning capabilities without training. In *Findings of the Association for Computational Linguistics: ACL 2026*. ArXiv:2510.09354.

A Algorithm and random-target control

Algorithm 1 One decoding step inside the CoT

Require: prompt x , tokens $y_{<t}$, strength α , top- p ; sampler (T, p, k)

- 1: $z_{\text{base}} \leftarrow M(x, y_{<t})$
- 2: $z_{\text{brief}} \leftarrow D(x, y_{<t}; \text{brief}); \quad z_{\text{verbose}} \leftarrow D(x, y_{<t}; \text{verbose})$
- 3: $S \leftarrow \mathcal{V}_p(\text{softmax}(z_{\text{base}})) \triangleright$ top- p set at temperature one
- 4: $g \leftarrow z_{\text{brief}} - z_{\text{verbose}}$
- 5: $z \leftarrow z_{\text{base}} + \alpha g$
- 6: $z[c] \leftarrow z_{\text{base}}[c]$ for $c \in \{</think>, \text{EOS}\}$
- 7: $z[v] \leftarrow -\infty$ for $v \notin S \quad \triangleright$ the mask is applied last
- 8: **return** a token sampled from z with (T, p, k) , appended to all three streams

Where the contrast applies. The contrast is added only while the large model writes its CoT, and the answer after `</think>` is decoded by the large model alone. Inside the CoT, the algorithm

restores the large model’s own logits for `</think>` and the end-of-sequence token, so the CoT ends only where the large model itself would consider ending it. At such a point the contrast changes the competition. By lowering the tokens that open another round of checking, it lets the large model’s own `</think>` win more often. The mask is applied last, so the candidates are the top- p set of the large model’s distribution at temperature one and the contrast never brings in a token outside it. The method has one hyperparameter, the strength α .

Random-target control. The control differs from Algorithm 1 in one step. After g is formed, it draws a uniformly random permutation of the positions in the top- p set S and assigns the entries of g at those positions to the permuted positions. Entries outside S are masked anyway. The permutation is seeded by the run’s seed and the step index. Everything else, including α , the restoration of the two logits and the mask, is identical to DoubtLess.

B Prompts

Every stream receives the problem followed by one instruction line. On GPQA-Diamond the answer is the letter of the chosen option, and on LiveCodeBench the instruction asks for a program in a code block. The brief and the verbose thinker append one sentence each, and the large model’s own stream carries neither. The concise-prompt setting gives the large model the brief sentence.

Instruction (all streams)

Please reason step by step, and put your final answer within `\boxed{}`.

Brief thinker (appended)

Be concise: go straight to the answer with the fewest steps. Once you have a result, do NOT re-derive, re-verify, or repeat any calculation, and do not second-guess it.

Verbose thinker (appended)

Be extremely thorough: double-check every step, re-verify each calculation in more than one way, and exhaustively consider alternative cases before you commit to a final answer.

NoWait: blocked words

wait, alternatively, hmm, but, however, alternative, another, check, double-check, oh, maybe, verify, other, again, now, ah, any. Each word is blocked in every single-token surface form (capitalized, with a leading space, after a full stop, in capitals), 89 token ids in total, with a logit bias of -100 throughout the generation. The seventeen words are those of Wang et al. (2025).

C The logit difference as a ratio of probabilities

Write p_M , p_b and p_v for the softmax distributions of z_{base} , z_{brief} and z_{verbose} , and Z_M , Z_b , Z_v for their normalizers. For any token w , $z_{\text{base}}[w] = \log p_M(w) + \log Z_M$ and likewise for the two thinkers. The sampler divides z by the temperature T before the softmax, so

$$\exp\left(\left(z_{\text{base}}[w] + \alpha(z_{\text{brief}}[w] - z_{\text{verbose}}[w])\right)/T\right) = Z_M^{1/T} \left(\frac{Z_b}{Z_v}\right)^{\alpha/T} p_M(w)^{1/T} \left[\frac{p_b(w)}{p_v(w)}\right]^{\alpha/T}.$$

The factor in front does not depend on w and cancels in the softmax, and $p_M(w)^{1/T}$ normalized is p_M^T , the distribution plain decoding samples from, which gives Equation (6). At $T=1$ the exponent is α . The top- p mask acts on z after this step, so it applies to the reweighted distribution. Restoring the `</think>` and end-of-sequence logits after this step also removes the factor $(Z_b/Z_v)^{\alpha/T}$ from these two tokens alone, so against every other candidate their odds are multiplied by $e^{-\alpha\Delta/T}$, where $\Delta = \log Z_b - \log Z_v$. On the plain-decoding CoTs of the first six problems of AIME 2025 (one run each, 42,076 positions), Δ has mean -0.04 and standard deviation 0.11 , so at $\alpha/T=8.3$ this factor is 1.33 at the median and lies between 0.54 and 3.58 on nine positions in ten. The random-target control permutes the values of $z_{\text{brief}} - z_{\text{verbose}}$, each of which contains Δ , within the top- p set and restores the same two logits, so it carries the same factor. It changes the thinking length by 1% over the seven benchmarks (Table 7), so the shortening under DoubtLess comes from which tokens the small model pushes down.

D Implementation details

The large model runs in vLLM 0.29 (Kwon et al., 2023) with tensor parallelism. We load the small model into the same vLLM model as the large one,

so that a single request drives all three streams of Figure 2. The small model keeps one copy of its linear weights but two copies of its attention layers, one for the brief thinker and one for the verbose thinker. The two thinkers go through the linear layers together, as a doubled batch, and through attention separately. Under tensor parallelism the small model is split across the GPUs by attention head, like the large model. The three streams differ only in the instruction sentence, which sits between the problem and the last few tokens of the chat template (Appendix B). Everything else, the prompt, those template tokens and the generated tokens, uses the same slot mapping in the three streams’ paged KV caches, and we keep it in one paged key–value cache. The request’s block table is allocated for the large model’s sequence, and the small model’s attention layers, which have the same cache layout (eight key–value heads of dimension 128 in bfloat16, split across the GPUs), write their keys and values into the same blocks. vLLM then allocates, accounts for and frees the three streams as one, with no padding and no duplicated copies. The keys and values of the instruction sentence itself, at most 64 tokens per thinker, live in a small buffer per request that is allocated together with the model. The position IDs are those of each stream’s own sequence, so the thinkers see the template tokens and the generated tokens shifted by the length of their sentence. A key stores its own position, so the order of the blocks does not have to match the order of the positions. A thinker’s attention is computed in two parts, paged attention over the request’s blocks and dense attention over the instruction buffer, which are merged by their log-sum-exp (vLLM’s cascade-attention kernel does exactly this). The result is the attention over the full sequence. In the first step all three streams prefill the problem. The small model then reads its two instruction sentences against the cached problem and fills the buffer, after which decoding proceeds with every sampled token fed to all three streams. The forward pass returns the three hidden states. At the logits step we apply the three output heads, form $z_{\text{base}} + \alpha(z_{\text{brief}} - z_{\text{verbose}})$ for the requests that are still inside their CoT, restore the `</think>` and end-of-sequence logits, mask to the large model’s top- p set (the random-target control permutes the difference within that set here) and pass z to vLLM’s sampler, which draws with the request’s temperature, top- p , top- k and seed. Whether a request has left its CoT is tracked on the GPU from the tokens

fed to the model, so asynchronous scheduling and CUDA graphs work as usual. After `</think>` the contrast is no longer applied. The small model runs on a second CUDA stream and overlaps the large model’s forward pass, and one CUDA graph per decoding step covers both models. With one problem in flight, a decoding step is bound by reading the large model’s weights (65.5 GB in bfloat16), and the small model’s kernels run on the second stream while those weights stream, with vLLM’s fused kernels for its normalization and rotary embedding and a single call of its output head for both thinkers. We time each tokenized request from submission to the engine until the complete response text is returned, including prefill and decoding. Before a run we check that the two vocabularies have the same size (151,936 for Qwen3) and that the instruction, `<think>` and `</think>` encode to the same ids with both tokenizers.

NoWait. The blocked words are the seventeen of Wang et al. (2025), listed in Appendix B. The paper expands each word to every vocabulary token that contains it as a substring and then removes unreasonable matches by hand (its example is *Ohio* for *oh*). On the Qwen3 vocabulary the substring rule matches 918 tokens, most of them unrelated words such as *know*, *many*, *company* and *button*, and the filtered list is not published. We block the 89 single-token forms of the words: the word itself, with a leading space, capitalized, in upper case and after a period, which are the forms the paper gives as its example for *wait*. The bias is -100 , the most negative value vLLM accepts, applied to the whole generation, with a 32,768-token budget as in the paper (24,576 on GPQA-Diamond) and the sampling settings of Appendix E. For Qwen3-32B over five seeds, the paper reports chains 13–16% shorter on AMC 2023, AIME 2024 and AIME 2025 with accuracy changes of -0.8 , $+2.0$ and -2.7 points, and on GPQA-Diamond chains 15% shorter at -6.1 points. Our plain-decoding chains have the same lengths as the paper’s on the three mathematics benchmarks (within 5%). NoWait removes more in our runs (21–24%), and its accuracy cost falls on AIME 2025 and HMMT, while GPQA-Diamond loses 0.3 points (Table 7). An independent run of NoWait on Qwen3-32B in the ThinkBrake paper (Song et al., 2026) loses 9 points on AIME 2024 and 12 on AIME 2025 at 11–12% fewer tokens. Our runs show the same direction, with the largest loss on AIME 2025.

DEER. DEER only decides when the CoT ends. A DEER run is therefore the plain-decoding run of the same problem and seed, cut at the first point where the rule fires, followed by `</think>` and an answer sampled with the budget that remains. A run in which the rule never fires is the plain run itself. We evaluate it on the stored plain-decoding runs of Table 1, in the same engine and with the same sampler and seeds. For DEER we follow the authors’ released vLLM implementation with its default settings. The transition point is the string *Wait*. At each of the first ten, the text before it is extended with the inducer `\n**Final Answer**\n\boxed` (on GPQA-Diamond, the released variant . . . I believe the final answer, rather than the option, is `\boxed`) and a trial answer of at most 20 tokens is decoded greedily and stopped at `</think>`. Its confidence is the geometric mean of the token probabilities from the second trial token on, and is zero unless the trial reached `</think>`. The block is closed with `\n</think>\n\n` when the confidence exceeds 0.95, and in any case once the thinking reaches $0.8 \times$ the budget (the released `think_ratio`), at the segment boundary that precedes it by at most 50 tokens. After the tenth trial no further trial is made, and the chain runs to its own end or to the $0.8 \times$ cap. The released code covers mathematics and GPQA. On LiveCodeBench the inducer opens a python code block and the trial may run to 1,024 tokens, with the same `</think>` requirement. The DEER paper also reports Qwen3-32B under the same sampler (temperature 0.6, top- p 0.95, budget 32,768; its Table 8): thinking tokens fall by 10% on AIME 2024, 9% on AIME 2025, 22% on AMC 2023 and 11% on OlympiadBench, with accuracy within one point of plain decoding. Our runs give 8%, 10%, 20% and 12%, with accuracy within 2.2 points (Table 7).

ThinkBrake. ThinkBrake (Song et al., 2026) ends the CoT at a sentence boundary when the margin between the most likely next token other than `</think>` and `</think>`, on the temperature-scaled logits, is at most τ . Its paper uses $\tau=0.1$ for every Qwen3 model. The paper does not say in which state the margin is read. Its only released code (for its tool-use benchmark) appends a period and a single newline to each prefix and reads `</think>` there, which is the state in which Qwen3 closes its CoT on its own. In our runs every natural close follows a single-newline token, whereas a

sentence inside the CoT mostly ends in a blank-line token, after which `</think>` is never near the top. Read in the state the model actually wrote, the rule therefore does not fire. We therefore follow the released code. At every sentence boundary, a token that ends a sentence and is followed by whitespace, or a token that ends with a newline, the sentence’s last token is replaced by its single-newline form (`.\n\n` becomes `.\n`) and the margin is read there. Where the rule fires, the run keeps its CoT up to that sentence, followed by the single newline, `</think>` and a freshly sampled answer. Like DEER, ThinkBrake is evaluated on the stored plain-decoding runs. It removes 21% of the thinking tokens over the seven benchmarks at a cost of 2.3 points: 37% on GPQA-Diamond and 44% on LiveCodeBench, where it costs 2.0 and 4.7 points, and 3–4% on the five others (Table 7). The paper reports 4–6% fewer thinking tokens on AIME 2024 and 2025 and 20% on GPQA-Diamond for Qwen3-32B with a 16k budget.

ASC. The published version of ASC learns the steering vector by gradient descent, which requires an additional training run. We follow the training-free version in the authors’ released code, which adds the mean of the last-token differences between verbose and concise CoTs, scaled by a strength, to the output of one decoder layer at every forward pass, so at every generated token and at the last prompt token. The calibration set is the first 100 problems of MATH-500 in an order that cycles through its subjects. For each, Qwen3-32B writes one CoT under its plain instruction and one under the brief thinker’s sentence, with the sampling settings of Appendix E and a 16,384-token budget, and both are read back after the plain instruction. The authors write the concise CoTs with GPT-4o, and we use the large model’s own. The vector is added after layer 57. Its strength ρ is the norm of the added vector relative to the mean norm of the hidden state at that layer, and $\rho=0.2$ is chosen on AIME 2026 (Table 6).

E Experimental setup details

Table 3: Benchmarks: problems $\times$ seeds per problem, thinking budget in tokens, answer type and license.

Benchmark	$n \times \text{seeds}$	Budget	Answer	License
AIME 2024	30×3	32k	integer	MAA
AIME 2025	30×3	32k	integer	MAA
HMMT Feb. 2025	30×3	32k	expression	HMMT
AMC 2023	40×3	32k	integer	MAA
GPQA-Diamond	198×3	24k	letter	CC BY 4.0
OlympiadBench	100×3	32k	expression	Apache 2.0
LiveCodeBench	100×3	32k	program	MIT

Benchmarks, budgets and licenses. AIME 2024 and AIME 2025 are used in full (both contests of each year, 30 problems), and so are HMMT February 2025 (30), AMC 2023 (40) and GPQA-Diamond (198). OlympiadBench’s 100 problems and LiveCodeBench’s 100 are drawn round-robin across subjects and across difficulty levels respectively. The thinking budget is 32,768 tokens except 24,576 on GPQA-Diamond. Under plain decoding the share of runs that reach it is 13% on AIME 2025, 3% on AIME 2024, 4% on HMMT, 0% on AMC 2023, 0.3% on GPQA-Diamond, 4% on OlympiadBench and 2% on LiveCodeBench. LiveCodeBench problems are those of release 6 (January to April 2025), GPQA-Diamond is the 198-question diamond split, and OlympiadBench’s problems are its text-only mathematics problems in English, so every problem in this paper is in English. The competition problems (AIME, AMC, HMMT) are the property of their organizers and are used for evaluation only, as in the work we compare with. GPQA-Diamond is released under CC BY 4.0, OlympiadBench under Apache 2.0 and LiveCodeBench under MIT.

Scoring. The visible answer after `</think>` is scored, never the CoT. Integer answers (AIME, AMC) must match exactly. LaTeX answers (HMMT, OlympiadBench) are compared with `math-verify` (Kydliček, 2025). An answer whose parts are boxed one by one is compared, as one list, with an answer key of several parts; an answer key that names what it gives ($m_{\max} = n^2 - n - 1$) is compared by its right side; and functions are compared after renaming their variable, so that $f(n) = n$ matches $f(z) = z$. We correct one OlympiadBench answer key, which gives $(10^3 + 1)^2$ itself where the problem asks for the sum of its digits. Multiple-choice answers are matched by letter, and an answer that spells out an option’s text or value, also in L^AT_EX, is mapped back to its letter first. Live-

CodeBench programs are executed against the test cases. A run that reaches its budget without ending its CoT scores zero.

Sampling, seeds and intervals. All settings sample with temperature 0.6, top- p 0.95 and top- k 20, the settings Qwen3 recommends for thinking mode. The seed of a run is derived from the problem index, the seed index and the setting, and the engine seeds the sampler of each run with it. Intervals are percentile bootstraps with 4,000 resamples over problems.

Compute. All runs use vLLM 0.29, with the small model’s two passes beside the large model on the same GPUs. The experiments in this paper took about 2,000 A100 80GB GPU hours and 300 B200 GPU hours.

F Strength and small-model size

Table 4: AIME 2025, 30 problems, three seeds each: DoubtLess (0.6B) at three strengths and at a negative one ($\alpha = -5$ adds the verbose minus the brief thinker). Accuracy, share of runs reaching the budget, mean thinking tokens, change in thinking tokens against plain decoding over all problems with its 95% interval, and accuracy change in points.

Setting	Acc.	Budget	Tokens	Token change	Acc. change
$\alpha = -5$	73.3	21.1	19,676	+26% [+17, +39]	+1.1
Plain decoding	72.2	13.3	15,606	–	–
$\alpha = 2.5$	74.4	4.4	11,964	–23% [–29, –17]	+2.2
$\alpha = 5$ (default)	72.2	1.1	11,092	–29% [–37, –21]	0.0
$\alpha = 10$	64.4	3.3	9,999	–36% [–44, –28]	–7.8

Table 5: AIME 2025, three seeds: DoubtLess with three small models of the same family at $\alpha = 5$, and with Qwen3-32B itself as both thinkers at $\alpha = 1$. Accuracy, share of runs reaching the budget, mean thinking tokens, change in thinking tokens over all problems, and accuracy change in points.

Small model	Acc.	Budget	Tokens	Token change	Acc. change
Qwen3-0.6B	72.2	1.1	11,092	–29% [–37, –21]	0.0
Qwen3-1.7B	50.0	0.0	6,584	–58% [–64, –51]	–22.2
Qwen3-4B	52.2	0.0	6,461	–59% [–64, –53]	–20.0
Qwen3-32B itself	73.3	4.4	11,590	–26% [–32, –20]	+1.1

Choosing the strength. AIME 2026 was released after Qwen3 and appears nowhere else in this paper. Table 6 sweeps α on its 30 problems with three seeds each: DoubtLess removes 21–23% of the thinking tokens at $\alpha \leq 3.5$, 29% at $\alpha = 5$ and 32–33% above it, while accuracy stays within noise of plain decoding up to $\alpha = 7$ and falls by 4.4 points at $\alpha = 10$. $\alpha = 5$ is where the removal levels off. For ASC we sweep the strength the same

way, and $\rho=0.2$ removes the most, 9%, while larger strengths remove less (Table 6).

Table 6: AIME 2026 (30 problems, released after Qwen3), three seeds each: plain decoding, DoubtLess (0.6B) at five strengths and ASC at three. Accuracy, share of runs reaching the budget, mean thinking tokens, change in thinking tokens against plain decoding over all problems with its 95% interval, and accuracy change in points.

Setting	Acc.	Budget	Tokens	Token change	Acc. change
Plain decoding	68.9	8.9	13,543	–	–
$\alpha=2.5$	76.7	3.3	10,474	–23% [–30, –16]	+7.8
$\alpha=3.5$	67.8	5.6	10,662	–21% [–30, –13]	–1.1
$\alpha=5$ (default)	72.2	4.4	9,605	–29% [–38, –20]	+3.3
$\alpha=7$	68.9	3.3	9,275	–32% [–40, –22]	0.0
$\alpha=10$	64.4	2.2	9,135	–33% [–42, –23]	–4.4
ASC $\rho=0.2$ (default)	78.9	5.6	12,267	–9% [–16, –2]	+10.0
ASC $\rho=0.4$	74.4	4.4	12,446	–8% [–18, +1]	+5.6
ASC $\rho=0.8$	67.8	4.4	13,478	0% [–8, +9]	–1.1

G Every setting on every benchmark

For each problem i , let ℓ_i and ℓ'_i be the mean thinking length over its runs under plain decoding and under a method. The change of the method against plain decoding is $(\sum_i \ell'_i - \sum_i \ell_i) / \sum_i \ell_i$, with a percentile bootstrap over problems for the interval. It uses every problem, so it does not depend on which problems a method solves. Table 7 lists, for the five settings, DEER, ThinkBrake and ASC on the seven benchmarks, the accuracy, the share of runs that reach the budget, the mean thinking length, this change, and the change over the problems both the method and plain decoding solved in every run, the measure used in earlier work. Figure 9 draws the thinking length under each of the five settings on every benchmark.

H Other model families

EXAONE-4.0-32B and EXAONE-4.0-1.2B share the EXAONE tokenizer and open the CoT with the same tag as Qwen3. Granite-4.2-30B and Granite-4.2-3B (IBM Granite Team, 2026) share the Granite tokenizer. The instruction and the two thinker sentences are identical to Appendix B for all three families. The two other families use the sampling settings, thinking budget and strength ($\alpha=5$) of the main table, with three seeds per problem, and each family’s own chat template with thinking enabled. The concise prompt in Table 8 is the setting of the main experiments, the brief thinker’s sentence appended to the large model’s own instruction, with three seeds per problem.

I Where the removed tokens come from

Each chain is split at the first automatic match to the correct answer in the CoT: for integer answers the first occurrence of the correct value as a standalone number, and for multiple-choice benchmarks the first occurrence of the correct letter in a phrase such as “the answer is B”. Problems whose answer is a LaTeX expression are not split. Table 9 gives, for AIME 2025, the mean number of thinking tokens before and after that point under each setting, over the correct runs in which the answer can be located, and Table 10 the frequency of some words per thousand thinking tokens. Figure 5 draws both. Table 11 repeats the split on every benchmark for the runs that both plain decoding and DoubtLess solve, so that the two methods are compared on the same runs. LiveCodeBench is absent because the model writes its program after the CoT, and HMMT and OlympiadBench cover their integer-answer problems only.

Where the two instructions disagree. For Figure 7 the small model reads each correct plain-decoding chain on AIME 2025 twice, once after the brief thinker’s prompt and once after the verbose thinker’s, the two prompts DoubtLess uses. At every position of the CoT we record the KL divergence from the brief to the verbose next-token distribution, computed over the brief thinker’s 64 most likely tokens, and the difference between the two log-probabilities of the token the large model wrote next. Positions are aligned at the first automatic match to the correct answer and averaged over runs in bins of 250 tokens, keeping the bins that at least ten runs reach. Panel (b) pools every token whose text, without surrounding spaces, is the given word. The 39 correct runs of at most 12,000 thinking tokens enter, from 16 problems. The shading of Figure 3 uses the same two readings: a token is shaded red when Equation (6) down-weights it by a factor of five or more and green when it up-weights it by five or more, that is, when $|\log p_{\text{brief}} - \log p_{\text{verbose}}| \geq 0.2$ at $\alpha=5$ and temperature 0.6.

Human validation of answer boundaries. Two annotators independently marked the first statement of the correct target answer in all 39 traces, allowing tentative statements and equivalent expressions while excluding incidental numbers. We repeated the analysis using each annotation set separately. The post-/pre-answer mean divergence ratios are

Table 7: Every setting on every benchmark: accuracy and its change against plain decoding in points (paired over problems), share of runs that reach the budget (for DEER and ThinkBrake, in the answer after the thinking is closed), mean thinking tokens, change in thinking tokens against plain decoding over all problems, and the same change over the problems that both the setting and plain decoding solved in every run (their number in parentheses). 95% bootstrap intervals over problems. The last block pools the seven benchmarks.

Setting	Acc. (%)	Acc. change	Budget (%)	Tokens	Token change, all	Token change, both solved
<i>AIME 2024</i> (30 problems, 3 seeds each, budget 32,768)						
Plain decoding	81.1	—	3.3	12,152	—	—
Concise prompt	83.3	+2.2 [−5.6, +8.9]	1.1	8,517	−30% [−38, −21]	−28% [−37, −20] (20)
NoWait	77.8	−3.3 [−12.2, +4.4]	2.2	9,575	−21% [−32, −10]	−26% [−40, −13] (18)
DoubtLess (0.6B)	83.3	+2.2 [−6.7, +11.1]	0.0	8,816	−27% [−37, −18]	−34% [−41, −27] (18)
Random-target control	81.1	0.0 [−7.8, +7.8]	1.1	12,036	−1% [−10, +9]	+8% [−1, +18] (18)
DEER	82.2	+1.1 [+0.0, +3.3]	0.0	11,133	−8% [−12, −5]	−11% [−19, −5] (20)
ThinkBrake	80.0	−1.1 [−3.3, +0.0]	2.2	11,635	−4% [−9, −1]	−2% [−3, −1] (20)
ASC	82.2	+1.1 [−5.6, +7.8]	0.0	10,734	−12% [−20, −3]	−6% [−16, +7] (18)
<i>AIME 2025</i> (30 problems, 3 seeds each, budget 32,768)						
Plain decoding	72.2	—	13.3	15,606	—	—
Concise prompt	75.6	+3.3 [−5.6, +12.2]	1.1	11,180	−28% [−37, −19]	−23% [−39, −5] (16)
NoWait	57.8	−14.4 [−24.4, −4.4]	5.6	11,939	−23% [−31, −15]	−38% [−48, −24] (9)
DoubtLess (0.6B)	72.2	0.0 [−11.1, +10.0]	1.1	11,092	−29% [−37, −21]	−32% [−39, −26] (15)
Random-target control	75.6	+3.3 [−3.3, +11.1]	14.4	14,992	−4% [−12, +3]	−2% [−8, +6] (16)
DEER	74.4	+2.2 [+0.0, +5.6]	4.4	14,087	−10% [−14, −6]	−8% [−17, −3] (18)
ThinkBrake	72.2	0.0 [−3.3, +3.3]	10.0	15,091	−3% [−6, −1]	−1% [−2, +0] (17)
ASC	73.3	+1.1 [−7.8, +10.0]	11.1	14,341	−8% [−15, +1]	−15% [−21, −7] (15)
<i>HMMT Feb. 2025</i> (30 problems, 3 seeds each, budget 32,768)						
Plain decoding	53.3	—	4.4	16,817	—	—
Concise prompt	46.7	−6.7 [−16.7, +3.3]	10.0	14,551	−13% [−23, −4]	−34% [−55, −4] (6)
NoWait	46.7	−6.7 [−14.4, +1.1]	6.7	13,630	−19% [−25, −13]	−34% [−49, −23] (6)
DoubtLess (0.6B)	56.7	+3.3 [−3.3, +11.1]	0.0	11,586	−31% [−38, −25]	−29% [−49, −11] (9)
Random-target control	53.3	0.0 [−10.0, +8.9]	10.0	17,201	+2% [−5, +10]	+9% [−4, +27] (6)
DEER	53.3	0.0 [+0.0, +0.0]	2.2	15,739	−6% [−9, −4]	−6% [−18, −1] (9)
ThinkBrake	52.2	−1.1 [−3.3, +0.0]	3.3	16,144	−4% [−6, −2]	−1% [−3, +0] (9)
ASC	51.1	−2.2 [−13.3, +7.8]	11.1	17,303	+3% [−7, +15]	−10% [−29, +11] (8)
<i>AMC 2023</i> (40 problems, 3 seeds each, budget 32,768)						
Plain decoding	96.7	—	0.0	6,130	—	—
Concise prompt	95.8	−0.8 [−6.7, +5.0]	0.8	4,203	−31% [−42, −20]	−39% [−46, −31] (32)
NoWait	94.2	−2.5 [−5.8, +0.8]	0.8	4,651	−24% [−39, −7]	−41% [−46, −34] (35)
DoubtLess (0.6B)	97.5	+0.8 [−1.7, +3.3]	0.0	4,063	−34% [−41, −27]	−33% [−41, −25] (36)
Random-target control	98.3	+1.7 [−1.7, +5.0]	0.8	6,605	+8% [−1, +17]	+3% [−3, +9] (36)
DEER	97.5	+0.8 [+0.0, +2.5]	0.0	4,875	−20% [−29, −13]	−20% [−29, −12] (37)
ThinkBrake	97.5	+0.8 [+0.0, +2.5]	0.0	5,919	−3% [−5, −2]	−4% [−5, −2] (37)
ASC	98.3	+1.7 [+0.0, +4.2]	0.0	5,458	−11% [−19, −3]	−15% [−21, −8] (37)
<i>GPQA-Diamond</i> (198 problems, 3 seeds each, budget 24,576)						
Plain decoding	68.7	—	0.3	7,354	—	—
Concise prompt	67.7	−1.0 [−4.4, +2.4]	0.0	4,429	−40% [−43, −37]	−48% [−53, −43] (97)
NoWait	68.4	−0.3 [−3.5, +3.0]	1.0	5,782	−21% [−25, −17]	−28% [−33, −23] (92)
DoubtLess (0.6B)	66.3	−2.4 [−5.6, +0.8]	0.0	3,995	−46% [−48, −43]	−43% [−48, −39] (97)
Random-target control	70.0	+1.3 [−1.7, +4.4]	0.5	7,319	0% [−3, +2]	−1% [−6, +3] (99)
DEER	68.7	0.0 [−0.5, +0.5]	0.0	7,200	−2% [−3, −1]	−4% [−6, −2] (110)
ThinkBrake	66.7	−2.0 [−5.2, +1.3]	0.7	4,650	−37% [−41, −33]	−23% [−27, −19] (95)
ASC	69.9	+1.2 [−1.7, +4.0]	1.0	7,058	−4% [−7, −1]	−6% [−11, −1] (103)
<i>OlympiadBench</i> (100 problems, 3 seeds each, budget 32,768)						
Plain decoding	70.7	—	3.7	12,246	—	—
Concise prompt	68.7	−2.0 [−6.0, +2.0]	1.7	9,227	−25% [−29, −20]	−36% [−40, −31] (56)
NoWait	69.7	−1.0 [−5.0, +2.7]	1.7	10,117	−17% [−23, −11]	−16% [−26, −7] (54)
DoubtLess (0.6B)	69.0	−1.7 [−6.3, +2.7]	1.0	8,951	−27% [−31, −22]	−32% [−38, −25] (55)
Random-target control	70.7	0.0 [−3.7, +3.3]	2.0	12,239	0% [−5, +5]	−3% [−8, +1] (57)
DEER	71.3	+0.7 [−0.7, +2.0]	0.7	10,762	−12% [−17, −8]	−15% [−23, −9] (63)
ThinkBrake	67.3	−3.3 [−7.0, −0.3]	3.0	11,770	−4% [−5, −3]	−3% [−5, −2] (60)
ASC	69.0	−1.7 [−6.3, +2.3]	2.3	11,708	−4% [−9, +1]	−5% [−11, +2] (55)
<i>LiveCodeBench</i> (100 problems, 3 seeds each, budget 32,768)						
Plain decoding	66.3	—	2.0	11,511	—	—
Concise prompt	65.7	−0.7 [−4.3, +2.7]	1.3	10,628	−8% [−12, −3]	−9% [−14, −5] (52)
NoWait	62.3	−4.0 [−8.0, −0.3]	0.3	9,489	−18% [−22, −13]	−25% [−30, −21] (48)
DoubtLess (0.6B)	66.0	−0.3 [−4.0, +3.3]	1.0	9,659	−16% [−21, −11]	−15% [−22, −7] (51)
Random-target control	66.3	0.0 [−4.0, +3.7]	1.0	10,901	−5% [−9, −1]	0% [−5, +5] (52)
DEER	61.7	−4.7 [−7.7, −2.0]	0.7	9,509	−17% [−24, −12]	−10% [−16, −5] (46)
ThinkBrake	61.7	−4.7 [−8.3, −1.3]	0.7	6,408	−44% [−49, −39]	−21% [−25, −18] (49)
ASC	66.0	−0.3 [−4.7, +4.0]	1.3	11,080	−4% [−9, +2]	−4% [−11, +3] (51)
<i>All seven benchmarks, pooled</i> (528 problems, three seeds each)						
Plain decoding	70.8	—	2.4	10,254	—	—
Concise prompt	69.8	−1.0 [−2.9, +0.9]	1.3	7,686	−25% [−27, −23]	−33% [−37, −30] (279)
NoWait	68.1	−2.7 [−4.5, −0.8]	1.6	8,231	−20% [−22, −17]	−27% [−30, −23] (262)
DoubtLess (0.6B)	69.9	−0.9 [−2.8, +1.0]	0.4	7,120	−31% [−33, −28]	−33% [−36, −30] (281)
Random-target control	71.6	+0.8 [−0.9, +2.5]	2.3	10,140	−1% [−3, +1]	0% [−3, +2] (284)
DEER	70.3	−0.5 [−1.2, +0.2]	0.6	9,236	−10% [−12, −8]	−10% [−13, −8] (303)
ThinkBrake	68.4	−2.3 [−3.9, −0.8]	1.8	8,071	−21% [−24, −19]	−10% [−12, −8] (287)
ASC	71.0	+0.2 [−1.6, +2.0]	2.3	9,784	−5% [−7, −2]	−7% [−10, −5] (287)

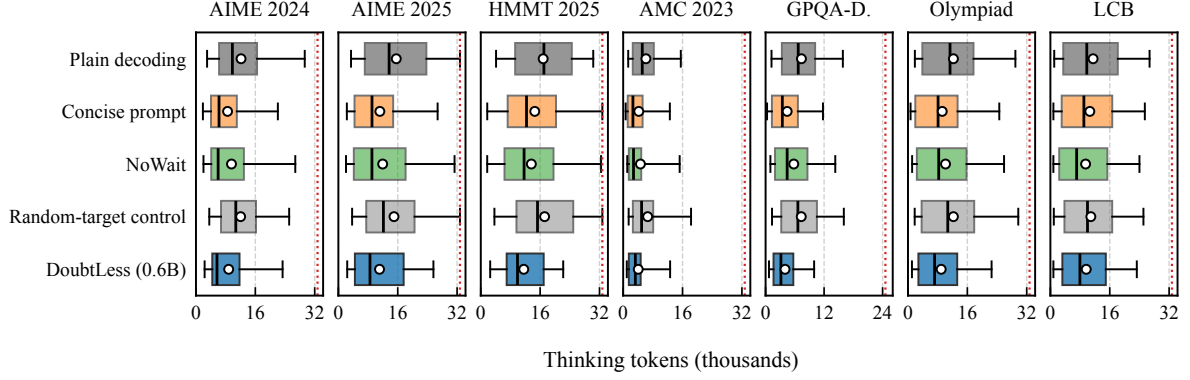


Figure 9: Thinking length under each method on each benchmark: box, quartiles; whiskers, 5th to 95th percentile; dot, mean; dotted line, the budget.

Table 8: DoubtLess removes about three times as much thinking as the concise prompt on EXAONE-4.0 and Granite-4.2, with unchanged accuracy. Same instructions and strength for every family. AIME 2025 and AMC 2023, 70 problems, three seeds each; each large model is steered by the small model below it. Accuracy (%) and thinking-token change against plain decoding, the latter with 95% intervals.

	Plain	Concise		DoubtLess	
Model	Acc.	Acc.	Tokens	Acc.	Tokens
Qwen3-32B	86.2	87.1	−29%	86.7	−31%
<i>Qwen3-0.6B</i>			[−36, −22]		[−36, −25]
EXAONE-4.0-32B	88.6	88.6	−8%	90.5	−25%
<i>EXAONE-4.0-1.2B</i>			[−11, −4]		[−31, −19]
Granite-4.2-30B	89.0	88.1	−14%	91.4	−40%
<i>Granite-4.2-3B</i>			[−21, −9]		[−48, −32]

Table 9: AIME 2025: mean thinking tokens before and after the first occurrence of the correct answer, over the correct runs in which the answer can be located (last column).

Setting	Before	After	Share after	Runs
Plain decoding	7,624	3,553	32%	65
Concise prompt	6,096	2,551	30%	68
NoWait	5,478	1,334	20%	52
DoubtLess (0.6B)	6,023	1,710	22%	65
Random-target control	6,716	3,851	36%	68

7.1 and 7.4, compared with 7.4 under automatic matching, and the share of the total divergence after the boundary is 84% under all three. Under either annotation set, mean divergence is larger after the boundary in all 39 traces.

Retained error-correcting verification. On AIME 2025 problem 11, a DoubtLess run initially rejects the positive root of $34y^2 - y - 32 = 0$, treating its rounded approximation as an excluded

Table 10: AIME 2025: occurrences per thousand thinking tokens of some words, pooled over all runs of a setting.

Word	Plain	Concise	NoWait	DoubtLess	Random
Wait	3.2	3.2	0.0	3.6	3.2
Hmm	0.1	0.1	0.0	0.1	0.1
But	2.7	2.7	0.0	3.5	2.8
Alternatively	1.1	1.2	0.0	1.3	1.0
However	1.5	1.4	0.0	1.3	1.6
verify	0.2	0.2	0.0	0.2	0.2
double-check	0.0	0.0	0.1	0.0	0.0
Therefore	6.1	5.2	6.8	5.3	6.2
So	2.5	3.1	3.2	3.2	2.8
Let me	4.0	4.3	6.3	3.8	3.9

interval endpoint. An algebraic check then uses $x = 34y^2 = y + 32$ to establish that $32 < x < 33$, restoring the root. The final answer includes both roots and uses their sum, $1/34$. Both annotators verified the error, its correction, and its effect on the final sum. This run uses 17,196 thinking tokens, compared with 20,866 under plain decoding.

J The self-debugging agent

The agent solves each of the 164 HumanEval+ problems (Liu et al., 2023) in at most two turns. In the first turn the model receives the function to complete with the instruction “Complete this Python function. Return the FULL function in one python block”, thinks, and writes the program. The program is run against the HumanEval+ tests. If it fails, the model receives the test’s error message with the instruction “Your program failed the tests with this error: ... Fix the program. Return the FULL function in one python block” and gets one more turn. Every turn has its own CoT with a budget of 4,096 tokens and 1,024 tokens for the program. DoubtLess is applied inside each CoT

Table 11: Plain decoding and DoubtLess (0.6B) on the same runs, those that both settings solve and in which the answer can be located (integer answers as a standalone number, letters as the answer of a phrase). Mean thinking tokens before and after the first occurrence of the correct answer under each setting, the change of each part, and the share of the removed tokens that lie after the answer. HMMT and OlympiadBench cover their integer-answer problems only; on LiveCodeBench the program is written after the CoT, so no split is possible.

Benchmark	Runs	Plain decoding		DoubtLess (0.6B)		Change		Removed after answer
		Before	After	Before	After	Before	After	
AIME 2024	68	5,751	3,301	5,186	1,474	-10%	-55%	76%
AIME 2025	56	6,310	3,700	5,183	1,630	-18%	-56%	65%
HMMT Feb. 2025	19	4,000	4,973	3,573	1,382	-11%	-72%	89%
AMC 2023	115	1,880	3,974	1,666	2,330	-11%	-41%	89%
GPQA-Diamond	346	2,833	2,799	1,991	1,179	-30%	-58%	66%
OlympiadBench	21	1,525	8,639	1,014	7,029	-34%	-19%	76%

exactly as in the single-turn runs, with the same two thinker sentences appended to the last user message, NoWait blocks the same 89 token ids, the concise prompt appends the brief sentence to the last user message, and the random-target control is applied as in the single-turn runs. DEER and ThinkBrake run inside each turn as it is decoded, since an early exit in the first turn changes the program that the repair turn sees. DEER uses the code inducer and 1,024-token trials of LiveCodeBench (Appendix D) and cuts the CoT at $0.8 \times$ the turn’s thinking budget. ThinkBrake first decodes the turn as plain decoding with the same seed, reads its rule at the sentence boundaries of that CoT as in the single-turn runs, and closes the CoT at the first boundary where it fires, followed by a freshly sampled program. Under plain decoding 16–20% of the turns reach the budget of 4,096 thinking tokens without closing their CoT and so write no program. DoubtLess halves the number of such turns, and DEER’s cut leaves none. The random-target control leaves the thinking per turn within 1% of plain decoding and the pass rate within 1.2 points. Three seeds per problem. The pass rate counts a problem as solved if the program passes all tests after the first or the second turn, averaged over seeds, and thinking tokens are averaged over turns and seeds.

K Per-problem results on AIME 2025

Figure 10 plots, for every problem, its mean thinking length under DoubtLess and under the random-target control against plain decoding. The percentage removed does not depend on how long plain decoding thinks on a problem. The problems lie along a line through the origin, the ones plain decoding finishes fastest and slowest lose a similar percentage (30% and 29% for the fastest and the slowest third), and the random-target control stays

on the diagonal. DoubtLess shifts the whole distribution of thinking length down, and the share of runs that reach the 32,768-token budget falls from 13% to 1% (Figure 9, Table 7). Table 12 lists every problem of AIME 2025 under the five settings: the mean thinking length over the three seeds and the number of seeds under which the answer was correct.

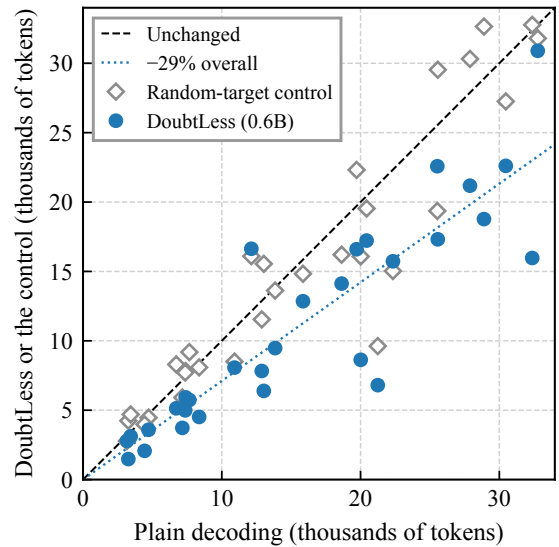


Figure 10: DoubtLess removes a similar percentage on short and on long problems, and the random-target control stays on the diagonal. AIME 2025: mean thinking tokens of each problem over its three runs, under DoubtLess (0.6B) and the control against plain decoding. The dotted line is the overall change.

Table 12: AIME 2025, per problem: mean thinking tokens over the three seeds and (correct runs / runs) under each setting.

Problem	Answer	Plain decoding	Concise prompt	NoWait	DoubtLess (0.6B)	Random-target
1	70	3,266 (3/3)	1,532 (3/3)	1,882 (3/3)	1,479 (3/3)	4,239 (3/3)
2	588	12,881 (3/3)	10,077 (3/3)	8,088 (3/3)	7,827 (3/3)	11,541 (3/3)
3	16	7,159 (3/3)	4,369 (3/3)	3,585 (3/3)	3,726 (3/3)	5,914 (3/3)
4	117	8,367 (3/3)	5,931 (3/3)	4,924 (2/3)	4,514 (3/3)	8,085 (3/3)
5	279	10,917 (3/3)	6,712 (3/3)	5,512 (2/3)	8,070 (3/3)	8,503 (3/3)
6	504	3,163 (3/3)	2,495 (3/3)	2,158 (3/3)	2,791 (3/3)	2,781 (3/3)
7	821	12,129 (1/3)	9,160 (0/3)	9,046 (1/3)	16,634 (1/3)	16,098 (2/3)
8	77	4,721 (3/3)	2,695 (3/3)	2,761 (2/3)	3,589 (3/3)	4,465 (3/3)
9	62	19,714 (3/3)	8,776 (3/3)	9,864 (3/3)	16,599 (1/3)	22,313 (2/3)
10	81	20,435 (3/3)	19,167 (1/3)	16,574 (1/3)	17,223 (0/3)	19,532 (2/3)
11	259	25,566 (2/3)	19,240 (2/3)	20,289 (1/3)	17,315 (2/3)	29,533 (1/3)
12	510	25,531 (1/3)	11,788 (3/3)	20,443 (1/3)	22,579 (2/3)	19,359 (2/3)
13	204	27,884 (0/3)	14,204 (0/3)	16,318 (0/3)	21,178 (0/3)	30,312 (0/3)
14	60	32,768 (0/3)	25,255 (2/3)	32,239 (0/3)	30,898 (0/3)	31,793 (0/3)
15	735	30,467 (0/3)	16,114 (0/3)	23,952 (0/3)	22,607 (0/3)	27,253 (0/3)
16	468	3,434 (3/3)	3,284 (3/3)	2,031 (3/3)	3,117 (3/3)	4,694 (3/3)
17	49	4,439 (3/3)	2,002 (3/3)	2,642 (3/3)	2,070 (3/3)	3,976 (3/3)
18	82	22,334 (1/3)	17,321 (2/3)	12,291 (2/3)	15,731 (3/3)	15,033 (3/3)
19	106	6,719 (3/3)	4,006 (3/3)	3,307 (3/3)	5,137 (3/3)	8,299 (3/3)
20	336	18,634 (3/3)	17,015 (3/3)	20,833 (2/3)	14,123 (3/3)	16,205 (3/3)
21	293	7,397 (3/3)	4,807 (3/3)	15,336 (2/3)	5,922 (3/3)	7,817 (3/3)
22	237	7,663 (2/3)	6,409 (3/3)	6,524 (3/3)	5,738 (3/3)	9,185 (3/3)
23	610	15,851 (3/3)	14,868 (3/3)	12,346 (2/3)	12,853 (3/3)	14,832 (3/3)
24	149	7,364 (3/3)	3,748 (3/3)	3,804 (1/3)	4,989 (2/3)	7,711 (3/3)
25	907	13,027 (3/3)	11,294 (2/3)	15,062 (1/3)	6,391 (3/3)	15,547 (3/3)
26	113	20,013 (2/3)	12,529 (2/3)	10,519 (0/3)	8,631 (2/3)	16,079 (3/3)
27	19	21,241 (2/3)	6,542 (3/3)	13,971 (2/3)	6,804 (3/3)	9,619 (2/3)
28	248	32,371 (0/3)	27,215 (0/3)	24,093 (0/3)	15,969 (1/3)	32,768 (0/3)
29	104	13,843 (3/3)	21,053 (3/3)	12,692 (3/3)	9,476 (3/3)	13,620 (3/3)
30	240	28,893 (0/3)	25,781 (0/3)	25,096 (0/3)	18,772 (0/3)	32,647 (0/3)

L Worked examples

AIME 2025 problem 1 (I-1): *Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b .* Both methods answer 70. Under plain decoding the CoT runs to 3,414 thinking tokens, and with DoubtLess (0.6B) it ends at 1,461. Both are the runs of the first seed in the main experiment, decoded under the same sampling settings, and are shown in full.

Plain decoding.

Okay, so I need to find all integer bases $b > 9$ where the number 17_b divides 97_b . Then, sum all those bases. Let me start by understanding what these numbers represent in base b .

First, let me recall how numbers are represented in different bases. In base b , the number 17_b means $1 \times b + 7$, right? Similarly, 97_b would be $9 \times b + 7$. So, translating both numbers to base 10 (decimal), we have:

$$17_b = 1 \times b + 7 = b + 7$$

$$97_b = 9 \times b + 7 = 9b + 7$$

So the problem is asking for bases $b > 9$ such that $b + 7$ divides $9b + 7$. In other words, $(9b + 7) \div (b + 7)$ should result in an integer.

Let me write that as a division problem. Let me denote $N = 9b + 7$ and $D = b + 7$. We need $D \mid N$, meaning there exists some integer k such that $N = k \times D$. So:

$$9b + 7 = k(b + 7)$$

Our goal is to find all integers $b > 9$ such that this equation holds for some integer k .

Alternatively, we can perform the division and find when the remainder is zero. Let me try polynomial division or algebraic manipulation.

Let me rearrange the equation:

$$9b + 7 = k(b + 7)$$

Let me solve for k :

$$k = \frac{9b + 7}{b + 7}$$

To find integer values of k , we can perform the division. Let me do that.

Divide $9b + 7$ by $b + 7$. Let me think about how to divide these linear expressions.

Dividing $9b + 7$ by $b + 7$, the leading term is $9b / b = 9$. Multiply divisor by 9: $9(b + 7) = 9b + 63$. Subtract this from the dividend:

$$(9b + 7) - (9b + 63) = 9b + 7 - 9b - 63 = -56$$

Therefore, the division gives:

$$\frac{9b + 7}{b + 7} = 9 - \frac{56}{b + 7}$$

So, for this to be an integer, $\frac{56}{b + 7}$ must be an integer. Therefore, $b + 7$ must be a positive divisor of 56.

But since $b > 9$, then $b + 7 > 9 + 7 = 16$. Therefore, $b + 7$ is a divisor of 56 greater than 16.

So, first, let me list all positive divisors of 56, then pick those greater than 16, subtract 7 to get b , check if $b > 9$, then sum them up.

First, factorize 56. 56 factors into 1, 2, 4, 7, 8, 14, 28, 56. Let me confirm:

Yes, 56 divided by 1 is 56

56 divided by 2 is 28

Divided by 4 is 14

Divided by 7 is 8

Then the divisors are 1, 2, 4, 7, 8, 14, 28, 56.

Now, we need divisors of 56 greater than 16. Looking at the list:

From the divisors: 28 and 56 are greater than 16.

Wait, 14 is 14, which is less than 16, so the next ones are 28 and 56. Therefore, $b + 7 \in \{28, 56\}$. Therefore, solving for b :

$$\text{Case 1: } b + 7 = 28 \implies b = 28 - 7 = 21$$

$$\text{Case 2: } b + 7 = 56 \implies b = 56 - 7 = 49$$

So the possible bases are 21 and 49. Both are greater than 9, so they satisfy the condition. Therefore, the sum is $21 + 49 = 70$.

Wait, but let me check if there are any other divisors. Wait, 56's divisors are only up to 56. But perhaps negative divisors? But since $b + 7$ is a positive quantity (as base is positive integer greater than 9), we don't need to consider negative divisors.

But let me double-check my logic here. The key step was when I did the division and found that $\frac{9b + 7}{b + 7} = 9 - \frac{56}{b + 7}$. Therefore, for this expression to be integer, $\frac{56}{b + 7}$ must be integer. Therefore, $b + 7$ divides 56. Therefore, $b + 7$ is a positive divisor of 56, and since $b > 9$, $b + 7 > 16$. Therefore, divisors of 56 greater than 16 are 28 and 56, leading to $b=21$ and 49. Therefore, sum is 70.

But let me check if there are other divisors. For example, maybe 56 divided by something else? Wait, divisors are only those numbers. Let me check 56's divisors again:

Start from 1:

$$1 \times 56 = 56$$

$$2 \times 28 = 56$$

$$4 \times 14 = 56$$

$$7 \times 8 = 56$$

So divisors are 1, 2, 4, 7, 8, 14, 28, 56. So yes, only those. Therefore, divisors above 16 are 28 and 56. So that seems right.

But let me check if these bases actually work. Let me verify with $b=21$:

$$\text{Compute } 17_{21} = 1 \times 21 + 7 = 28$$

$$\text{Compute } 97_{21} = 9 \times 21 + 7 = 189 + 7 = 196$$

Check if 28 divides 196: $196 / 28 = 7$. Yes, that's an integer. So that works.

Now check $b=49$:

$$17_{49} = 1 \times 49 + 7 = 56$$

$$97_{49} = 9 \times 49 + 7 = 441 + 7 = 448$$

Check if 56 divides 448: $448 / 56 = 8$. Yes, that's an integer. Perfect. So both bases work.

Therefore, the sum is $21 + 49 = 70$.

Wait, but hold on. Let me think again. Is there a possibility that there are more divisors if we consider other factorizations?

Wait, for example, if I had considered that maybe $b + 7$ is a divisor of 56, but perhaps I missed some? But 56 is $2^3 \times 7^1$. Its divisors are indeed as listed. So no, there are no more divisors. So 28 and 56 are the only ones above 16.

But let me think again about the equation. Suppose I didn't do the division, but instead set up the congruence.

We need $b + 7$ divides $9b + 7$. So, $b + 7 \mid 9b + 7$. Which implies that $b + 7 \mid 9b + 7 - 9(b + 7)$. Let me compute that:

$$9b + 7 - 9(b + 7) = 9b + 7 - 9b - 63 = -56$$

Therefore, $b + 7 \mid -56$. Since divisibility is unaffected by sign, this implies $b + 7 \mid 56$. Which is the same conclusion as before. Therefore, same result. Therefore, same answer.

Therefore, the answer is $21 + 49 = 70$. Therefore, the sum is 70.

But let me check if there's a possibility of other divisors if we consider that $b + 7$ could be less than or equal to 16 but somehow the original problem allows it? Wait, but the problem states that base $b > 9$. However, in the number 97_b , the digits must be valid in base b . In base b , digits must be from 0 to $b - 1$. Therefore, in 97_b , the digits are 9 and 7, so we must have $b > 9$, since digit 9 is present. Which is already given in the problem statement.

Therefore, our initial condition $b > 9$ is correct.

But just to check, for example, if we had considered divisors between 7 and 16, but since $b + 7$ must be greater than 16, they are excluded. However, just for thoroughness, let me check if any of the smaller divisors would give a base $b > 9$. For example, take divisor 14: $b + 7 = 14$ implies $b = 7$. But 7 is not greater than 9, so invalid. Similarly, divisor 8: $b + 7 = 8$ implies $b = 1$, invalid. Similarly, 7: $b = 0$, nope. 4: $b = -3$, nope. 2: $b = -5$, nope. 1: $b = -6$. So none of those work. The next divisor is 28, which gives $b = 21$, which is valid. Then 56 gives 49. So yeah, only those two.

Therefore, the answer is $21 + 49 = 70$. Therefore, the sum is 70. Therefore, I think that's the correct answer.

But let me try another approach to confirm. Let's suppose that $b + 7$ divides $9b + 7$. Therefore, there exists integer k such that:

$$9b + 7 = k(b + 7)$$

Rearranging:

$$9b + 7 = kb + 7k$$

Bring all terms to left-hand side:

$$9b + 7 - kb - 7k = 0$$

Factor:

$$b(9 - k) + 7(1 - k) = 0$$

Rearranged:

$$b(9 - k) = 7(k - 1)$$

Therefore:

$$b = \frac{7(k - 1)}{9 - k}$$

Since $b > 9$, and b must be a positive integer. Let me note that denominator is $9 - k$. Let me write this as:

Multiply numerator and denominator by -1 :

$$b = \frac{7(1 - k)}{k - 9}$$

Alternatively, maybe better to write:

Let me consider that $9 - k$ must divide $7(k - 1)$. Also, since $b > 9$, the right-hand side must be integer and greater than 9.

Alternatively, maybe express b as:

Let me write:

$$b = \frac{7(k - 1)}{9 - k}$$

Let me note that $9 - k$ must divide $7(k - 1)$. Also, since b must be positive, the numerator and denominator must have same sign.

Case 1: Both numerator and denominator positive.

So:

$$7(k - 1) > 0 \text{ and } 9 - k > 0$$

First inequality: $k - 1 > 0$ implies $k > 1$

Second inequality: $9 - k > 0$ implies $k < 9$

Therefore, k is integer between 2 and 8 inclusive.

Case 2: Both numerator and denominator negative.

So:

$$7(k - 1) < 0 \text{ and } 9 - k < 0$$

First inequality: $k - 1 < 0$ implies $k < 1$

Second inequality: $9 - k < 0$ implies $k > 9$

But k cannot be both less than 1 and greater than 9, so no solution here. Therefore, only Case 1 is possible: k from 2 to 8 inclusive.

So let me check each integer k from 2 to 8 and see if b is integer and greater than 9.

Let me tabulate:

For $k=2$:

$b = 7(2 - 1)/(9 - 2) = 7 \cdot 1/7 = 1$. But 1 is not > 9 . Disregard.

$k=3$:

$7(3 - 1)/(9 - 3) = 7 \cdot 2/6 = 14/6 = 7/3 \approx 2.333...$ Not integer. Disregard.

$k=4$:

$7(4 - 1)/(9 - 4) = 7 \cdot 3/5 = 21/5 = 4.2$. Not integer. Disregard.

$k=5$:

$7(5 - 1)/(9 - 5) = 7 \cdot 4/4 = 28/4 = 7$. $b=7$, which is not > 9 . Disregard.

$k=6$:

$7(6 - 1)/(9 - 6) = 7 \cdot 5/3 \approx 35/3 \approx 11.666...$ Not integer. Disregard.

$k=7$:

$7(7 - 1)/(9 - 7) = 7 \cdot 6/2 = 42/2 = 21$. $b=21$, which is > 9 . So that's valid. So this is one solution.

$k=8$:

$7(8 - 1)/(9 - 8) = 7 \cdot 7/1 = 49/1 = 49$. $b=49$, which is > 9 . Valid. Another solution.

Therefore, only $k=7$ and $k=8$ give valid bases $b=21$ and 49 . Which matches our previous results. Therefore, sum is $21 + 49 = 70$. Therefore, same answer.

Therefore, I think that is solid. Therefore, the answer is 70.

But just to make sure, let me check for $k=6$ again. $35/3$ is not integer. Correct. For $k=5, 7$. Not greater than 9. So yes, only $k=7$ and 8. Therefore, answer is 70. Therefore, I can be confident.

****Final Answer****
 $\boxed{70}$

With DoubtLess (0.6B), $\alpha=5$.

Okay, let me try to figure out this problem. So, we need to find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b . Hmm, let me break this down step by step.

First, I remember that numbers written in different bases convert to base 10 by multiplying each digit by the base raised to the power of its position. For example, 17_b would be $1 \times b + 7$ in base 10, right? Similarly, 97_b would be $9 \times b + 7$. Let me confirm that. Yeah, since the rightmost digit is the units place (b^0), and the next is b^1 , so yes, that makes sense.

So, translating both numbers to base 10:

$$\begin{aligned} - 17_b &= 1 \times b + 7 = b + 7 \\ - 97_b &= 9 \times b + 7 = 9b + 7 \end{aligned}$$

The problem states that 17_b divides 97_b . In other words, $b + 7$ divides $9b + 7$. So, mathematically, this means that $(9b + 7) \div (b + 7)$ should result in an integer. Alternatively, there exists some integer k such that $9b + 7 = k \times (b + 7)$.

Our goal is to find all integer bases $b > 9$ for which this divisibility condition holds, then sum those bases.

Let me think about how to approach this equation. Maybe I can express $9b + 7$ divided by $b + 7$ as a quotient plus some remainder. If the remainder is zero, then it's divisible. Alternatively, perform polynomial division or manipulate the equation algebraically.

Let me try to perform the division. Let me write $\frac{9b + 7}{b + 7}$. Let's do polynomial division or simplify it.

First, divide $9b$ by b , which is 9. Multiply $(b + 7)$ by 9: that gives $9b + 63$. Subtract that from $9b + 7$:

$$(9b + 7) - (9b + 63) = 9b + 7 - 9b - 63 = -56.$$

$$\text{Therefore, } \frac{9b + 7}{b + 7} = 9 - \frac{56}{b + 7}.$$

For this to be an integer, $\frac{56}{b + 7}$ must be an integer. Therefore, $b + 7$ must be a positive divisor of 56.

But since $b > 9$, then $b + 7 > 16$. Therefore, we need to find all divisors of 56 that are greater than 16, then subtract 7 to get the corresponding bases, and sum them up.

Wait, let me check. If $b + 7$ divides 56, then $b + 7$ is a positive divisor of 56. However, since $b > 9$, then $b + 7 > 16$. Therefore, we need to find all divisors of 56 that are greater than 16, then subtract 7

from each of them to get valid bases b , then sum those bases.

First, let me list all positive divisors of 56. The prime factorization of 56 is $2^3 \times 7^1$. Therefore, the divisors are:

1, 2, 4, 7, 8, 14, 28, 56.

So divisors greater than 16 are 28 and 56. Therefore, $b + 7 = 28$ or $b + 7 = 56$. Therefore:

If $b + 7 = 28$, then $b = 21$.

If $b + 7 = 56$, then $b = 49$.

Therefore, the bases are 21 and 49. Their sum is $21 + 49 = 70$.

Wait, is that all? Let me verify if there are any other divisors. The divisors of 56 are indeed 1, 2, 4, 7, 8, 14, 28, 56. The next one after 14 is 28, which is greater than 16. So yes, only 28 and 56. Therefore, bases 21 and 49, sum is 70.

But let me check with actual numbers to make sure.

For base 21:

$$17_{21} = 1 \times 21 + 7 = 28.$$

$$97_{21} = 9 \times 21 + 7 = 189 + 7 = 196.$$

Check if 28 divides 196: $196 / 28 = 7$. Yes, that's an integer.

For base 49:

$$17_{49} = 1 \times 49 + 7 = 56.$$

$$97_{49} = 9 \times 49 + 7 = 441 + 7 = 448.$$

Check if 56 divides 448: $448 / 56 = 8$. Yes, that's also an integer. Perfect.

Therefore, the bases are 21 and 49, sum is 70. So the answer is 70.

But let me just make sure there isn't a negative divisor or something? But since $b > 9$, and bases are positive integers greater than the digits they contain. Since the digits here are 9 and 7, the base has to be greater than 9. So base must be at least 10, but we have the condition $b > 9$, so starting from 10 upwards. But since we have $b + 7$ divides 56, and divisors of 56 are positive, as negative divisors would lead to negative $b + 7$, which would make b negative, which isn't allowed. So only positive divisors. Therefore, yes, 28 and 56 are the only ones. Hence, 21 and 49. Sum is 70.

Therefore, the answer is $\boxed{70}$.

****Final Answer****
 $\boxed{70}$