

000 001 NEMOTRON-CC-MATH: A 133 BILLION-TOKEN- 002 SCALE HIGH QUALITY MATH PRETRAINING DATASET 003 004

005 **Anonymous authors**

006 Paper under double-blind review

007 008 ABSTRACT 009

011 Pretraining large language models (LLMs) on high-quality, structured data such
012 as mathematics and code substantially enhances reasoning capabilities. However,
013 existing math-focused datasets built from Common Crawl suffer from degraded
014 quality due to brittle extraction heuristics, lossy HTML-to-text conversion, and
015 the failure to reliably preserve mathematical structure. In this work, we introduce
016 Nemotron-CC-Math, a large-scale, high-quality mathematical corpus con-
017 structed from Common Crawl using a novel, domain-agnostic pipeline specifically
018 designed for robust scientific text extraction.

019 Unlike previous efforts, our pipeline recovers math across various formats (e.g.,
020 MathJax, KaTeX, MathML) by leveraging layout-aware rendering with lynx and
021 a targeted LLM-based cleaning stage. This approach preserves the structural in-
022 tegrity of equations and code blocks while removing boilerplate, standardizing
023 notation into L^AT_EX representation, and correcting inconsistencies.

024 We collected a large, high-quality math corpus, namely Nemotron-CC-Math-3+
025 (133B tokens) and Nemotron-CC-Math-4+ (52B tokens). Notably, Nemotron-
026 CC-Math-4+ not only surpasses all prior open math datasets—including Mega-
027 Math, FineMath, and OpenWebMath—but also contains 5.5 \times more tokens than
028 FineMath-4+, which was previously the highest-quality math pretraining dataset.
029 When used to pretrain a Nemotron-T 8B model, our corpus yields +4.8 to +12.6
030 gains on MATH and +4.6 to +14.3 gains on MBPP+ over strong baselines, while
031 also improving general-domain performance on MMLU and MMLU-Stem.

032 We present the first pipeline to reliably extract scientific content—including math—
033 from noisy web-scale data, yielding measurable gains in math, code, and general
034 reasoning, and setting a new state of the art among open math pretraining corpora.
035 To support open-source efforts, we will release our code and datasets¹.
036

037 1 INTRODUCTION 038

039 The rapid advancement of large language models (LLMs) has sparked a growing interest in impro-
040 ving their mathematical reasoning capabilities. Recent studies indicate that pretraining on carefully
041 curated domain-specialized data—such as mathematics (Paster et al., 2024; Han et al., 2024; Wang
042 et al., 2024b; Azerbayev et al., 2024) and code (Kocetkov et al., 2022; Lozhkov et al., 2024; Li
043 et al., 2023)—substantially improves domain-specific accuracy, general knowledge and reasoning
044 abilities (Muennighoff et al., 2023; Aryabumi et al., 2024; Lewkowycz et al., 2022; Shao et al.,
045 2024). This suggests that high-quality mathematical data plays a pivotal role not only in solving
046 math problems but also in strengthening broader reasoning skills.

047 Math capabilities of models like O1 (OpenAI, 2024) and DeepSeek-R1 (Guo et al., 2025) critically
048 depend on access to large-scale, high-quality mathematical pretraining data. Unfortunately, datasets
049 used in pretraining SOTA models like DeepSeekMath (Shao et al., 2024), Minerva (Lewkowycz
050 et al., 2022) and Qwen-2.5-Math (Yang et al., 2024) are not publicly released. Meanwhile, open-
051 source alternatives such as OpenWebMath (OWM) (Paster et al., 2024), FineMath (Allal et al.,
052 2025), InfiIMMWebMath (Han et al., 2024) and MathPile (Wang et al., 2024b) remain limited in

053 ¹Code and data will be released upon acceptance.

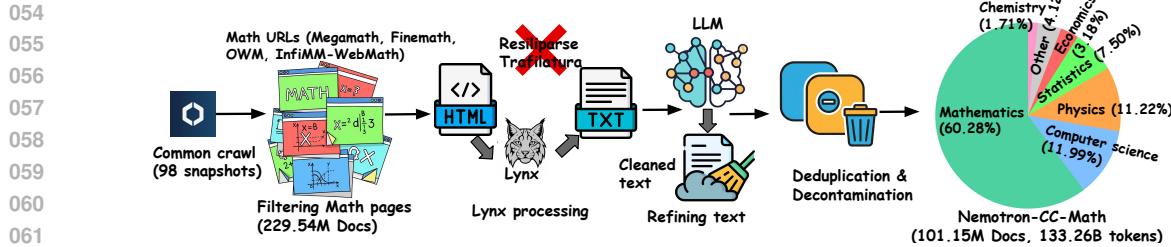


Figure 1: Overview of the Nemotron-CC-Math construction pipeline. Starting from Common Crawl snapshots, we extract math-related URLs using curated datasets (e.g., MegaMath, FineMath). After fetching 229.54M webpages, we render pages through Lynx and apply LLM-based cleaning, quality filtering, and deduplication (see §2.1). We visualize the topic distribution of our data (Right).

both scale and fidelity—largely due to brittle extraction pipelines that degrade content quality and fail to preserve mathematical equations and structure (see Appendix A.3).

While Common Crawl forms a primary source for large-scale pretraining (Penedo et al., 2023; Tang et al., 2024; Su et al., 2025), its value for mathematical pretraining remains underexploited. Existing math-specific extraction pipelines (Paster et al., 2024; Zhou et al., 2025) are not well-suited to fully leverage this resource. In particular, current methods frequently fail to detect or accurately extract equations, either omitting them altogether or corrupting their structure (Han et al., 2024; Allal et al., 2025). This severely compromises content fidelity. Mathematical notation on the web appears in a wide range of formats—including MathML, L^AT_EX, and dynamically rendered scripts—whose representations continue to evolve over time (see Figure 2). Compounding this challenge, HTML pages in Common Crawl often lack associated stylesheets and JavaScript resources, preventing proper rendering and further obstructing reliable equation recovery. These limitations collectively hinder the construction of high-quality mathematical pretraining corpora that capture the breadth and variety of real-world mathematical content.

To bridge this gap, we propose a modular, scalable, and domain-agnostic framework for reliably extracting mathematically rich content from raw web data, enabling the construction of a large-scale, diverse, and high-fidelity math corpus. Our multi-stage extraction and filtering pipeline ensures quality at scale (see Figure 1). In the first stage, HTML documents are rendered into text using the Lynx text-based browser², which preserves mathematical equations and symbols with high accuracy. In the second stage, a lightweight LLM normalizes heterogeneous math representations into L^AT_EX while discarding boilerplate and irrelevant content. This LLM-based approach allows us to avoid the brittle, heuristic-based rules employed in previous pipelines (Paster et al., 2024), resulting in more reliable and consistent extraction of mathematical content. Subsequently, we apply a quality classifier to retain the high-quality pages, followed by deduplication to eliminate redundancy. In addition, we perform thorough contamination detection against downstream benchmarks (see § 2.4), ensuring that any overlapping or duplicated samples are identified and removed from the corpus.

By leveraging the scale of Common Crawl and the rigor of our pipeline, we present Nemotron-CC-Math—the highest quality open-source math corpus to date, comprising of 133B tokens, where its highest quality subset (Nemotron-CC-Math-4+) totals 53B tokens. Our pipeline is optimized for performance using Polars and Ray, enabling us to process terabytes of HTML content efficiently. To facilitate future research, we release both the dataset and our full pipeline implementation.

Our contributions are as follows:

- We reviewed prior data extraction pipelines, and show that they fail to accurately extract math and code content, often stripping math equations and code snippets (Appendix A.3).
- We introduce a scalable and modular pipeline for extracting high-quality mathematical content from Common Crawl, explicitly addressing the longstanding challenge of HTML math variability—including L^AT_EX, MathML, Unicode, and inline or malformed equations.
- To our knowledge, this is the first work to employ the text-based browser Lynx for HTML-to-plain-text conversion with preservation of math and code formatting, and to introduce LLM-based standardization of mathematical representations across the content.

²<https://lynx.invisible-island.net/>

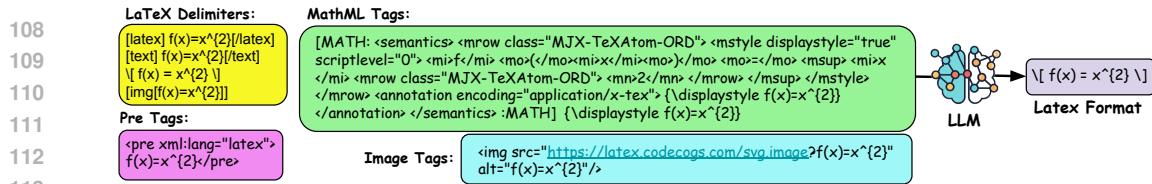


Figure 2: Mathematical expressions on HTML pages appear in diverse formats—LaTeX within custom delimiters, `<pre>` blocks, image tags, and MathML. These variations challenge standard text extraction pipelines, which often fail to recover the underlying \LaTeX equations correctly. To address this, we use an LLM to standardize all mathematical representations into a unified \LaTeX format.

- We will release Nemotron-CC-Math, a dataset of 133B tokens of high-quality and diverse math-rich web documents extracted from Common Crawl, whose 4+ subset contains $5.5 \times$ more tokens than FineMath-4+, the previous best math pretraining dataset.
- We will open-source our full pipeline (extraction, processing and scoring) to ensure reproducibility and support its application to other domains.
- We thoroughly analyze Nemotron-CC-Math by examining its composition, including statistics on webpage types, subject areas, and the most frequent source domains.
- Through extensive experiments and detailed quality analysis (§3.3), we demonstrate that models pretrained on our dataset outperform those trained on existing pretraining math datasets across a range of benchmarks, including math, code, and general knowledge tasks.

2 THE COLLECTION OF NEMOTRON-CC-MATH

We construct the Nemotron-CC-Math corpus from Common Crawl³, a large-scale web archive extensively used in recent LLM training (Dubey et al., 2024; Hui et al., 2024; DeepSeek-AI et al., 2025). Common Crawl contains over 300B documents across more than 6M WARC files (each contains over 1GB of compressed content). Our goal is to build a pipeline that can process the technical content from Common Crawl correctly. We apply our pipeline to math domain to assemble a high-quality, large-scale corpus of mathematical content from Common Crawl. To achieve this, we designed a robust and highly scalable data processing pipeline capable of operating at the full scale of Common Crawl, as illustrated in Figure 1.

Prior efforts such as OWM (Paster et al., 2024) and DeepSeekMath (Shao et al., 2024) rely on lightweight classifiers to identify technical pages. We initially explored a similar approach but found fundamental limitations in achieving meaningful improvements: first, mathematical content constitutes $< 1\%$ of Common Crawl, making manual ground truth annotation extremely difficult; second, since classifiers must run on all Common Crawl documents, only very efficient methods like FastText with simplified HTML parsing are viable. This creates an inherently high-bias setup with no straightforward path to improvement—attempts to increase recall for technical content invariably lead to drastic drops in precision. Rather than refining such classifiers for marginal gains, we leverage community-filtered datasets: extracting URLs from OWM, InfiMM-WebMath (Han et al., 2024), FineMath (Allal et al., 2025), and MegaMath (Zhou et al., 2025), including all major subsets. This approach allows us to benefit from the diverse filtering strategies employed by different research groups while avoiding the limitations of any single classifier.

We then retrieve the original HTML from 98 Common Crawl snapshots (2014-2024) for these URLs, enabling fine-grained extraction that preserves mathematical expressions, symbols, and formatting—often degraded in prior processing (Appendix A.3). This process yields 229.54M high-quality web-pages spanning a diverse range of mathematical content.

2.1 RELIABLE TEXT EXTRACTION FOR SCIENTIFIC CONTENT

2.1.1 LIMITATIONS OF PRIOR WORK

Extracting mathematical content from raw HTML presents a significant challenge for text extraction pipelines. Unlike natural language, which often follows consistent structural patterns, math equa-

³<https://commoncrawl.org/>

162 tions appear in highly variable forms across the web (see Figure 2). These variations stem from the
 163 absence of standardized conventions for embedding math in HTML, as well as the diversity of ren-
 164 dering engines (e.g., MathJax, KaTeX, MathML, image-based representations, and custom plugins).
 165 Moreover, websites frequently evolve their rendering strategies, making any fixed set of heuristics
 166 fragile in practice. As a result, existing extraction pipelines often fail to reliably extract scientific
 167 content, with equations either missed entirely, mis-parsed, or distorted. Preserving formatting is
 168 equally important: the indentation and layout of code blocks and the placement of mathematical
 169 symbols often carry semantic meaning, and losing this structure severely degrades the value of the
 170 extracted content for downstream modeling.

171 Existing content extraction tools such as JUSTTEXT (Endrédy & Novák, 2013), TRAFILATURA (Bar-
 172 baresi, 2021), and RESILIPARSE (Bevendorff et al., 2018)—used in large-scale dataset construction
 173 pipelines including The Pile (Gao et al., 2020), FineMath (Allal et al., 2025), and RefinedWeb
 174 (Penedo et al., 2023)—were designed primarily for general-purpose boilerplate removal and nar-
 175 rative text extraction. While effective for general documents, they often strip or corrupt equations,
 176 miss inline L^AT_EX equations changing semantics, and flatten (or miss) code blocks requiring strict
 177 indentation (e.g., Python). These shortcomings limit their usability for building high-quality math
 178 or code datasets. Examples are provided in Appendix A.3.

179 2.1.2 OUR TEXT EXTRACTION PIPELINE

180 The diversity of mathematical representations on the web necessitates using large language models
 181 to faithfully convert technical HTML content into a format suitable for LLM pretraining. Since
 182 raw HTML from WARC files is too verbose for direct LLM processing, and traditional parsers risk
 183 losing critical information, we employ `lynx`, a text-based browser that renders web pages into plain
 184 text while preserving mathematical equations and code formatting. Unlike DOM-based parsers used
 185 in prior work (Paster et al., 2024; Zhou et al., 2025; Allal et al., 2025), `lynx` executes HTML layout
 186 rules to produce output that mirrors the human-perceived page structure, reliably capturing equations
 187 and maintaining code indentation.

188 While `lynx` preserves the structural layout of web pages, its output includes boilerplate elements
 189 such as navigation bars and redundant headers. To refine this output, we apply an LLM cleanup pass
 190 using Phi-4 (Abdin et al., 2024a)(14B parameters), which preserves primary content and references
 191 while removing non-essential content. LLM additionally standardizes mathematical expressions
 192 into consistent L^AT_EX format (Figure 2), and corrects typographical errors. This two-stage pipeline—
 193 structural preservation via `lynx` followed by semantic refinement via an LLM—yields high-quality,
 194 coherent text suitable for large-scale mathematical corpora. Ablation studies (§3.2) show that this
 195 cleanup task is simple enough for smaller models to perform effectively. Qualitative comparisons
 196 with prior work and the full cleanup prompt are provided in Appendices A.3 and A.5, respectively.

197 2.2 QUALITY CLASSIFICATION

198 To support the later stages of training where data fidelity is especially important (Hu et al., 2024; Ab-
 199 din et al., 2024b), we further filtered our Nemotron-CC-Math to retain only its highest-scoring sub-
 200 set, Nemotron-CC-Math-4+. We employed the FineMath classifier (Allal et al., 2025) which assigns
 201 a 5-point scale score to each page, focusing on identifying content with mathematical reasoning and
 202 material suited to middle- and high-school levels. The data distribution for each quality score is
 203 provided in Appendix §A.7. After classification, we also performed deduplication and decon-
 204 tamination (see §2.3 and §2.4). We developed two variants of Nemotron-CC-Math: Nemotron-CC-
 205 Math-4+ (52.32B tokens, 45M documents) with scores 4-5 and Nemotron-CC-Math-3+ (133.26B
 206 tokens, 101M documents) with scores 3-5.

207 2.3 FUZZY DEDUPLICATION

208 Removing near-duplicate documents is essential for efficient and stable model training, and reduc-
 209 ing the risk of memorization (Lee et al., 2022; Tokpanov et al., 2024). We applied fuzzy deduplica-
 210 tion using the NeMo-Curator framework, which uses a MinHash-based Locality Sensitive Hashing
 211 (LSH) (Broder, 2000) to efficiently detect duplicates. The probability that two documents with Jac-
 212 card similarity S hash to the same bucket is $P = 1 - (1 - S^b)^r$, where b is the number of hash
 213 functions per band and r is the number of bands. With $r=20$ bands and $b=13$ hash functions per

Table 1: Comparison of Nemotron-CC-Math with math pretraining datasets. Nemotron-CC-Math-4+ is $5.5\times$ larger than the highest-quality open math dataset (FineMath-4+) with a permissive license, and substantially outperforms FineMath across math, code, and knowledge tasks (Table 2).

Dataset	Open Source	#Documents (M)	#Tokens (B)	Source
Minerva (Lewkowycz et al., 2022)	✗	-	38.50	arXiv, Web
MathMix (Lightman et al., 2023)	✗	-	1.50	Unknown
DeepSeekMath (Shao et al., 2024)	✗	-	120	CommonCrawl
ProofPile (Azerbayev et al., 2023)	✓	2.04	8.30	arXiv, Textbooks, Formal Math Libraries, StackExchange, ProofWiki, MATH
ProofPile-2 (Azerbayev et al., 2024)	✓	11.20	55	OpenWebMath, ArXiv, AlgebraicStack
AMPS (Hendrycks et al., 2021b)	✓	5.10	0.70	Khan Academy, Synthetic data
MathPile (Wang et al., 2024b)	✓	0.73	9.50	arXiv, Textbooks, ProofWiki, Wikipedia, StackExchange, CommonCrawl
OpenWebMath (Paster et al., 2024)	✓	6.30	14.70	CommonCrawl
InfIMM-WebMath-4+ (Han et al., 2024)	✓	6.30	8.50	CommonCrawl
FineMath-4+ (Allal et al., 2025)	✓	6.70	9.60	CommonCrawl
MegaMath-Pro (Zhou et al., 2025)	✓	15	15.10	CommonCrawl
Nemotron-CC-Math-4+ (Ours)	✓	45.10	52.32	CommonCrawl
InfIMM-WebMath-3+ (Han et al., 2024)	✓	13.90	20.50	CommonCrawl
FineMath-3+ (Allal et al., 2025)	✓	21.40	34	CommonCrawl
MegaMath-Web (Zhou et al., 2025)	✓	106.50	263.90	CommonCrawl
Nemotron-CC-Math-3+ (Ours)	✓	101.15	133.26	CommonCrawl

band, our setup targets a Jaccard similarity threshold of 0.8. Pairwise similarity is computed using 24-character n-grams, and LSH uses concurrent shuffling of five bands to identify duplicates.

2.4 DECONTAMINATION

The source documents used in Nemotron-CC-Math are from mostly pre-decontaminated datasets. However, we follow a more thorough decontamination procedure as outlined in Yang et al. (2023). We embed all the documents in Nemotron-CC-Math using the Qwen2.5B 32B model (Qwen et al., 2025) as well as all the prompts and answers from our evaluation benchmarks: MMLU (Hendrycks et al., 2021a), MMLU-Pro (Wang et al., 2024a), MATH (Hendrycks et al., 2021b), and GSM8K (Cobbe et al., 2021). We remove all documents with a cosine similarity above 0.9 to any benchmark prompt or answer, resulting in the removal of less than 0.002% of all documents.

3 EXPERIMENTS

Datasets We compare Nemotron-CC-Math to existing prior math pretraining datasets, including Megamath, OWM, and FineMath. Table 1 summarizes the dataset statistics.

Experimental Setup Math and code abilities generally arise only after extensive training; following Blakeney et al. (2024); Dubey et al. (2024); OLMo et al. (2024); Allal et al. (2025) to estimate the quality of different math pretraining datasets, we run annealing ablations on a mid training checkpoint of Nemotron-T 8B model (NVIDIA et al., 2025). The base model was pretrained on 9T tokens using a mixture of general-domain and math-focused corpora (see Appendix A.8 for a detailed breakdown). In each ablation, the target math dataset is upweighted to constitute 30% of the total data blend, while the weights of all other datasets are correspondingly downweighted to make up the remaining 70%. This controlled adjustment isolates the contribution of the math data while preserving overall blend composition (See Appendix A.4 for hyper-parameters). We consider two controlled ablations:

- **100B Token Ablation:** This setting targets compact, high-quality math datasets, typically below 30B tokens. For each run, the mathematical portion of the blend is replaced with a single candidate dataset—such as FineMath-4+, MegaMath-Pro, or OWM—enabling direct comparison with Nemotron-CC-Math-4+. The modified blends are trained for 100B tokens to evaluate performance under a consistent compute budget.
- **300B Token Ablation:** To fairly assess larger math datasets, including FineMath-3+ and MegaMath-Web, we apply the same replacement and proportional adjustment strategy but extend the total annealing budget to 300B tokens. This configuration also tests whether increased scale can offset dataset quality differences.

Benchmarks We evaluate model performance across a diverse suite of benchmarks spanning knowledge understanding, code, and mathematical reasoning tasks. Knowledge understanding is assessed using MMLU datasets, including MMLU-Pro (Wang et al., 2024a), MMLU, and MMLU-STEM (Hendrycks et al., 2021a) with results reported as exact match (EM) accuracy. Code generation quality is measured on four tasks-MBPP (Austin et al., 2021), and HumanEval (Chen et al., 2021) and their EvalPlus variants, HumanEval+ and MBPP+ (Liu et al., 2023). For code tasks, following Guo et al. (2025), to improve the stability, we report the avg@20 which reports the average accuracy from generating 20 samples for each prompt. To produce these samples, we apply nucleus sampling with a temperature of 0.6 and a top- p value of 0.95. Mathematical reasoning is evaluated on the GMS8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) benchmarks, with greedy decoding and using Math-Verify⁴ for symbolic matching. We run evaluations of all models ourselves using lm-evaluation-harness⁵.

3.1 PRETRAINING EXPERIMENTS RESULTS

Results Tables 2 compare Nemotron-T 8B models pretrained with different math datasets at 100B and 300B tokens, respectively. Across all benchmarks, models using the curated Nemotron-CC-Math consistently match or outperform competing datasets—including OWM, MegaMath, and FineMath—on knowledge, code, and math tasks.

At 100B tokens, Nemotron-CC-Math-4+ achieves the top results in every math task—e.g., 40.6 on MATH (+4.8 vs. FineMath-4+, +6.6 vs. MegaMath-Pro) and 76.27 on GMS8K (+0.3 vs. FineMath-4+, +4.85 vs. OWM). It also leads most code benchmarks (e.g., 34.82 on HumanEval+, +2.3 vs. OWM) and all knowledge tasks (e.g., 38.49 on MMLU-Pro, +2.1 vs. MegaMath-Pro).

At 300B tokens, Nemotron-CC-Math-3+ extends these gains-reaching 44.2 on MATH (+9.6 vs. FineMath-3+, +12.6 vs. MegaMath-Web) and 80.06 on GMS8K (+0.6 vs. FineMath-3+, +3.6 vs. OWM). Code scores also improve substantially, with 37.16 on HumanEval+ (+3.0 vs. FineMath-3+) and 43.51 on MBPP+ (+4.6 vs. MegaMath-Web, +14.32 vs. Finemath-3+). Knowledge remains best or near-best across MMLU variants, with a top score of 64.26 on MMLU-STEM.

Although we do not explicitly target the code domain, it is noteworthy that the curated Nemotron-CC-Math datasets substantially improve code performance. Upon analysis, we find that Nemotron-CC-Math-3+ and Nemotron-CC-Math-4+ contain approximately 4.3M and 1.44M samples with code snippets⁶. In contrast to prior datasets, which often fail to capture code content, our curation pipeline retains code snippets in full, preserving syntax and structure. We attribute the observed code improvements to this incidental yet high-quality code data. Overall, results show that high-quality curated math data in pretraining boosts performance in math reasoning, code, and general knowledge. Comparing 100B and 300B token results, gains scale with more pretraining. This highlights the value of high-quality math data for improving LLMs across specialized and general domains.

3.2 ABLATION ON MODEL CHOICE

To ablate the model choice for the task of boilerplate removal from rendered web pages, we sampled 7M documents and evaluated several instruction-tuned LLMs including DeepSeek-V3 (Liu et al., 2024), Qwen2.5-32B/Instruct, Qwen2.5-72B/Instruct (Team, 2024), and Phi-4 across knowledge, coding, and math benchmarks.

Table 3 presents the results. Surprisingly, despite its significantly smaller size (14B parameters), Phi-4 performs competitively across all domains, often matching or exceeding the results of much larger models such as DeepSeek-V3 (671B) and Qwen2.5-72B-Instruct (72B). In particular, Phi-4 achieves the best performance on math tasks (e.g., 79.98 EM on GMS8K and 40.6 EM on MATH) and leads or matches the performance in several code-related benchmarks.

Given the marginal differences in performance and the substantial gap in computational cost, we selected Phi-4 as the default model for all experiments in this paper. Our findings indicate that the task of webpage boilerplate removal does not require excessively large models, and smaller instruction-tuned models can yield efficient and effective results.

⁴<https://github.com/huggingface/math-verify>

⁵<https://github.com/EleutherAI/lm-evaluation-harness>.

⁶We filter out examples enclosed within triple backticks indicating a code block (e.g., ““python ... ““).

Models Trained on 100B Tokens					
	Benchmark (Metric)	OWM	MegaMath (Pro)	FineMath (4+)	Nemotron-CC-Math (4+)
Knowledge	MMLU-Pro (EM)	35.49	36.41	36.74	38.49
	MMLU (EM)	65.62	66.81	66.73	67.55
	MMLU-Stem (EM)	58.83	60.86	61.62	62.67
Code	HumanEval+ (avg@20)	32.53	31.01	32.16	34.82
	MBPP+ (avg@20)	43.76	46.03	28.88	45.11
	MBPP (avg@20)	53.11	52.51	53.42	53.48
	HumanEval (avg@20)	37.07	35.91	37.77	38.93
Math	MATH (EM)	29.20	34.00	35.80	40.60
	GMS8K (EM)	71.42	73.46	75.97	76.27
Models Trained on 300B Tokens					
	Benchmark (Metric)	OWM	MegaMath (Web)	FineMath (3+)	Nemotron-CC-Math (3+)
Knowledge	MMLU-Pro (EM)	35.00	36.33	39.57	39.32
	MMLU (EM)	65.20	65.44	67.92	68.20
	MMLU-Stem (EM)	59.20	59.88	62.29	64.26
Code	HumanEval+ (avg@20)	33.54	32.29	34.18	37.16
	MBPP+ (avg@20)	37.59	38.89	29.19	43.51
	MBPP (avg@20)	52.22	53.05	57.57	56.15
	HumanEval (avg@20)	38.32	36.34	37.80	40.30
Math	MATH (EM)	34.20	31.60	34.60	44.20
	GMS8K (EM)	76.42	78.24	79.45	80.06

Table 2: Evaluation results for models trained with different math datasets using either 100B or 300B tokens. NEMOTRON-CC-MATH variants consistently outperform or obtain comparable results to OpenWebMath, MegaMath, and FineMath baselines across knowledge, code, and math tasks. Math performance improves with a longer token horizon, showing Nemotron-CC-Math continues to scale effectively with increased training. Code results use average accuracy over 20 generations; all other tasks use exact match (EM). Bold indicates the best result in each row.

3.3 LLM-AIDED QUALITY ASSESSMENT OF SCIENTIFIC CONTENT FIDELITY

We first performed an overlap analysis across OWM, MegaMath-Pro, FineMath, and our Nemotron-CC-Math datasets, identifying 97,788 shared samples. As this joint subset is far too small for meaningful side-by-side pretraining experiments, we use it as a rigorous, shared basis for comparative quality assessment. To complement standard benchmark results, we conduct an LLM-aided quality assessment to directly measure how effectively Common-Crawl-derived mathematical datasets

	Benchmark (Metric)	DeepSeek-V3	Qwen2.5-32B	Qwen2.5-72B	Phi-4
Knowledge	MMLU-Pro (EM)	38.82	39.65	39.65	38.49
	MMLU (EM)	67.68	67.01	67.73	67.54
	MMLU-Stem (EM)	62.96	61.88	62.73	63.24
Code	HumanEval+ (avg@20)	28.54	28.35	29.63	30.40
	MBPP+ (avg@20)	45.99	38.58	41.34	41.88
	MBPP (avg@20)	53.91	53.83	54.09	55.39
	HumanEval (avg@20)	32.10	31.92	35.21	34.09
Math	MATH (EM)	36.60	38.80	38.60	40.60
	GMS8K (EM)	75.51	74.00	73.92	79.98

Table 3: Model choice ablation. We compare DeepSeek-V3 (671B), Qwen2.5-32B/72B, and Phi-4 (14B) across benchmarks. Despite its smaller size, Phi-4 performs competitively—often leading in math—demonstrating smaller models can efficiently clean webpages without losing performance.

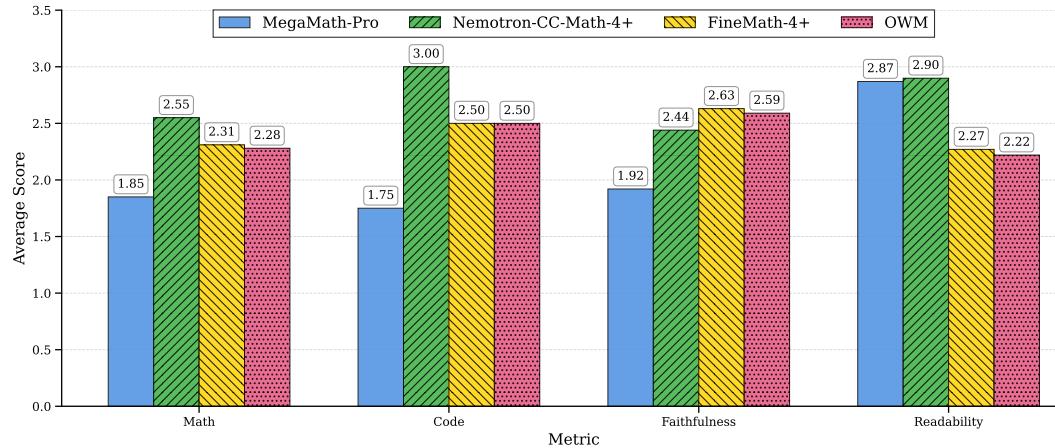


Figure 3: LLM-Aided quality assessment results comparing the cleaned webpage quality across four datasets: Ours (Nemotron-CC-Math), OWM, FineMath, and MegaMath. The LLM judge scored 100 randomly sampled documents from each dataset on four dimensions. Our method achieves the strongest performance in preserving mathematical expressions and code structure, with competitive faithfulness and the highest readability. These results highlight the effectiveness of the Lynx → LLM pipeline in retaining scientific structure while improving textual clarity.

preserve scientific content during extraction and cleaning. This assessment focuses on four critical dimensions essential for high-quality mathematical pretraining corpora:

- **Math Preservation (0–3 or N/A):** Correctness and completeness of mathematical expressions unified to \LaTeX .
- **Code Preservation (0–3 or N/A):** Structural and semantic integrity, syntax, and functional behavior of preserved code blocks.
- **Faithfulness (0–3):** Preservation of core scientific content integrity without omission or meaning alteration.
- **Readability (0–3):** Overall clarity, organization, coherence, and textual flow of the output document.

Methodology We randomly sampled 100 webpages from the shared subset. We employed OpenAI gpt-5.1 as an automated judge to assess the conversion quality. The judge was provided with the original raw text (extracted via Lynx) and the converted document from each dataset, guided by a detailed scoring rubric and evaluation instructions (see Appendix A.6). We report the mean score for each dataset across the 100 samples and four dimensions.

Results Figure 3 summarizes the assessment results. Nemotron-CC-Math demonstrates superior fidelity in preserving the underlying scientific structure, achieving the highest scores in both Math and Code Preservation:

- **Math Preservation (2.55):** Our score significantly surpasses all baselines (OWM: 2.28, FineMath: 2.31, MegaMath: 1.85). This validates that our combined Lynx-based extraction and LLM-driven \LaTeX normalization more reliably preserves mathematical content than prior heuristic HTML-to-text pipelines.
- **Code Preservation (3.00):** Nemotron-CC-Math achieved the maximum score, substantially outperforming all competitors (OWM/FineMath: 2.50, MegaMath: 1.75). This confirms the effectiveness of the Lynx-based approach in retaining code formatting and indentation, often lost by DOM-based extractors.
- **Faithfulness (2.44) and Readability (2.90):** Although our faithfulness score is marginally lower than FineMath (2.63) and OWM (2.59)—a consequence of the intentional LLM-based cleanup that may involve compressing or rewriting contextual details—Nemotron-CC-Math achieved the highest readability. This demonstrates that our pipeline successfully trades a minimal loss in literal text preservation for superior textual coherence and clarity.

432	Domain	#Documents (M)	Document %	433	Domain	#Characters (B)	Characters %
434	mathhelpforum.com	8.54	8.44	435	mathhelpforum.com	17.11	3.67
436	jiskha.com	5.33	5.26	437	jiskha.com	12.52	2.69
438	physicsforums.com	2.82	2.78	439	mathforum.org	8.96	1.92
440	math.stackexchange.com	2.38	2.35	441	physicsforums.com	8.19	1.76
442	mathforum.org	2.38	2.35	443	math.stackexchange.com	6.96	1.49
444	openstudy.com	1.88	1.86	445	mathoverflow.net	6.78	1.45
446	forums.wolfram.com	1.51	1.49	447	nrich.maths.org	6.24	1.34
448	mathoverflow.net	1.33	1.32	449	scribd.com	4.47	0.96
450	nrich.maths.org	1.13	1.12	451	educator.com	3.58	0.77
452	mathisfunforum.com	0.76	0.75	453	forums.wolfram.com	3.35	0.72
454	coursehero.com	0.76	0.75	455	docplayer.net	3.20	0.69
456	brilliant.org	0.68	0.67	457	en.wikipedia.org	3.07	0.66
458	gmatclub.com	0.65	0.65	459	openstudy.com	3.10	0.66
460	chegg.com	0.58	0.57	461	gmatclub.com	2.73	0.59
462	gradesaver.com	0.54	0.53	463	mathisfunforum.com	2.72	0.58
464	socratic.org	0.49	0.49	465	coursehero.com	2.31	0.50
466	purplemath.com	0.45	0.44	467	slideplayer.com	2.06	0.44
468	physics.stackexchange.com	0.44	0.43	469	hindawi.com	1.98	0.42
470	betterlesson.com	0.41	0.41	471	softmath.com	1.95	0.42
472	brainmass.com	0.40	0.40	473	archive.org	1.96	0.42

Table 4: Comparison of Most Common Domains by Document (left) and Character Count (right).

In conclusion, these results confirm that Nemotron-CC-Math offers the optimal balance between structural fidelity and textual clarity. By moving beyond brittle heuristic pipelines, our `Lynx` → LLM approach captures essential scientific content and structure while yielding coherent, high-quality text, establishing Nemotron-CC-Math as a substantially higher-quality scientific corpus compared to existing Common-Crawl-based math pretrainig datasets.

3.4 DATASET ANALYSIS

Data Composition We measured domain distribution by document and character count. Table 4 shows the top twenty domains by each metric. Similar to prior work (Paster et al., 2024), the most common sources are discussion forums, Q&A sites, and educational resources. Overall, the dataset spans 980,922 unique domains, with the top 100 domains accounting for 36.46% of characters and 43.22% of documents.

Topic Distribution To characterize the dataset, we randomly sampled 150,000 documents and classified them into mathematics, physics, statistics, chemistry, economics, computer science, or other using the Qwen3-30B-A3B-Instruct-2507 model (see Appendix A.2 for the prompt). Figure 1 shows the results. Most documents pertain to mathematics, with the remainder distributed across the other scientific domains; approximately 4.12% fall outside these categories.

4 RELATED WORKS

High-quality math pretraining datasets are essential for improving LLM reasoning. OWM compiles 14.7B tokens from Common Crawl but relies on brittle heuristics and Resliparse for HTML rendering, often stripping or corrupting formulas and code. FineMath inherits these issues, building its 54B-token corpus using the OWM pipeline. Similarly, MegaMath faces similar challenges.

MathPile (Wang et al., 2024b) aggregates 9.5B tokens from sources including arXiv (85%), textbooks, and forums but much of the content remains in raw `LATEX` form, limiting usability for LLM pretraining. InfiMM-Web-Math (Han et al., 2024) is 40B tokens multimodal dataset pairing images with math text. Proof-Pile (Azerbayev et al., 2023) is a 8.3B-token dataset collected from various sources such as arXiv, formal math libraries, Wikipedia and Stack Exchange. Proof-Pile-2 (Azerbayev et al., 2024) is a 55B-token dataset combining arXiv, OWM, and Algebraic-Stack mathematical code. Additionally, auxiliary datasets include AMPS (Hendrycks et al., 2021b) with Khan Academy problems and Mathematica-generated content, and NaturalProofs (Welleck et al., 2021), covering theorems and proofs from ProofWiki and the Stacks project.

Proprietary datasets like WebMath (OpenAI) (Polu & Sutskever, 2020), MathMix (Lightman et al., 2023), DeepSeekMath (Shao et al., 2024), and Minerva’s Math Web Pages (Lewkowycz et al., 2022) advance math reasoning but lack public access, limiting transparency. We release Nemotron-CC-Math openly to foster community progress.

486 **5 CONCLUSION**

488 We present a scalable, domain-agnostic pipeline for extracting high-quality technical content from
 489 Common Crawl, focusing here on the mathematical domain. By integrating robust HTML-to-text
 490 conversion with LLM-based domain-aware cleaning, our approach addresses longstanding chal-
 491 lenges in web-scale extraction of structured technical data.

492 Applied to the math domain, our pipeline produced Nemotron-CC-Math, whose 4+ subset is $5.5 \times$
 493 larger than the previous highest-quality math set, FineMath-4+. Pretraining on Nemotron-CC-Math
 494 improves math reasoning (+12.6 MATH), code generation (+14.3 MBPP+), and general knowledge
 495 (+5.1 MMLU-Stem), outperforming prior math datasets.

496 Importantly, the modular, domain-agnostic design enables application to other technical fields. As
 497 LLMs advance, our pipeline offers a crucial tool for generating targeted, high-quality pretraining
 498 data to drive model capabilities.

500 **REFERENCES**

502 Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar,
 503 Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 techni-
 504 cal report. *arXiv preprint arXiv:2412.08905*, 2024a.

505 Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany
 506 Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, Alon Benhaim, Misha
 507 Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu
 508 Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ro-
 509 nen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng
 510 Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karam-
 511 patziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi
 512 Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi,
 513 Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant,
 514 Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi,
 515 Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia
 516 Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt,
 517 Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang,
 518 Cyril Zhang, Jianwen Zhang, Li Lyra Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren
 519 Zhou. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*,
 520 abs/2404.14219, 2024b. URL <https://doi.org/10.48550/arXiv.2404.14219>.

521 Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo,
 522 Lewis Tunstall, Andrés Marafioti, Hynek Kydliček, Agustín Piqueres Lajarín, Vaibhav Srivastav,
 523 Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo
 524 Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and
 525 Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model,
 526 2025. URL <https://arxiv.org/abs/2502.02737>.

527 Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh
 528 Fadaee, Ahmet Üstün, and Sara Hooker. To code or not to code? exploring impact of code in
 529 pre-training. In *ICLR*, 2024.

531 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan,
 532 Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with
 533 Large Language Models, 2021. URL <https://arxiv.org/abs/2108.07732>.

534 Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and
 535 Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathemat-
 536 ics. *arXiv preprint arXiv:2302.12433*, 2023.

538 Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer,
 539 Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model
 for mathematics. In *ICLR*, 2024.

540 Adrien Barbaresi. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discov-
 541 ery and Extraction. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the*
 542 *Association for Computational Linguistics and the 11th International Joint Conference on Natu-*
 543 *ral Language Processing: System Demonstrations*, pp. 122–131. Association for Computational
 544 Linguistics, 2021. URL <https://aclanthology.org/2021.acl-demo.15>.

545 Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic ChatNoir: Search En-
 546 gine for the ClueWeb and the Common Crawl. In Leif Azzopardi, Allan Hanbury, Gabriella Pasi,
 547 and Benjamin Piwowarski (eds.), *Advances in Information Retrieval. 40th European Conference*
 548 *on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York,
 549 March 2018. Springer.

550 Cody Blakeney, Mansheej Paul, Brett W Larsen, Sean Owen, and Jonathan Frankle. Does your data
 551 spark joy? performance gains from domain upsampling at the end of training. *arXiv preprint*
 552 *arXiv:2406.03476*, 2024.

553 Andrei Z Broder. Identifying and filtering near-duplicate documents. In *Annual symposium on*
 554 *combinatorial pattern matching*, pp. 1–10. Springer, 2000.

555 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared
 556 Kaplan, et al. Evaluating Large Language Models Trained on Code, 2021. URL <https://arxiv.org/abs/2107.03374>.

557 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 558 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
 559 Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL <https://arxiv.org/abs/2110.14168>.

560 DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Cheng-
 561 gang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang,
 562 Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting
 563 Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui
 564 Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi
 565 Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li,
 566 Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang,
 567 Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun
 568 Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan
 569 Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J.
 570 Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang,
 571 Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng
 572 Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shut-
 573 ing Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanja Zhao,
 574 Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue
 575 Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xi-
 576 aokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin
 577 Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang,
 578 Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang
 579 Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui
 580 Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying
 581 Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu,
 582 Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan
 583 Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F.
 584 Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda
 585 Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao,
 586 Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li,
 587 Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL
 588 <https://arxiv.org/abs/2412.19437>.

589 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha
 590 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.
 591 *arXiv preprint arXiv:2407.21783*, 2024.

594 István Endrédy and Attila Novák. More effective boilerplate removal-the goldminer algorithm.
595 *Polibits*, 48:79–83, 12 2013. doi: 10.17562/PB-48-10.

596

597 Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason
598 Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text
599 for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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

603

604 Xiaotian Han, Yiren Jian, Xuefeng Hu, Haogeng Liu, Yiqi Wang, Qihang Fan, Yuang Ai, Huaibo
605 Huang, Ran He, Zhenheng Yang, and Quanzeng You. Infimm-webmath-40b: Advancing multi-
606 modal pre-training for enhanced mathematical reasoning, 2024. URL <https://arxiv.org/abs/2409.12568>.

607

608 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Ja-
609 cob Steinhardt. Measuring massive multitask language understanding, 2021a. URL <https://arxiv.org/abs/2009.03300>.

610

611 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
612 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,
613 2021b.

614

615 Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang,
616 Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models
617 with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

618

619 Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun
620 Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men,
621 Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xu-
622 ancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL
623 <https://arxiv.org/abs/2409.12186>.

624

625 Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferran-
626 dis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of
627 permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

628

629 Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Moham-
630 mad Shoeybi, and Bryan Catanzaro. Reducing Activation Recomputation in Large Transformer
631 Models, 2022. URL <https://arxiv.org/abs/2205.05198>.

632

633 Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-
634 Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *ACL*,
635 pp. 8424–8445, 2022.

636

637 Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ra-
638 masesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative
639 reasoning problems with language models. *NeurIPS*, 35:3843–3857, 2022.

640

641 Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou,
642 Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with
643 you! *arXiv preprint arXiv:2305.06161*, 2023.

644

645 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee,
646 Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *CoRR*,
647 abs/2305.20050, 2023. doi: 10.48550/arXiv.2305.20050. URL <https://doi.org/10.48550/arXiv.2305.20050>.

648

649 Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao,
650 Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint*
651 *arXiv:2412.19437*, 2024.

648 Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by
 649 ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation.
 650 *arXiv preprint arXiv:2305.01210*, 2023. doi: <https://doi.org/10.48550/arXiv.2305.01210>. URL
 651 <https://arxiv.org/abs/2305.01210>.

652 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint*
 653 *arXiv:1711.05101*, 2017.

654 Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane
 655 Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The
 656 next generation. *arXiv preprint arXiv:2402.19173*, 2024.

658 Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Noua-
 659 mane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language
 660 models, 2023.

661 NVIDIA, :, Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad
 662 Bercovich, Aleksander Ficek, Alexis BJORLIN, Ali Taghibakhshi, Amala Sanjay Deshmukh,
 663 Ameya Sunil Mahabaleshwarkar, Andrew Tao, Anna Shors, Ashwath Aithal, Ashwin Poojary,
 664 Ayush Dattagupta, Balaram Buddharaju, Bobby Chen, Boris Ginsburg, Boxin Wang, Brandon
 665 Norick, Brian Butterfield, Bryan Catanzaro, Carlo del Mundo, Chengyu Dong, Christine Har-
 666 vey, Christopher Parisien, Dan Su, Daniel Korzekwa, Danny Yin, Daria Gitman, David Mosal-
 667 lanezhad, Deepak Narayanan, Denys Fridman, Dima Rekesh, Ding Ma, Dmytro Pykhtar, Dong
 668 Ahn, Duncan Riach, Dusan Stosic, Eileen Long, Elad Segal, Ellie Evans, Eric Chung, Erick
 669 Galinkin, Evelina Bakhturina, Ewa Dobrowolska, Fei Jia, Fuxiao Liu, Gargi Prasad, Gerald Shen,
 670 Guilin Liu, Guo Chen, Haifeng Qian, Helen Ngo, Hongbin Liu, Hui Li, Igor Gitman, Ilia Kar-
 671 manov, Ivan Moshkov, Izik Golan, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jarno Seppa-
 672 nen, Jason Lu, Jason Sewall, Jiaqi Zeng, Jiaxuan You, Jimmy Zhang, Jing Zhang, Jining Huang,
 673 Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jon Barker, Jonathan Cohen, Joseph
 674 Jennings, Jupinder Parmar, Karan Sapra, Kari Briski, Kateryna Chumachenko, Katherine Luna,
 675 Keshav Santhanam, Kezhi Kong, Kirthi Sivamani, Krzysztof Pawelec, Kumar Anik, Kunlun Li,
 676 Lawrence McAfee, Leon Derczynski, Lindsey Pavao, Luis Vega, Lukas Voegtle, Maciej Bala,
 677 Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Markus Kliegl,
 678 Marta Stepniewska-Dziubinska, Matthieu Le, Matvei Novikov, Mehrzad Samadi, Michael Ander-
 679 sch, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mike Ranzinger, Mikolaj Blaz, Misha
 680 Smelyanskiy, Mohamed Fawzy, Mohammad Shoeybi, Mostofa Patwary, Nayeon Lee, Nima
 681 Tajbakhsh, Ning Xu, Oleg Rybakov, Oleksii Kuchaiev, Olivier Delalleau, Osvald Nitski, Parth
 682 Chadha, Pasha Shamis, Paulius Micikevicius, Pavlo Molchanov, Peter Dykas, Philipp Fischer,
 683 Pierre-Yves Aquilanti, Piotr Bialecki, Prasoon Varshney, Pritam Gundecha, Przemek Tredak,
 684 Rabeeh Karimi, Rahul Kandu, Ran El-Yaniv, Raviraj Joshi, Roger Waleffe, Ruoxi Zhang, Sabrina
 685 Kavanaugh, Sahil Jain, Samuel Kriman, Sangkug Lym, Sanjeev Satheesh, Saurav Muralidha-
 686 ran, Sean Narendhiran, Selvaraj Anandaraj, Seonmyeong Bak, Sergey Kashirsky, Seungju Han,
 687 Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Sharon Clay, Shelby Thomas,
 688 Shrimai Prabhulmoye, Shubham Pachori, Shubham Toshniwal, Shyamala Prayaga, Siddhartha
 689 Jain, Sirshak Das, Slawek Kierat, Somshubra Majumdar, Song Han, Soumye Singhal, Sriharsha
 690 Niverty, Stefania Alborghetti, Suseela Panguluri, Swetha Bhendigeri, Syeda Nahida Akter, Szy-
 691 mon Migacz, Tal Shiri, Terry Kong, Timo Roman, Tomer Ronen, Trisha Saar, Tugrul Konuk,
 692 Tuomas Rintamaki, Tyler Poon, Ushnish De, Vahid Noroozi, Varun Singh, Vijay Korthikanti,
 693 Vitaly Kurin, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenliang Dai, Wonmin Byeon, Xiaowei
 694 Ren, Yao Xu, Yejin Choi, Yian Zhang, Ying Lin, Yoshi Suhara, Zhiding Yu, Zhiqi Li, Zhiyu Li,
 695 Zhongbo Zhu, Zhuolin Yang, and Zijia Chen. Nemotron-h: A family of accurate and efficient
 696 hybrid mamba-transformer models, 2025. URL <https://arxiv.org/abs/2504.03624>.

697 Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita
 698 Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint*
 699 *arXiv:2501.00656*, 2024.

700 OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.

701 Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open
 702 dataset of high-quality mathematical web text. *ICLR*, 2024.

702 Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli,
 703 Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refined-
 704 web dataset for falcon llm: Outperforming curated corpora with web data only. *NeurIPS*, 36:
 705 79155–79172, 2023.

706 Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving.
 707 *arXiv preprint arXiv:2009.03393*, 2020.

708 Qwen, ;, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 709 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 710 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 711 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 712 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 713 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 714 URL <https://arxiv.org/abs/2412.15115>.

715 Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory Optimiza-
 716 tions Toward Training Trillion Parameter Models, 2020. URL <https://arxiv.org/abs/1910.02054>.

717 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 718 Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical
 719 reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

720 Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan
 721 Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Par-
 722 allelism, 2020. URL <https://arxiv.org/abs/1909.08053>.

723 Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary,
 724 Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a
 725 refined long-horizon pretraining dataset. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova,
 726 and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association
 727 for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, Vienna, Austria, July
 728 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. URL <https://aclanthology.org/2025.acl-long.123/>.

729 Liping Tang, Nikhil Ranjan, Omkar Pangarkar, Xuezhi Liang, Zhen Wang, Li An, Bhaskar Rao,
 730 Linghao Jin, Huijuan Wang, Zhoujun Cheng, Suqi Sun, Cun Mu, Victor Miller, Xuezhe Ma, Yue
 731 Peng, Zhengzhong Liu, and Eric P. Xing. Txt360: A top-quality llm pre-training dataset requires
 732 the perfect blend, 2024. URL <https://huggingface.co/spaces/LLM360/Txt360>.

733 Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

734 Yury Tokpanov, Paolo Glorioso, Quentin Anthony, and Beren Millidge. Zyda-2: a 5 trillion token
 735 high-quality dataset. *arXiv preprint arXiv:2411.06068*, 2024.

736 Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming
 737 Ren, Aaran Arulraj, Xuan He, Ziyuan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi
 738 Fan, Xiang Yue, and Wenhui Chen. Mmlu-pro: A more robust and challenging multi-task language
 739 understanding benchmark, 2024a. URL <https://arxiv.org/abs/2406.01574>.

740 Zengzhi Wang, Xuefeng Li, Rui Xia, and Pengfei Liu. Mathpile: A billion-token-scale pretraining
 741 corpus for math. *NeurIPS*, 37:25426–25468, 2024b.

742 Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun
 743 Cho. Naturalproofs: Mathematical theorem proving in natural language. *arXiv preprint
 744 arXiv:2104.01112*, 2021.

745 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jian-
 746 hong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-math technical report: Toward mathematical
 747 expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

748 Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking
 749 benchmark and contamination for language models with rephrased samples, 2023.

756 Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong
757 Liu, and Eric P Xing. Megamath: Pushing the limits of open math corpora. *arXiv preprint*
758 *arXiv:2504.02807*, 2025.
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

810
811

A APPENDIX

812
813

A.1 USE OF LARGE LANGUAGE MODELS (LLMs)

814
815 We used a large language model (LLM) solely to aid in polishing the writing of this paper, includ-
816 ing improving grammar, clarity, and flow. The LLM was not involved in the research ideation,
817 methodology, experiments, analysis, or interpretation of results.818
819

A.2 PROMPT USED FOR TOPIC CLASSIFICATION

820
821 We employ the following prompt to classify documents into a predefined set of categories. Dur-
822 ing classification, the large language model (LLM) occasionally produces category labels that fall
823 outside the predefined taxonomy: mathematics, computer science, physics, statistics, economics,
824 chemistry, and other. To maintain consistency and reduce label fragmentation, any out-of-taxonomy
825 label is reassigned to the category other, ensuring a coherent and structured category distribution.

826
827 You are a topic classification assistant.
 828 Given the following document text, identify its main topic from this list only:
 829 – Mathematics
 830 – Computer Science
 831 – Physics
 832 – Statistics
 833 – Chemistry
 834 – Economics
 835 – Other

836 Choose the single most relevant category from the list.
 837 Document:
 838 {text}

839 Your output should be only 1 word. Finish your response right after category and do not add any
 840 explanation.

841
842

A.3 QUALITATIVE EXAMPLES

843
844 This section presents qualitative comparisons among OpenWebMath (OWM), MegaMath-Pro,
845 FineMath-4+, and Nemotron-CC-Math-4+, highlighting differences in content quality.846
847

A.3.1 DEGENERATE CASES IN MEGAMATH-PRO DATASET

848
849 We identified a subset of degenerate generations within the MegaMath-Pro dataset. Representative
850 examples are presented below to illustrate this phenomenon. Notably, these samples achieve unex-
851 pectedly high scores on both mathematical and language scores, raising concerns about the dataset’s
852 overall reliability for pretraining LLMs. For each example, we provide the associated metadata. The
853 excerpts shown correspond to the initial portion of each generation; in every case, the text extends
854 over several additional pages, repeating the final sentence displayed.855
856

A.3.2 SIDE BY SIDE COMPARISON BETWEEN OUR DATASET AND PRIOR WORK

857
858 We observe that our pipeline not only keep the math equations but also keep the codes and their
859 formatting. We observe that previous pipeline in most cases are not keeping codes or lose their
860 formatting. This is specifically important for languages like python. To highlight this difference, we
861 provide two sets of examples demonstrating both code and mathematical equations. Notably, inline
862 equations are often removed in prior work, such as MegaMath.

864
865

A degenerate sample from MegaMath-Pro

866
867
868
869
870

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

871
872
873
874

The Integral Calculator is a free online tool for calculating the value of a definite integral. The Integral Calculator, part of the graphing calculator, helps with one variable calculus. The Integral Calculator supports definite and indefinite integrals (antiderivatives) as well as integrating functions with many variables.

875
876
877
878

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

879
880
881
882

The Integral Calculator is a free online tool for calculating the value of a definite integral. The Integral Calculator, part of the graphing calculator, helps with one variable calculus. The Integral Calculator supports definite and indefinite integrals (antiderivatives) as well as integrating functions with many variables.

883
884
885
886

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

887
888
889
890

The Integral Calculator is a free online tool for calculating the value of a definite integral. The Integral Calculator, part of the graphing calculator, helps with one variable calculus. The Integral Calculator supports definite and indefinite integrals (antiderivatives) as well as integrating functions with many variables.

891
892
893
894
895

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

896
897
898
899

The Integral Calculator is a free online tool for calculating the value of a definite integral. The Integral Calculator, part of the graphing calculator, helps with one variable calculus. The Integral Calculator supports definite and indefinite integrals (antiderivatives) as well as integrating functions with many variables.

900
901
902
903

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

904
905
906
907

The Integral Calculator is a free online tool for calculating the value of a definite integral. The Integral Calculator, part of the graphing calculator, helps with one variable calculus. The Integral Calculator supports definite and indefinite integrals (antiderivatives) as well as integrating functions with many variables.

908
909
910
911

The Integral Calculator is able to calculate integrals online of the composition of common functions, using integral properties, the different mechanisms of integration and calculation online. The Integral Calculator is a simple online calculator that computes the definite and indefinite integrals. The Integral Calculator will show you a graphical version of your input while you type.

912
913

The Integral Calculator is a free online tool for calculating the value of a definite integral.

914

Meta:915
916
917

URL: <http://031c82c.netsolhost.com/yvcmr4/article.php?c08ee4=complex-integration-calculator>

Math Score: 0.9996713399887085

Lang Score: 0.7893766164779663

WARC Filename: CC-MAIN-2022-21/segments/1652662531762.30/warc/CC-MAIN-20220520061824-00605.warc.gz

918
919

A degenerate sample from MegaMath-Pro

920 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 921 The angle will be calculated and displayed. The angle will be calculated and displayed.
 922 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 923 The angle will be calculated and displayed. The angle will be calculated and displayed.
 924 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 925 The angle will be calculated and displayed. The angle will be calculated and displayed.
 926 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 927 The angle will be calculated and displayed. The angle will be calculated and displayed.
 928 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 929 The angle will be calculated and displayed. The angle will be calculated and displayed.
 930 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 931 The angle will be calculated and displayed. The angle will be calculated and displayed.
 932 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 933 The angle will be calculated and displayed. The angle will be calculated and displayed.
 934 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 935 The angle will be calculated and displayed. The angle will be calculated and displayed.
 936 The angle will be calculated and displayed. The angle will be calculated and displayed.
 937 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 938 The angle will be calculated and displayed. The angle will be calculated and displayed.
 939 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 940 The angle will be calculated and displayed. The angle will be calculated and displayed.
 941 The angle will be calculated and displayed. The angle will be calculated and displayed.
 942 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 943 The angle will be calculated and displayed. The angle will be calculated and displayed.
 944 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 945 The angle will be calculated and displayed. The angle will be calculated and displayed.
 946 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 947 The angle will be calculated and displayed. The angle will be calculated and displayed.
 948 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 949 The angle will be calculated and displayed. The angle will be calculated and displayed.
 950 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 951 The angle will be calculated and displayed. The angle will be calculated and displayed.
 952 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 953 The angle will be calculated and displayed. The angle will be calculated and displayed.
 954 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 955 The angle will be calculated and displayed. The angle will be calculated and displayed.
 956 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 957 The angle will be calculated and displayed. The angle will be calculated and displayed.
 958 The angle will be calculated and displayed. The angle will be calculated and displayed.
 959 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 960 The angle will be calculated and displayed. The angle will be calculated and displayed.
 961 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 962 The angle will be calculated and displayed. The angle will be calculated and displayed.
 963 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 964 The angle will be calculated and displayed. The angle will be calculated and displayed.
 965 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 966 The angle will be calculated and displayed. The angle will be calculated and displayed.
 967 The angle will be calculated and displayed. Use the law of cosines to find one of the angles.
 968 The angle will be calculated and displayed. The angle will be calculated and displayed.
 969

Meta:**URL:** <http://102theking.com/regina-ward-sb0lp/how-to-find-an-angle-without-an-angle-finder-ebb037>**Math Score:** 0.9885214567184448**Lang Score:** 0.8642399311065674**WARC Filename:** CC-MAIN-2021-31/segments/1627046154796.71/warc/CC-MAIN-20210804045226-00255.warc.gz

972
973

A degenerate sample from MegaMath-Pro

974
975
976

The equation of the axis of symmetry in a vertical parabola is equal to the x-coordinate of the vertex. The axis of symmetry always passes through the vertex of the parabola. The x-coordinate of the vertex is equal to the formula.

977
978
979
980
981
982

To learn about the axis of symmetry, watch this tutorial! The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number. A parabola is the graph of a quadratic function. Each parabola has a line of symmetry. Also known as the axis of symmetry, this line divides the parabola into mirror images. The line of symmetry is always a vertical line of the form $x = n$, where n is a real number.

983
984
985

When graphing, we want to include certain special points in the graph. The y-intercept is the point where the graph intersects the y-axis. The x-intercepts are the points where the graph intersects the x-axis. The vertex is the point that defines the minimum or maximum of the graph.

986

The axis of symmetry for an equation with x^2 is the vertical line that passes through the vertex. The axis of symmetry is the line $x = h$, where (h, k) is the vertex of the parabola.

988
989

The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number.

990
991
992

The equation of the axis of symmetry in a vertical parabola is equal to the x-coordinate of the vertex. The axis of symmetry always passes through the vertex of the parabola. The x-coordinate of the vertex is equal to the formula.

993
994

To learn about the axis of symmetry, watch this tutorial! The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number.

995
996
997

A parabola is the graph of a quadratic function. Each parabola has a line of symmetry. Also known as the axis of symmetry, this line divides the parabola into mirror images. The line of symmetry is always a vertical line of the form $x = n$, where n is a real number.

998
999
1000
1001

When graphing, we want to include certain special points in the graph. The y-intercept is the point where the graph intersects the y-axis. The x-intercepts are the points where the graph intersects the x-axis. The vertex is the point that defines the minimum or maximum of the graph.

1002
1003
1004

The axis of symmetry for an equation with x^2 is the vertical line that passes through the vertex. The axis of symmetry is the line $x = h$, where (h, k) is the vertex of the parabola.

1005
1006
1007
1008

The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number.

1009
1010
1011

The equation of the axis of symmetry in a vertical parabola is equal to the x-coordinate of the vertex. The axis of symmetry always passes through the vertex of the parabola. The x-coordinate of the vertex is equal to the formula.

1012
1013
1014

To learn about the axis of symmetry, watch this tutorial! The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number.

1015
1016
1017

A parabola is the graph of a quadratic function. Each parabola has a line of symmetry. Also known as the axis of symmetry, this line divides the parabola into mirror images. The line of symmetry is always a vertical line of the form $x = n$, where n is a real number.

1018
1019
1020

When graphing, we want to include certain special points in the graph. The y-intercept is the point where the graph intersects the y-axis. The x-intercepts are the points where the graph intersects the x-axis. The vertex is the point that defines the minimum or maximum of the graph.

1021

The axis of symmetry for an equation with x^2 is the vertical line that passes through the vertex. The axis of symmetry is the line $x = h$, where (h, k) is the vertex of the parabola.

1022
1023
1024

The axis of symmetry is the line that divides the graph into two perfect halves. The axis of symmetry is always a vertical line of the form $x = n$, where n is a real number. The equation of the axis of symmetry in a vertical parabola is equal to the x-coordinate of the vertex.

Meta:

URL: <http://1798091312.srv040122.webreus.net/q8oh94/c5d31c-vertical-symmetry-graph>

Math Score: 0.9988757371902466

Lang Score: 0.9151933789253235

WARC Filename: CC-MAIN-2021-25/segments/1623488273983.63/warc/CC-MAIN-20210621120456-00277.warc.gz

Lynx output

1026 #10000 Terabyte
 1027
 1028 10000 Terabyte
 1029
 1030 about opensource disclaimer
 1031
 1032 (BUTTON)
 1033 about opensource disclaimer
 1034
 1035

1036 Detailed explanation of a smart solution to an algo problem beating 99.9% submission
 1037

1038 Written on January 7th, 2018 by @10000TB

1039 [attachments_article_algorithm_col_slide_lamparas-colgantes-algorithm-slide-03.jpg]

1040 This post is about a coding problem and why the solution I pasted down below is smart.
 1041 Problem:

1042 Given two sparse matrices A and B, return the result of AB.

1043 You may assume that A's column number is equal to B's row number.

1044 Example:

1045 A = [
 1046 [1, 0, 0],
 1047 [-1, 0, 3]
 1048]

1049 B = [
 1050 [7, 0, 0],
 1051 [0, 0, 0],
 1052 [0, 0, 1]
 1053]

1054
 1055
 1056
 1057 AB = | 1 0 0 | | 7 0 0 | | 7 0 0 |
 1058 | -1 0 3 | x | 0 0 0 | = | -7 0 3 |
 1059 | 0 0 1 |

1060
 1061 If it is of your interest, I would recommend you take a few minutes to think about how you would
 1062 approach this problem!

1063
 1064 The main focus of this post is to 1) explain in detail why the provided solution is smart and 2) make
 1065 some improvements/tweaks in the code of the smart solution to show you which part is really
 1066 essential, 3) also I will briefly mention why Sparse Matrix Manipulation can help make some
 1067 improvements on top of the smart solution.

1068 a) Originally, the normal way to calculate the multiplication of two metrics A, and B is as follow: We
 1069 take the all values from the first line of A, and all values from the first column of B, and
 1070 multiply the corresponding values and sum them up, the final sum is the value for the location of
 1071 first column and first row in final result matrix. Similarly, the value at [i][j] of result matrix C,
 1072 which is C[i][j] is calculated as:

1073 C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + ... A[i][K]B[K][j]

1074 (which is the sum of each multiplication of corresponding K values from row i of A and K values from
 1075 column j of B)

1076 The Key is: if we calculate it this way, we finishing calculating the final value for the result matrix at
 1077 once

1078 Then a brute force solution is as follow:

1079

```

1080
1081
1082     public class Solution {
1083         public int[][] multiply(int[][] A, int[][] B) {
1084             int m = A.length, n = A[0].length, nB = B[0].length;
1085             int[][] C = new int[m][nB];
1086
1087             for(int i = 0; i < m; i++) {
1088                 for (int j = 0; j < nB; j++) {
1089                     for(int k = 0; k < n; k++) {
1090                         C[i][j] += A[i][k] * B[k][j];
1091                     }
1092                 }
1093             }
1094         }
1095
1096         b) The smart solution: the key part of smart solution is that: it does not calculate the final result at
1097             once, and it takes each value from A, and calculate and partial sum and accumulate it into the
1098             final spot:
1099             For example, for each value A[i][k], if it is not zero, it will be used at most nB times ( n is B[0].
1100                 length ), which can be illustrated as follow: Generally for the following equation:
1101                 C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + ... A[i][k]B[k][j] .... A[i][K]B[K][j]
1102                 j can be from 0 to nB, if we write all of them down, it will like following:
1103                 For i from 0 to nB:
1104                     C[ i ][ 0 ]=A[ i ][0]B[0][0] + A[i][1]B[1][0] + A[i][2]B[2][0] + ... A[i][k]B[k][0] .... A[i][K]B[K]
1105                         [0]
1106                     C[ i ][ 1 ]=A[ i ][0]B[0][1] + A[i][1]B[1][1] + A[i][2]B[2][1] + ... A[i][k]B[k][0] .... A[i][K]B[K]
1107                         [1]
1108                     ...
1109                     C[ i ][ nB ]=A[ i ][0]B[0][nB] + A[i][1]B[1][nB] + A[i][2]B[2][nB] + ... A[i][k]B[k][nB] .... A[i][K]
1110                         *[B[K][nB]
1111             As you can see from above: for the same value A[i][k] from the first matrix, it will be used at most
1112             nB times if A[i][k] is not zero. And the smart solution is taking advantage of that!!!, the smart
1113             solution can be described as:
1114             For each value A[i][k] in matrix A, if it is not zero, we calculate A[i][k] * B[k][j] and accumulate it
1115             into C[ i ][ j ] (Key part: the C[ i ][ j ] by now is not the final value in the result matrix !!
1116             Remember, in the brute force solution, the final value of C[i][j], takes sum of all multiplication
1117             values of K corresponding values from A and B? here C[ i ][ j ] is only sum of some
1118             multiplication values, NOT ALL until the program is done)
1119             BY NOW, it is very clear that, if the value A[ i ][ k ] from matrix is zero, we skip a For-loop-
1120             calculation, which is a loop iterating nB times, and this is the key part of why the smart solution
1121             is smart!!!
1122             The smart solution code is as follow:
1123             public class Solution {
1124                 public int[][] multiply(int[][] A, int[][] B) {
1125                     int m = A.length, n = A[0].length, nB = B[0].length;
1126                     int[][] C = new int[m][nB];
1127
1128                     for(int i = 0; i < m; i++) {
1129                         for(int k = 0; k < n; k++) {
1130                             if (A[i][k] != 0) {
1131                                 for (int j = 0; j < nB; j++) {
1132                                     if (B[k][j] != 0) C[i][j] += A[i][k] * B[k][j];
1133                                 }
1134                             }
1135                         }
1136                     }
1137                     return C;
1138                 }
1139             }
1140
1141             (Credit:@yavinci; I am having a different version of the solution, so I am directly referencing the
1142             original version as a reference to demonstrate how mine is different).

```

1134

1135

1136 Based on the discussion above, the inner checking (if ($B[k][j] \neq 0$)) is actually not necessary,
 1137 because whether or not we have that check, we still iterate nB times, (since the operation $C[i][j]$
 1138 $\leftarrow A[i][k] * B[k][j]$; inside the if–check is $O(1)$ time)

1139

1139 So the smart solution can also be written as follow by removing the check (which is my version):
 1140

```
1140 public class Solution {
1141     public int[][] multiply(int[][] A, int[][] B) {
1142         int m = A.length, n = A[0].length, nB = B[0].length;
1143         int[][] C = new int[m][nB];
1144
1145         for(int i = 0; i < m; i++) {
1146             for(int k = 0; k < n; k++) {
1147                 if (A[i][k] != 0) {
1148                     for (int j = 0; j < nB; j++) {
1149                         if (B[k][j] != 0) C[i][j] += A[i][k] * B[k][j];
1150                     }
1151                 }
1152             }
1153         }
1154     }
1155 }
```

1154

1155 c) "Sparse matrix manipultion" helps, if we compress the first sparse matrix into rows of lists(in
 1156 each row list, it contains (value, index) pair), we actually don't need to go over all values in a
 1157 row in matrix A when are calculating the final result matrix. But Overall, it does not help
 1158 improve run time algorithmatically!!

1159

1160 References:

1161 1. Image credit: attachments_article_algorithm_col_slide_lamparas–colgantes–algorithm–slide–03.
 1162 jpg

1163

1162 Please enable JavaScript to view the comments powered by Disqus.

1163 Feel free to share!

1164

1165 You may also enjoy...

1166

1167 Please enable JavaScript to view the comments powered by Disqus. comments powered by Disqus

1168

1168 10000 Terabyte | An unserious blog
 1169 10000tb.org xuehaohu.com david–hu.org

1170

1171

Meta:

1172

1173

URL: <http://10000tb.org/Detailed-Explanation-of-Easiest-JAVA-solutions-Beating-ninety-nine-point-nine.html>

1174

1175

WARC Filename:

1176

CC-MAIN-2021-31/segments/1627046154500.32/warc/CC-MAIN-20210804013942-00074.warc.gz

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

```

1188 A sample from Nemotron-CC-Math
1189
1190
1191 # 10000 Terabyte
1192
1193 ## Detailed Explanation of a Smart Solution to an Algorithm Problem
1194
1195 *Written on January 7th, 2018 by @10000TB*
1196
1197 This post is about a coding problem and why the solution provided is smart. The problem is as follows:
1198
1199 **Problem:** Given two sparse matrices  $\begin{pmatrix} A \end{pmatrix}$  and  $\begin{pmatrix} B \end{pmatrix}$ , return the result of  $\begin{pmatrix} AB \end{pmatrix}$ . You may
1200 assume that  $\begin{pmatrix} A \end{pmatrix}$ 's column number is equal to  $\begin{pmatrix} B \end{pmatrix}$ 's row number.
1201
1202 **Example:**  

1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

1242
1243
1244     \[
1245     C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + \ldots + A[i][K]B[K][j]
1246     \]
1247
1248 The brute force solution is as follows:
1249
1250     ````java
1251     public class Solution {
1252         public int[][] multiply(int[][] A, int[][] B) {
1253             int m = A.length, n = A[0].length, nB = B[0].length;
1254             int[][] C = new int[m][nB];
1255
1256             for (int i = 0; i < m; i++) {
1257                 for (int j = 0; j < nB; j++) {
1258                     for (int k = 0; k < n; k++) {
1259                         C[i][j] += A[i][k] * B[k][j];
1260                     }
1261                 }
1262             }
1263         }
1264     }
1265
1266
1267     ````
```

b) The Smart Solution

The key part of the smart solution is that it does not calculate the final result at once. Instead, it takes each value from \mathcal{A} , calculates a partial sum, and accumulates it into the final spot.

For example, for each value $\mathcal{A}[i][k]$, if it is not zero, it will be used at most nB times (n is $B[0].length$). Generally, for the following equation:

```

1270
1271     \[
1272     C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + \ldots + A[i][k]B[k][j] + \ldots + A[i][K]
1273     B[K][j]
1274 \]
```

j can be from 0 to nB . If we write all of them down, it will look like this:

For i from 0 to nB :

```

1278
1279     \[
1280     C[i][0] = A[i][0]B[0][0] + A[i][1]B[1][0] + A[i][2]B[2][0] + \ldots + A[i][k]B[k][0] + \ldots + A[i][
1281     K]B[K][0]
1282 \]
1283
1284     \[
1285     C[i][1] = A[i][0]B[0][1] + A[i][1]B[1][1] + A[i][2]B[2][1] + \ldots + A[i][k]B[k][1] + \ldots + A[i][
1286     K]B[K][1]
1287 \]
1288
1289     \[
1290     \ldots
1291     \]
1292
1293     \[
1294     C[i][nB] = A[i][0]B[0][nB] + A[i][1]B[1][nB] + A[i][2]B[2][nB] + \ldots + A[i][k]B[k][nB] + \ldots
1295     + A[i][K]B[K][nB]
1296 \]
```

1296
1297
1298
1299
1300
1301
1302
1303

As you can see, for the same value $\langle A[i][k] \rangle$ from the first matrix, it will be used at most $\langle nB \rangle$ times if $\langle A[i][k] \rangle$ is not zero. The smart solution takes advantage of this by calculating $\langle A[i][k] \times B[k][j] \rangle$ and accumulating it into $\langle C[i][j] \rangle$. Note that $\langle C[i][j] \rangle$ is not the final value in the result matrix at this point. In the brute force solution, the final value of $\langle C[i][j] \rangle$ is the sum of all multiplication values of corresponding values from $\langle A \rangle$ and $\langle B \rangle$; here, $\langle C[i][j] \rangle$ is only the sum of some multiplication values, not all, until the program is done.

1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324

```
```java
public class Solution {
 public int[][] multiply(int[][] A, int[][] B) {
 int m = A.length, n = A[0].length, nB = B[0].length;
 int[][] C = new int[m][nB];

 for (int i = 0; i < m; i++) {
 for (int k = 0; k < n; k++) {
 if (A[i][k] != 0) {
 for (int j = 0; j < nB; j++) {
 if (B[k][j] != 0) C[i][j] += A[i][k] * B[k][j];
 }
 }
 }
 }
 return C;
 }
}```
```

(Credit: @yavinci; I am having a different version of the solution, so I am directly referencing the original version as a reference to demonstrate how mine is different.)

1325  
1326  
1327  
1328

Based on the discussion above, the inner checking  $\langle (if (B[k][j] \neq 0)) \rangle$  is actually not necessary because whether or not we have that check, we still iterate  $\langle nB \rangle$  times. The operation  $\langle C[i][j] += A[i][k] \times B[k][j] \rangle$  inside the if-check is  $\langle O(1) \rangle$  time. So the smart solution can also be written as follows by removing the check (which is my version):

1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349

```
```java
public class Solution {
    public int[][] multiply(int[][] A, int[][] B) {
        int m = A.length, n = A[0].length, nB = B[0].length;
        int[][] C = new int[m][nB];

        for (int i = 0; i < m; i++) {
            for (int k = 0; k < n; k++) {
                if (A[i][k] != 0) {
                    for (int j = 0; j < nB; j++) {
                        C[i][j] += A[i][k] * B[k][j];
                    }
                }
            }
        }
        return C;
    }
}```
```

c) Sparse Matrix Manipulation

Sparse matrix manipulation helps if we compress the first sparse matrix into rows of lists (in each row list, it contains (value, index) pairs). We actually don't need to go over all values in a row in matrix $\langle A \rangle$ when calculating the final result matrix. However, overall, it does not help improve runtime algorithmically.

1350

1351

1352

References

1353

1. Image credit: attachments_article_algorithm_col_slide_lamparas-colgantes-algorithm-slide-03.jpg

1354

Meta:

1355

URL:

1356

<http://10000tb.org/Detailed-Explanation-of-Easiest-JAVA-solutions-Beating-ninety-nine-point-nine.html>

1357

WARC Filename:

1358

CC-MAIN-2021-31/segments/1627046154500.32/warc/CC-MAIN-20210804013942-00074.warc.gz

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404 A sample from OpenWebMath

1405

1406

1407

1408

1409

1410

1411

1412

1413

Detailed explanation of a smart solution to an algo problem beating 99.9% submission

This post is about a coding problem and why the solution I pasted down below is smart.

Problem:

Given two sparse matrices A and B, return the result of AB.

You may assume that A's column number is equal to B's row number.

Example:

A = [
[1, 0, 0],
[-1, 0, 3]
]B = [
[7, 0, 0],
[0, 0, 0],
[0, 0, 1]
]AB =
$$\begin{vmatrix} 1 & 0 & 0 \\ -1 & 0 & 3 \end{vmatrix} \times \begin{vmatrix} 7 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 7 & 0 & 0 \\ -7 & 0 & 3 \\ 0 & 0 & 1 \end{vmatrix}$$

If it is of your interest, I would recommend you take a few minutes to think about how you would approach this problem!

The main focus of this post is to 1) explain in detail why the provided solution is smart and 2) make some improvements/tweaks in the code of the smart solution to show you which part is really essential, 3) also I will briefly mention why Sparse Matrix Manipulation can help make some improvements on top of the smart solution.

a) Originally, the normal way to calculate the multiplication of two metrics A, and B is as follow: We take the all values from the first line of A, and all values from the first column of B, and multiply the corresponding values and sum them up, the final sum is the value for the location of first column and first row in final result matrix. Similarly, the value at [i][j] of result matrix C, which is C[i][j] is calculated as:

C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + ... A[i][K]B[K][j]
(which is the sum of each multiplication of corresponding K values from row i of A and K values from column j of B)

The key is: if we calculate it this way, we finishing calculating the final value for the result matrix at once

Then a brute force solution is as follow:

b) The smart solution: the key part of smart solution is that: it does not calculate the final result at once, and it takes each value from A, and calculate and partial sum and accumulate it into the final spot: For example, for each value A[i][k], if it is not zero, it will be used at most nB times (n is B[0].length), which can be illustrated as follow: Generally for the following equation:

C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + ... A[i][k]B[k][j] A[i][K]B[K][j]

j can be from 0 to nB, if we write all of them down, it will like following:

For i from 0 to nB:

```

1458
1459 C[ i ][ 0 ]=A[ i ][0]
1460 B[0][0] + A[i][1]B[1][0] + A[i][2]B[2][0] + ... A[i][k]B[k][0] .... A[i][K]B[K][0]
1461 C[ i ][ 1 ]=A[ i ][0]
1462 B[0][1] + A[i][1]B[1][1] + A[i][2]B[2][1] + ... A[i][k]B[k][0] .... A[i][K]B[K][1]
1463
1464 C[ i ][ nB ]=A[ i ][0]
1465 B[0][nB] + A[i][1]B[1][nB] + A[i][2]B[2][nB] + ... A[i][k]B[k][nB] .... A[i][K]*B[K][nB]
1466
1467 As you can see from above: for the same value A[i][k] from the first matrix, it will be used at most
1468 nB times if A[i][k] is not zero. And the smart solution is taking advantage of that!!!, the smart
1469 solution can be described as:
1470
1471 For each value A[i][k] in matrix A, if it is not zero, we calculate A[i][k] * B[k][j] and accumulate it
1472 into C[ i ][ j ] (Key part: the C[ i ][ j ] by now is not the final value in the result matrix !!
1473 Remember, in the brute force solution, the final value of C[i][j], takes sum of all multiplication
1474 values of K corresponding values from A and B? here C[ i ][ j ] is only sum of some
1475 multiplication values, NOT ALL until the program is done)
1476
1477 BY NOW, it is very clear that, if the value A[ i ][ k ] from matrix is zero, we skip a For-loop-
1478 calculation, which is a loop iterating nB times, and this is the key part of why the smart solution
1479 is smart!!!
1480
1481 The smart solution code is as follow:
1482
1483 (Credit:@yavinci; I am having a different version of the solution, so I am directly referencing the
1484 original version as a reference to demonstrate how mine is different).
1485
1486 Based on the discussion above, the inner checking (if (B[k][j] != 0) is actually not necessary,
1487 because whether or not we have that check, we still iterate nB times, ( since the operation C[i][j]
1488 ] += A[i][k] * B[k][j]; inside the if-check is O(1) time)
1489
1490 So the smart solution can also be written as follow by removing the check ( which is my version ):
1491
1492 c) "Sparse matrix manipulation" helps, if we compress the first sparse matrix into rows of lists( in
1493 each row list, it contains ( value, index ) pair ), we actually don't need to go over all values in a
1494 row in matrix A when are calculating the final result matrix. But Overall, it does not help
1495 improve run time algorithmatically!!
1496
1497 References:
1498
1499 1. Image credit: attachments.article.algorithm.col_slide_lamparas-colgantes-algorithm-slide-03.jpg
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

```

```

1512 A sample from MegaMath-Pro
1513
1514
1515 # Detailed Explanation of a Smart Solution to a Matrix Multiplication Problem
1516 The problem involves multiplying two sparse matrices A and B. We are given that A's column number
1517 is equal to B's row number.
1518
1519 ## Problem Statement
1520
1521 Given two sparse matrices A and B, return the result of AB.
1522
1523 Example:
1524 A = [
1525 [ 1, 0, 0],
1526 [-1, 0, 3]
1527 ]
1528 B = [
1529 [ 7, 0, 0 ],
1530 [ 0, 0, 0 ],
1531 [ 0, 0, 1 ]
1532 ]
1533 The result AB is calculated as follows:
1534 AB = [
1535 [ 7, 0, 0 ],
1536 [ -7, 0, 3 ]
1537 ]
1538 ## Normal Approach to Matrix Multiplication
1539
1540 The normal way to calculate the multiplication of two matrices A and B is to take the values from the
1541 first line of A and the first column of B, multiply the corresponding values, and sum them up. The
1542 final sum is the value for the location of the first column and first row in the final result matrix.
1543
1544 C[i][j] = A[i][0]*B[0][j] + A[i][1]*B[1][j] + A[i][2]*B[2][j] + ... + A[i][K]*B[K][j]
1545
1546 ## Brute Force Solution
1547
1548 The brute force solution involves calculating the final result at once.
1549
1550 ## Smart Solution
1551
1552 The smart solution does not calculate the final result at once. Instead, it takes each value from A,
1553 calculates a partial sum, and accumulates it into the final spot.
1554
1555 For each value A[i][k], if it is not zero, it will be used at most nB times (where n is B[0].length). The
1556 smart solution can be described as follows:
1557
1558 For each value A[i][k] in matrix A, if it is not zero, calculate A[i][k] * B[k][j] and accumulate it into C[i]
1559 [j].
1560
1561 The key part of the smart solution is that it skips a For-loop calculation if the value A[i][k] from matrix
1562 A is zero. This is the key part of why the smart solution is smart.
1563
1564 ## Smart Solution Code
1565
1566 The smart solution code can be written as follows:
1567
1568 ## Improvement Using Sparse Matrix Manipulation
1569
1570 Sparse matrix manipulation can help improve the solution by compressing the first sparse matrix into
1571 rows of lists (in each row list, it contains a (value, index) pair). This allows us to avoid going over
1572 all values in a row in matrix A when calculating the final result matrix. However, this does not
1573 improve the runtime algorithmatically.

```

```

1566
1567
1568    ## Key Takeaways
1569
1570        * The smart solution does not calculate the final result at once.
1571        * The smart solution skips a For-loop calculation if the value A[i][k] from matrix A is zero.
1572        * Sparse matrix manipulation can help improve the solution, but it does not improve the runtime
1573            algorithmatically.
1574
1575    ## Example Multiple Choice Question
1576
1577        What is the key advantage of the smart solution?
1578
1579            A) It calculates the final result at once.
1580            B) It skips a For-loop calculation if the value A[i][k] from matrix A is zero.
1581            C) It uses sparse matrix manipulation to improve runtime.
1582            D) It has a higher time complexity than the brute force solution.
1583
1584        Answer: B) It skips a For-loop calculation if the value A[i][k] from matrix A is zero.

```

Meta:
URL:

<http://10000tb.org/Detailed-Explanation-of-Easiest-JAVA-solutions-Beating-ninety-nine-point-nine.html>

WARC Filename:

CC-MAIN-2021-31/segments/1627046154500.32/warc/CC-MAIN-20210804013942-00074.warc.gz

```

1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619

```

1620 A sample document from Finemath-4+

1621

1622 # Detailed explanation of a smart solution to an algo problem beating 99.9% submission

1623

1624 This post is about a coding problem and why the solution I pasted down below is smart.

1625

1626 Problem:

1627

1628 Given two sparse matrices A and B, return the result of AB.

1629

1630 You may assume that A's column number is equal to B's row number.

1631

1632 Example:

1633

1634 $A = \begin{bmatrix} 1, 0, 0 \\ -1, 0, 3 \end{bmatrix}$

1635

1636

1637

1638 $B = \begin{bmatrix} 7, 0, 0 \\ 0, 0, 0 \\ 0, 0, 1 \end{bmatrix}$

1639

1640

1641

1642

1643 $AB = \begin{vmatrix} 1 & 0 & 0 \\ -1 & 0 & 3 \end{vmatrix} \times \begin{vmatrix} 7 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 7 & 0 & 0 \\ -7 & 0 & 3 \\ 0 & 0 & 1 \end{vmatrix}$

1644

1645

1646

1647

1648 If it is of your interest, I would recommend you take a few minutes to think about how you would approach this problem!

1649

1650 The main focus of this post is to 1) explain in detail why the provided solution is smart and 2) make some improvements/tweaks in the code of the smart solution to show you which part is really essential,

1651 3) also I will briefly mention why Sparse Matrix Manipulation can help make some improvements

1652 on top of the smart solution.

1653

1654 a) Originally, the normal way to calculate the multiplication of two matrices A, and B is as follows: We

1655 take all the values from the first line of A, and all values from the first column of B, and

1656 multiply the corresponding values and sum them up, the final sum is the value for the location of

1657 first column and first row in final result matrix. Similarly, the value at [i][j] of result matrix C,

1658 which is $C[i][j]$ is calculated as:

1659

1660 $C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + \dots + A[i][K]B[K][j]$

1661 (which is the sum of each multiplication of corresponding K values from row i of A and K values from

1662 column j of B)

1663 The key is: if we calculate it this way, we finish calculating the final value for the result matrix at

1664 once

1665

1666 Then a brute force solution is as follows:

1667

1668 b) The smart solution: the key part of smart solution is that: it does not calculate the final result at once,

1669 and it takes each value from A, and calculate and partial sum and accumulate it into the final spot:

1670 For example, for each value $A[i][k]$, if it is not zero, it will be used at most nB times (n is $B[0].length$),

1671 which can be illustrated as follows: Generally for the following equation:

1672

1673 $C[i][j] = A[i][0]B[0][j] + A[i][1]B[1][j] + A[i][2]B[2][j] + \dots + A[i][k]B[k][j] + \dots + A[i][K]B[K][j]$

j can be from 0 to nB , if we write all of them down, it will look like the following:

For i from 0 to nB :

1674

1675

C[i][0]=A[i][0]

B[0][0] + A[i][1]B[1][0] + A[i][2]B[2][0] + ... A[i][k]B[k][0] A[i][K]B[K][0]

1676

C[i][1]=A[i][0]

B[0][1] + A[i][1]B[1][1] + A[i][2]B[2][1] + ... A[i][k]B[k][0] A[i][K]B[K][1]

1677

1678

C[i][nB]=A[i][0]

B[0][nB] + A[i][1]B[1][nB] + A[i][2]B[2][nB] + ... A[i][k]B[k][nB] A[i][K]*B[K][nB]

1679

1680

As you can see from above: for the same value A[i][k] from the first matrix, it will be used at most

1681 nB times if A[i][k] is not zero. And the smart solution is taking advantage of that!!!, the smart

1682

1683

solution can be described as:

1684

1685 For each value A[i][k] in matrix A, if it is not zero, we calculate A[i][k] * B[k][j] and accumulate it

1686 into C[i][j] (Key part: the C[i][j] by now is not the final value in the result matrix !!)

1687 Remember, in the brute force solution, the final value of C[i][j], takes sum of all multiplication

1688 values of K corresponding values from A and B? here C[i][j] is only sum of some

1689 multiplication values, NOT ALL until the program is done)

1690

1691 BY NOW, it is very clear that, if the value A[i][k] from matrix is zero, we skip a For-loop–

1692 calculation, which is a loop iterating nB times, and this is the key part of why the smart solution

1693

1694

The smart solution code is as follow:

1695

1696

(Credit:@yavinci; I am having a different version of the solution, so I am directly referencing the original version as a reference to demonstrate how mine is different).

1697

1698

Based on the discussion above, the inner checking (if (B[k][j] != 0) is actually not necessary, because whether or not we have that check, we still iterate nB times, (since the operation C[i][j] += A[i][k] * B[k][j]; inside the if–check is O(1) time)

1699

1700

So the smart solution can also be written as follow by removing the check (which is my version):
c) "Sparse matrix manipulation" helps, if we compress the first sparse matrix into rows of lists(in each row list, it contains (value, index) pair), we actually don't need to go over all values in a row in matrix A when are calculating the final result matrix. But Overall, it does not help improve run time algorithmatically!!

1701

1702

References:

1703

1704

1. Image credit: attachments.article.algorithm.col_slide_lamparas-colgantes-algorithm-slide-03.jpg

1705

1706

Meta:

1707

1708

URL:

<http://10000tb.org/Detailed-Explanation-of-Easiest-JAVA-solutions-Beating-ninety-nine-point-nine.html>

1709

1710

WARC Filename:

1711

1712

CC-MAIN-2021-31/segments/1627046154500.32/warc/CC-MAIN-20210804013942-00074.warc.gz

1713

1714

1715

1716

1717

1718

1719

1720

1721

1722

1723

1724

1725

1726

1727

1728	A sample document from MegaMath-Web
1729	
1730	
1731	### Example 1: Calculating Heat Transfer Through Conduction: Conduction Rate Through an Ice Box
1732	A Styrofoam ice box has a total area of 0.950 m ² and walls with an average thickness of 2.50
1733	cm. The box contains ice, water, and canned beverages atThe inside of the box is kept cold by
1734	melting ice. How much ice melts in one day if the ice box is kept in the trunk of a car at
1735	**Strategy**
1736	This question involves both heat for a phase change (melting of ice) and the transfer of heat by
1737	conduction. To find the amount of ice melted, we must find the net heat transferred. This value can
1738	be obtained by calculating the rate of heat transfer by conduction and multiplying by time.
1739	**Solution**
1740	– Identify the knowns.
1741	– Identify the unknowns. We need to solve for the mass of the ice,We will also need to solve for the net
1742	heat transferred to melt the ice,
1743	– Determine which equations to use. The rate of heat transfer by conduction is given by
1744	$[latex]\boldsymbol{=}[/latex][/math]$
1745	– The heat is used to melt the ice:
1746	– Insert the known values:
1747	$[latex]\boldsymbol{=}\textbf{13.3}\textbf{ J/s}[/math]$
1748	– Multiply the rate of heat transfer by the time ():
1749	– Set this equal to the heat transferred to melt the ice:Solve for the mass
1750	$[latex]\frac{Q}{L_f} = \frac{1.15 \times 10^6 \textbf{ J}}{334 \times 10^3 \textbf{ J/kg}}[/math]$
1751	**Discussion**
1752	The result of 3.44 kg, or about 7.6 lbs, seems about right, based on experience. You might expect to use
1753	about a 4 kg (7–10 lb) bag of ice per day. A little extra ice is required if you add any warm food or
1754	beverages.
1755	Inspecting the conductivities in Table 3 shows that Styrofoam is a very poor conductor and thus a good
1756	insulator. Other good insulators include fiberglass, wool, and goose-down feathers. Like
1757	Styrofoam, these all incorporate many small pockets of air, taking advantage of air’s poor thermal
1758	conductivity.
1759	Substance
1760	Thermal conductivity k (J/s.m.°C)
1761	--- ---
1762	Silver 420
1763	Copper 390
1764	Gold 318
1765	Aluminum 220
1766	Steel iron 80
1767	Steel (stainless) 14
1768	Ice 2.2
1769	Glass (average) 0.84
1770	Concrete brick 0.84
1771	Water 0.6
1772	Fatty tissue (without blood) 0.2
1773	Asbestos 0.16
1774	Plasterboard 0.16
1775	Wood 0.08–0.16
1776	Snow (dry) 0.10
1777	Cork 0.042
1778	Glass wool 0.042
1779	Wool 0.04
1780	Down feathers 0.025
1781	Air 0.023
	Styrofoam 0.010
	Table 3. Thermal Conductivities of Common Substances ^{1}

1782

1783

1784

1785

1786

1787

1788

1789

1790

1791

1792

1793

1794

1795

1796

1797

1798

A combination of material and thickness is often manipulated to develop good insulators—the smaller the conductivity and the larger the thickness the better. The ratio of will thus be large for a good insulator. The ratio is called the factor. The rate of conductive heat transfer is inversely proportional to the larger the value of the better the insulation. factors are most commonly quoted for household insulation, refrigerators, and the like—unfortunately, it is still in non-metric units of $\text{ft}^2 \cdot ^\circ\text{F} \cdot \text{h/Btu}$, although the unit usually goes unstated (1 British thermal unit [Btu] is the amount of energy needed to change the temperature of 1.0 lb of water by 1.0 $^\circ\text{F}$). A couple of representative values are a factor of 11 for 3.5-in-thick fiberglass batts (pieces) of insulation and a factor of 19 for 6.5-in-thick fiberglass batts. Walls are usually insulated with 3.5-in batts, while ceilings are usually insulated with 6.5-in batts. In cold climates, thicker batts may be used in ceilings and walls.

Note that in Table 3, the best thermal conductors—silver, copper, gold, and aluminum—are also the best electrical conductors, again related to the density of free electrons in them. Cooking utensils are typically made from good conductors.

Example 2: Calculating the Temperature Difference Maintained by a Heat Transfer: Conduction Through an Aluminum Pan

Water is boiling in an aluminum pan placed on an electrical element on a stovetop. The sauce pan has a bottom that is 0.800 cm thick and 14.0 cm in diameter. The boiling water is evaporating at the rate of 1.00 g/s. What is the temperature difference across (through) the bottom of the pan?

Strategy

Conduction through the aluminum is the primary method of heat transfer here, and so we use the equation for the rate of heat transfer and solve for the temperature difference.

Solution

– Identify the knowns and convert them to the SI units.

The thickness of the pan, the area of the pan, and the thermal conductivity,

– Calculate the necessary heat of vaporization of 1 g of water:

– Calculate the rate of heat transfer given that 1 g of water melts in one second:

– Insert the knowns into the equation and solve for the temperature difference:

$$[\text{latex size}="2"] \boldsymbol{\left(\frac{Q}{t} \right)} = \frac{8.00 \times 10^{-3} \text{ m}^2}{(220 \text{ J/s}) \cdot (0.800 \text{ cm})} \cdot (1.54 \times 10^{-2} \text{ m}^2)$$

Discussion

The value for the heat transfer is typical for an electric stove. This value gives a remarkably small temperature difference between the stove and the pan. Consider that the stove burner is red hot while the inside of the pan is nearly because of its contact with boiling water. This contact effectively cools the bottom of the pan in spite of its proximity to the very hot stove burner. Aluminum is such a good conductor that it only takes this small temperature difference to produce a heat transfer of 2.26 kW into the pan.

Conduction is caused by the random motion of atoms and molecules. As such, it is an ineffective mechanism for heat transport over macroscopic distances and short time distances. Take, for example, the temperature on the Earth, which would be unbearably cold during the night and extremely hot during the day if heat transport in the atmosphere was to be only through conduction. In another example, car engines would overheat unless there was a more efficient way to remove excess heat from the pistons.

Check Your Understanding

1: How does the rate of heat transfer by conduction change when all spatial dimensions are doubled?

1834

1835

1836

1837

1838

1839

Summary

- Heat conduction is the transfer of heat between two objects in direct contact with each other.
- The rate of heat transfer(energy per unit time) is proportional to the temperature differenceand the contact areaand inversely proportional to the distancebetween the objects:

$$[tex]\boldsymbol{=}[/tex]$$

1843

1844

Meta:

1845

URL:

1846

<http://pressbooks-dev.oer.hawaii.edu/collegephysics/chapter/14-5-conduction/>

1847

WARC Filename:

1848

crawl-data/CC-MAIN-2019-04/segments/1547583658844.27/warc/CC-MAIN-20190117062012-20190117084012-00486.warc.gz

1849

Lang score: 0.767403781414032

1850

Math score: 0.5140737295150757

1851

1852

1853

1854

1855

1856

1857

1858

1859

1860

1861

1862

1863

1864

1865

1866

1867

1868

1869

1870

1871

1872

1873

1874

1875

1876

1877

1878

1879

1880

1881

1882

1883

1884

1885

1886

1887

1888

1889

1890	A sample document from Nemotron-CC-Math					
1891						
1892	### Example 1: Calculating Heat Transfer Through Conduction: Conduction Rate Through an Ice Box					
1893						
1894	A Styrofoam ice box has a total area of 0.950 m^2 and walls with an average thickness of 2.50 cm. The box contains ice, water, and canned beverages at 0°C . The inside of the box is kept cold by melting ice. How much ice melts in one day if the ice box is kept in the trunk of a car at 35.0°C ?					
1895						
1896						
1897						
1898						
1899	**Strategy**					
1900	This question involves both heat for a phase change (melting of ice) and the transfer of heat by conduction. To find the amount of ice melted, we must find the net heat transferred. This value can be obtained by calculating the rate of heat transfer by conduction and multiplying by time.					
1901						
1902						
1903	**Solution**					
1904	1. Identify the knowns.					
1905	$\begin{aligned} A &= 0.950 \text{ m}^2, d = 2.50 \text{ cm} = 0.0250 \text{ m}, T_1 = 0^\circ\text{C}, \\ T_2 &= 35.0^\circ\text{C}, t = 1 \text{ day} = 24 \text{ hours} = 86,400 \text{ s}. \end{aligned}$					
1906						
1907						
1908	2. Identify the unknowns. We need to solve for the mass of the ice, m . We will also need to solve for the net heat transferred to melt the ice, Q .					
1909						
1910						
1911	3. Determine which equations to use. The rate of heat transfer by conduction is given by					
1912	$\frac{Q}{t} = \frac{kA(T_2 - T_1)}{d}$					
1913						
1914						
1915	4. The heat is used to melt the ice: $Q = mL_f$.					
1916						
1917	5. Insert the known values:					
1918	$\begin{aligned} \frac{Q}{t} &= \frac{(0.010 \text{ J/s}) \cdot (0.950 \text{ m}^2) \cdot (35.0^\circ\text{C} - 0^\circ\text{C})}{0.0250 \text{ m}} = 13.3 \text{ J/s}. \end{aligned}$					
1919						
1920						
1921						
1922	6. Multiply the rate of heat transfer by the time ($1 \text{ day} = 86,400 \text{ s}$):					
1923	$Q = (Q/t)t = (13.3 \text{ J/s})(86,400 \text{ s}) = 1.15 \times 10^6 \text{ J}$					
1924						
1925						
1926	7. Set this equal to the heat transferred to melt the ice: $Q = mL_f$. Solve for the mass m :					
1927	$m = \frac{Q}{L_f} = \frac{1.15 \times 10^6 \text{ J}}{334 \times 10^3 \text{ J/kg}} = 3.44 \text{ kg}$					
1928						
1929						
1930						
1931						
1932	**Discussion**					
1933	The result of 3.44 kg, or about 7.6 lbs, seems about right, based on experience. You might expect to use about a 4 kg (7–10 lb) bag of ice per day. A little extra ice is required if you add any warm food or beverages.					
1934						
1935						
1936	Inspecting the conductivities in Table 3 shows that Styrofoam is a very poor conductor and thus a good insulator. Other good insulators include fiberglass, wool, and goose-down feathers. Like Styrofoam, these all incorporate many small pockets of air, taking advantage of air's poor thermal conductivity.					
1937						
1938						
1939						
1940	**Substance Thermal Conductivity**					
1941						
1942	<table border="1"> <thead> <tr> <th>Substance</th> <th>Thermal conductivity (k) (J/s·m·°C)</th> </tr> </thead> <tbody> <tr> <td>Silver</td> <td>420</td> </tr> </tbody> </table>		Substance	Thermal conductivity (k) (J/s·m·°C)	Silver	420
Substance	Thermal conductivity (k) (J/s·m·°C)					
Silver	420					
1943						

1944		
1945	Copper	390
1946	Gold	318
1947	Aluminum	220
1948	Steel iron	80
1949	Steel (stainless)	14
1950	Ice	2.2
1951	Glass (average)	0.84
1952	Concrete brick	0.84
1953	Water	0.6
1954	Fatty tissue (without blood)	0.2
1955	Asbestos	0.16
1956	Plasterboard	0.16
1957	Wood	0.08–0.16
1958	Snow (dry)	0.10
1959	Cork	0.042
1960	Glass wool	0.042
1961	Wool	0.04
1962	Down feathers	0.025
1963	Air	0.023
1964	Styrofoam	0.010

Table 3. Thermal Conductivities of Common Substances

A combination of material and thickness is often manipulated to develop good insulators—the smaller the conductivity (k) and the larger the thickness (d) , the better. The ratio of (d/k) will thus be large for a good insulator. The ratio (d/k) is called the (R) factor. The rate of conductive heat transfer is inversely proportional to (R) . The larger the value of (R) , the better the insulation. (R) factors are most commonly quoted for household insulation, refrigerators, and the like—unfortunately, it is still in non-metric units of $\text{ft}^2 \cdot \text{F.h/Btu}$, although the unit usually goes unstated (1 British thermal unit [Btu] is the amount of energy needed to change the temperature of 1.0 lb of water by 1.0 °F). A couple of representative values are an (R) factor of 11 for 3.5-in-thick fiberglass batts (pieces) of insulation and an (R) factor of 19 for 6.5-in-thick fiberglass batts. Walls are usually insulated with 3.5-in batts, while ceilings are usually insulated with 6.5-in batts. In cold climates, thicker batts may be used in ceilings and walls.

Figure 4. The fiberglass batt is used for insulation of walls and ceilings to prevent heat transfer between the inside of the building and the outside environment.

Note that in Table 3, the best thermal conductors—silver, copper, gold, and aluminum—are also the best electrical conductors, again related to the density of free electrons in them. Cooking utensils are typically made from good conductors.

Example 2: Calculating the Temperature Difference Maintained by a Heat Transfer: Conduction Through an Aluminum Pan

Water is boiling in an aluminum pan placed on an electrical element on a stovetop. The saucepan has a bottom that is 0.800 cm thick and 14.0 cm in diameter. The boiling water is evaporating at the rate of 1.00 g/s. What is the temperature difference across (through) the bottom of the pan?

Strategy

Conduction through the aluminum is the primary method of heat transfer here, and so we use the equation for the rate of heat transfer and solve for the temperature difference.

$$\begin{aligned} T_2 - T_1 &= \frac{Q}{t} \left(\frac{d}{kA} \right) \\ \end{aligned}$$

Solution

1. Identify the knowns and convert them to the SI units.

- The thickness of the pan, $(d = 0.800, \text{cm} = 8.0 \times 10^{-3}, \text{m})$,
- The area of the pan, $(A = \pi(0.14/2)^2, \text{m}^2 = 1.54 \times 10^{-2}, \text{m}^2)$,
- The thermal conductivity, $(k = 220, \text{J/s} \cdot \text{m} \cdot \text{circ}^\circ\text{C})$.

1998
 1999
 2000 2. Calculate the necessary heat of vaporization of 1 g of water:
 2001
$$Q = mL_v = (1.00 \times 10^{-3} \text{ kg})(2256 \times 10^3 \text{ J/kg}) = 2256 \text{ J}$$

 2002
 2003
 2004
 2005 3. Calculate the rate of heat transfer given that 1 g of water evaporates in one second:
 2006
$$Q/t = 2256 \text{ J/s or } 2.26 \text{ kW}$$

 2007
 2008
 2009 4. Insert the knowns into the equation and solve for the temperature difference:
 2010
$$T_2 - T_1 = \frac{Q}{t} \left(\frac{d}{kA} \right) = (2256 \text{ J/s}) \frac{8.00 \times 10^{-3} \text{ m}}{(220 \text{ J/s}) \cdot (2 \text{ m}) \cdot (1.54 \times 10^{-2} \text{ m}^2)} = 5.33^\circ\text{C}$$

 2011
 2012
 2013
 2014
 2015 **Discussion**
 2016
 2017 The value for the heat transfer ($Q/t = 2.26 \text{ kW}$ or 2256 J/s) is typical for an
 2018 electric stove. This value gives a remarkably small temperature difference between the stove
 2019 and the pan. Consider that the stove burner is red hot while the inside of the pan is nearly
 2020 (100°C) because of its contact with boiling water. This contact effectively
 2021 cools the bottom of the pan in spite of its proximity to the very hot stove burner. Aluminum is
 2022 such a good conductor that it only takes this small temperature difference to produce a heat
 2023 transfer of 2.26 kW into the pan.
 2024
 2025 Conduction is caused by the random motion of atoms and molecules. As such, it is an ineffective
 2026 mechanism for heat transport over macroscopic distances and short time distances. Take, for
 2027 example, the temperature on the Earth, which would be unbearably cold during the night and
 2028 extremely hot during the day if heat transport in the atmosphere was to be only through
 2029 conduction. In another example, car engines would overheat unless there was a more efficient
 2030 way to remove excess heat from the pistons.
 2031
 2032 **### Check Your Understanding**
 2033
 2034 1: How does the rate of heat transfer by conduction change when all spatial dimensions are doubled?
 2035
 2036
 2037 **### Summary**
 2038
 2039 – Heat conduction is the transfer of heat between two objects in direct contact with each other.
 2040 – The rate of heat transfer (Q/t) (energy per unit time) is proportional to the temperature
 2041 difference ($T_2 - T_1$) and the contact area (A) and inversely proportional to the distance
 2042 (d) between the objects:
 2043
$$\frac{Q}{t} = \frac{kA(T_2 - T_1)}{d}$$

 2044
 2045 **Meta:**
 2046 **URL:**
 2047 <http://pressbooks-dev.oer.hawaii.edu/collegephysics/chapter/14-5-conduction/>
 2048 **WARC Filename:**
 2049 <crawl-data/CC-MAIN-2019-04/segments/1547583658844.27/warc/CC-MAIN-20190117062012-20190117084012-00486.warc.gz>
 2050
 2051

2052 A sample document from OpenWebMath
 2053
 2054
 2055 **### Example 1: Calculating Heat Transfer Through Conduction: Conduction Rate Through an Ice Box**
 2056 A Styrofoam ice box has a total area of 0.950 m² and walls with an average thickness of 2.50
 2057 cm. The box contains ice, water, and canned beverages atThe inside of the box is kept cold by
 2058 melting ice. How much ice melts in one day if the ice box is kept in the trunk of a car at
 2059
 2060 **Strategy**
 2061 This question involves both heat for a phase change (melting of ice) and the transfer of heat by
 2062 conduction. To find the amount of ice melted, we must find the net heat transferred. This value can
 2063 be obtained by calculating the rate of heat transfer by conduction and multiplying by time.
 2064
 2065 **Solution**
 2066 1. Identify the knowns.
 2067 2. Identify the unknowns. We need to solve for the mass of the ice,We will also need to solve for the net
 2068 heat transferred to melt the ice,
 2069 3. Determine which equations to use. The rate of heat transfer by conduction is given by
 2070 $\text{boldsymbol}\{=}\$$
 2071 4. The heat is used to melt the ice:
 2072 5. Insert the known values:
 2073 $\text{boldsymbol}\{=}\$ \text{boldsymbol}\{=\text{13.3}\text{textrbf}\{ J/s\}\}.$
 2074 6. Multiply the rate of heat transfer by the time ():
 2075 7. Set this equal to the heat transferred to melt the ice:Solve for the mass
 $[\text{textrsize="2"]}\text{boldsymbol}\{\frac{Q}{L_{\text{textrbf}\{ f\}}}\}[\text{textrsize="2"]}\text{boldsymbol}\{\frac{1.15\times10^6\text{textrbf}\{ J\}}{334\times10^3\text{textrbf}\{ J/kg\}}\}[\text{textr]$
 2076
 2077 **Discussion**
 2078 The result of 3.44 kg, or about 7.6 lbs, seems about right, based on experience. You might expect to use
 2079 about a 4 kg (7–10 lb) bag of ice per day. A little extra ice is required if you add any warm food or
 2080 beverages.
 2081 Inspecting the conductivities in Table 3 shows that Styrofoam is a very poor conductor and thus a good
 2082 insulator. Other good insulators include fiberglass, wool, and goose-down feathers. Like
 2083 Styrofoam, these all incorporate many small pockets of air, taking advantage of air's poor thermal
 2084 conductivity.
 2085
 2086 **Substance Thermal conductivity**
 2087 k (J/s.m.°C)
 2088 Silver 420
 2089 Copper 390
 2090 Gold 318
 2091 Aluminum 220
 2092 Steel iron 80
 2093 Steel (stainless) 14
 2094 Ice 2.2
 2095 Glass (average) 0.84
 2096 Concrete brick 0.84
 2097 Water 0.6
 2098 Fatty tissue (without blood) 0.2
 2099 Asbestos 0.16
 2100 Plasterboard 0.16
 2101 Wood 0.08–0.16
 2102 Snow (dry) 0.10
 2103 Cork 0.042
 2104 Glass wool 0.042
 2105 Wool 0.04
 2106 Down feathers 0.025
 2107 Air 0.023
 2108 Styrofoam 0.010
 2109
 2110 **Table 3. Thermal Conductivities of Common Substances**
 2111
 2112 A combination of material and thickness is often manipulated to develop good insulators—the smaller the
 2113 conductivityand the larger the thicknessthe better. The ratio ofwill thus be large for a good insulator. The
 2114 ratiois called thefactor. The rate of conductive heat transfer is inversely proportional to

2106

2107

The larger the value of the better the insulation. Factors are most commonly quoted for household insulation, refrigerators, and the like—unfortunately, it is still in non-metric units of $\text{ft}^2 \cdot ^\circ\text{F} \cdot \text{h} / \text{Btu}$, although the unit usually goes unstated (1 British thermal unit [Btu] is the amount of energy needed to change the temperature of 1.0 lb of water by 1.0 $^\circ\text{F}$). A couple of representative values are an factor of 11 for 3.5-in-thick fiberglass batts (pieces) of insulation and an factor of 19 for 6.5-in-thick fiberglass batts. Walls are usually insulated with 3.5-in batts, while ceilings are usually insulated with 6.5-in batts. In cold climates, thicker batts may be used in ceilings and walls.

2114

2115

2116

2117

Note that in Table 3, the best thermal conductors—silver, copper, gold, and aluminum—are also the best electrical conductors, again related to the density of free electrons in them. Cooking utensils are typically made from good conductors.

2118

2119

Example 2: Calculating the Temperature Difference Maintained by a Heat Transfer: Conduction Through an Aluminum Pan

2120

2121

2122

Water is boiling in an aluminum pan placed on an electrical element on a stovetop. The sauce pan has a bottom that is 0.800 cm thick and 14.0 cm in diameter. The boiling water is evaporating at the rate of 1.00 g/s. What is the temperature difference across (through) the bottom of the pan?

2123

Strategy

Conduction through the aluminum is the primary method of heat transfer here, and so we use the equation for the rate of heat transfer and solve for the temperature difference.

$$[\text{latex size="2"}] \boldsymbol{Q} = \frac{d}{kA} \Delta T$$

2127

Solution

1. Identify the knowns and convert them to the SI units.

The thickness of the pan, the area of the pan, and the thermal conductivity,

2. Calculate the necessary heat of vaporization of 1 g of water:

3. Calculate the rate of heat transfer given that 1 g of water melts in one second:

4. Insert the knowns into the equation and solve for the temperature difference:

$$[\text{latex size="2"}] \boldsymbol{Q} = \frac{d}{kA} \Delta T$$

$$[\text{latex size="2"}] \boldsymbol{Q} = \frac{0.800 \times 10^{-3} \text{ m}}{220 \text{ J/s} \cdot \text{m}^2} \cdot (1.54 \times 10^{-2} \text{ m}^2) \Delta T$$

2135

Discussion

The value for the heat transfer is typical for an electric stove. This value gives a remarkably small temperature difference between the stove and the pan. Consider that the stove burner is red hot while the inside of the pan is nearly because of its contact with boiling water. This contact effectively cools the bottom of the pan in spite of its proximity to the very hot stove burner. Aluminum is such a good conductor that it only takes this small temperature difference to produce a heat transfer of 2.26 kW into the pan.

2142

2143

2144

2145

2146

2147

Conduction is caused by the random motion of atoms and molecules. As such, it is an ineffective mechanism for heat transport over macroscopic distances and short time distances. Take, for example, the temperature on the Earth, which would be unbearably cold during the night and extremely hot during the day if heat transport in the atmosphere was to be only through conduction. In another example, car engines would overheat unless there was a more efficient way to remove excess heat from the pistons.

2148

2149

1: How does the rate of heat transfer by conduction change when all spatial dimensions are doubled?

2150

Summary

– Heat conduction is the transfer of heat between two objects in direct contact with each other.

– The rate of heat transfer (energy per unit time) is proportional to the temperature difference and the contact area and inversely proportional to the distance between the objects:

$$Q = \frac{d}{kA} \Delta T$$

2155

Meta:

2156

2157

2158

2159

URL: <http://pressbooks-dev.oer.hawaii.edu/collegephysics/chapter/14-5-conduction/>

WARC Path:

s3://commoncrawl/crawl-data/CC-MAIN-2019-04/segments/1547583658844.27/warc/CC-MAIN-20190117062012-20190117084012-00486.warc.gz

2160
2161

A sample document lynx output

2162

Example 1: Calculating Heat Transfer Through Conduction: Conduction Rate Through an Ice Box

2163

2164

A Styrofoam ice box has a total area of 0.950 m² and walls with an average thickness of 2.50 cm. The box contains ice, water, and canned beverages at 0°C. The inside of the box is kept cold by melting ice. How much ice melts in one day if the ice box is kept in the trunk of a car at 35.0°C?

2165

2166

2167

Strategy

2168

2169

2170

2171

2172

2173

This question involves both heat for a phase change (melting of ice) and the transfer of heat by conduction. To find the amount of ice melted, we must find the net heat transferred. This value can be obtained by calculating the rate of heat transfer by conduction and multiplying by time.

2174

Solution

2175

2176

2177

2178

2179

2180

2181

2182

2183

2184

2185

2186

2187

2188

2189

2190

2191

2192

2193

1. Identify the knowns.

$$A = 0.950 \text{ m}^2; d = 2.50 \text{ cm} = 0.0250 \text{ m}; T_1 = 0^\circ\text{C}; T_2 = 35.0^\circ\text{C}; t = 1 \text{ day} = 24 \text{ hours} = 86,400 \text{ s}.$$

2. Identify the unknowns. We need to solve for the mass of the ice, m . We will also need to solve for the net heat transferred to melt the ice, Q .

3. Determine which equations to use. The rate of heat transfer by conduction is given by

$$\frac{Q}{t} = \frac{kA(T_2 - T_1)}{d}.$$

4. The heat is used to melt the ice: $Q = mL_f$.

5. Insert the known values:

$$\frac{Q}{t} = \frac{(0.010 \text{ J/s}) \cdot (0.950 \text{ m}^2) \cdot (35.0^\circ\text{C} - 0^\circ\text{C})}{0.0250 \text{ m}} = 13.3 \text{ J/s}.$$

6. Multiply the rate of heat transfer by the time ($1 \text{ day} = 86,400 \text{ s}$):

$$Q = (Q/t)t = (13.3 \text{ J/s})(86,400 \text{ s}) = 1.15 \times 10^6 \text{ J}.$$

7. Set this equal to the heat transferred to melt the ice: $Q = mL_f$. Solve for the mass m :

$$m = \frac{Q}{L_f} = \frac{1.15 \times 10^6 \text{ J}}{334 \times 10^3 \text{ J/kg}} = 3.44 \text{ kg}.$$

Discussion

2194

2195

2196

The result of 3.44 kg, or about 7.6 lbs, seems about right, based on experience. You might expect to use about a 4 kg (7-10 lb) bag of ice per day. A little extra ice is required if you add any warm food or beverages.

2197

Inspecting the conductivities in Table 3 shows that Styrofoam is a very poor conductor and thus a good insulator. Other good insulators include fiberglass, wool, and goose-down feathers. Like Styrofoam, these all incorporate many small pockets of air, taking advantage of air's poor thermal conductivity.

2198

Substance Thermal conductivity

2199

k (J/s·m·°C)

2200

Silver 420

2201

Copper 390

2202

Gold 318

2203

Aluminum 220

2204

Steel iron 80

2205

Steel (stainless) 14

2206

Ice 2.2

2207

Glass (average) 0.84

2208

Concrete brick 0.84

2209

Water 0.6

2210

Fatty tissue (without blood) 0.2

2211

Asbestos 0.16

2212

Plasterboard 0.16

2213

Wood 0.08–0.16

Snow (dry) 0.10

Cork 0.042

Glass wool 0.042

2214
 2215 Wool 0.04
 2216 Down feathers 0.025
 2217 Air 0.023
 2218 Styrofoam 0.010

2219 Table 3. Thermal Conductivities of Common Substances¹

2220
 2221 A combination of material and thickness is often manipulated to develop good insulators—the
 2222 smaller the conductivity \boldsymbol{k} and the larger the thickness \boldsymbol{d} , the
 2223 better. The ratio of $\boldsymbol{d/k}$ will thus be large for a good insulator. The ratio \boldsymbol{R}
 2224 is called the \boldsymbol{R} factor. The rate of conductive heat transfer is
 2225 inversely proportional to \boldsymbol{R} . The larger the value of \boldsymbol{R} , the better
 2226 the insulation. \boldsymbol{R} factors are most commonly quoted for household insulation,
 2227 refrigerators, and the like—unfortunately, it is still in non-metric units of $\text{ft}^2 \cdot ^\circ\text{F} \cdot \text{h} / \text{Btu}$, although
 2228 the unit usually goes unstated (1 British thermal unit [Btu] is the amount of energy needed to
 2229 change the temperature of 1.0 lb of water by 1.0°F). A couple of representative values are an \boldsymbol{R}
 2230 factor of 11 for 3.5-in-thick fiberglass batts (pieces) of insulation and an \boldsymbol{R}
 2231 factor of 19 for 6.5-in-thick fiberglass batts. Walls are usually insulated with
 2232 3.5-in batts, while ceilings are usually insulated with 6.5-in batts. In cold climates, thicker
 2233 batts may be used in ceilings and walls.

2234 The figure shows two thick rectangular pieces of fiberglass batt lying one upon the other.

2235 Figure 4. The fiberglass batt is used for insulation of walls and ceilings to prevent heat transfer
 2236 between the inside of the building and the outside environment.

2237 Note that in Table 3, the best thermal conductors—silver, copper, gold, and aluminum—are also the
 2238 best electrical conductors, again related to the density of free electrons in them. Cooking
 2239 utensils are typically made from good conductors.

2240 Example 2: Calculating the Temperature Difference Maintained by a Heat Transfer: Conduction
 2241 Through an Aluminum Pan

2242 Water is boiling in an aluminum pan placed on an electrical element on a stovetop. The sauce pan
 2243 has a bottom that is 0.800 cm thick and 14.0 cm in diameter. The boiling water is evaporating at
 2244 the rate of 1.00 g/s. What is the temperature difference across (through) the bottom of the pan?

2245 Strategy

2246 Conduction through the aluminum is the primary method of heat transfer here, and so we use the
 2247 equation for the rate of heat transfer and solve for the temperature difference[.]
 2248
$$\boldsymbol{T_2 - T_1} = \frac{\boldsymbol{Q}}{\boldsymbol{A} \cdot \boldsymbol{t}} = \frac{\boldsymbol{Q}}{\boldsymbol{\pi(0.14/2)^2} \cdot \boldsymbol{t}} = \frac{\boldsymbol{Q}}{\boldsymbol{1.54 \times 10^{-2} \cdot t}}$$

2249
 2250 Solution

- 2251 1. Identify the knowns and convert them to the SI units.
 2252 The thickness of the pan, $\boldsymbol{d} = 0.800 \text{ cm} = 8.0 \times 10^{-3} \text{ m}$,
 2253 the area of the pan, $\boldsymbol{A} = \pi(0.14/2)^2 \text{ m}^2 = 1.54 \times 10^{-2} \text{ m}^2$,
 2254 and the thermal conductivity, $\boldsymbol{k = 220 \text{ J/s} \cdot \text{m}^{-1}}$.
 2255
- 2256 2. Calculate the necessary heat of vaporization of 1 g of water:

$$\boldsymbol{Q = mL_v} = (1.00 \times 10^{-3} \text{ kg})(2256 \times 10^3 \text{ J/kg}) = 2256 \text{ J}$$
.
- 2257 3. Calculate the rate of heat transfer given that 1 g of water melts in one second:

$$\boldsymbol{Q/t} = 2.26 \text{ J/s}$$
.
- 2258 4. Insert the knowns into the equation and solve for the temperature difference:

$$\boldsymbol{T_2 - T_1} = \frac{\boldsymbol{Q}}{\boldsymbol{A} \cdot \boldsymbol{t}} = \frac{(2256 \text{ J/s})}{(1.54 \times 10^{-2} \text{ m}^2) \cdot (8.0 \times 10^{-3} \text{ m})} = 5.33^\circ\text{C}$$
.

2259 Discussion

2260 The value for the heat transfer $\boldsymbol{Q/t} = 2.26 \text{ J/s}$ is
 2261 typical for an electric stove. This value gives a remarkably small temperature difference
 2262 between the stove and the pan. Consider that the stove burner is red hot while the inside of the
 2263 pan is nearly 100°C because of its contact with boiling water.

2268

2269

2270 This contact effectively cools the bottom of the pan in spite of its proximity to the very hot stove
 2271 burner. Aluminum is such a good conductor that it only takes this small temperature difference
 2272 to produce a heat transfer of 2.26 kW into the pan.

2273

2274 Conduction is caused by the random motion of atoms and molecules. As such, it is an ineffective
 2275 mechanism for heat transport over macroscopic distances and short time distances. Take, for
 2276 example, the temperature on the Earth, which would be unbearably cold during the night and
 2277 extremely hot during the day if heat transport in the atmosphere was to be only through
 2278 conduction. In another example, car engines would overheat unless there was a more efficient
 2279 way to remove excess heat from the pistons.

2280

2281

Check Your Understanding

2282

2283

1: How does the rate of heat transfer by conduction change when all spatial dimensions are doubled
 ?

2284

2285

Summary

2286

2287

2288

2289

2290

2291

2292

2293

2294

2295

2296

2297

2298

2299

2300

2301

2302

2303

2304

2305

2306

2307

2308

2309

2310

2311

2312

2313

2314

2315

2316

2317

2318

2319

2320

2321

* Heat conduction is the transfer of heat between two objects in direct contact with each other.

* The rate of heat transfer $\boldsymbol{Q/t}$ (energy per unit time) is proportional to the temperature difference $\boldsymbol{T_2-T_1}$ and the contact area \boldsymbol{A} and inversely proportional to the distance \boldsymbol{d} between the objects:

$$\boldsymbol{\frac{Q}{t} = \frac{kA(T_2-T_1)}{d}}$$

Meta:

URL:

<http://pressbooks-dev.oer.hawaii.edu/collegephysics/chapter/14-5-conduction/>

WARC Path:

crawl-data/CC-MAIN-2019-04/segments/1547583658844.27/warc/CC-MAIN-20190117062012-20190117084012-00486.warc.gz

A.4 HYPER-PARAMETERS

For phase 1 training, we trained a transformer model on a token horizon of 9 trillion tokens. We used a sequence length of 8192 and global batch size of 768 (6291456 tokens per batch). we used a peak learning rate of 6×10^{-4} , and warmup over 8.3 billion toknes; we used cosine learning rate decay with a minimum value equal to 1% of the peak value, and weight decay of 0.1. We use AdamW optimizer (Loshchilov & Hutter, 2017) with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$, and a gradient clipping threshold of 1.0.

We pre-train our model using Megatron-LM⁷; we rely on Transformer Engine⁸ for FP8 support. We use 8-way tensor model parallelism (Shoeybi et al., 2020) with sequence parallelism (Korthikanti et al., 2022) for additional memory savings, and 768-way data parallelism with optimizer state distributed over the data-parallel replicas (Rajbhandari et al., 2020). We trained the Nemotron-T 8B transformer model on 2048 NVIDIA H100 GPUs.

In Phase 2 training, annealing experiments were conducted with total token counts of 100 billion and 300 billion. We employed a linear learning rate decay schedule with no warmup phase, using an initial learning rate of 2×10^{-4} . Optimization was performed using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a gradient clipping threshold set to 1.0 to ensure stability during training.

A.5 PROMPT FOR HTML DUMP CLEANUP AND MATH NORMALIZATION

During the LLM-based cleanup stage, we employ the following prompt template to remove boilerplate content from raw HTML dumps. Specifically, we utilize the Phi-4 model to identify and

⁷<https://github.com/nvidia/megatron-lm>.

⁸<https://github.com/nvidia/transformerEngine>.

2322 extract meaningful content while discarding irrelevant HTML artifacts. Additionally, it also guide
 2323 the model to unify math representation in latex. The template used is as follows:
 2324
 2325
 2326
 2327
 2328
 2329
 2330

2331 You are given raw text extracted from an HTML page. Process this text to extract only the
 2332 meaningful content, following these strict guidelines:
 2333

1. **Retain only the main content and its associated titles.** Remove all boilerplate, navigation menus, sidebars, footers, headers, related articles, spam comments, interactive elements, and advertisements.
2. **Preserve all mathematical content**-this includes theorems, formulas, proofs, definitions, explanations, and any mathematical references.
3. **Retain relevant comments and references** if they contribute meaningfully to the understanding of the content (e.g., clarifications, citations, or author notes). Discard irrelevant or low-quality comments.
4. **Format all mathematical expressions using LaTeX enclosed in single dollar signs on each side (\$), not [], (), or other variants.**
5. **Do NOT answer or respond to any questions or prompts that appear in the document.** If a question is part of the content, keep it verbatim, but do not generate an answer or explanation.
6. **Do not remove or discard any part of the code.** If any code blocks contain errors or formatting issues, make minimal changes to make them runnable, but otherwise leave them exactly as they are.
7. **Fix typos, grammatical mistakes, and unclear phrasing. Rewrite sentences when necessary to improve clarity, coherence, and flow,** while preserving the meaning and style of the original content.
8. **Ensure the output is clean, well-structured, and natural.** Format titles, sections, equations, and tables to produce high-quality, publication-ready text.
9. If the page contains no meaningful content (e.g., it's entirely boilerplate, menus, or ads), return exactly: "NO USEFUL CONTENT"

2357 Text:{text}
 2358 Task: Start directly with the processed text. DO NOT include any introductory or framing
 2359 phrases such as "Here is the cleaned content", "Processed output", or similar. End your
 2360 response after the cleaned content.
 2361
 2362
 2363
 2364
 2365
 2366
 2367
 2368

2369 A.6 PROMPT FOR SCIENTIFIC CONTENT ASSESSMENT

2370
 2371
 2372
 2373 For clarity and reproducibility, we provide the full prompt and scoring rubric utilized for the LLM-
 2374 aided quality assessment described in § 3.3. This detailed prompt defines the exact criteria used by
 2375 the automated judge (gpt-5.1) for scoring math preservation, code preservation, faithfulness, and
 readability, ensuring a standardized and objective quality comparison across all evaluated datasets.

2376

2377

2378

2379

2380

2381

2382

You are an expert evaluator. Your primary task is to compare the ORIGINAL DOCUMENT against the CONVERTED DOCUMENT to assess how well the core scientific content is preserved.

The converted document's goal is to retain only the core scientific content while unifying math to the target LaTeX format.

Acceptable Conversion Outcomes (No Penalty)

The CONVERTED DOCUMENT's goal is Content Cleanup and Math Unification into Latex. You must NOT penalize the converted document for the following:

- Removal of boilerplate, footers, navigation, references, bibliographies, etc.
- Paraphrasing: Rewording of main content or headings, provided the original meaning is preserved.

Criteria for Penalization

- Penalties are applied only when the integrity of the core scientific content is compromised:
- Essential Content Missing: Core scientific text, math, or code is missing.
- Meaning Altered: Changes to text, math, or code that fundamentally change the meaning or alter program behavior.

You must evaluate 4 dimensions using this scoring scale:

- 0 = Not preserved / severely corrupted
- 1 = Partially preserved / major issues
- 2 = Mostly preserved / minor issues
- 3 = Perfectly preserved
- N/A = Category not applicable because the ORIGINAL contains no content of that type

Definitions:

- **Math Preservation (0–3 or “N/A”):** Evaluate the correctness AND retaining of the math equations and expressions. Note that original math in web can appear in various forms such as:
 - MathJax / KaTeX
 - MathML
 - SVG/PNG equation images
 - Inline symbolic expressions in text

All math must be converted to proper LaTeX in the CONVERTED DOCUMENT and retained with its meaning intact.

Do NOT penalize:

- harmless formatting differences (whitespace, line breaks, equivalent LaTeX forms, etc)
- conversion from any math format (MathML, KaTeX, images, etc.) into LaTeX

Penalize when:

- math equations are missing, incomplete, replaced with prose, or altered in meaning.
- MathML, MathJax, or other math formats are not retained or not converted to LaTeX

2430

- Non-standard LaTeX or unnecessary commands are used that could misrepresent the math (e.g., `\displaystyle` in inline math)

2431

- IMPORTANT: If the original contains multiple equations and the converted document keeps only one, or removes mathematical structure, the score MUST be 1 or lower.

2432

- Score 0 only when math equations are stripped, or severely corrupted.

2433

- **Code Preservation (0–3 or “N/A”):** correctness, completeness, syntax, indentation, parsability, and structural fidelity.
 - Minor whitespace changes or equivalent code rewrites are **NOT** penalized when **functionality is preserved**.
 - **Penalize only** when missing main code content or changes that **alter program behavior or functionality**.

2434

- **Faithfulness (0–3):** This dimension assesses the overall integrity and inclusion of the core scientific content.
 - **Penalize Only If:**
 - * **Main content is missing.**
 - * **Meaning is altered** (e.g., hallucinations, fabricated steps, meaning-changing rewrites).
 - * **Scientific integrity is compromised.**
 - **No Penalty For:**
 - * **Missing boilerplate, navigation, references, bibliographies, etc.**
 - * **Correcting corrupted math or code fragments to their intended meaning.**
 - * **Minor changes in non-essential sections.**
 - * **Paraphrased headings/content when scientific integrity is preserved.**

2435

- **Readability (0–3):** This score evaluates the overall clarity, organization, and coherence of the converted document, including:
 - Logical structure and organization
 - Clear, descriptive section headings
 - Proper paragraphing and spacing
 - Smooth flow and coherence
 - Absence of jarring formatting or fragmentation
 - Whether the text reads like a clean, human-written explanation
 - **Scoring guide:**
 - * 3 = Highly readable (well-structured, coherent, clearly sectioned; polished and easy to follow)
 - * 2 = Mostly readable (minor structural or coherence flaws)
 - * 1 = Hard to read (poor structure, missing context, unclear or disorganized)
 - * 0 = Very poor readability (fragmented, confusing, incoherent, minimal structure)
 - **Readability does NOT affect correctness of content, but reflects presentation quality.**

2436

2437

2438

2439

2440

2441

2442

2443

2444

2445

2446

2447

2448

2449

2450

2451

2452

2453

2454

2455

2456

2457

2458

2459

2460

2461

2462

2463

2464

2465

2466

2467

2468

2469

2470

2471

2472

2473

2474

2475

2476

2477

2478

2479

2480

2481

2482

2483

Rules:

- Use “N/A” ONLY if the ORIGINAL document contains no math or no code, respectively.
- Output must be STRICT JSON. No commentary before or after.
- The “notes” list should contain only **meaningful issues**, written as short natural language strings. **Do NOT include trivial formatting changes or minor spacing differences.**

2484
 2485 • Include meaningful notes. If no meaningful issues, output [].
 2486 • Do NOT escape JSON with backticks.
 2487

2488 Here is an example of the EXACT output format you must produce:

```
2489     {
  2490         "math": 3,
  2491         "code": "N/A",
  2492         "faithfulness": 3,
  2493         "readability": 3,
  2494         "notes": [
  2495             "Missing the 'Related Work' section from the original document",
  2496             "Paragraph order slightly altered in Section 2"
  2497         ]
  2498     }
```

2499 Here are the documents to evaluate:

2500 ORIGINAL DOCUMENT:

```
2501     {original_text}
```

2503 CONVERTED DOCUMENT:

```
2505     {text}
```

2508 A.7 NEMOTRON-CC-MATH CORPUS STATISTICS BEFORE QUALITY FILTERING

2510 To ensure full transparency of our data construction pipeline, we report the scale of the raw corpus
 2511 prior to applying the quality-based filtering and deduplication steps described in §2.2 and §2.3,
 2512 respectively. Table 5 summarizes the number of documents and total tokens associated with each
 2513 quality score assigned during the initial Common Crawl extraction stage. Following the practice
 2514 established by the FineMath dataset, documents assigned scores of 1 and 2—which correspond to
 2515 the lowest-quality portions of the corpus—were removed before subsequent processing.

2516 Score	2517 # Documents	2517 # Tokens (B)
2518 Score 1	35,171,234	78.71
2519 Score 2	68,120,438	125.96
2520 Score 3	64,171,676	92.99
2521 Score 4	48,312,962	56.79
2522 Score 5	227,127	0.29

2523 Table 5: Corpus statistics prior to quality filtering.

2526 A.8 DATA MIXTURES USED DURING PRE-TRAINING EXPERIMENTS.

2528 To evaluate the value of our data, we setup a pretraining experiment. We used the same mixture as
 2529 used in NVIDIA et al. (2025). The data mixture spans eight broad content categories: web crawl,
 2530 mathematics, Wikipedia, code, academic publications, high quality crawl subset (Crawl++), multi-
 2531 lingual corpora, and synthetic instruction-style datasets. The Crawl++ category aggregates curated
 2532 web-derived sources such as OpenWebText, BigScience, and Reddit. The multilingual component
 2533 covers nine languages: Spanish, German, French, Italian, Portuguese, Chinese, Japanese, Korean,
 2534 and Russian. To construct the mixtures, NVIDIA et al. (2025) applied uniform weighting within
 2535 datasets of the same quality tier, and they assigned greater weight to datasets of higher quality.

2536 Following NVIDIA et al. (2025), we adopt a phased pretraining strategy. Phase 1 emphasizes data
 2537 diversity by leveraging a broad and heterogeneous mixture of sources. In contrast, Phases 2 pri-
 2538 marily focus on higher-quality datasets, such as Wikipedia and academic corpora, to refine model

2538

2539

2540

2541

2542

2543

2544

2545

2546

2547

2548

2549

2550

2551

2552

2553

2554

2555

2556

2557

2558

2559

2560

2561

2562

2563

2564

2565

2566

2567

2568

2569

2570

2571

2572

2573

2574

2575

2576

2577

2578

2579

2580

2581

2582

2583

2584

2585

2586

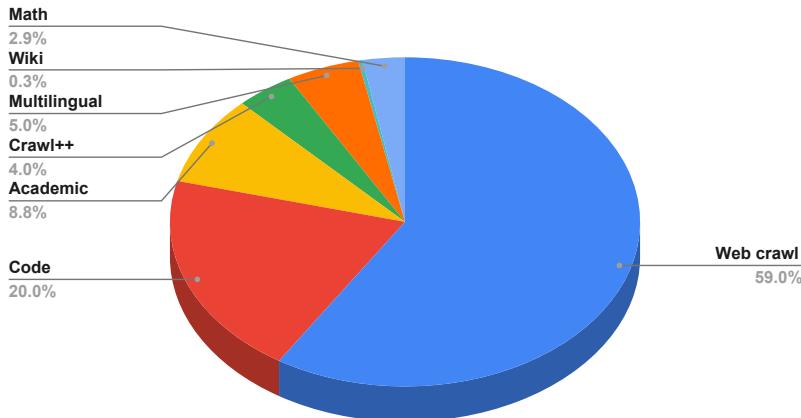
2587

2588

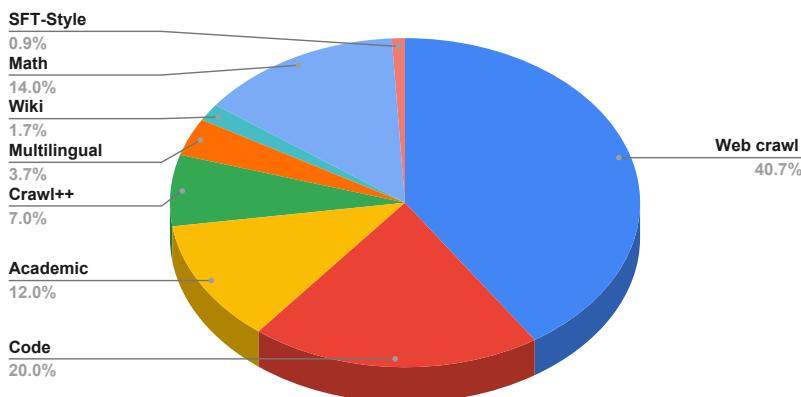
2589

2590

2591



(a) Phase 1 data mixture.



(b) Phase 2 data mixture.

Figure 4: Data mixtures for each phase of pretraining experiments presented in Table 2.

performance. The data mixtures used in each phase 1 and phase 2 are shown in Figure 4. We begin by pretraining a Nemotron-T 8B transformer model using Phase 1 mixture for a total of 9 trillion tokens. To assess the value of each of the math datasets, we then conduct a series of annealing experiments using the phase 2 mixture as a base. In each variant, we substitute the math dataset with a target dataset under evaluation, assigning it a fixed weight of 30%. The remaining 70% of the mixture is rebalanced proportionally among the other data sources to maintain a consistent total. Table 2 show the results for model trained on an additional 100 and 300 billion token budget.