Mercury: An Efficiency Benchmark for LLM Code Synthesis

Du ***Mingzhe***

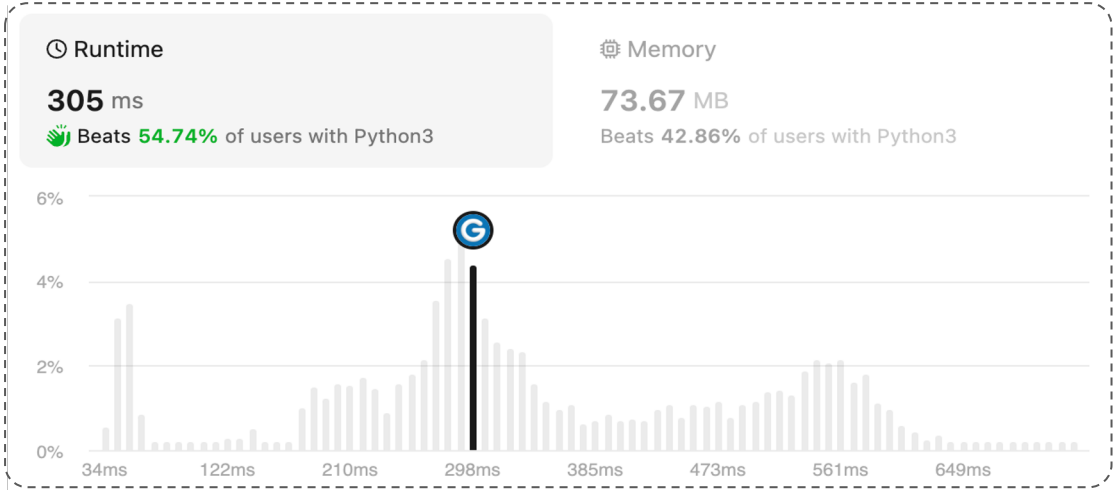
Paper Link: https://arxiv.org/abs/2402.07844

What is Code Efficiency?

```python
# Sort an array of integers in ascending order.

# Quick Sort - 121 ms
def sortArray(self, nums):
    def quicksort(nums, l, r):
        if r - l <= 1: return
        pivot = partition(nums, l, r)
        quicksort(nums, l, pivot)
        quicksort(nums, pivot + 1, r)
    quicksort(nums, 0, len(nums))
    return nums

# Bubble Sort - 5714 ms
def sortArray(self, nums):
    i = 0
    while i < len(nums)-1:
        j = i + 1
        while j < len(nums):
            if nums[i] > nums[j]:
                nums[i],nums[j] = nums[j],nums[i]
            j += 1
        i += 1
    return nums
```



$$Beyond@K = \frac{\sum_{N,K}^{n=0,k=0}[1-\frac{max(R_A^n)-R_k^n}{max(R_A^n)-min(R_A^n)}]}{N \cdot K}$$

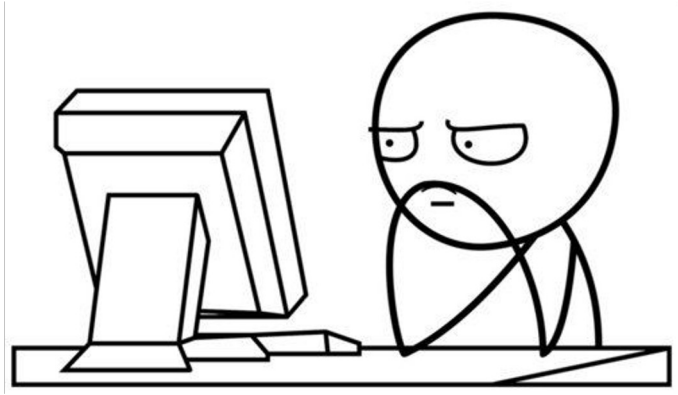
Why Efficiency is Important?

One Example - GTA Online

Slow Game Loading.

Lasted **7** Years.

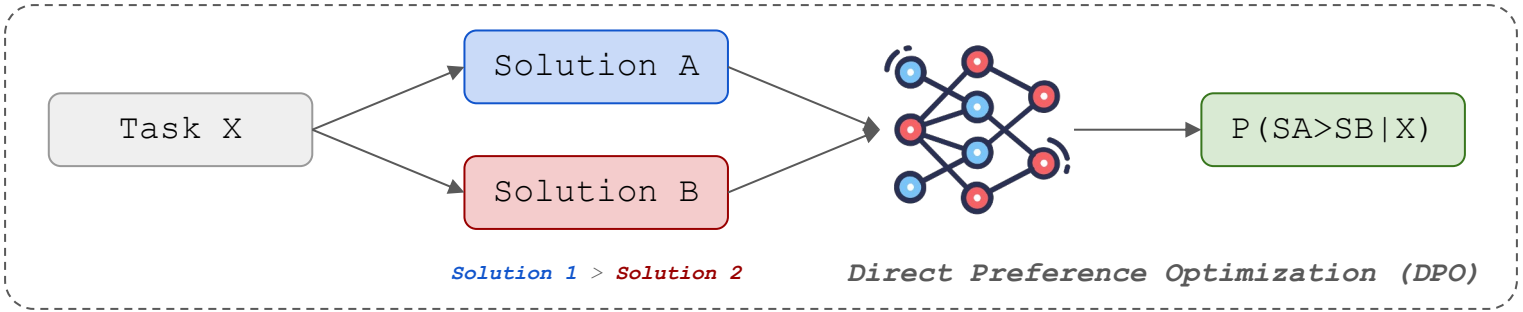
6 Mins to **1** Mins.



3,046,557 Active Monthly Users.

7 * 5 * 3,046,557 = **106,629,495** mins are wasted!

How to Improve Efficiency?



Neural network icons created by Vectors Tank - Flaticon

Results

Model Name	Mercury-easy			Mercury-medium			Mercury-hard			Mercury-Average		
	B@1	B@3	B@5	B@1	B@3	B@5	B@1	B@3	B@5	B@1	B@3	B@5
deepseek-coder-6.7b	63.55	72.55	75.19	58.80	65.70	67.28	45.72	48.52	50.58	55.99	62.22	64.33
+ SFT	57.70	72.57	75.18	54.87	65.24	66.77	41.87	47.54	49.62	51.42	61.74	63.83
+ DPO	58.29	73.03	75.65	55.86	66.15	67.77	43.70	49.52	51.54	52.56	62.87	64.96
CodeLlama-7b	46.49	55.33	56.37	36.08	46.02	49.05	6.83	9.46	9.48	29.72	36.79	38.12
+ SFT	42.57	52.48	54.54	37.48	40.54	45.48	5.11	3.97	4.86	28.23	32.22	34.79
+ DPO	45.90	55.87	56.83	40.58	46.54	49.57	8.32	10.41	10.47	31.45	37.47	38.78
CodeLlama-13b	42.30	48.47	52.79	39.95	53.32	55.60	7.94	14.61	15.75	29.88	38.50	41.09
+ SFT	42.21	54.90	56.36	35.54	44.47	47.71	3.10	8.78	9.68	26.81	35.92	37.76
+ DPO	40.08	54.50	59.67	40.99	59.19	61.18	13.70	19.16	20.54	31.40	43.97	46.85
CodeLlama-34b	62.50	74.35	77.58	54.02	65.84	67.17	23.17	24.02	28.51	46.45	54.55	57.61
+ SFT	64.49	78.51	83.99	53.88	69.47	71.00	15.17	22.09	27.52	44.37	56.48	60.69
+ DPO	71.62	84.24	87.89	63.15	74.81	76.14	27.30	32.19	36.41	53.88	63.57	66.68
deepseek-coder-33b	67.92	76.61	78.78	67.18	73.34	76.27	52.36	58.79	59.43	62.40	69.52	71.41
+ SFT	68.58	79.85	83.04	67.16	76.10	79.38	45.09	55.55	57.87	60.15	70.41	73.33
+ DPO	**77.83**	**86.06**	**88.83**	**75.15**	**83.08**	**85.20**	60.52	**66.70**	**67.64**	**71.10**	**78.53**	**80.48**
gpt-4-preview-1106	70.54	-	-	62.53	-	-	**70.06**	-	-	67.84	-	-

Table 4: Code Efficiency Evaluation Results Categorized by Difficulty Levels. The notation $B@K$ represents the $Beyond@K$ score with K generated solutions. **The bolded value** indicates the top performance for each metric, while the underlined values denote the most effective approaches among the original model and the baselines.

Thank you 😀