# Deterministic Monotone Min-Plus Product and Convolution

**Ce Jin**
University of California Berkeley, CA, USA

**Jaewoo Park**
University of California San Diego, La Jolla, CA, USA

**Barna Saha**
University of California San Diego, La Jolla, CA, USA

**Yinzhan Xu**
University of California San Diego, La Jolla, CA, USA

## Abstract

The *Monotone Min-Plus Product* problem is a useful primitive that has seen many algorithmic applications over the past decade. It also generalizes various other structured Min-Plus products studied in the literature, such as Bounded Difference Min-Plus Product and Bounded Integer Min-Plus Product. In this problem, we are given two $n \times n$ integer matrices $A$ and $B$, where each row of $B$ is a monotone non-decreasing sequence of integers from $\{1, \ldots, n\}$, and the goal is to compute their Min-Plus product, defined as the $n \times n$ matrix $C$ with $C_{i,j} = \min_k\{A_{i,k} + B_{k,j}\}$. The fastest known algorithm for this task [Chi, Duan, Xie, and Zhang, STOC'22] runs in $n^{(\omega+3)/2+o(1)} = \mathcal{O}(n^{2.686})$ time, significantly improving over the brute-force cubic algorithm. However, its main disadvantage is that it requires *randomization*, which is then inherited by all downstream applications.

Our main result is a *deterministic* algorithm for Monotone Min-Plus product with the same running time $n^{(\omega+3)/2+o(1)} = \mathcal{O}(n^{2.686})$ as its randomized counterpart, improving upon the previous deterministic bound $\mathcal{O}(n^{2.875})$ [Gu, Polak, Vassilevska Williams, and Xu, ICALP'21]. Our derandomization also applies to previously studied extensions and variants (e.g., [Dürr, IPL'23]), including rectangular matrices, bounded range $[n^\mu]$, and column-monotone matrices. As an immediate consequence, we derandomize state-of-the-art algorithms for multiple problems, including Language Edit Distance, RNA Folding, Optimum Stack Generation, unweighted Tree Edit Distance, Batched Range Mode, and Approximate All-Pairs Shortest Paths.

Our techniques also yield a deterministic algorithm for the *Monotone Min-Plus Convolution* problem that runs in $n^{1.5+o(1)}$ time, nearly matching the best-known randomized time complexity $\widetilde{\mathcal{O}}(n^{1.5})$ [Chi, Duan, Xie, and Zhang, STOC'22]. This algorithm can be used to derandomize state-of-the-art algorithms for Jumbled Indexing for binary strings and several variants of Knapsack.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Min-plus product, min-plus convolution, monotone matrices, fine-grained complexity, deterministic algorithms

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2026.119

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2605.07150`

**Funding** *Ce Jin*: Supported by the Miller Research Fellowship at the Miller Institute for Basic Research in Science, UC Berkeley.
*Jaewoo Park*: Supported by NSF HDR TRIPODS Phase II grant 2217058 (EnCORE Institute).
*Barna Saha*: Supported by NSF HDR TRIPODS Phase II grant 2217058 (EnCORE Institute).
*Yinzhan Xu*: Supported by NSF HDR TRIPODS Phase II grant 2217058 (EnCORE Institute).



53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 119; pp. 119:1–119:23

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

One of the most fundamental problems in algorithm design is the All-Pairs Shortest Paths problem (APSP), which asks us to compute pairwise distances in an $n$-node weighted graph. APSP is known to be asymptotically equivalent to *Min-Plus product* [27], where we are given two $n \times n$ matrices $A$ and $B$ and need to compute an $n \times n$ matrix $A \star B$ such that

$$(A \star B)_{i,j} := \min_{1 \le k \le n} \{A_{i,k} + B_{k,j}\}.$$

The fastest known algorithm for these problems runs in $n^3/2^{\Omega(\sqrt{\log n})}$ time, due to Williams [29], and further improving this running time remains a major open problem. In fact, a popular hypothesis in fine-grained complexity is that APSP, or equivalently Min-Plus product, does not admit an $\mathcal{O}(n^{3-\varepsilon})$-time algorithm for any $\varepsilon > 0$; see [26] for a survey on fine-grained complexity.

Despite the difficulty of designing faster algorithms for Min-Plus product, researchers have identified several structured cases where it admits truly subcubic-time algorithms. For instance, *bounded integer Min-Plus product*, where both matrices have entries from $[M]$[1] for some integer $M$, can be computed in $\widehat{\mathcal{O}}(Mn^{\omega})$ time [2].[2,3] A generalization of bounded integer Min-Plus product is *bounded difference Min-Plus product*, where adjacent entries in the input matrices differ by at most $M$. Bringmann, Grandoni, Saha, and Vassilevska Williams [7] first studied this setting and gave an $\mathcal{O}(n^{2.825})$-time randomized algorithm and an $\mathcal{O}(n^{2.861})$-time deterministic algorithm for the case $M = \mathcal{O}(1)$. They used bounded difference Min-Plus product as a tool to obtain the first subcubic-time algorithms for Language Edit Distance, RNA Folding, and Optimum Stack Generation.

An important setting that further generalizes bounded difference Min-Plus product, and is the focus of this paper, is *monotone Min-Plus product*; see [19] for a discussion of this generalization. An $n \times n$ matrix $B$ is called row-monotone if

- for every $1 \le i, j \le n$, we have $B_{i,j} \in [n]$;
- for every $1 \le i \le n$ and $1 \le j < n$, we have $B_{i,j} \le B_{i,j+1}$.

Column-monotone matrices are defined analogously. The task of computing the Min-Plus product between an arbitrary integer matrix $A$ and a row-monotone matrix $B$ is called the monotone Min-Plus product problem. The importance of monotone Min-Plus product is reflected in its wide range of applications, including Language Edit Distance [7], RNA Folding [7], Optimum Stack Generation [7], Batched Range Mode [28, 19, 14], unweighted Tree Edit Distance [23, 14, 24], Single-Source Replacement Paths [19, 14], approximate APSP [13, 14, 25], and $k$-Dyck Edit Distance [17, 14].

The fastest known algorithm for monotone Min-Plus product, due to Chi, Duan, Xie, and Zhang [11], runs in $\widehat{\mathcal{O}}(n^{(3+\omega)/2}) = \mathcal{O}(n^{2.686})$ time. Unfortunately, this running time is achieved by a randomized algorithm, making the fastest algorithms for applications of monotone Min-Plus product randomized as well. More frustratingly, for many of these algorithms, the only source of randomness is the black-box call to monotone Min-Plus product; see Tables 1 and 2 for examples. Thus, prior to our work, obtaining deterministic

[1] For a positive integer $m$, $[m]$ denotes $\{1, 2, \ldots, m\}$.

[2] In this paper, we use $\widehat{\mathcal{O}}$ to hide $n^{o(1)}$ factors and $\widetilde{\mathcal{O}}$ to hide $\operatorname{poly}\log n$ factors.

[3] $\omega \le 2.372$ [1] denotes the square matrix multiplication exponent; that is, $\omega$ is the smallest constant such that the product of two $n \times n$ matrices can be computed in $\widehat{\mathcal{O}}(n^{\omega})$ arithmetic operations. More generally, we use $\omega(a, b, c)$ to denote the smallest constant such that the product of an $n^a \times n^b$ matrix and an $n^b \times n^c$ matrix can be computed in $\widehat{\mathcal{O}}(n^{\omega(a,b,c)})$ arithmetic operations.

**Table 1** Applications where the monotone Min-Plus product oracle is the only source of randomization in the fastest known randomized algorithms. Previous best deterministic running times are also listed. The $\widehat{\mathcal{O}}$-notation is omitted.

| Problem | Randomized Runtime | Deterministic Runtime |
|---|---|---|
| Language Edit Distance | $n^{(3+\omega)/2} \leq n^{2.686}$ [7, 11] | $n^{2.861}$ [7] |
| RNA Folding | $n^{(3+\omega)/2} \leq n^{2.686}$ [7, 11] | $n^{2.861}$ [7] |
| Optimum Stack Generation | $n^{(3+\omega)/2} \leq n^{2.686}$ [7, 11] | $n^{2.861}$ [7] |
| Batched Range Mode | $n^{(3+2\omega)/(3+\omega)} \leq n^{1.442}$ [14] | $n^{(18+2\omega)/(13+\omega)} \leq n^{1.480}$ [18] |
| Unweighted Tree Edit Distance | $n^{(3+\omega)/2} \leq n^{2.686}$ [24] | $n^{2.964}$ [23] |
| $k$-Dyck Edit Distance | $n + k^{4.545}$ [17, 14] | $n + k^{4.854}$ [17] |

**Table 2** Fastest known algorithms for $+2k$-approximate APSP on unweighted undirected graphs. The algorithms of [25] use the monotone Min-Plus product oracle as their only randomized component. The $\widehat{\mathcal{O}}$-notation is omitted.

| Additive error $+2k$ | Runtime for $+2k$-approximate APSP |
|---|---|
| $+2$ | $n^{2.22548}$ (randomized) [21] |
| $+4$ | $n^{2.14613}$ (deterministic) [21] |
| $+6$ | $n^{2.10260}$ (deterministic) [21] |
| $+8$ | $n^{2.07727}$ (randomized) [25] |
| $+10$ | $n^{2.06194}$ (randomized) [25] |
| $\cdots$ | $\cdots$ (randomized) [25] |

algorithms for these applications required replacing the randomized monotone Min-Plus product algorithm with a slower deterministic one, such as the $\mathcal{O}(n^{2.875})$-time algorithm of [19], or the $\mathcal{O}(n^{2.861})$-time bounded difference Min-Plus product algorithm of [7].

▶ Remark. We briefly explain how some of the randomized running times in Tables 1 and 2 are obtained. The randomized running times for Language Edit Distance, RNA Folding, and Optimum Stack Generation in Table 1 are obtained by replacing the bounded difference Min-Plus product algorithm of [7] with the monotone Min-Plus product algorithm of [11]. For every positive integer $k$, the algorithm of [25] for $+2k$-approximate APSP in Table 2 runs in $\widehat{\mathcal{O}}(n^{2+x/(k+1)})$ time, where $x$ is the solution to $1 + 2x = \omega(1 - \frac{k-1}{k+1}x, 1 - x, 1 - \frac{k-2}{k+1}x)$, and uses the monotone Min-Plus product oracle as its only source of randomness. For additive error $+2k$ with $2k \geq 8$, this is the fastest known algorithm. The work of [21] improves the running times of the $+4$- and $+6$-approximation algorithms of [25] using deterministic algorithms, and further improves the running time for $+2$-approximation using a randomized algorithm whose randomness does not come from the monotone Min-Plus product oracle.

A problem closely related to Min-Plus product is Min-Plus convolution. In the *Min-Plus convolution* problem, the input consists of two length-$n$ arrays $A = (A_1, A_2, \ldots, A_n)$ and $B = (B_1, B_2, \ldots, B_n)$, and the output is an array $A \diamond B$ indexed by $k = 2, \ldots, 2n$, where

$$(A \diamond B)_k = \min_{1 \leq i \leq k-1} \{A_i + B_{k-i}\},$$

with out-of-bounds entries set to $\infty$. The fastest known algorithm for Min-Plus convolution runs in $n^2/2^{\Theta(\sqrt{\log n})}$ time, which can be obtained by combining Williams' APSP algorithm [29] with the reduction in [3]. A popular hypothesis in fine-grained complexity is the Min-Plus convolution hypothesis, which postulates that Min-Plus convolution cannot be solved in $\mathcal{O}(n^{2-\varepsilon})$ time for any $\varepsilon > 0$; see [12, 22]. It is known that the Min-Plus convolution hypothesis implies the Min-Plus product hypothesis [3].

**Table 3** Applications of monotone Min-Plus convolution for which, after applying known derandomizations where needed, the only remaining source of randomization is the monotone Min-Plus convolution oracle. The table also lists the previously best-known deterministic running times. For Knapsack problems, $w_{\max}$ and $p_{\max}$ bound item weights and profits, respectively; $W$ is the capacity and $OPT$ is the optimal profit. The $\widetilde{\mathcal{O}}$-notation is omitted.

| Problem | Randomized Runtime | Deterministic Runtime |
|---|---|---|
| Jumbled Indexing for Binary Strings | $n^{1.5}$ [9, 11] | $n^{1.864}$ [9] |
| Unbounded Knapsack | $n + w_{\max}\sqrt{p_{\max}}$ [6] | $n + (p_{\max} + w_{\max})^{1.864}$ [5] |
| 0-1 Knapsack | $n + W\sqrt{OPT}$ [6] | $n + W \cdot OPT^{0.937}$ [5, 8] |
| Weakly-Approximate Unbounded Knapsack | $n + 1/\varepsilon^{1.5}$ [5] | $n + 1/\varepsilon^{1.864}$ [5] |

Similar to Min-Plus product, Min-Plus convolution admits significantly faster algorithms when the input is structured. A length-$n$ array $A$ is called monotone if

- for every $1 \le i \le n$, we have $A_i \in [n]$;
- for every $1 \le i < n$, we have $A_i \le A_{i+1}$.

When both input arrays are monotone, Chan and Lewenstein [9] gave an $\mathcal{O}(n^{1.859})$-time randomized algorithm and an $\mathcal{O}(n^{1.864})$-time deterministic algorithm. Chi, Duan, Xie, and Zhang [11] improved the randomized running time to $\widetilde{\mathcal{O}}(n^{1.5})$. Bringmann, Dürr, and Polak [6] generalized this result to a randomized $\widetilde{\mathcal{O}}(n^{1+\mu/2})$-time algorithm for arrays with entries in $[n^\mu]$ and with only one of the input arrays being monotone.

Monotone Min-Plus convolution also has a wide range of applications, including Jumbled Indexing [9], Knapsack problems with bounded weights and profits [5, 6], and weak approximation schemes for Knapsack problems [5, 10].[4] Due to the gap between the best-known randomized and deterministic algorithms for monotone Min-Plus convolution, the best-known running times for all these applications are achieved by randomized algorithms. For some applications, the monotone Min-Plus convolution subroutine is the only remaining source of randomness; see Table 3.

▶ Remark. We explain how some of the running times in Table 3 are obtained. The randomized running time for Jumbled Indexing for Binary Strings follows by replacing the monotone Min-Plus convolution algorithm in [9] with the faster randomized algorithm of [11]. For Unbounded Knapsack, the randomized $\widetilde{\mathcal{O}}(n + (w_{\max} + p_{\max})^{1.5})$-time algorithm of Bringmann and Cassis [5], combined with the rectangular and $[n^\mu]$-bounded generalization of [6], yields the refined bounds $\widetilde{\mathcal{O}}(n + w_{\max}\sqrt{p_{\max}})$ and $\widetilde{\mathcal{O}}(n + p_{\max}\sqrt{w_{\max}})$.

For 0-1 Knapsack, the randomized $\widetilde{\mathcal{O}}(n + W\sqrt{OPT})$ bound follows from [6, Theorem 12]. We state this bound, rather than their stronger $\widetilde{\mathcal{O}}(n + W\sqrt{p_{\max}})$ bound, because it is the one to which our derandomization applies. The deterministic 0-1 Knapsack bound shown in the table is not explicitly stated in the literature. The 0-1 Knapsack algorithm of [5] has another source of randomness, namely the random partitioning technique originating from [4, 12]. This additional source of randomness was recently derandomized by [8]; see [8, Appendix A] for a deterministic reduction from 0-1 Knapsack to Min-Plus convolution in [12]. The same derandomization also applies to the random partitioning step used in [5]. More specifically, if monotone Min-Plus convolution with $[n^\mu]$-bounded inputs can be solved deterministically in $\widetilde{\mathcal{O}}(n^{1+\alpha\mu})$ time, then the results of [8, 5] imply a deterministic 0-1 Knapsack algorithm

[4] These Knapsack results use Max-Plus convolution instead of Min-Plus convolution, but the two are equivalent up to a sign change.

running in $\widetilde{\mathcal{O}}(n + W \cdot OPT^{\alpha})$ time. The previously known deterministic algorithm for $[n^{\mu}]$-bounded monotone Min-Plus convolution can be obtained using the deterministic $\mathcal{O}(n^{1.864})$-time algorithm for the $[n]$-bounded case [9] as a black box, together with blocking and balancing (details omitted). This yields a deterministic $\widetilde{\mathcal{O}}(n^{1+2\mu/(4-1.864)}) = \widetilde{\mathcal{O}}(n^{1+0.937\mu})$-time algorithm for $[n^{\mu}]$-bounded monotone Min-Plus convolution, and therefore yields the deterministic $\widetilde{\mathcal{O}}(n + W \cdot OPT^{0.937})$ bound shown in the table.

## 1.1 Our Results

We close the gap between the deterministic and randomized running times for monotone Min-Plus product and monotone Min-Plus convolution. Our work is inspired by the randomized algorithm of Chi, Duan, Xie, and Zhang [11]. However, our presentation is more streamlined due to several reductions. Moreover, our algorithm for monotone Min-Plus product applies to rectangular matrices and to the case where the entries of $B$ lie in an arbitrary range $[n^{\mu}]$, rather than only in $[n]$. This matches the extensions studied in the randomized setting by [14, 17, 25], which generalized the framework of [11].

▶ **Theorem 1.** *There is a deterministic algorithm that, given an $n^a \times n^b$ integer matrix $A$ and an $n^b \times n^c$ row-monotone matrix $B$ with entries in $[n^{\mu}]$, computes their Min-Plus product in time $\widehat{\mathcal{O}}(n^{a+c} + n^{b+c} + n^{(a+b+\mu+\omega(a,b,c))/2})$.*

Note that the terms $n^{a+c}$ and $n^{b+c}$ above are necessary due to input and output sizes. As in [11], our algorithm also extends to the case where $B$ is column-monotone. Owing to our more streamlined algorithm, this extension is more modular than in prior approaches.

▶ **Theorem 2.** *There is a deterministic algorithm that, given an $n^a \times n^b$ integer matrix $A$ and an $n^b \times n^c$ column-monotone matrix $B$ with entries in $[n^{\mu}]$, computes their Min-Plus product in time $\widehat{\mathcal{O}}\left(n^{a+b} + n^{b+c} + n^{(a+c+\mu+\omega(a,b,c))/2}\right)$.*

Very recently, Fischer [15] showed, unless the APSP hypothesis fails, there is no $\mathcal{O}(n^{2.5-\varepsilon})$-time algorithm for column-monotone Min-Plus product for any $\varepsilon > 0$ when $a = b = c = \mu = 1$. Thus, if $\omega = 2$, our algorithm is near-optimal in this case under the APSP hypothesis.

As in [11], our techniques also extend to monotone Min-Plus convolution. Our algorithm for monotone Min-Plus convolution applies when the array entries lie in $[n^{\mu}]$. Its running time essentially matches the best-known randomized running time [11, 6].

▶ **Theorem 3.** *There is a deterministic algorithm that, given two length-$n$ monotone arrays $A$ and $B$, both with entries in $[n^{\mu}]$, computes their Min-Plus convolution in time $\widehat{\mathcal{O}}\left(n^{1+\mu/2}\right)$.*

Using the reduction of [6, Theorem 8] from the case where both arrays are monotone to the case where only one array is monotone, our algorithm also applies when only one of the two arrays is monotone.

Our algorithms directly derandomize the best randomized algorithms for many problems.

▶ **Corollary 4.** *All problems in Tables 1 and 3, as well as $+2k$-approximate APSP for $2k \geq 8$ in Table 2, can be solved by deterministic algorithms with the listed randomized running times, up to an $n^{o(1)}$ factor.*

In fact, we can derandomize the $+2$-approximate APSP algorithm of [21] shown in Table 2. This algorithm does not use monotone Min-Plus product, so its derandomization is not an immediate consequence of our algorithms; nevertheless, it can be derandomized using standard techniques. A proof can be found in the full version.

## 2 Technical Overview

In this technical overview, we focus on row-monotone Min-Plus product in the parameter regime originally considered by [11], where both input matrices have dimension $n \times n$ and entries from $[n]$. The extensions to rectangular matrices, larger entry ranges, column-monotone matrices, and monotone Min-Plus convolution are handled in the main body.

Our algorithm is a derandomization of the algorithm of [11]. The only source of randomization in their algorithm is to pick a random prime $Q$ from some appropriate range. Hence, it may appear straightforward to derandomize their algorithm because there is now a standard recipe for derandomizing the choice of a prime (see, e.g., [9, 16]): Instead of choosing a prime from a relatively large range, one can take $Q$ to be the product of several smaller primes from a smaller range, which are found one by one. For each such prime, one can afford to deterministically enumerate all primes in the smaller range and choose one that works. However, this approach faces several difficulties when applied to the algorithm of [11].

**A.** The intuition for why [11] needs a random prime $Q$ is the following. There is an implicit set $S$ of roughly $n^3$ nonzero integers, and we want only a small number of integers in $S$ to be divisible by $Q$; call this quantity $f_Q$. When $Q$ is a random prime from $[M]$, the expected value of $f_Q$ can be upper bounded by roughly $n^3/M$. If we aim to use the standard approach to derandomize the choice of $Q$, we would keep finding primes $p_i$ from a smaller range $[R]$, and maintain $Q = \prod_{i=1}^{t} p_i$, where $t$ is the current number of primes we have determined. For each additional prime, we expect $f_Q$ to drop by a factor of roughly $R$. In theory, one can show that such a prime exists, and even a random prime likely works. However, in order for a deterministic algorithm to pick $p_t$, it would naively need to compute the value of $f_Q$ for each choice of $p_t$. Unfortunately, there does not seem to be an efficient algorithm for this task, which marks the first difficulty in applying this derandomization approach.

**B.** In addition to the previous role, the prime $Q$ in [11] also serves another role: it is used to separate an integer $x$ into a high-order part ($\lfloor x/Q \rfloor$) and a low-order part ($x \bmod Q$). If we use the standard approach to derandomize the choice of $Q$ by multiplying small primes one by one, the high-order parts and low-order parts of the integers keep changing, making it difficult to derandomize the algorithm.

### Standard Reductions

To address Item B, we separate the two roles played by $Q$. As it turns out, it suffices to pick an arbitrary integer from an appropriate range to separate the integers into high-order parts and low-order parts. Let $M = \Theta(n^{(3-\omega)/2})$ be such an integer. By standard reductions, it suffices to solve the following problem.

▶ **Problem 1.** *Let $A, B, C$ be $n \times n$ integer matrices with nonnegative entries bounded by $\mathcal{O}(n)$, where both $B$ and $C$ are row-monotone. In addition, all entries $x$ of the matrices satisfy $x \bmod M \le M/10$. For every $i, j \in [n]$, determine whether there exists $k \in [n]$ such that $A_{i,k} + B_{k,j} = C_{i,j}$.*

At a high level, the reduction has two steps. First, by recursively halving the entries of $A$ and $B$, the product of the halved instance gives an approximation to each entry of $A \star B$: if $C'$ is the product of the halved instance, then each true value $(A \star B)_{i,j}$ must be one of $2C'_{i,j}$, $2C'_{i,j} + 1$, or $2C'_{i,j} + 2$. Thus, a verifier for candidate values can recover the full product by testing these three candidates. Second, using a standard residue-shifting reduction similar to [11], we transform the verification instance into constantly many promised instances in which the relevant residues of $A$, $B$, and $C$ lie in a small interval modulo $M$.

These standard reductions were also present in the algorithm of [11], appearing as a recursive step. By explicitly applying the standard reductions first, we move the recursive step outside the main algorithm, making the algorithm more streamlined. In addition, this helps unify the algorithms for the row-monotone and column-monotone cases. As it turns out, the column-monotone case can also be reduced to a version of Problem 1, where for every $i, k$, we need to determine the existence of such a $j$; see Section 5 for details.

### Main algorithm

Define $A^{\text{high}}_{i,k} = \lfloor A_{i,k}/M \rfloor$, and define $B^{\text{high}}$ and $C^{\text{high}}$ analogously. The algorithm first finds a modulus $Q$ satisfying $M \le Q \le Mn^{o(1)}$; we defer the additional properties required of $Q$, as well as the description of how to find it, until later in the overview. Then the algorithm computes the following quantities:

- $s_{i,j}$, the number of $k \in [n]$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$;
- $s'_{i,j}$, the number of $k \in [n]$ such that $A_{i,k}+B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$.

It is not difficult to see that $A_{i,k} + B_{k,j} = C_{i,j}$ if and only if $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} = C^{\text{high}}_{i,j}$. Hence, for each pair $(i, j)$, there exists a witness $k$ if and only if $s_{i,j} > s'_{i,j}$. Therefore, it suffices to compute $s_{i,j}$ and $s'_{i,j}$ for every pair $(i, j)$.

Using an approach similar to [2], one can compute all values $s_{i,j}$ in $\widehat{\mathcal{O}}(Qn^{\omega}) = \widehat{\mathcal{O}}(n^{(3+\omega)/2})$ time. To discuss how to compute $s'_{i,j}$, we define segments, similarly to [11], and a special type of segment called active segments. Let $\ell_{\max} := \lceil \log(M/20) \rceil$.

▶ **Definition 5.** *For $0 \le \ell \le \ell_{\max}$, a level-$\ell$ segment is a tuple $(i, k, [j_0, j_1])$ such that, for every $j_0 \le j \le j_1$, we have $\lfloor B_{k,j_0}/2^{\ell} \rfloor = \lfloor B_{k,j}/2^{\ell} \rfloor$ and $\lfloor C_{i,j_0}/2^{\ell} \rfloor = \lfloor C_{i,j}/2^{\ell} \rfloor$, and $[j_0, j_1]$ cannot be further extended.*

▶ **Definition 6.** *For $0 \le \ell \le \ell_{\max}$, a level-$\ell$ segment $(i, k, [j_0, j_1])$ is called active with respect to $Q$ if $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} - C^{\text{high}}_{i,j_0} \neq 0$ and $A_{i,k} + B_{k,j_0} - C_{i,j_0} \bmod Q \in [-4 \cdot 2^{\ell}, 4 \cdot 2^{\ell}]$. Let $S_{\ell}(Q)$ be the set of active level-$\ell$ segments.*

Since $B$ and $C$ are row-monotone, the number of level-$\ell$ segments is $\mathcal{O}(n^3/2^{\ell})$.

We will find a modulus $Q$ so that, for every $0 \le \ell \le \ell_{\max}$, $|S_{\ell}(Q)| \le \widehat{\mathcal{O}}(n^3/Q)$. For now, assume that we have such a modulus $Q$. We show how to compute $s'_{i,j}$ under this assumption.

To see why active segments help compute $s'_{i,j}$, consider a triple $(i, j, k)$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$; that is, $k$ contributes one to $s'_{i,j}$. Let $(i, k, [j_0, j_1])$ be the unique level-0 segment where $j \in [j_0, j_1]$. It is not difficult to see that this segment is active. Hence, we proceed by computing $S_0(Q)$:

- First, we find $S_{\ell_{\max}}(Q)$ by enumerating all level-$\ell_{\max}$ segments and checking whether each segment is active. The number of level-$\ell_{\max}$ segments is $\mathcal{O}(n^3/2^{\ell_{\max}}) = \mathcal{O}(n^{(3+\omega)/2})$. Hence, this part takes $\widetilde{\mathcal{O}}(n^{(3+\omega)/2})$ time.
- Suppose we have computed the set $S_{\ell+1}(Q)$ of active level-$(\ell+1)$ segments. For every segment $L \in S_{\ell+1}(Q)$, we enumerate all level-$\ell$ segments contained in $L$ and check whether they are active. Here, a segment $(i, k, [j_0, j_1])$ is contained in another segment $(i', k', [j'_0, j'_1])$ if $i = i'$, $k = k'$, and $[j_0, j_1] \subseteq [j'_0, j'_1]$. It can be shown that every segment in $S_{\ell}(Q)$ is contained in some segment in $S_{\ell+1}(Q)$, so this procedure correctly computes $S_{\ell}(Q)$. Moreover, each level-$(\ell+1)$ segment contains only a constant number of level-$\ell$ segments. By the assumption $|S_{\ell}(Q)| \le \widehat{\mathcal{O}}(n^3/Q)$ for every $\ell$, this step takes $\widehat{\mathcal{O}}(n^3/Q) = \widehat{\mathcal{O}}(n^{(3+\omega)/2})$ time.

Recall that the segments in $S_0(Q)$ contain all triples $(i,j,k)$ that can contribute to $s'_{i,j}$. For every level-0 segment $(i,k,[j_0,j_1]) \in S_0(Q)$, the values $B_{k,j}$ are constant over $j \in [j_0,j_1]$, and the values $C_{i,j}$ are also constant over this interval. Hence, either all triples $(i,k,j)$ with $j \in [j_0,j_1]$ contribute to $s'_{i,j}$, or none of them do. We can therefore use a simple data structure to compute all values $s'_{i,j}$ from $S_0(Q)$ in $\widetilde{\mathcal{O}}(|S_0(Q)|) = \widehat{\mathcal{O}}(n^{(3+\omega)/2})$ time.

### Finding a Good Modulus

It remains to find a modulus $Q \in [M, Mn^{o(1)}]$ such that $|S_\ell(Q)| \le \widehat{\mathcal{O}}(n^3/Q)$ for every $0 \le \ell \le \ell_{\max}$. To do so, consider the following multiset of numbers for every level $\ell$:

$$\mathcal{X}_\ell := \{A_{i,k}+B_{k,j_0}-C_{i,j_0}-s : (i,k,[j_0,j_1]) \text{ is a level-}\ell \text{ segment}, s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]\} \cap (\mathbb{Z}\setminus\{0\}),$$

and let $X_{\ell,Q}$ denote the number of elements in $\mathcal{X}_\ell$ that are divisible by $Q$. Clearly, $|\mathcal{X}_\ell| \le \mathcal{O}(n^3/2^\ell)\cdot\mathcal{O}(2^\ell) = \mathcal{O}(n^3)$. When $Q \ge M$ (and hence $Q > 4\cdot 2^\ell$), $X_{\ell,Q}$ is exactly the number of level-$\ell$ segments $(i,k,[j_0,j_1])$ such that $A_{i,k}+B_{k,j_0}-C_{i,j_0} \bmod Q \in [-4\cdot 2^\ell, 4\cdot 2^\ell]$ and $A_{i,k}+B_{k,j_0}-C_{i,j_0} \notin [-4\cdot 2^\ell, 4\cdot 2^\ell]$. Using the assumptions in Problem 1, it is not difficult to show that one can equivalently replace the second condition with $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} \ne C^{\text{high}}_{i,j_0}$. Hence, $X_{\ell,Q}$ is exactly $|S_\ell(Q)|$.

To summarize, we have several multisets $\mathcal{X}_\ell$, each containing $\mathcal{O}(n^3)$ nonzero integers, and we need to find a modulus $Q$ such that, for every $\ell$, the number of elements of $\mathcal{X}_\ell$ divisible by $Q$ is at most $\widehat{\mathcal{O}}(n^3/Q)$. This is exactly the setting where the standard recipe of constructing $Q$ as a product of smaller primes applies. More precisely, we construct $Q$ as the product of primes $p_1, p_2, \ldots, p_t$ from $[R/2, R]$, for some relatively small $R = n^{o(1)}$. Let $P$ be the set of primes in $[R/2,R]$. At each step, we find an additional prime $p_i \in P$ by enumerating all primes in $P$ and choosing a suitable one. We stop once the product of the chosen primes exceeds $M$. By a standard probability analysis, there exist primes $p_1, p_2, \ldots, p_t \in P$ such that if $Q_t = \prod_{i=1}^t p_i$, then $X_{\ell,Q_t}$ is bounded by roughly $n^3/R^t$ for every $\ell$. Since $R^t \approx Q_t$, this gives $X_{\ell,Q_t} \lesssim n^3/Q_t$ for every $\ell$, as desired. Hence, it remains to find these suitable primes deterministically.

In this overview, we illustrate our approach by showing how to find the first prime $p \in P$ such that $X_{\ell,p} = \widetilde{\mathcal{O}}(n^3/R)$ for every $\ell$. The same idea extends easily to finding the remaining primes. Since $R = n^{o(1)}$, we can afford to enumerate all primes in $P$ and choose one that works. However, even when $p$ is fixed, it is unclear how to compute $X_{\ell,p}$ efficiently: the straightforward approach requires enumerating all level-$\ell$ segments, which takes $\mathcal{O}(n^3/2^\ell)$ time. When $\ell$ is small, this bound can be as high as $\mathcal{O}(n^3)$. This reflects the first difficulty mentioned in Item A, although the setting here is already somewhat different.

Instead of computing $X_{\ell,p}$, we take a more indirect approach. Let

$$\mathcal{Y}_\ell := \{A_{i,k}+B_{k,j_0}-C_{i,j_0}-s : (i,k,[j_0,j_1]) \text{ is a level-}\ell \text{ segment}, s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]\}.$$

This multiset has the same definition as $\mathcal{X}_\ell$, except that we do not remove the zero elements. Let $Y_{\ell,p}$ denote the number of elements in $\mathcal{Y}_\ell$ that are divisible by $p$. As it turns out, one can compute $Y_{\ell,p}$ efficiently in $\widehat{\mathcal{O}}(2^\ell n^\omega) = \widehat{\mathcal{O}}(Mn^\omega)$ time, following the high-level idea of [2]. Next, observe that $Y_{\ell,p} - X_{\ell,p}$ is exactly the number of pairs $((i,k,[j_0,j_1]),s)$ such that $A_{i,k}+B_{k,j_0}-C_{i,j_0}-s=0$, where $(i,k,[j_0,j_1])$ is a level-$\ell$ segment and $s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]$. In particular, $Y_{\ell,p} - X_{\ell,p}$ does not depend on $p$; denote this quantity by $Z_\ell$.

In our algorithm, we compute $Y^*_\ell := \min_{p\in P} Y_{\ell,p}$ for every $\ell$, and then choose a prime $p_0 \in P$ minimizing $\max_\ell\{Y_{\ell,p_0} - Y^*_\ell\}$. This prime $p_0$ can be found deterministically in $Mn^{\omega+o(1)}$ time, since this procedure only uses the computable values $Y_{\ell,p}$, rather than the values $X_{\ell,p}$. Note that $Y^*_\ell - Z_\ell$ is exactly $X^*_\ell := \min_{p\in P} X_{\ell,p}$. Hence, $p_0$ minimizes

$$\max_{0\le\ell\le\ell_{\max}}\{Y_{\ell,p} - Y^*_\ell\} = \max_{0\le\ell\le\ell_{\max}}\{X_{\ell,p} - X^*_\ell\}.$$

By a standard probabilistic argument, $X^*_\ell \le \widetilde{\mathcal{O}}(n^3/R)$ for every $\ell$. Thus, $p_0$ minimizes $\max_{0\le\ell\le\ell_{\max}}\{X_{\ell,p}\}$ up to a $\widetilde{\mathcal{O}}(n^3/R)$ additive error. Again by a standard probabilistic argument,

$$\min_{p\in P}\max_{0\le\ell\le\ell_{\max}}\{X_{\ell,p}\} \le \widetilde{\mathcal{O}}(n^3/R).$$

Therefore, $X_{\ell,p_0} \le \widetilde{\mathcal{O}}(n^3/R)$ for every $\ell$, as desired.

## 3 Preliminaries

### Notations

For integers $a$ and $M \ge 1$, we write $a \bmod M$ for the unique integer $b \in \{0, 1, \ldots, M-1\}$ such that $a \equiv b \pmod M$. For a positive integer $x$, we write $[x] := \{1, 2, \ldots, x\}$. For integers $x \le y$, we write $[x, y] := \{x, x+1, \ldots, y\}$.

### Min-Plus Product

Let $A$ be an $n^a \times n^b$ integer matrix and let $B$ be an $n^b \times n^c$ integer matrix. The *Min-Plus product* of $A$ and $B$ is the $n^a \times n^c$ matrix $A \star B$ defined by

$$(A \star B)_{i,j} := \min_{1\le k\le n^b}\{A_{i,k} + B_{k,j}\} \qquad \text{for } (i,j) \in [n^a] \times [n^c].$$

### Min-Plus Convolution

Let $A$ and $B$ be arrays of length $n$ indexed from 1. The *Min-Plus convolution* of $A$ and $B$ is an array $A \diamond B$ indexed by $2, \ldots, 2n$, defined by

$$(A \diamond B)_k := \min_{\substack{1\le i\le k-1\\ 1\le i, k-i\le n}}\{A_i + B_{k-i}\}, \qquad \text{for } k \in [2, 2n].$$

### Monotonicity

An $n \times m$ matrix $A$ is *row-monotone* if for every $i \in [n]$ and $j \in [m-1]$, $A_{i,j} \le A_{i,j+1}$.[5] It is *column-monotone* if for every $i \in [n-1]$ and $j \in [m]$, $A_{i,j} \le A_{i+1,j}$. An array $A$ of length $n$ is *monotone* if for every $i \in [n-1]$, $A_i \le A_{i+1}$.

### Rectangular Matrix Multiplication

For $a, b, c \ge 0$, let $\omega(a, b, c)$ denote an exponent such that the product of an $n^a \times n^b$ matrix and an $n^b \times n^c$ matrix can be computed in $\widehat{\mathcal{O}}(n^{\omega(a,b,c)})$ arithmetic operations. The best known upper bound is $\omega(1, 1, 1) < 2.372$ [1]. It is known that $\omega(a, b, c) = \omega(a, c, b)$ [20].

### Polynomial Multiplication

Let $P, Q \in \mathbb{Z}[x, y]$ be bivariate polynomials whose degree in $x$ is at most $d_x$ and whose degree in $y$ is at most $d_y$. We can compute $P + Q$ and $P - Q$ in $\mathcal{O}(d_x d_y)$ time, and compute $P \cdot Q$ in $\widetilde{\mathcal{O}}(d_x d_y)$ time using FFT. Let $P(x) \in (\mathbb{Z}[x])^{n^a \times n^b}$ and $Q(x) \in (\mathbb{Z}[x])^{n^b \times n^c}$ be polynomial matrices whose entries have degree at most $D$. Then the matrix product $P(x) \cdot Q(x)$ can be computed in $\widehat{\mathcal{O}}(D\, n^{\omega(a,b,c)})$ time.

[5] Unlike in the introduction, where boundedness is included for readability, monotonicity here only refers to the ordering condition; boundedness is stated separately.

### Prime Number Theorem

Let $\pi(x)$ denote the number of primes at most $x$. The *prime number theorem* states that $\pi(x) = x/\log x \pm o(x/\log x)$. As a corollary, for any sufficiently large integer $R$, the number of primes in the interval $[R/2, R]$ is $\pi(R) - \pi(R/2-1) = \Theta(R/\log R)$.

## 4 Deterministic Monotone Min-Plus Product

In this section, we present a deterministic algorithm for row-monotone Min-Plus product, whose definition we recall below.

▶ **Problem 2.** *Let $A$ be an $n^a \times n^b$ integer matrix. Let $B$ be an $n^b \times n^c$ row-monotone integer matrix with entries in $[n^\mu]$. Given $(A, B)$ as input, compute $A \star B$.*

### 4.1 Reduction to a Verification Problem

We start with a standard reduction that transforms the problem into a simpler form. Consider the following verification problem.

▶ **Problem 3.** *Let $A$ be an $n^a \times n^b$ integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $B$ be an $n^b \times n^c$ row-monotone integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $C$ be an $n^a \times n^c$ row-monotone integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Given $(A, B, C)$ as input, for every $(i, j) \in [n^a] \times [n^c]$, decide whether there exists $k \in [n^b]$ such that $A_{i,k} + B_{k,j} = C_{i,j}$.*

▶ **Lemma 7.** *If there is a deterministic algorithm for Problem 3 that runs in $T(n^a, n^b, n^c; n^\mu)$ time, then there is a deterministic algorithm for Problem 2 that runs in $\widetilde{\mathcal{O}}(T(n^a, n^b, n^c; n^\mu))$ time.*

**Proof.** First, by a simple observation of [11], we may assume that all entries of $A$ are nonnegative integers bounded by $\mathcal{O}(n^\mu)$: for each $i$, subtract $\delta_i$ from every entry in the $i$-th row of $A$, where $\delta_i$ is the minimum value in that row. Then any entry of $A$ greater than $2n^\mu$ can be set to $2n^\mu + 1$, since it can never participate in an optimal solution. After computing the Min-Plus product of the modified matrix $A$ and $B$, we simply add $\delta_i$ back to the $i$-th row of the result.

Let $A'_{i,j} := \lfloor A_{i,j}/2 \rfloor$ and $B'_{i,j} := \lfloor B_{i,j}/2 \rfloor$. Flooring preserves row-monotonicity, so the rows of $B'$ are also row-monotone. We compute $C' := A' \star B'$ recursively. The base case occurs when both $A$ and $B$ are zero matrices, in which case we can compute $A \star B$ trivially. Now define three $n^a \times n^c$ candidate matrices $C^{(0)}, C^{(1)}, C^{(2)}$ by $C^{(s)}_{i,j} := 2C'_{i,j} + s$ for $s \in \{0, 1, 2\}$. Since the rows of $B'$ are monotone, one can prove that the rows of $C'$ are also monotone. Therefore, the rows of each $C^{(s)}$ are monotone as well, and we can run the algorithm for Problem 3 on the instance $(A, B, C^{(s)})$ for each $s \in \{0, 1, 2\}$.

Since $(x-1)/2 \le \lfloor x/2 \rfloor \le x/2$ and $C' = A' \star B'$, we have

$$C'_{i,j} = \min_k \left\{ \left\lfloor \frac{A_{i,k}}{2} \right\rfloor + \left\lfloor \frac{B_{k,j}}{2} \right\rfloor \right\} \le \min_k \left\{ \frac{A_{i,k} + B_{k,j}}{2} \right\} = \frac{(A \star B)_{i,j}}{2},$$

and

$$C'_{i,j} = \min_k \left\{ \left\lfloor \frac{A_{i,k}}{2} \right\rfloor + \left\lfloor \frac{B_{k,j}}{2} \right\rfloor \right\} \ge \min_k \left\{ \frac{A_{i,k} + B_{k,j} - 2}{2} \right\} = \frac{(A \star B)_{i,j} - 2}{2}.$$

Together, these inequalities imply $(A \star B)_{i,j} - 2C'_{i,j} \in \{0,1,2\}$. Thus, for each fixed pair $(i,j)$, there exists at least one $s \in \{0,1,2\}$ such that $C^{(s)}_{i,j} = (A \star B)_{i,j}$, and we can recover $(A \star B)_{i,j}$ by taking the minimum value among $C^{(s)}_{i,j}$ for which the verifier accepts $(i,j)$ on $(A, B, C^{(s)})$. Moreover, if the verifier accepts $(i,j)$ on $(A, B, C^{(s)})$ with witness $k$, then $C^{(s)}_{i,j} = A_{i,k} + B_{k,j} \geq (A \star B)_{i,j}$. Therefore, taking the minimum accepted value among the $C^{(s)}_{i,j}$ returns exactly $(A \star B)_{i,j}$.

For the running time, we make $\mathcal{O}(1)$ calls to the $T(n^a, n^b, n^c; n^\mu)$-time verifier per recursion level. The recursion depth is $\mathcal{O}(\log(n^\mu)) = \mathcal{O}(\log n)$. The additional costs of forming $A'$, $B'$, and the matrices $C^{(s)}$ are dominated by $T(n^a, n^b, n^c; n^\mu)$. Hence, the total running time is $\widetilde{\mathcal{O}}(T(n^a, n^b, n^c; n^\mu))$. ◀

Using an approach similar to that of [11, Lemma 3.5], we further reduce Problem 3 to the following problem and focus on solving it efficiently. Since the reduction is standard, we defer the proof to the full version.

▶ **Problem 4.** *Let $(A, B, C)$ be an instance of Problem 3, and let $M$ be a positive integer that is a multiple of 100 and satisfies $M = \mathcal{O}(n^\mu)$. Additionally, assume that for every $i \in [n^a]$, $k \in [n^b]$, and $j \in [n^c]$, we have $A_{i,k} \bmod M$, $B_{k,j} \bmod M$, $C_{i,j} \bmod M \leq M/10$. Given $(A, B, C)$ as input, for every $(i,j) \in [n^a] \times [n^c]$, decide whether there exists $k \in [n^b]$ such that $A_{i,k} + B_{k,j} = C_{i,j}$.*

▶ **Lemma 8.** *If there is a $T(n^a, n^b, n^c; n^\mu, M)$-time algorithm for Problem 4 for every such $M$, then there is an $\widetilde{\mathcal{O}}(T(n^a, n^b, n^c; n^\mu, M))$-time algorithm for Problem 3.*

## 4.2 Algorithm for Problem 4

Suppose we are given an instance $(A, B, C)$ of Problem 4 with $M = \Theta(n^d)$ for some constant $0 \leq d \leq \mu$. Define

$$A^{\text{high}}_{i,k} = \left\lfloor \frac{A_{i,k}}{M} \right\rfloor, \qquad B^{\text{high}}_{k,j} = \left\lfloor \frac{B_{k,j}}{M} \right\rfloor, \qquad C^{\text{high}}_{i,j} = \left\lfloor \frac{C_{i,j}}{M} \right\rfloor.$$

We also define

$$A^{\text{low}}_{i,k} = A_{i,k} \bmod M, \qquad B^{\text{low}}_{k,j} = B_{k,j} \bmod M, \qquad C^{\text{low}}_{i,j} = C_{i,j} \bmod M.$$

The high-level outline of our algorithm is as follows:

1. Find a "good" modulus $Q$ with $M \leq Q \leq Mn^{o(1)}$. The definition of "good" is deferred to Definition 16.
2. For every $(i,j) \in [n^a] \times [n^c]$, compute $s_{i,j}$, the number of $k \in [n^b]$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$.
3. For every $(i,j) \in [n^a] \times [n^c]$, compute $s'_{i,j}$, the number of $k \in [n^b]$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$.
4. For each $(i,j) \in [n^a] \times [n^c]$, output YES if and only if $s_{i,j} > s'_{i,j}$.

We first prove two simple facts that bound $|A_{i,k} + B_{k,j} - C_{i,j}|$, depending on whether $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} = C^{\text{high}}_{i,j}$, and then prove the correctness of the algorithm assuming that our subroutines are correct.

▶ **Fact 9.** *For every $(i,k,j) \in [n^a] \times [n^b] \times [n^c]$, if $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$, then $|A_{i,k} + B_{k,j} - C_{i,j}| \geq 7M/10$.*

**Proof.** We decompose $A_{i,k}+B_{k,j}-C_{i,j} = M\cdot\big(A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}-C^{\text{high}}_{i,j}\big)+\big(A^{\text{low}}_{i,k}+B^{\text{low}}_{k,j}-C^{\text{low}}_{i,j}\big)$. If $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}-C^{\text{high}}_{i,j}\neq 0$, then $|A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}-C^{\text{high}}_{i,j}|\geq 1$. By the promise of Problem 4, $|A^{\text{low}}_{i,k}+B^{\text{low}}_{k,j}-C^{\text{low}}_{i,j}|\leq |A^{\text{low}}_{i,k}|+|B^{\text{low}}_{k,j}|+|C^{\text{low}}_{i,j}|\leq 3M/10$. Therefore, using $|a+b|\geq|a|-|b|$, we have

$$\begin{aligned}\big|A_{i,k}+B_{k,j}-C_{i,j}\big| &\geq M\cdot\big|A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}-C^{\text{high}}_{i,j}\big|-\big|A^{\text{low}}_{i,k}+B^{\text{low}}_{k,j}-C^{\text{low}}_{i,j}\big|\\ &\geq M-3M/10=7M/10.\end{aligned}$$

◀

▶ **Fact 10.** *For every $(i,k,j)\in[n^a]\times[n^b]\times[n^c]$, if $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}=C^{\text{high}}_{i,j}$, then $|A_{i,k}+B_{k,j}-C_{i,j}|\leq 3M/10$.*

**Proof.** Since $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}-C^{\text{high}}_{i,j}=0$, we have $|A_{i,k}+B_{k,j}-C_{i,j}|=|A^{\text{low}}_{i,k}+B^{\text{low}}_{k,j}-C^{\text{low}}_{i,j}|\leq 3M/10$ by the same argument as in Fact 9. ◀

Recall that $s_{i,j}$ is the number of $k\in[n^b]$ such that $A_{i,k}+B_{k,j}\equiv C_{i,j} \pmod Q$, and $s'_{i,j}$ is the number of $k\in[n^b]$ such that $A_{i,k}+B_{k,j}\equiv C_{i,j} \pmod Q$ and $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}\neq C^{\text{high}}_{i,j}$. We now prove the correctness of our algorithm.

▶ **Lemma 11** (Correctness). *For every $(i,j)\in[n^a]\times[n^c]$, there exists $k\in[n^b]$ such that $A_{i,k}+B_{k,j}=C_{i,j}$ if and only if $s_{i,j}>s'_{i,j}$.*

**Proof.** Fix $(i,j)$ and suppose there exists $k\in[n^b]$ such that $A_{i,k}+B_{k,j}=C_{i,j}$. Then $A_{i,k}+B_{k,j}\equiv C_{i,j} \pmod Q$, so this $k$ contributes one to $s_{i,j}$. Moreover, under the promise of Problem 4, we have $A^{\text{low}}_{i,k}+B^{\text{low}}_{k,j}\leq 2M/10$, so adding the low parts creates no carry across a multiple of $M$. Hence, $C^{\text{high}}_{i,j}=\lfloor C_{i,j}/M\rfloor=\lfloor(A_{i,k}+B_{k,j})/M\rfloor=A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}$. Therefore, this $k$ does not contribute to $s'_{i,j}$, and so $s_{i,j}>s'_{i,j}$.

Conversely, suppose $s_{i,j}>s'_{i,j}$. Then there exists $k\in[n^b]$ such that $A_{i,k}+B_{k,j}\equiv C_{i,j} \pmod Q$ and $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j}=C^{\text{high}}_{i,j}$. By Fact 10, we have $|A_{i,k}+B_{k,j}-C_{i,j}|\leq 3M/10<Q$. Since $A_{i,k}+B_{k,j}-C_{i,j}$ is a multiple of $Q$ and has magnitude strictly less than $Q$, it must be zero. Therefore, $A_{i,k}+B_{k,j}=C_{i,j}$. ◀

In the following sections, we describe how to compute $s_{i,j}$ and $s'_{i,j}$ efficiently, and how to construct a suitable modulus $Q$. To this end, we introduce several objects used by the subroutines and their analysis. Let $\ell_{\max}$ be such that $M/20\leq 2^{\ell_{\max}}<M/10$.

▶ **Definition 12** (Segments). *For $0\leq\ell\leq\ell_{\max}$, a level-$\ell$ segment is a triple $(i,k,[j_0,j_1])$ such that, for every $j\in[j_0,j_1]$, $\lfloor B_{k,j_0}/2^\ell\rfloor=\lfloor B_{k,j}/2^\ell\rfloor$ and $\lfloor C_{i,j_0}/2^\ell\rfloor=\lfloor C_{i,j}/2^\ell\rfloor$, and $[j_0,j_1]$ cannot be extended further.*

▶ **Fact 13.** *For $0\leq\ell\leq\ell_{\max}$, the total number of level-$\ell$ segments is $\mathcal{O}(n^{a+b+\mu}/2^\ell)$.*

▶ **Definition 14** (Active segment). *For $0\leq\ell\leq\ell_{\max}$, a level-$\ell$ segment $(i,k,[j_0,j_1])$ is called* active *with respect to $Q$ if $A^{\text{high}}_{i,k}+B^{\text{high}}_{k,j_0}\neq C^{\text{high}}_{i,j_0}$ and there exists $s\in[-4\cdot 2^\ell,4\cdot 2^\ell]$ such that $A_{i,k}+B_{k,j_0}-C_{i,j_0}\equiv s \pmod Q$.*

▶ **Definition 15** (Set of active segments). *For $0\leq\ell\leq\ell_{\max}$, let $S_\ell(Q)$ be the set of all level-$\ell$ segments that are active with respect to $Q$.*

▶ **Definition 16** (Good modulus). *We say that an integer $Q$ is a* good modulus *if $M\leq Q\leq Mn^{o(1)}$ and $|S_\ell(Q)|\leq\widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$ for all $0\leq\ell\leq\ell_{\max}$.*

## 4.3 Computing $s_{i,j}$ and $s'_{i,j}$

Recall that $s_{i,j}$ is the number of indices $k \in [n^b]$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$. We compute $s_{i,j}$ for all $(i,j) \in [n^a] \times [n^c]$ using polynomial matrix multiplication. The technique is standard, so we defer the proof to the full version.

▶ **Lemma 17** (Computing $s_{i,j}$). *We can compute $s_{i,j}$ for every $(i,j) \in [n^a] \times [n^c]$ in time $\widehat{\mathcal{O}}(Qn^{\omega(a,b,c)})$.*

Recall that $s'_{i,j}$ is the number of indices $k$ such that $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$. Suppose that $Q$ is a good modulus as defined in Definition 16; we show how to find such a modulus in the next section. For simplicity, let $S_\ell$ denote the set of all active level-$\ell$ segments, suppressing the dependence on $Q$. We first prove a property used by our subroutine, and then prove two lemmas describing the procedure for computing $s'_{i,j}$.

▶ **Lemma 18.** *For all $0 \le \ell < \ell_{\max}$ and $(i,k,[j'_0,j'_1]) \in S_\ell$, the unique level-$(\ell+1)$ segment $(i,k,[j_0,j_1])$ with $[j'_0,j'_1] \subseteq [j_0,j_1]$ is active, i.e., $(i,k,[j_0,j_1]) \in S_{\ell+1}$.*

**Proof.** Fix $j \in [j_0,j_1]$. By definition of a level-$(\ell+1)$ segment, we have $\lfloor B_{k,j_0}/2^{\ell+1} \rfloor = \lfloor B_{k,j}/2^{\ell+1} \rfloor$. Hence, $|B_{k,j} - B_{k,j_0}| < 2^{\ell+1} < M/10$. Moreover, $|B^{\text{low}}_{k,j} - B^{\text{low}}_{k,j_0}| \le M/10$ by the promise of Problem 4. Hence, if $B^{\text{high}}_{k,j} \neq B^{\text{high}}_{k,j_0}$, then

$$|B_{k,j} - B_{k,j_0}| = |(B^{\text{high}}_{k,j} - B^{\text{high}}_{k,j_0})M + (B^{\text{low}}_{k,j} - B^{\text{low}}_{k,j_0})| \ge M - |B^{\text{low}}_{k,j} - B^{\text{low}}_{k,j_0}| \ge 9M/10,$$

which contradicts the previous bound. Therefore, $B^{\text{high}}_{k,j}$ is constant for $j \in [j_0,j_1]$, and by a similar argument, $C^{\text{high}}_{i,j}$ is constant for $j \in [j_0,j_1]$. Now, since $(i,k,[j'_0,j'_1]) \in S_\ell$, we have $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j'_0} - C^{\text{high}}_{i,j'_0} \neq 0$. Because $j'_0 \in [j_0,j_1]$, we conclude that $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} - C^{\text{high}}_{i,j_0} \neq 0$, satisfying the high-part condition of being active.

For the congruence condition, note that $[j'_0,j'_1] \subseteq [j_0,j_1]$ and $(i,k,[j_0,j_1])$ is a level-$(\ell+1)$ segment. Hence, $|B_{k,j_0} - B_{k,j'_0}| < 2^{\ell+1}$ and $|C_{i,j_0} - C_{i,j'_0}| < 2^{\ell+1}$, which implies $|(A_{i,k} + B_{k,j_0} - C_{i,j_0}) - (A_{i,k} + B_{k,j'_0} - C_{i,j'_0})| < 2 \cdot 2^{\ell+1}$. Since $(i,k,[j'_0,j'_1]) \in S_\ell$, we have $A_{i,k} + B_{k,j'_0} - C_{i,j'_0} \equiv s' \pmod{Q}$ for some $s' \in [-4 \cdot 2^\ell, 4 \cdot 2^\ell]$. Take

$$s = s' + (A_{i,k} + B_{k,j_0} - C_{i,j_0}) - (A_{i,k} + B_{k,j'_0} - C_{i,j'_0}).$$

Then $|s| < 4 \cdot 2^\ell + 2 \cdot 2^{\ell+1} = 4 \cdot 2^{\ell+1}$ and $(A_{i,k} + B_{k,j_0} - C_{i,j_0}) \equiv s - s' + (A_{i,k} + B_{k,j'_0} - C_{i,j'_0}) \equiv s \pmod{Q}$. Thus, the level-$(\ell+1)$ segment $(i,k,[j_0,j_1])$ is active. ◀

▶ **Lemma 19** (Computing $S_0$). *We can compute $S_0$ in $\widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$ time.*

**Proof.** We first compute all level-$\ell_{\max}$ segments as follows. For each $(i,k) \in [n^a] \times [n^b]$, we use binary search on $j$ to group maximal consecutive indices on which $\lfloor B_{k,j}/2^{\ell_{\max}} \rfloor$ and $\lfloor C_{i,j}/2^{\ell_{\max}} \rfloor$ are both constant. We then enumerate all those level-$\ell_{\max}$ segments and filter the active ones to obtain $S_{\ell_{\max}}$. Next, for $\ell = \ell_{\max} - 1, \ldots, 0$, we construct $S_\ell$ from $S_{\ell+1}$ as follows. Every segment in $S_{\ell+1}$ refines into $\mathcal{O}(1)$ level-$\ell$ subsegments, since decreasing $\ell$ by 1 can split a maximal constant block into at most two blocks for $B$ and at most two blocks for $C$. We use binary search to compute these subsegments. For each resulting subsegment, we test whether it is active at level $\ell$ and, if so, insert it into $S_\ell$.

For correctness, note that every active level-$\ell$ segment is a subsegment of an active level-$(\ell+1)$ segment by Lemma 18; the correctness therefore follows by induction.

By Fact 13, computing all level-$\ell_{\max}$ segments via binary search and constructing $S_{\ell_{\max}}$ takes $\widetilde{\mathcal{O}}(n^{a+b+\mu}/2^{\ell_{\max}}) = \widetilde{\mathcal{O}}(n^{a+b+\mu}/M) = \widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$ time. For the refinement, we note that $Q$ is a good modulus, so for each $\ell$, we have $|S_\ell| = \widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$, and each active segment generates only $\mathcal{O}(1)$ subsegments. There are $\mathcal{O}(\log M) = \widetilde{\mathcal{O}}(1)$ levels, so the total time spent in this refinement step is $\widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$. ◀

▶ **Lemma 20** (Computing $s'_{i,j}$)**.** *We can compute $s'_{i,j}$ for all $(i,j) \in [n^a] \times [n^c]$ in time $\widehat{\mathcal{O}}(n^{a+b+\mu}/Q + n^{a+c})$.*

**Proof.** We first compute the set $S_0$ using Lemma 19. To compute all $s'_{i,j}$, we aggregate the contributions of active level-0 segments in $S_0$ as follows. For each fixed $i \in [n^a]$, we maintain a difference array $D_i$ of length $n^c + 1$, indexed by $1, \ldots, n^c + 1$, and initialized to zero. For each active level-0 segment $(i, k, [j_0, j_1]) \in S_0$, we test whether $A_{i,k} + B_{k,j_0} \equiv C_{i,j_0} \pmod{Q}$. If the test succeeds, we increment $D_i[j_0]$ by one and decrement $D_i[j_1 + 1]$ by one. After processing all active level-0 segments, we compute prefix sums and set $s'_{i,j} = \sum_{1 \le t \le j} D_i[t]$ for all $j = 1, \ldots, n^c$.

For correctness, fix any triple $(i, k, j)$ counted in $s'_{i,j}$. Then we have $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$. Let $(i, k, [j_0, j_1])$ be the unique level-0 segment such that $j \in [j_0, j_1]$. Since $(i, k, [j_0, j_1])$ is a level-0 segment, the values $B_{k,j}$ and $C_{i,j}$ are constant over all $j \in [j_0, j_1]$. Since $j \in [j_0, j_1]$, we have $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} = A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j} = C^{\text{high}}_{i,j_0}$ and $A_{i,k} + B_{k,j_0} \equiv A_{i,k} + B_{k,j} \equiv C_{i,j} \equiv C_{i,j_0} \pmod{Q}$. Hence, $(i, k, [j_0, j_1])$ is an active level-0 segment in $S_0$, and the algorithm performs the range update for $(i, k, [j_0, j_1])$. This update increases the reconstructed value $s'_{i,j}$ by one for this $k$ exactly when $j \in [j_0, j_1]$.

Conversely, whenever the algorithm performs a range update for $(i, k, [j_0, j_1]) \in S_0$, we have $A_{i,k} + B_{k,j_0} \equiv C_{i,j_0} \pmod{Q}$. Since $(i, k, [j_0, j_1])$ is a level-0 segment, the values $B_{k,j}$ and $C_{i,j}$ are constant over all $j \in [j_0, j_1]$. Therefore, $A_{i,k} + B_{k,j} \equiv C_{i,j} \pmod{Q}$ for every $j \in [j_0, j_1]$. Moreover, the segment is active, so $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j} \neq C^{\text{high}}_{i,j}$ for every $j \in [j_0, j_1]$, and the update contributes exactly the intended triples to $s'_{i,j}$.

For the running time, scanning all segments in $S_0$ and performing the constant-time test and update takes $\mathcal{O}(|S_0|) = \widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$. The total time to compute prefix sums is $\mathcal{O}(n^{a+c})$, since for each fixed $i$ we scan $j = 1, \ldots, n^c$ once. Together with the time to compute $S_0$, which is $\widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$ by Lemma 19, this yields the claimed running time. ◀

## 4.4 Finding a Good Modulus $Q$

We now construct a *good modulus* $Q$, namely an integer $Q$ with $M \le Q \le Mn^{o(1)}$ such that the number of active level-$\ell$ segments is $\widehat{\mathcal{O}}(n^{a+b+\mu}/Q)$ for every $0 \le \ell \le \ell_{\max}$.

Let $4 \le R \le n^{o(1)}$ be a parameter to be fixed later, and let $\mathcal{P}$ be the set of primes in $[R/2, R]$. We take $Q$ to be a product of primes $p_1, p_2, \ldots, p_T$ with $p_t \in \mathcal{P}$ for all $1 \le t \le T$. We choose these primes inductively: at step $t \ge 1$, we select a prime $p_t \in \mathcal{P}$ and multiply it into the current modulus $Q_{t-1}$, obtaining

$$Q_t := \prod_{i=1}^{t} p_i, \qquad \text{where } Q_0 := 1.$$

For the analysis, define $X_{\ell,t}$ as the number of pairs $((i, k, [j_0, j_1]), s)$, where $(i, k, [j_0, j_1])$ is a level-$\ell$ segment and $s \in [-4 \cdot 2^\ell, 4 \cdot 2^\ell]$, such that

$$A_{i,k} + B_{k,j_0} - C_{i,j_0} \equiv s \pmod{Q_t} \quad \text{and} \quad A_{i,k} + B_{k,j_0} - C_{i,j_0} \neq s.$$

Our inductive invariant for $Q_t$ is that

$$X_{\ell,t} = n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t \qquad \text{for all } 0 \le \ell \le \ell_{\max}. \tag{1}$$

Later, we choose $R$ so that $X_{\ell,T} = \widehat{\mathcal{O}}(n^{a+b+\mu}/Q_T)$. Then, by the following lemma, $Q_T$ is a good modulus.

▶ **Lemma 21.** *For every $0 \le \ell \le \ell_{\max}$, we have $|S_\ell(Q_T)| \le X_{\ell,T}$.*

**Proof.** Let $(i,k,[j_0,j_1]) \in S_\ell(Q_T)$. By definition of an active segment, there exists an integer $s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]$ such that $A_{i,k} + B_{k,j_0} - C_{i,j_0} \equiv s \pmod{Q_T}$ and $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} \neq C^{\text{high}}_{i,j_0}$. We claim that $A_{i,k} + B_{k,j_0} - C_{i,j_0} \neq s$. Indeed, since $A^{\text{high}}_{i,k} + B^{\text{high}}_{k,j_0} \neq C^{\text{high}}_{i,j_0}$, Fact 9 implies $|A_{i,k} + B_{k,j_0} - C_{i,j_0}| \ge 7M/10$. On the other hand, $|s| \le 4\cdot 2^\ell \le 4\cdot 2^{\ell_{\max}} < 4M/10 < 7M/10$, so $A_{i,k} + B_{k,j_0} - C_{i,j_0} \neq s$. Therefore, every active segment in $S_\ell(Q_T)$ yields a pair $((i,k,[j_0,j_1]),s)$ counted by $X_{\ell,T}$, and hence $|S_\ell(Q_T)| \le X_{\ell,T}$. ◀

We now explain how to choose $p_t$ so that the invariant Equation (1) continues to hold. For the base case $t = 0$, we have $Q_0 := 1$. Since $X_{\ell,0}$ counts level-$\ell$ segment–shift pairs, we trivially have $X_{\ell,0} \le n^{a+b+\mu}$ by Fact 13, and thus $Q_0$ satisfies the invariant. Now assume Equation (1) holds for $t-1$ and that $Q_{t-1}$ is fixed. For any prime $p \in \mathcal{P}$, let $X_{\ell,t}(p)$ denote the value of $X_{\ell,t}$ when $Q_t$ is set to $Q_{t-1}\cdot p$. Our goal is to find a prime $p_t \in \mathcal{P}$ such that Equation (1) holds with respect to $X_{\ell,t}(p_t)$.

First, such a prime exists; applying the following lemma inductively yields the entire sequence $p_1, \ldots, p_T$. The proof is standard, so we defer it to the full version.

▶ **Lemma 22.** *There exists $p^* \in \mathcal{P}$ such that, for every $0 \le \ell \le \ell_{\max}$,*

$$X_{\ell,t}(p^*) = X_{\ell,t-1} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right). \tag{2}$$

*Moreover, for every $0 \le \ell \le \ell_{\max}$, if $p$ is chosen uniformly at random from $\mathcal{P}$, then $\mathbb{E}_{p\sim\mathcal{P}}[X_{\ell,t}(p)] = X_{\ell,t-1}\cdot\mathcal{O}(\log n/R)$.*

We now discuss how to deterministically select these primes. For this, it is convenient to track the following additional quantities.

1. Let $Y_{\ell,t}$ denote the number of pairs $((i,k,[j_0,j_1]),s)$, where $(i,k,[j_0,j_1])$ is a level-$\ell$ segment and $s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]$, satisfying
$$A_{i,k} + B_{k,j_0} - C_{i,j_0} \equiv s \pmod{Q_t}.$$
2. Let $Z_\ell$ denote the number of pairs $((i,k,[j_0,j_1]),s)$, where $(i,k,[j_0,j_1])$ is a level-$\ell$ segment and $s \in [-4\cdot 2^\ell, 4\cdot 2^\ell]$, satisfying
$$A_{i,k} + B_{k,j_0} - C_{i,j_0} = s.$$

As with $X_{\ell,t}(p)$, let $Y_{\ell,t}(p)$ denote the value of $Y_{\ell,t}$ when the modulus is $Q_{t-1}\cdot p$.

As discussed in Section 2, we can afford to compute $Y_{\ell,t}(p)$ for all $p \in \mathcal{P}$ and $0 \le \ell \le \ell_{\max}$. Computing $Y_{\ell,t}(p)$ is essentially the same as computing $s_{i,j}$: we reduce modular counting to rectangular matrix multiplication over a polynomial ring. The only difference is that we count *segments* rather than all column indices. More formally, we have the following lemma.

▶ **Lemma 23** (Computing $Y_{\ell,t}(p)$)**.** *We can compute $Y_{\ell,t}(p)$ for all primes $p \in \mathcal{P}$ and all levels $0 \le \ell \le \ell_{\max}$ in time $\widehat{\mathcal{O}}(R^2 Q_{t-1} n^{\omega(a,b,c)})$.*

**Proof.** Fix a prime $p \in \mathcal{P}$ and a level $\ell$. We present an algorithm to compute $Y_{\ell,t}(p)$ for the modulus $Q' = Q_{t-1} \cdot p$ in $\widehat{\mathcal{O}}(Q'\, n^{\omega(a,b,c)})$ time. Since there are $\ell_{\max} = \mathcal{O}(\log M) = \mathcal{O}(\log n)$ levels, $|\mathcal{P}| = \Theta(R/\log R) = \widetilde{\mathcal{O}}(R)$, and $Q' = Q_{t-1} \cdot p \le Q_{t-1}R$, the total running time to compute $Y_{\ell,t}(p)$ for all $p \in \mathcal{P}$ and all levels is $\widehat{\mathcal{O}}(|\mathcal{P}| \cdot \ell_{\max} \cdot (RQ_{t-1})\, n^{\omega(a,b,c)}) = \widehat{\mathcal{O}}(R^2 Q_{t-1} n^{\omega(a,b,c)})$.

Recall that $Y_{\ell,t}(p)$ counts pairs $((i,k,[j_0,j_1]),s)$ where $(i,k,[j_0,j_1])$ is a level-$\ell$ segment, $s \in [-4\cdot 2^\ell,\, 4\cdot 2^\ell]$, and $A_{i,k} + B_{k,j_0} - C_{i,j_0} \equiv s \pmod{Q'}$. A level-$\ell$ segment $(i,k,[j_0,j_1])$ is a maximal interval on which both $\lfloor B_{k,j}/2^\ell \rfloor$ and $\lfloor C_{i,j}/2^\ell \rfloor$ are constant. Therefore, $[j_0,j_1]$ starts at column $j$ if and only if either $j = 1$, or at least one of $\lfloor B_{k,j}/2^\ell \rfloor$ and $\lfloor C_{i,j}/2^\ell \rfloor$ changes at $j$. Hence, it suffices to check each segment exactly once via its start column $j_0$. More concretely, define an $n^b \times n^c$ boundary indicator matrix $I^B$ by

$$I^B_{k,1} := 1, \qquad I^B_{k,j} := \mathbf{1}\big[\lfloor B_{k,j-1}/2^\ell \rfloor \neq \lfloor B_{k,j}/2^\ell \rfloor\big] \quad (j \ge 2),$$

and an $n^a \times n^c$ boundary indicator matrix $I^C$ by

$$I^C_{i,1} := 1, \qquad I^C_{i,j} := \mathbf{1}\big[\lfloor C_{i,j-1}/2^\ell \rfloor \neq \lfloor C_{i,j}/2^\ell \rfloor\big] \quad (j \ge 2).$$

Then it suffices to count, for each $(i,j) \in [n^a] \times [n^c]$, the indices $k \in [n^b]$ with $I^B_{k,j} = 1$ or $I^C_{i,j} = 1$, grouped by the residue class of $A_{i,k} + B_{k,j} - C_{i,j} \pmod{Q'}$, and then weight each residue class by the number of admissible shifts $s \in [-4\cdot 2^\ell,\, 4\cdot 2^\ell]$ in that class.

To do this, we use the polynomial matrix multiplication technique. Work over the ring $\mathcal{R} := \mathbb{F}[x]/(x^{Q'} - 1)$, where $\mathbb{F}$ is a field of sufficiently large characteristic. Construct an $n^a \times n^b$ matrix $A'$ and two $n^b \times n^c$ matrices $B'$ and $B^{\mathrm{bdry}}$ by

$$A'_{i,k}(x) := x^{A_{i,k} \bmod Q'}, \qquad B'_{k,j}(x) := x^{B_{k,j} \bmod Q'}, \qquad B^{\mathrm{bdry}}_{k,j}(x) := I^B_{k,j} \cdot x^{B_{k,j} \bmod Q'}.$$

Compute the matrix products $D^{\mathrm{all}} := A' \cdot B'$ and $D^{\mathrm{bdry}} := A' \cdot B^{\mathrm{bdry}}$ over $\mathcal{R}$ using fast rectangular matrix multiplication, and finally construct an $n^a \times n^c$ matrix $D$ defined by

$$D_{i,j}(x) := \begin{cases} D^{\mathrm{all}}_{i,j}(x), & \text{if } I^C_{i,j} = 1, \\ D^{\mathrm{bdry}}_{i,j}(x), & \text{if } I^C_{i,j} = 0. \end{cases}$$

For each $r \in \{0,\dots,Q'-1\}$ and $(i,j)$, let $U_{i,j}(r)$ be the coefficient of $x^r$ in $D_{i,j}(x)$. Then $U_{i,j}(r)$ is the number of indices $k$ such that $A_{i,k} + B_{k,j} \equiv r \pmod{Q'}$ and either $I^B_{k,j} = 1$ or $I^C_{i,j} = 1$. Next, define

$$W(r) := \big|\big\{s \in [-4\cdot 2^\ell,\, 4\cdot 2^\ell] :\ s \equiv r \pmod{Q'}\big\}\big|.$$

Note that the pair $((i,k,[j_0,j_1]),s)$ is counted in $Y_{\ell,t}(p)$ exactly when $A_{i,k} + B_{k,j_0} - C_{i,j_0} \equiv s \pmod{Q'}$. Equivalently, if $r \equiv A_{i,k} + B_{k,j_0} \pmod{Q'}$, then $s \equiv r - C_{i,j_0} \pmod{Q'}$. Thus, grouping by $r$, the contribution of a fixed $(i,j)$ to $Y_{\ell,t}(p)$ is

$$\sum_{r=0}^{Q'-1} U_{i,j}(r) \cdot W\big((r - C_{i,j}) \bmod Q'\big).$$

Therefore, summing over $i \in [n^a]$ and $j \in [n^c]$, we recover $Y_{\ell,t}(p)$.

For the running time, arithmetic in $\mathcal{R}$ reduces exponents modulo $Q'$, so the cost of the rectangular matrix product is $\widehat{\mathcal{O}}(Q'\, n^{\omega(a,b,c)})$, and we compute $\mathcal{O}(1)$ such products. Extracting $U_{i,j}(r)$ for all $(i,j)$ and $r$ costs $\mathcal{O}(Q'\, n^{a+c})$, computing $W(r)$ for all $r$ costs $\mathcal{O}(Q')$, and evaluating the triple sum costs $\mathcal{O}(Q'\, n^{a+c})$. Hence, for fixed $p$ and $\ell$ we can compute $Y_{\ell,t}(p)$ in $\widehat{\mathcal{O}}(Q'\, n^{\omega(a,b,c)})$ time. ◀

▶ **Lemma 24.** *We can find a prime $p_t \in \mathcal{P}$ satisfying Equation (1); that is, $X_{\ell,t}(p_t) = n^{a+b+\mu} \cdot \mathcal{O}(\log^2 n/R)^t$ for all $0 \le \ell \le \ell_{\max}$. Moreover, $p_t$ can be found in time $\widehat{\mathcal{O}}(R^2 Q_{t-1} n^{\omega(a,b,c)})$.*

**Proof.** We run the algorithm in Lemma 23 to compute $Y_{\ell,t}(p)$ for all primes $p \in \mathcal{P}$ and all levels $0 \le \ell \le \ell_{\max}$. For each $\ell$, define $Y^*_{\ell,t} := \min_{p\in\mathcal{P}} Y_{\ell,t}(p)$, and choose $p_t \in \mathcal{P}$ that minimizes

$$\Phi(p) := \max_{0\le\ell\le\ell_{\max}} \{Y_{\ell,t}(p) - Y^*_{\ell,t}\}.$$

For correctness, we first note that $Y^*_{\ell,t} \ge Z_\ell$. Since $X_{\ell,t}(p) = Y_{\ell,t}(p) - Z_\ell$ for all $p \in \mathcal{P}$ and $X_{\ell,t}(p) \ge 0$, we have $Y_{\ell,t}(p) \ge Z_\ell$ for every $p \in \mathcal{P}$, and thus $Y^*_{\ell,t} \ge Z_\ell$.

Now, let $p^* \in \mathcal{P}$ be a prime satisfying Equation (2), which exists by Lemma 22. Then, for every $\ell$,

$$Y_{\ell,t}(p^*) - Y^*_{\ell,t} \le Y_{\ell,t}(p^*) - Z_\ell = X_{\ell,t}(p^*) = X_{\ell,t-1} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right) = n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t,$$

where the last equality follows from the inductive hypothesis Equation (1). Therefore, $\Phi(p^*) = n^{a+b+\mu} \cdot \mathcal{O}(\log^2 n/R)^t$, and by the choice of $p_t$ minimizing $\Phi$, we obtain

$$Y_{\ell,t}(p_t) - Y^*_{\ell,t} \le \Phi(p_t) \le \Phi(p^*) \le n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t \qquad \text{for all } 0 \le \ell \le \ell_{\max}.$$

Next, define $X^*_{\ell,t} := \min_{p\in\mathcal{P}} X_{\ell,t}(p)$. By minimality, Lemma 22, and the inductive hypothesis Equation (1),

$$X^*_{\ell,t} \le \mathop{\mathbb{E}}_{p\sim\mathcal{P}}[X_{\ell,t}(p)] = X_{\ell,t-1} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right) = n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t.$$

Also, since $X_{\ell,t}(p) = Y_{\ell,t}(p) - Z_\ell$ for all $p \in \mathcal{P}$ and $Z_\ell$ is independent of $p$, we have

$$Y^*_{\ell,t} - Z_\ell = \min_{p\in\mathcal{P}}\{Y_{\ell,t}(p) - Z_\ell\} = \min_{p\in\mathcal{P}} X_{\ell,t}(p) = X^*_{\ell,t}.$$

Combining the above bounds, for every $0 \le \ell \le \ell_{\max}$ we get

$$X_{\ell,t}(p_t) = Y_{\ell,t}(p_t) - Z_\ell = \big(Y^*_{\ell,t} - Z_\ell\big) + \big(Y_{\ell,t}(p_t) - Y^*_{\ell,t}\big) \le X^*_{\ell,t} + \Phi(p_t) = n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t.$$

For the running time, computing $Y_{\ell,t}(p)$ for all $p \in \mathcal{P}$ and all $0 \le \ell \le \ell_{\max}$ via Lemma 23 dominates, giving $\widehat{\mathcal{O}}\big(R^2 Q_{t-1} n^{\omega(a,b,c)}\big)$ time. The remaining steps, namely taking minima and selecting $p_t$, take at most $\mathcal{O}(|\mathcal{P}|\ell_{\max}) = \widetilde{\mathcal{O}}(R \log n)$ additional time, which is lower order. ◀

It remains to set the parameters so that the above procedure yields a good modulus.

▶ **Fact 25.** *Let $R = 2^{\Theta\left(\sqrt{\log n/\log\log n}\right)}$ and $T = \mathcal{O}(\log n/\log R)$. Then $(\log n)^{2T} = n^{o(1)}$.*

▶ **Lemma 26** (Computing a good modulus)**.** *We can compute a good modulus $Q$ in time $\widehat{\mathcal{O}}(M n^{\omega(a,b,c)})$.*

**Proof.** By applying Lemma 24 inductively for $T$ steps, we obtain primes $p_1, \ldots, p_T \in \mathcal{P}$ and a modulus $Q_T := \prod_{t=1}^T p_t$ such that Equation (1) holds. Since $p_t \in [R/2, R]$ for all $1 \le t \le T$, we have $(R/2)^T \le Q_T \le R^T$. In particular, to ensure $Q_T \ge M$ it suffices to take

$T = \Theta(\log M/\log R) = \mathcal{O}(\log n/\log R)$. Moreover, if we take $T$ to be the *first* index such that $Q_T \geq M$, then $Q_{T-1} < M$ and hence $Q_T = Q_{T-1} \cdot p_T < M \cdot R$, so $Q_T$ overshoots $M$ by at most a factor of $R$. We set

$$R = 2^{\Theta\left(\sqrt{\log n/\log\log n}\right)}.$$

Then $R = n^{o(1)}$, and therefore $M \leq Q_T \leq M \cdot R = M \cdot n^{o(1)}$. Finally, by Fact 25, we have $(\log^2 n)^T = n^{o(1)}$, and so $(\log^2 n/R)^T = n^{o(1)}/Q_T$. Hence $Q_T$ satisfies $M \leq Q_T \leq Mn^{o(1)}$, and

$$X_{\ell,T} = n^{a+b+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^T = \frac{n^{a+b+\mu+o(1)}}{Q_T} \qquad \text{for all } 0 \leq \ell \leq \ell_{\max}.$$

Therefore, $Q_T$ is a good modulus.

For the running time, we apply Lemma 24 for $T$ steps. At step $t$, the cost of computing $p_t$ is $\widehat{\mathcal{O}}\big(R^2 Q_{t-1} n^{\omega(a,b,c)}\big)$. Since $Q_{t-1} \leq Q_{T-1} < M$, the total time is

$$\widehat{\mathcal{O}}\left(\sum_{t=1}^{T} R^2 Q_{t-1} n^{\omega(a,b,c)}\right) = \widehat{\mathcal{O}}\Big(TR^2 M\, n^{\omega(a,b,c)}\Big) = \widehat{\mathcal{O}}\Big(Mn^{\omega(a,b,c)}\Big),$$

where the last bound uses $TR^2 = n^{o(1)}$ for our choice of $R$. ◀

## 4.5 Combining the Results

▶ **Theorem 27.** *There is a deterministic $\widehat{\mathcal{O}}(Mn^{\omega(a,b,c)} + n^{a+b+\mu}/M)$-time algorithm for Problem 4.*

**Proof.** We first compute a good modulus $Q$ via Lemma 26 in time $\widehat{\mathcal{O}}(Mn^{\omega(a,b,c)})$. Next, we compute $s_{i,j}$ for all $(i,j) \in [n^a] \times [n^c]$ via Lemma 17 in time $\widehat{\mathcal{O}}(Qn^{\omega(a,b,c)}) = \widehat{\mathcal{O}}(Mn^{\omega(a,b,c)})$. Finally, we compute $s'_{i,j}$ for all $(i,j) \in [n^a] \times [n^c]$ via Lemma 20 in time $\widehat{\mathcal{O}}\big(n^{a+b+\mu}/Q + n^{a+c}\big) = \widehat{\mathcal{O}}\big(n^{a+b+\mu}/M + n^{a+c}\big)$. For each $(i,j) \in [n^a] \times [n^c]$, we output Yes if and only if $s_{i,j} > s'_{i,j}$; correctness follows from Lemma 11. The total running time is $\widehat{\mathcal{O}}(Mn^{\omega(a,b,c)} + n^{a+b+\mu}/M)$. ◀

**Proof of Theorem 1.** Our claimed running time is $\widehat{\mathcal{O}}\left(n^{a+c} + n^{b+c} + n^{(a+b+\mu+\omega(a,b,c))/2}\right)$. Let $d := (a+b+\mu-\omega(a,b,c))/2$. Since $\mu \geq 0$, we have $d \leq \mu$: indeed, $d > \mu$ would imply $\mu + \omega(a,b,c) < a+b$, contradicting the trivial lower bound $\omega(a,b,c) \geq a+b$. Furthermore, we may assume that $d \geq 0$: if $d < 0$, then $a+b+\mu < \omega(a,b,c)$. In this case, Lemma 17 of [19] gives an algorithm running in time $\widehat{\mathcal{O}}\left(n^{a+c} + n^{b+c} + n^{a+b+\mu}\right)$, which is dominated by our claimed bound. Hence, we may assume $0 \leq d \leq \mu$.

Now, by Lemmas 7 and 8, it suffices to solve Problem 4. By Theorem 27, there is a deterministic algorithm for Problem 4 that runs in time $\widehat{\mathcal{O}}(Mn^{\omega(a,b,c)} + n^{a+b+\mu}/M)$. We take $d$ as the exponent for $M$; that is, take $M = \Theta\left(n^{(a+b+\mu-\omega(a,b,c))/2}\right)$. Then the two terms balance, yielding

$$\widehat{\mathcal{O}}\left(Mn^{\omega(a,b,c)} + n^{a+b+\mu}/M\right) = \widehat{\mathcal{O}}\left(n^{(a+b+\mu+\omega(a,b,c))/2}\right),$$

which is dominated by our claimed bound. ◀

## 5 Extension to Column-Monotone Min-Plus Product

In this section, we present a deterministic algorithm for column-monotone Min-Plus product, whose definition we recall below.

▶ **Problem 5.** *Let $A$ be an $n^a \times n^b$ integer matrix. Let $B$ be an $n^b \times n^c$ column-monotone integer matrix with entries in $[n^\mu]$. Given $(A, B)$ as input, compute $A \star B$.*

We start by reducing column-monotone Min-Plus product to the following problem, which is a "rotated" version of Problem 3 (the differences have been highlighted).

▶ **Problem 3'.** *Let $A$ be an $n^a \times n^b$ integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $B$ be an $n^b \times n^c$ row-monotone integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $C$ be an $n^a \times n^c$ row-monotone integer matrix with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Given $(A, B, C)$ as input, **for every $(i, k) \in [n^a] \times [n^b]$, decide whether there exists $j \in [n^c]$** such that $A_{i,k} + B_{k,j} = C_{i,j}$.*

▶ **Theorem 28.** *There is a deterministic algorithm for Problem 3' that runs in $\widehat{\mathcal{O}}(n^{a+c} + n^{b+c} + n^{(a+b+\mu+\omega(a,b,c))/2})$ time.*

We defer the proof of Theorem 28 to the full version, as it largely repeats arguments from Section 4. Instead, we prove Theorem 2 assuming Theorem 28.

**Proof of Theorem 2 assuming Theorem 28.** Let $(A, B)$ be an instance of Problem 5. Similar to the proof of Lemma 7, we may assume that all entries of $A$ are nonnegative integers bounded by $\mathcal{O}(n^\mu)$. By an observation in [11], we may also assume that each row of $A$ is monotonically non-increasing: if there exist $i, k_1, k_2$ such that $k_1 < k_2$ and $A_{i,k_1} < A_{i,k_2}$, then $A_{i,k_1} + B_{k_1,j} < A_{i,k_2} + B_{k_2,j}$ for every $j$ by the monotonicity of $B$. Hence, replacing $A_{i,k_2}$ by $A_{i,k_1}$ does not change $A \star B$.

Now we follow the same recursive halving approach as in the proof of Lemma 7. Let $A'_{i,j} := \lfloor A_{i,j}/2 \rfloor$ and $B'_{i,j} := \lfloor B_{i,j}/2 \rfloor$. Flooring preserves monotonicity, so the columns of $B'$ are monotone and the rows of $A'$ are monotonically non-increasing. We compute $C' := A' \star B'$ recursively, with the trivial base case when both matrices are zero. Define three $n^a \times n^c$ candidate matrices by $C^{(s)}_{i,j} := 2C'_{i,j} + s$ for $s \in \{0, 1, 2\}$.

As in the proof of Lemma 7, it suffices to check, for each $s \in \{0, 1, 2\}$ and $(i, j) \in [n^a] \times [n^c]$, whether there exists $k \in [n^b]$ such that $A_{i,k} + B_{k,j} = C^{(s)}_{i,j}$. Let $W = \mathcal{O}(n^\mu)$ be the maximum entry of $A$, $B$, and $C^{(s)}$. This condition is equivalent to $(W - C^{(s)}_{i,j}) + B_{k,j} = W - A_{i,k}$. Thus, it can be solved by Problem 3' on the input $(W^{n^a \times n^c} - C^{(s)}, B^T, W^{n^a \times n^b} - A)$, where $W^{m \times p}$ denotes the $m \times p$ all-$W$ matrix. All entries are bounded by $\mathcal{O}(n^\mu)$, and both $B^T$ and $W^{n^a \times n^b} - A$ are row-monotone, so the input satisfies the requirements of Problem 3'. The dimensions are rotated as follows: the three matrices have dimensions $n^a \times n^c$, $n^c \times n^b$, and $n^a \times n^b$. Hence, by Theorem 28 and the symmetry $\omega(a, c, b) = \omega(a, b, c)$, each verifier call takes time $\widehat{\mathcal{O}}\big(n^{a+b} + n^{b+c} + n^{(a+c+\mu+\omega(a,b,c))/2}\big)$. Since there are only constantly many verifier calls per recursion level and the recursion depth is $\mathcal{O}(\log n)$, the asymptotic running time remains the same. ◀

## 6 Deterministic Monotone Min-Plus Convolution

In this section, we present a deterministic algorithm for monotone Min-Plus convolution, whose definition we recall below.

▶ **Problem 6.** *Let $A$ and $B$ be two monotone arrays of length $n$ with entries in $[n^\mu]$. Given $(A, B)$ as input, compute $A \diamond B$.*

We defer most details in this section to the full version, since they largely repeat the core arguments from Section 4.

As in Section 4, we first reduce Problem 6 to a corresponding verification problem and then to a promised version in which all entries have small residues modulo a parameter $M$.

▶ **Problem 7.** *Let $A$ and $B$ be two monotone arrays of length $n$ with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $C$ be an array of length $2n-1$ indexed from 2 to $2n$ with nonnegative entries bounded by $\mathcal{O}(n^\mu)$. Let $M$ be a positive integer that is a multiple of 100 and at most $\mathcal{O}(n^\mu)$. Additionally, assume that for every $i \in [n]$, we have $A_i \bmod M, B_i \bmod M \le M/10$, and for every $k \in [2, 2n]$, we have $C_k \bmod M \le M/10$. Given $(A, B, C)$ as input, for every $k \in [2, 2n]$, decide whether there exists $i \in [n]$ such that $k - i \in [n]$ and $A_i + B_{k-i} = C_k$.*

▶ **Lemma 29.** *If there is a $T(n; n^\mu, M)$-time deterministic algorithm for Problem 7 for any choice of $M$, then there is an $\widetilde{\mathcal{O}}(T(n; n^\mu, M))$-time deterministic algorithm for Problem 6.*

## 6.1 Algorithm for Problem 7

Suppose we are given an instance $(A, B, C)$ of Problem 7 with $M = \Theta(n^d)$ for some constant $0 \le d \le \mu$. Define $A_i^{\text{high}} = \lfloor A_i/M \rfloor$, $B_j^{\text{high}} = \lfloor B_j/M \rfloor$, and $C_k^{\text{high}} = \lfloor C_k/M \rfloor$. We also define $A_i^{\text{low}} = A_i \bmod M$, $B_j^{\text{low}} = B_j \bmod M$, and $C_k^{\text{low}} = C_k \bmod M$. The high-level outline of our algorithm is as follows:

1. Find a "good" modulus $Q$ with $M \le Q \le Mn^{o(1)}$. The definition of "good" is deferred to Definition 35.
2. For every $k \in [2, 2n]$, compute $s_k$, the number of indices $i \in [n]$ such that $k - i \in [n]$ and $A_i + B_{k-i} \equiv C_k \pmod{Q}$.
3. For every $k \in [2, 2n]$, compute $s'_k$, the number of indices $i \in [n]$ such that $k - i \in [n]$ and $A_i + B_{k-i} \equiv C_k \pmod{Q}$ and $A_i^{\text{high}} + B_{k-i}^{\text{high}} \ne C_k^{\text{high}}$.
4. For each $k \in [2, 2n]$, output YES if and only if $s_k > s'_k$.

▶ **Lemma 30** (Correctness). *For every $k \in [2, 2n]$, there exists $i \in [n]$ such that $k - i \in [n]$ and $A_i + B_{k-i} = C_k$ if and only if $s_k > s'_k$.*

Let $\ell_{\max}$ be such that $M/20 \le 2^{\ell_{\max}} < M/10$. We define segments, active segments, and the notion of a good modulus analogously to Section 4.

▶ **Definition 31** (Segments). *For $0 \le \ell \le \ell_{\max}$, a level-$\ell$ segment is a pair $([i_0, i_1], k)$ such that $[i_0, i_1] \subseteq [\max\{1, k-n\}, \min\{n, k-1\}]$, for every $i \in [i_0, i_1]$, $\lfloor A_{i_0}/2^\ell \rfloor = \lfloor A_i/2^\ell \rfloor$, and $\lfloor B_{k-i_0}/2^\ell \rfloor = \lfloor B_{k-i}/2^\ell \rfloor$, and $[i_0, i_1]$ cannot be extended further.*

▶ **Fact 32.** *For $0 \le \ell \le \ell_{\max}$, the number of level-$\ell$ segments is $\mathcal{O}(n^{1+\mu}/2^\ell)$.*

▶ **Definition 33** (Active segment). *For $0 \le \ell \le \ell_{\max}$, a level-$\ell$ segment $([i_0, i_1], k)$ is called* active *with respect to $Q$ if $A_{i_0}^{\text{high}} + B_{k-i_0}^{\text{high}} - C_k^{\text{high}} \ne 0$ and there exists $s \in [-4 \cdot 2^\ell, 4 \cdot 2^\ell]$ such that $A_{i_0} + B_{k-i_0} - C_k \equiv s \pmod{Q}$.*

▶ **Definition 34** (Set of active segments). *For $0 \le \ell \le \ell_{\max}$, let $S_\ell(Q)$ be the set of all active level-$\ell$ segments with respect to $Q$.*

▶ **Definition 35** (Good modulus). *We say that an integer $Q$ is a* good modulus *if $M \le Q \le Mn^{o(1)}$ and, for every $0 \le \ell \le \ell_{\max}$, $|S_\ell(Q)| \le \widehat{\mathcal{O}}(n^{1+\mu}/Q)$.*

Let $4 \le R \le n^{o(1)}$ be a parameter to be fixed later, and let $\mathcal{P}$ be the set of primes in $[R/2, R]$. We take $Q$ to be a product of primes $p_1, p_2, \ldots, p_T$, where $p_t \in \mathcal{P}$ for all $1 \le t \le T$. Let $Q_t$ be the modulus at step $t$. The natural invariant corresponding to Equation (1) is

$$X_{\ell,t} = n^{1+\mu} \cdot \mathcal{O}\left(\frac{\log^2 n}{R}\right)^t \qquad \text{for all } 0 \le \ell \le \ell_{\max}. \tag{3}$$

Here, $X_{\ell,t}$ counts pairs $(([i_0, i_1], k), s)$, where $([i_0, i_1], k)$ is a level-$\ell$ segment, $s \in [-4 \cdot 2^\ell, 4 \cdot 2^\ell]$, $A_{i_0} + B_{k-i_0} - C_k \equiv s \pmod{Q_t}$, and $A_{i_0} + B_{k-i_0} - C_k \neq s$. One can show that $|S_\ell(Q_T)| \le X_{\ell,T}$. Let $Y_{\ell,t}$ count pairs $(([i_0, i_1], k), s)$, where $([i_0, i_1], k)$ is a level-$\ell$ segment and $s \in [-4 \cdot 2^\ell, 4 \cdot 2^\ell]$, such that $A_{i_0} + B_{k-i_0} - C_k \equiv s \pmod{Q_t}$. For any prime $p \in \mathcal{P}$, let $X_{\ell,t}(p)$ (resp., $Y_{\ell,t}(p)$) denote the value of $X_{\ell,t}$ (resp., $Y_{\ell,t}$) when the modulus is $Q_t := Q_{t-1} \cdot p$. Fix $t$. As in Section 4.4, we compute $Y_{\ell,t}(p)$ for all primes $p \in \mathcal{P}$ and all levels $0 \le \ell \le \ell_{\max}$, and use them to find a suitable prime $p_t \in \mathcal{P}$ that satisfies Equation (3). Such a prime exists by a probabilistic argument, as in Lemma 22; we defer the details to the full version.

▶ **Lemma 36** (Computing $Y_{\ell,t}(p)$). *We can compute $Y_{\ell,t}(p)$ for all primes $p \in \mathcal{P}$ and all levels $0 \le \ell \le \ell_{\max}$ in $\widetilde{\mathcal{O}}(R^2 Q_{t-1} n)$ time.*

▶ **Lemma 37.** *We can find a prime $p_t \in \mathcal{P}$ satisfying Equation (3); that is, $X_{\ell,t}(p_t) = n^{1+\mu} \cdot \mathcal{O}(\log^2 n / R)^t$ for all $0 \le \ell \le \ell_{\max}$ in $\widetilde{\mathcal{O}}\big(R^2 Q_{t-1} n\big)$ time.*

By computing $T = \Theta(\log M / \log R)$ suitable primes with $R = 2^{\Theta\left(\sqrt{\log n / \log\log n}\right)}$, we obtain the following lemma.

▶ **Lemma 38** (Computing a good modulus). *We can compute a good modulus $Q$ in $\widehat{\mathcal{O}}(Mn)$ time.*

Let $Q$ be the good modulus obtained by this procedure. We use polynomial multiplication to compute $s_k$. The computation of $s'_k$ is analogous to that in Lemma 20.

▶ **Lemma 39** (Computing $s_k$). *We can compute $s_k$ for every $k \in [2, 2n]$ in time $\widetilde{\mathcal{O}}(Qn)$.*

▶ **Lemma 40.** *For all $0 \le \ell \le \ell_{\max} - 1$, and $([i'_0, i'_1], k) \in S_\ell(Q)$, the unique level-$(\ell+1)$ segment $([i_0, i_1], k)$ such that $[i'_0, i'_1] \subseteq [i_0, i_1]$ is active, i.e., $([i_0, i_1], k) \in S_{\ell+1}(Q)$.*

▶ **Lemma 41** (Computing $s'_k$). *We can compute $s'_k$ for all $k \in [2, 2n]$ in $\widehat{\mathcal{O}}\big(n^{1+\mu}/Q\big)$ time.*

**Proof sketch.** First, compute $S_0(Q)$ similarly to Lemma 19. Its correctness follows inductively using Lemma 40. Then compute the values $s'_k$ using $S_0(Q)$ as follows: for each segment $([i_0, i_1], k) \in S_0(Q)$, we test whether $A_{i_0} + B_{k-i_0} \equiv C_k \pmod{Q}$. If the test succeeds, we increment $s'_k$ by $i_1 - i_0 + 1$. For correctness, fix any $i \in [n]$ such that $k - i \in [n]$, $A_i + B_{k-i} \equiv C_k \pmod{Q}$, and $A_i^{\text{high}} + B_{k-i}^{\text{high}} \neq C_k^{\text{high}}$. Let $([i_0, i_1], k)$ be the unique level-0 segment containing $(i, k)$. Since $([i_0, i_1], k)$ is a level-0 segment, the values $A_i$ and $B_{k-i}$ are constant for all $i \in [i_0, i_1]$. Because $i \in [i_0, i_1]$, we have $A_{i_0}^{\text{high}} + B_{k-i_0}^{\text{high}} = A_i^{\text{high}} + B_{k-i}^{\text{high}} \neq C_k^{\text{high}}$ and $A_{i_0} + B_{k-i_0} \equiv A_i + B_{k-i} \equiv C_k \pmod{Q}$. Hence, $([i_0, i_1], k)$ is an active level-0 segment and thus belongs to $S_0(Q)$. Moreover, the above identities hold for every $i \in [i_0, i_1]$. For a fixed $k$, the level-0 segments form disjoint maximal intervals, so exactly $i_1 - i_0 + 1$ indices $i \in [n]$ in this segment satisfy these conditions. Therefore, the algorithm correctly increments $s'_k$ by exactly $i_1 - i_0 + 1$.

The time for computing $S_0(Q)$ is $\widehat{\mathcal{O}}(n^{1+\mu}/Q)$ by an argument analogous to that in Lemma 20. We scan all segments in $S_0(Q)$ and perform a constant-time test and update for each segment, so this final step takes $\widehat{\mathcal{O}}(|S_0(Q)|) = \widehat{\mathcal{O}}(n^{1+\mu}/Q)$ time. ◀

Combining these results, we obtain the following theorem, as well as Theorem 3.

▶ **Theorem 42.** *There is a deterministic algorithm that solves Problem 7 in time* $\widehat{\mathcal{O}}(Mn + n^{1+\mu}/M)$.

**Proof of Theorem 3.** By the reduction in Lemma 29, it suffices to solve Problem 7 efficiently. By Theorem 42, there is a deterministic algorithm that solves Problem 7 in time $\widehat{\mathcal{O}}(Mn + n^{1+\mu}/M)$. Take $M = \Theta\big(n^{\mu/2}\big)$. Then the two terms balance, yielding the claimed time bound $\widehat{\mathcal{O}}\big(n^{1+\mu/2}\big)$. ◀

## References


1 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039, 2025. doi:10.1137/1.9781611978322.63.

2 Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997. doi:10.1006/JCSS.1997.1388.

3 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/S00453-012-9734-3.

4 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1073–1084, 2017. doi:10.1137/1.9781611974782.69.

5 Karl Bringmann and Alejandro Cassis. Faster Knapsack algorithms via bounded monotone min-plus-convolution. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 31:1–31:21, 2022. doi:10.4230/LIPIcs.ICALP.2022.31.

6 Karl Bringmann, Anita Dürr, and Adam Polak. Even faster Knapsack via rectangular monotone min-plus convolution and balancing. In *Proceedings of the 32nd Annual European Symposium on Algorithms (ESA)*, pages 33:1–33:15, 2024. Full version at https://arxiv.org/abs/2404.05681. doi:10.4230/LIPIcs.ESA.2024.33.

7 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly subcubic algorithms for language edit distance and RNA folding via fast bounded-difference min-plus product. *SIAM J. Comput.*, 48(2):481–512, 2019. doi:10.1137/17M112720X.

8 Timothy M. Chan. Derandomizing pseudopolynomial algorithms for subset sum. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3600–3610, 2026. doi:10.1137/1.9781611978971.131.

9 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 31–40, 2015. doi:10.1145/2746539.2746568.

10 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Weakly approximating Knapsack in subquadratic time. In *Proceedings of the 52nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 51:1–51:18, 2025. doi:10.4230/LIPIcs.ICALP.2025.51.

11 Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1529–1542, 2022. doi:10.1145/3519935.3520057.

12 Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min, +)-convolution. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 22:1–22:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.22.

13 Mingyang Deng, Yael Kirkpatrick, Victor Rong, Virginia Vassilevska Williams, and Ziqian Zhong. New additive approximations for shortest paths and cycles. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 50:1–50:10, 2022. doi:10.4230/LIPIcs.ICALP.2022.50.

14 Anita Dürr. Improved bounds for rectangular monotone min-plus product and applications. *Inf. Process. Lett.*, 181:106358, 2023. doi:10.1016/J.IPL.2023.106358.

15 Nick Fischer. Universe reduction for apsp: Equivalence of three fine-grained hypotheses. In *Proceedings of the 58th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page to appear, 2026.

16 Nick Fischer, Piotr Kaliciak, and Adam Polak. Deterministic 3SUM-hardness. In *Proceedings of the 15th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 49:1–49:24, 2024. doi:10.4230/LIPIcs.ITCS.2024.49.

17 Dvir Fried, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, Ely Porat, and Tatiana Starikovskaya. An improved algorithm for the $k$-dyck edit distance problem. *ACM Trans. Algorithms*, 20(3):26, 2024. doi:10.1145/3627539.

18 Younan Gao and Meng He. Faster path queries in colored trees via sparse matrix multiplication and min-plus product. In *Proceedings of the 30th Annual European Symposium on Algorithms (ESA)*, pages 59:1–59:15, 2022. doi:10.4230/LIPIcs.ESA.2022.59.

19 Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster monotone min-plus product, range mode, and single source replacement paths. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198, pages 75:1–75:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.75.

20 Xiaohan Huang and Victor Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complex.*, 14(2):257–299, 1998. doi:10.1006/JCOM.1998.0476.

21 Ce Jin, Yael Kirkpatrick, Michał Stawarz, and Virginia Vassilevska Williams. Improved additive approximation algorithms for apsp. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3639–3651, 2026. doi:10.1137/1.9781611978971.133.

22 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, Warsaw, Poland, July 10-14, 2017*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.21.

23 Xiao Mao. Breaking the cubic barrier for (unweighted) tree edit distance. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 792–803, 2021. doi:10.1109/FOCS52979.2021.00082.

24 Jakob Nogler, Adam Polak, Barna Saha, Virginia Vassilevska Williams, Yinzhan Xu, and Christopher Ye. Faster weighted and unweighted tree edit distance and APSP equivalence. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*, pages 2167–2178, 2025. doi:10.1145/3717823.3718116.

25 Barna Saha and Christopher Ye. Faster approximate all pairs shortest paths. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4758–4827, 2024. doi:10.1137/1.9781611977912.170.

26 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.

27 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

28 Virginia Vassilevska Williams and Yinzhan Xu. Truly subcubic min-plus product for less structured matrices, with applications. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 12–29, 2020. doi:10.1137/1.9781611975994.2.

29 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.