



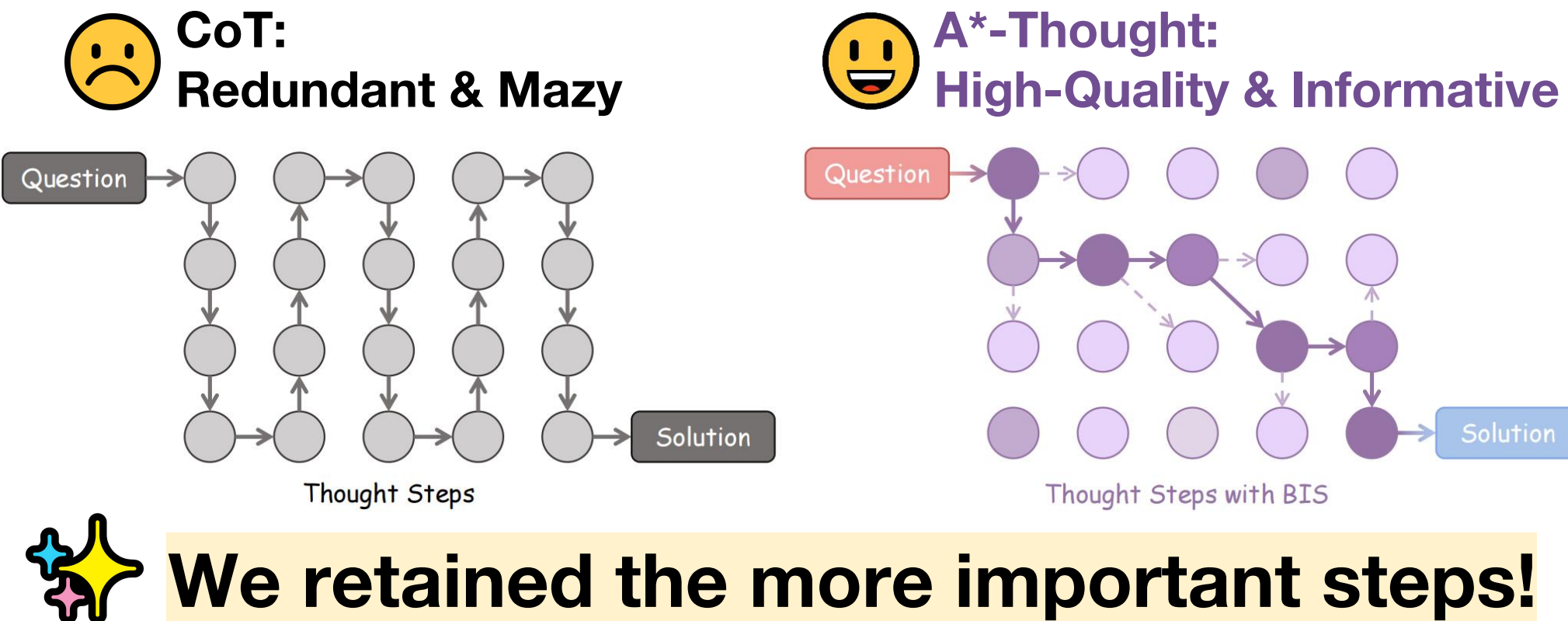
# A\*-Thought: Efficient Reasoning via Bidirectional Compression for Low-Resource Settings

Xiaoang Xu<sup>1</sup>, Shuo Wang<sup>2\*</sup>, Xu Han<sup>2,4,5</sup>, Zhenghao Liu<sup>3</sup>, Huijia Wu<sup>1</sup>, Peipei Li<sup>1</sup>, Zhiyuan Liu<sup>2,4,5</sup>, Maosong Sun<sup>2,4,5</sup>, Zhaofeng He<sup>1\*</sup>

<sup>1</sup> BUPT <sup>2</sup> Dept. of Comp. Sci. & Tech., Tsinghua University <sup>3</sup> Northeastern University <sup>4</sup> Institute for AI, Tsinghua University <sup>5</sup> Beijing National Research Center for Information Science and Technology



## Chain-of-Thought v.s. A\*-Thought



## Step-Level Bidirectional Importance Score (BIS)

**Attention-Level**

$$\text{ATTN}(\mathbf{y}|\mathbf{x}) = \frac{1}{H|\mathbf{y}||\mathbf{x}|} \sum_{h=1}^H \sum_{j=1}^{|\mathbf{y}|} \sum_{i=1}^{|\mathbf{x}|} a_h(y_j, x_i)$$

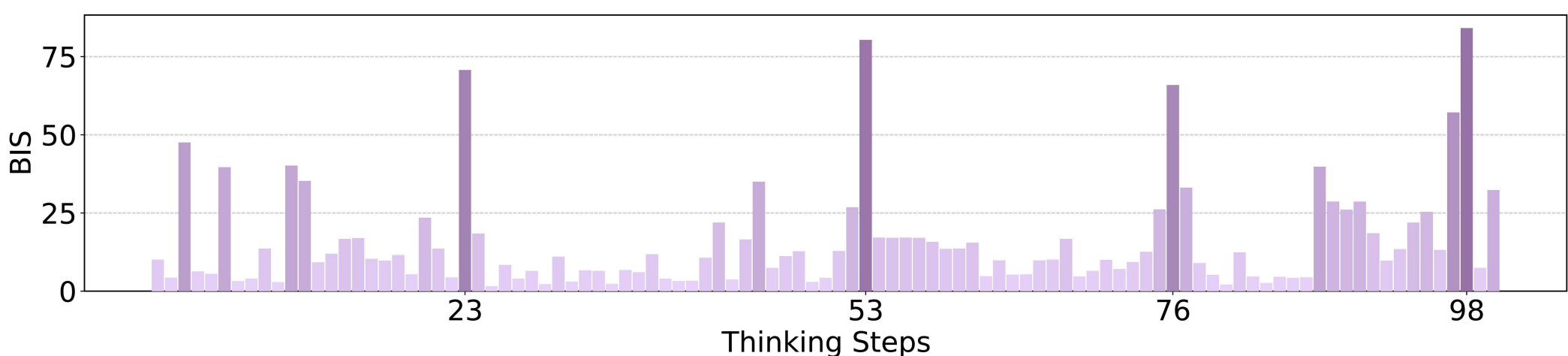
**Modle-Level**

$$\text{NLL}(\mathbf{y}|\mathbf{x}) = -\frac{1}{|\mathbf{y}|} \sum_{j=1}^{|\mathbf{y}|} \log P(y_j | \mathbf{x}, y_{<j})$$

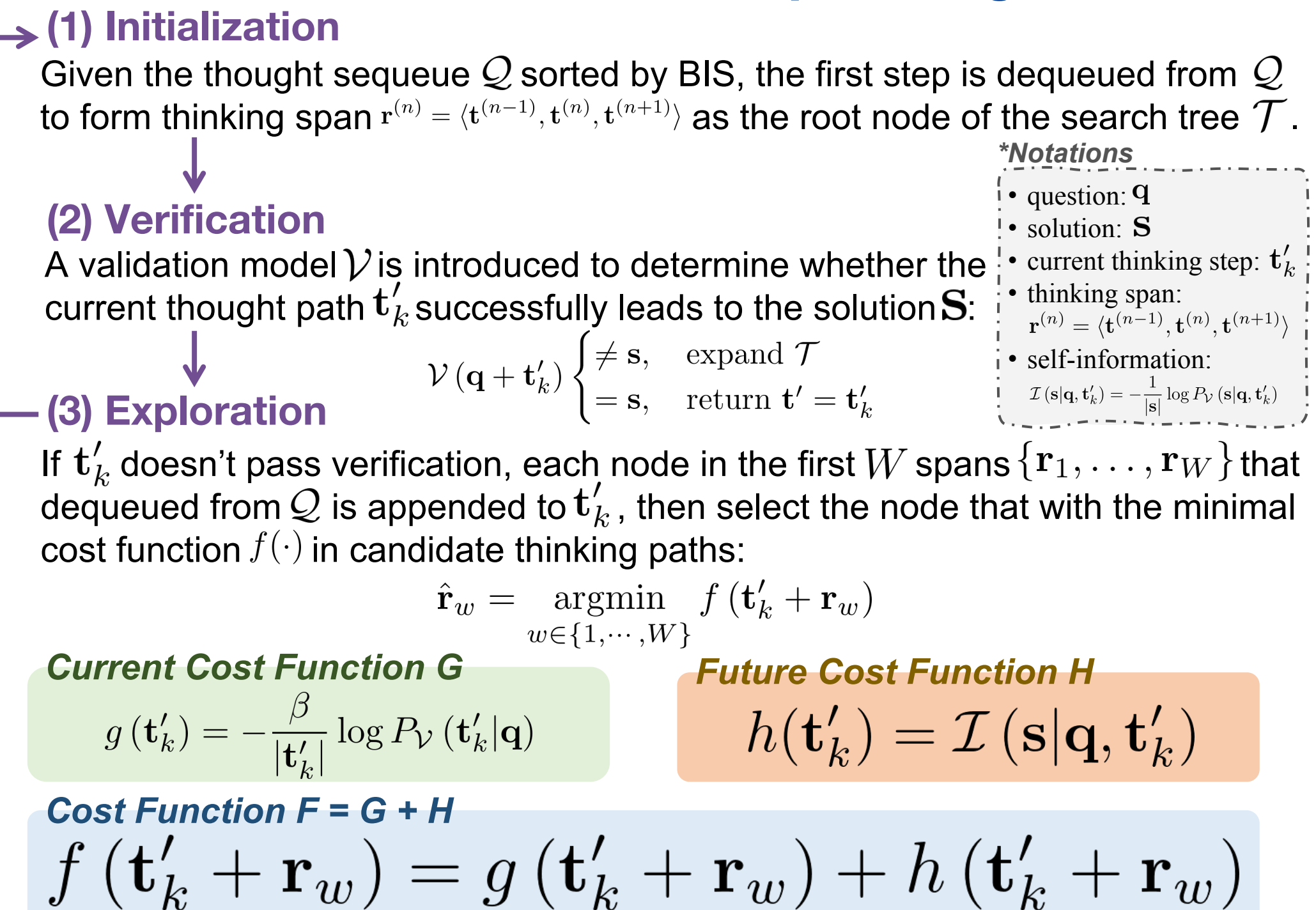
**Question-Solution Global Information**

$$\text{BIS}(\mathbf{t}^{(n)}) = \frac{(1 - \alpha) \text{ATTN}(\mathbf{q}|\mathbf{t}^{(n)}) + \alpha \text{ATTN}(\mathbf{s}|\mathbf{t}^{(n)})}{(1 - \alpha) \text{NLL}(\mathbf{q}|\mathbf{t}^{(n)}) + \alpha \text{NLL}(\mathbf{s}|\mathbf{t}^{(n)})}$$

Only a very few thinking steps have high BIS ... How to sample them?



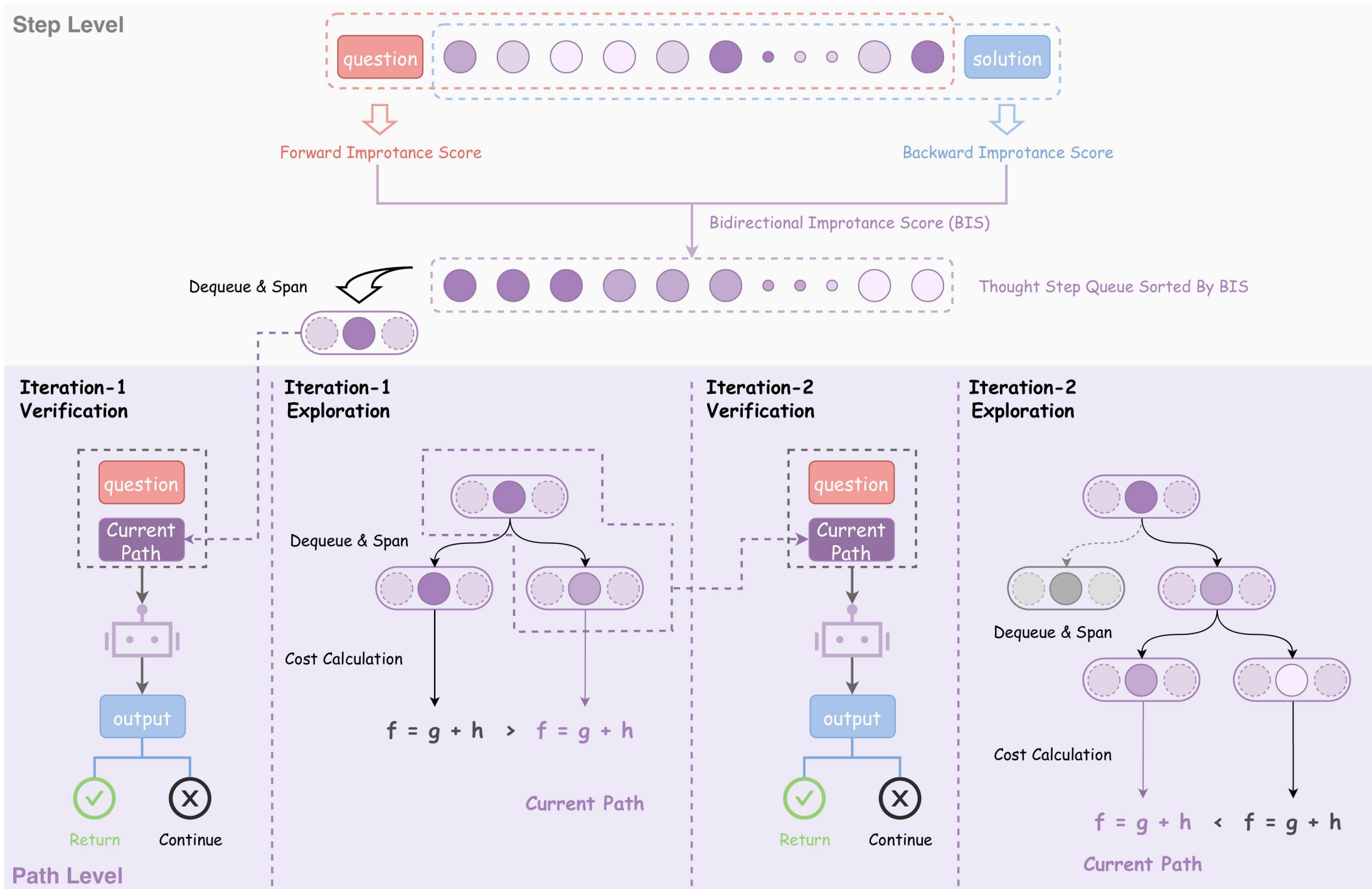
## Path-Level A\* Search for Compressing CoT



## A\*-Thought Framework

**1. How to identify high-quality and informative steps at the step-level?**

We design a **step-level bidirectional importance score (BIS)** to evaluate the criticality of individual sentences. This scoring mechanism serves to significantly enhance effectiveness of the A\* search procedure compared to standard sampling.



**2. How to effectively assemble individual thinking steps into a concise and effective reasoning path at the path-level?**

We introduce a **path-level A\* search algorithm** tailored for compressing lengthy CoTs from LRMs. It strategically considers both current path quality and estimated future costs to optimize LRM performance under stringent output length constraints.

## Efficiency & Performance

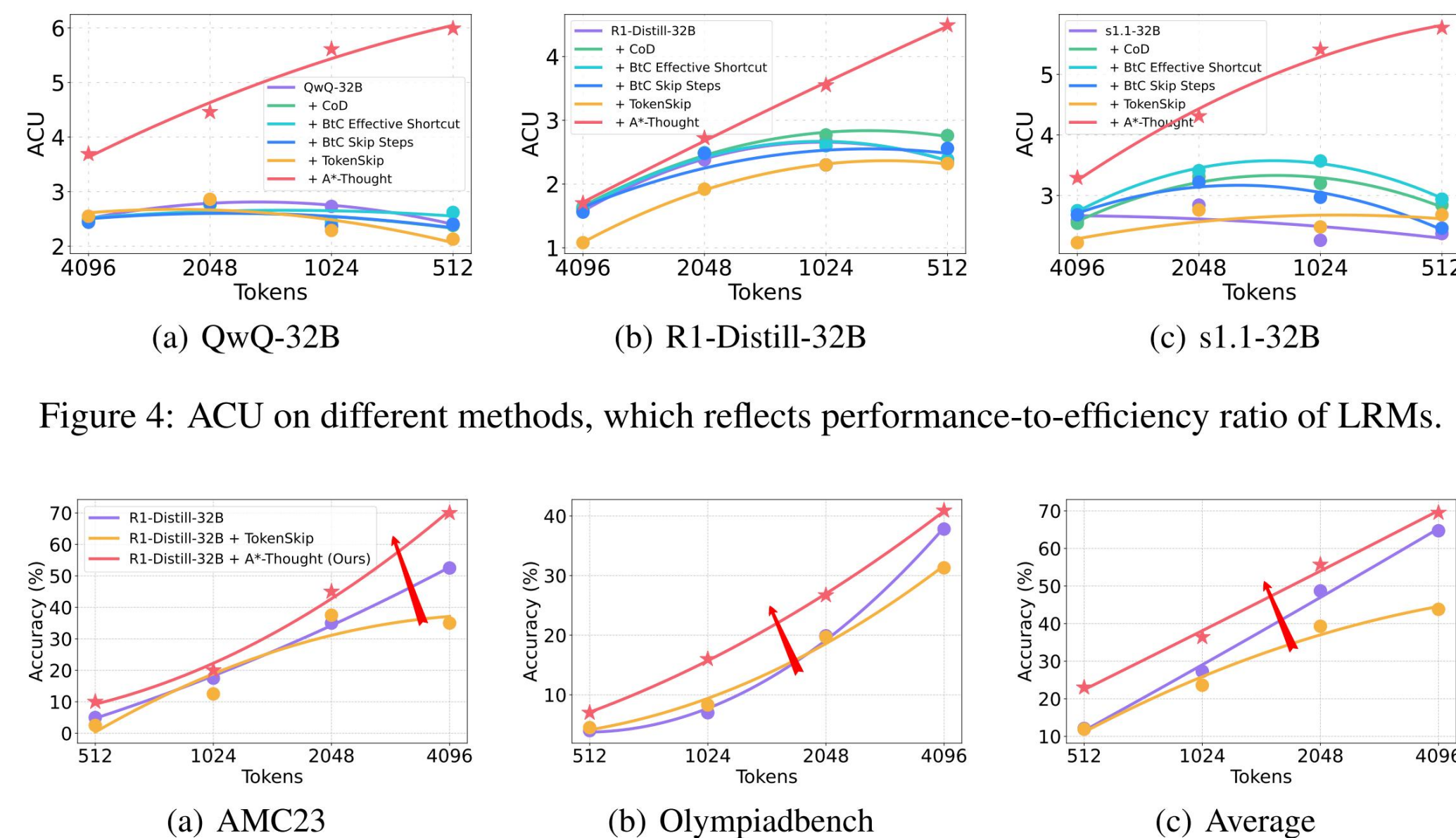


Figure 4: ACU on different methods, which reflects performance-to-efficiency ratio of LRMs.

Figure 5: Performance of R1-Distill-32B augmented using TokenSkip and A\*-Thought. "Average" denotes the average accuracy of the model in MATH500, AMC23, OlympiadBench, and GSM8K.

## Results

Table 1: Experimental results of different long-to-short methods across several benchmarks. The best results are shown in **bold**, and the second-best results are underlined.

Methods	MATH500		AMC23		OlympiadBench		GSM8K		Average		ACU
	Acc.( $\uparrow$ )	Len.( $\downarrow$ )	Acc.( $\uparrow$ )	Len.( $\downarrow$ )	Acc.( $\uparrow$ )	Len.( $\downarrow$ )	Acc.( $\uparrow$ )	Len.( $\downarrow$ )	Acc.( $\uparrow$ )	Len.( $\downarrow$ )	
<b>Budget: 512 Tokens</b>											
QwQ-32B	10.8	512.00	2.5	512.00	3.3	512.00	27.6	511.97	11.1	511.99	2.16
QwQ-32B w/ s1K-1.1	9.6	512.00	7.5	512.00	3.4	512.00	28.8	512.00	12.3	512.00	2.41
+ CoD	10.6	512.00	5.0	512.00	4.2	512.00	29.0	511.96	12.2	511.99	2.38
+ BtC Effective Shortcut	10.2	512.00	<u>12.5</u>	512.00	4.2	512.00	26.7	511.95	<u>13.4</u>	511.99	<u>2.62</u>
+ BtC Skip Steps	9.6	512.00	5.0	512.00	5.6	512.00	28.9	511.95	12.3	511.99	2.40
+ TokenSkip	10.8	511.05	2.5	512.00	3.9	512.00	26.4	508.11	10.9	510.79	2.13
+ A*-Thought	<b>33.2</b>	<b>491.92</b>	<b>15.0</b>	<b>508.60</b>	<b>12.0</b>	<b>509.74</b>	<b>57.4</b>	<b>451.76</b>	<b>29.4</b>	<b>490.51</b>	<b>5.99</b>
<b>Budget: 1024 Tokens</b>											
QwQ-32B	16.6	1016.85	15.0	1024.00	6.4	1023.93	49.1	951.96	21.8	1004.19	2.17
QwQ-32B w/ s1K-1.1	24.8	1023.52	17.5	1024.00	8.9	1023.94	60.1	999.80	27.8	1017.82	2.73
+ CoD	24.8	1023.37	5.0	1024.00	7.3	1023.64	60.1	996.84	24.3	1016.96	2.39
+ BtC Effective Shortcut	23.4	1022.88	7.5	1024.00	7.7	1023.92	61.3	1000.44	25.0	1017.81	2.45
+ BtC Skip Steps	23.4	1023.25	5.0	1024.00	7.6	1024.00	59.9	1000.93	24.0	1018.05	2.36
+ TokenSkip	22.4	995.96	12.5	1024.00	6.4	1019.61	49.7	934.74	22.8	993.58	2.29
+ A*-Thought	<b>50.8</b>	<b>858.28</b>	<b>37.5</b>	<b>928.25</b>	<b>22.3</b>	<b>954.74</b>	<b>81.9</b>	<b>688.69</b>	<b>48.1</b>	<b>857.49</b>	<b>5.61</b>
<b>Budget: 2048 Tokens</b>											
QwQ-32B	51.2	1844.96	25.0	1978.60	18.4	2021.95	80.4	1245.68	43.8	1772.80	2.47
QwQ-32B w/ s1K-1.1	60.0	1887.15	35.0	2000.95	23.3	2012.14	88.7	1474.00	51.8	1843.56	2.81
+ CoD	60.2	1894.54	30.0	2022.35	25.5	2018.02	89.5	1490.23	51.3	1856.29	2.76
+ BtC Effective Shortcut	60.8	1884.67	35.0	2004.65	23.7	2012.43	89.8	1473.25	52.3	1843.75	2.84
+ BtC Skip Steps	58.8	1884.96	35.0	2005.67	23.2	2013.05	89.2	1490.39	51.6	1848.52	2.79
+ TokenSkip	53.6	1685.34	35.0	1923.25	19.7	1943.68	86.7	1272.03	48.8	1706.08	2.86
+ A*-Thought	<b>69.2</b>	<b>1271.76</b>	<b>45.0</b>	<b>1540.30</b>	<b>30.3</b>	<b>1625.89</b>	<b>91.2</b>	<b>843.69</b>	<b>58.9</b>	<b>1320.41</b>	<b>4.46</b>
<b>Budget: 4096 Tokens</b>											
QwQ-32B	75.4	2798.67	55.0	3456.05	36.5	3645.22	85.8	1348.24	63.2	2812.05	2.25
QwQ-32B w/ s1K-1.1	79.6	2693.27	65.0	3485.95	<b>42.4</b>	3500.66	<b>95.2</b>	1624.11	<b>70.6</b>	2826.00	2.50
+ CoD	<b>80.2</b>	2719.00	60.0	3354.28	42.0	3488.67	95.0	1655.80	69.3	2804.44	2.47
+ BtC Effective Shortcut	79.6	2696.72	57.5	3355.43	42.4	3493.28	94.8	1636.45	68.6	2795.47	2.45
+ BtC Skip Steps	80.2	2710.83	57.5	3399.93	41.8	3494.41	94.9	1651.37	68.6	2814.14	2.44
+ TokenSkip	74.4	2336.29	52.5	3156.68	37.8	3289.44	94.8	1412.87	64.9	2548.82	2.55
+ A*-Thought	78.8	<b>1699.78</b>	<b>65.0</b>	<b>2385.85</b>	40.1	<b>2546.45</b>	93.1	<b>874.54</b>	<u>69.3</u>	<b>1876.66</b>	<b>3.69</b>

## Analysis

Table 4: Amount of the training data and the corresponding time.  $\rho$  denotes the compression rate.

Training Set	$\rho$	Training Time (s)		
		QwQ-32B	R1-Distill-32B	s1.1-32B
s1K-1.1	100.00%	13819.60	13784.32	13638.07
+ TokenSkip	85.41% ( $\downarrow$ 14.59%)	12846.14 ( $\downarrow$ 7.04%)	12700.96 ( $\downarrow$ 7.86%)	12679.52 ( $\downarrow$ 7.03%)
+ A*-Thought	31.31% ( $\downarrow$ 68.69%)	10468.20 ( $\downarrow$ 24.25%)	10809.50 ( $\downarrow$ 21.58%)	10550.03 ( $\downarrow$ 22.64%)

Mikka wants to order a pizza with two toppings. He has 8 different toppings to choose from. How many different pizzas could he order?

Okay, so Mikka wants to order a pizza with two different toppings, and there are 8 different toppings available. I need to figure out how many different pizzas he could order. Hmm, let me think.

First, when the problem says "two different toppings," that means Mikka is choosing two distinct toppings from the 8 available. The question is asking about the number of combinations possible. Since the order of the toppings on the pizza doesn't matter—like, a pizza with pepperoni and mushrooms is the same as a pizza with mushrooms and pepperoni—this is a combination problem, right? Not a permutation, because permutations care about the order, and combinations don't.

So, the formula for combinations is  $C(n, k) = \frac{n!}{k!(n-k)!}$ , where  $n$  is the total number of items, and  $k$  is the number of items you're choosing. In this case,  $n$  is 8 toppings, and  $k$  is 2 toppings. Let me plug those numbers into the formula.

$$C(8, 2) = \frac{8!}{2!(8-2)!} = \frac{8!}{2! \cdot 6!} = \frac{8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}{2 \times 1 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}$$

Wait, 8! is  $8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$ , but since there's a 6! in the denominator, we can cancel out the 6! part. So that leaves  $\frac{8 \times 7}{2 \times 1}$ . Let me compute that:  $8 \times 7$  is 56, and  $2 \times 1$  is 2. So 56 divided by 2 is 28. So, there are 28 different combinations.

But let me make sure I didn't make a mistake here. Sometimes, when I do combinations, I get confused between whether it's permutations or combinations. Let me double-check. If the problem had said that the order of the toppings matters, like if they were arranged in a specific way on the pizza, then it would be a permutation. But since the pizza is just a circle, and toppings are spread around, the order isn't important. So, yeah, combinations are correct.

Alternatively, I can think of it as Mikka choosing the first topping and then the second topping. For the first topping, he has 8 choices, and then for the second topping, he has 7 choices left. So,  $8 \times 7 = 56$ . But wait, this counts each pair twice because choosing topping A first and then topping B is the same as choosing topping B first and then topping A. So to correct for that overcounting, we divide by 2.  $56 / 2 = 28$ . Yep, that matches the combination result.

Wait, just to be thorough, let's consider an example with fewer toppings. Suppose there were only 3 toppings: A, B, and C. How many different two-topping pizzas could he make? Let's list them: AB, AC, BC. That's 3. According to the formula,  $C(3, 2) = \frac{3!}{2!(3-2)!} = \frac{3!}{2! \cdot 1!} = \frac{3 \times 2 \times 1}{2 \times 1 \times 1} = 3$ . Correct. So that works. If we tried the other method,  $\frac{3 \times 2}{2} = 3$ . Also correct. So the formula works here.

Therefore, returning to the original problem, with 8 toppings, the number of two-topping combinations is 28. So Mikka has 28 different pizza options.

**Final Answer** The number of different pizzas Mikka could order is **28**.

Mikka wants to order a pizza... Using the combination formula:

$$C(8, 2) = \frac{8!}{2!(8-2)!} = \frac{8 \times 7}{2 \times 1} = 28$$

Therefore, Mikka can order 28 different pizzas.

**Final Answer** The number of different pizzas Mikka could order is **28**.

Thus, the number of different pizzas Mikka could order is **28**.

Figure 7: Responses generated by QwQ-32B models trained with and without A\*-Thought. Red box represents the question, purple box represents the thinking process, blue box represents the solution.

Email for more questions: [xuxiaoang@bupt.edu.cn](mailto:xuxiaoang@bupt.edu.cn)