Seminar on Advanced Algorithms and Data Structures - Student Report

Witnesses for Boolean Matrix Multiplication and for Shortest Paths

Matteo Signer

April 1, 2018

1 Introduction

Consider the product C of two $n \times n$ boolean matrices, A and B. To be more precise, the multiplication is over the semiring $(\{0,1\}, \vee, \wedge)$, so we have $C_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}$. The *witnesses* of such a multiplication is an index k for each i, j where $C_{ij} = 1$ such that $A_{ik} = 1 = B_{kj}$. These witnesses can be considered a proof or example of $C_{ij} = 1$.

The utility of boolean matrix multiplication is clear, for example the transitive closure of a graph with n vertices can be computed using $\mathcal{O}(\log n)$ matrix multiplications. The witnesses come into play where one wants to find some k for all pairs (i, j) with $A_{ik} = 1 = B_{kj}$.

Of course, straightforward boolean without witnesses can be computed as fast as an integer matrix multiplication by selecting the entries that are non-zero in the result. This is known to be computable faster than the $O(n^3)$ running time of naive multiplication. Strassen was the first to produce such an algorithm that runs in $O(n^\omega)$ for $\omega = 2.807$[4], and at the time of the original paper, Coppersmith and Winograd had discovered an algorithm for $\omega = 2.376$[5].

More recent results provide algorithms for even smaller values of ω, such as Le Gall's algorithm which achieves $\omega = 2.373$[6].

These subcubic algorithms don't produce witnesses. We will construct an algorithm for computing the witnesses that runs in $\mathcal{O}(n^\omega (\log n)^{\mathcal{O}(1)})$.

2 Algorithm

We first look at a simpler, randomized algorithm called *BPWM*, as described by R. Seidel[2]. The algorithm shares many of the concepts and ideas we use in our final deterministic algorithm.

2.1 BPWM

The algorithm relies on one key observation.

For the boolean matrices A and B and their product C over the integers, we can find the witnesses for the entries with $C_{ij} = 1$ easily with the following method:

Take the matrix B' defined as $B'_{ij} = i \cdot B_{ij}$, i.e. multiply each row by its index. The product $C' = A \cdot B'$ contains witnesses for the entries where $C_{ij} = 1$. This is obvious, since if $C_{ij} = 1$, then there exists exactly one k where both A_{ik} and B_{kj}, and $B'_{kj} = k$ is the only contributor to C'_{ij} since for all other k, either A_{ik} or B'_{ik} are zero (as they were in the original matrices A and B).

Lets look at an example:

$$B = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad B' = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 2 & 0 & 0 & 2 \\ 0 & 3 & 3 & 3 \\ 4 & 0 & 0 & 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 2 & \underline{1} & 2 & 3 \\ \underline{1} & 0 & 0 & \underline{1} \\ 2 & 0 & \underline{1} & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad C' = \begin{bmatrix} 3 & \underline{3} & 4 & 6 \\ \underline{2} & 0 & 0 & \underline{2} \\ 5 & 0 & \underline{1} & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The highlighted entries take the value 1 in C, so they are witnesses for $A \cdot B$, which can easily be seen.

We also don't need to compute the matrix C, as it suffices to just check if A_{ik} and B_{kj} are both 1 to check if $k = C'_{ij}$ is a witness.

So we can compute the witnesses for the 1-entries efficiently. The idea now is to select a random subset of d of the columns of A and rows of B, say $s_1, \ldots, s_d$. This gives us two $n \times d / d \times n$ matrices X and Y with $X_{ij} = A_{is_i}$, $Y_{ij} = B_{s_ij}$. The entries that are non-zero in $Z = X \cdot Y$ are also non-zero in $A \cdot B$.

Using the above example with $d = 2$ columns/rows and $s_1 = 2$, $s_2 = 3$:

$$Y = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad Y' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 2 & 2 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad Z = \begin{bmatrix} \underline{1} & \underline{1} & \underline{1} & 2 \\ \underline{1} & 0 & 0 & \underline{1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad Z' = \begin{bmatrix} \underline{1} & \underline{2} & \underline{2} & 3 \\ \underline{1} & 0 & 0 & \underline{1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Because we got rid of some columns and rows, the entries of the product matrix are generally smaller and we find witnesses we didn't have before.

The witnesses in Z' are for the matrices X and Y, so we need to transform the witness $k = Z'_{ij}$ into s_k to be a witness for the product $A \cdot B$.

We want to maximize the chance that an entry of $X \cdot Y$ becomes 1. Intuitively, for a large C_{ij}, this should happen when we select few columns/rows, and for smaller entries of C we should take more. To put it crudely, the chance is maximized when $C_{ij} \cdot d \approx n$.

We thus pick $d = 1, 2, 4, \ldots, 2^{\lceil \log_2 n \rceil - 1}$, so we are at most a factor 2 away.

For each entry $c = C_{ij}$ (i.e. with c potential witnesses), $\frac{c \cdot d}{n}$ is between $\frac{1}{2}$ and 1 in one of these values of d.

Now we determine a lower bound on the probability that this entry in the X-Y-product becomes one in that iteration, and thus also for the entire loop.

Claim 1. *An iteration with $n/2 \leq c \cdot d \leq n$ finds a witness for an entry where $C_{ij} = c$ with at least probability $\frac{1}{2\mathrm{e}}$.*

Proof. There are c potential witnesses where $A_{ik} = 1 = B_{kj}$. The probability of hitting one of them is $\frac{c}{n}$.

The number of potential witnesses we hit with d columns is thus $X \sim B(d, \frac{c}{n})$, and we are successful if $X = 1$. Plugging in the probability distribution of the Binomial distribution gives us

$$\mathrm{P}[X = 1] = d\frac{c}{n}(1 - \frac{c}{n})^{d-1}.$$

Using our constraits $d\frac{c}{n} \geq \frac{1}{2}$ and $-\frac{c}{n} \geq -\frac{1}{d}$, we find

$$\mathrm{P}[X = 1] \geq \frac{1}{2}(1 - \frac{1}{d})^{d-1} \geq \frac{1}{2}\mathrm{e}^{-1}. \qquad \square$$

The probability is small, but larger than a constant. We can repeat the algorithm m times so the probability that a certain witness is *not* found is at most $(1 - \frac{1}{2\mathrm{e}})^m$. We can find all witnesses we didn't find in $\mathcal{O}(n)$ time per witness, so as to not influence our total running time, we want to find all but n witnesses. This is the expected case when the probability is at most $\frac{1}{n}$.

Solving for m gives $m \geq \log_{1-\frac{1}{2\mathrm{e}}} \frac{1}{n} = -\log n / \log(1 - \frac{1}{2\mathrm{e}})$ which is satisfied for example by $m = \lceil 3.42 \log_2 n \rceil$.

Concluding, we only have to spend $\mathcal{O}(n^\omega \log^2 n + w \cdot n)$ where w denotes the number of entries for which we haven't found a witness. This is $\mathcal{O}(n^\omega \log^2 n)$ in the expected case.

2.2 Deterministic algorithm

Now let's look at the final algorithm that takes $\tilde{\mathcal{O}}(n^\omega)$ [1] in the *worst* case. In this algorithm, we can find the witnesses for entries between 1 and $c = \lceil \log \log n + 9 \rceil$ in one go, and $\alpha = \frac{8}{2^c}$ is the fraction of witnesses we are willing to compute by hand.

```
L denotes the set of positive entries of C for which we haven't found a witness yet
while L ≠ ∅ do
    R ← (r_ij = 1)_{i,j=1}^n
    for ⌈1 + 3 log_{4/3} n⌉ iterations do
        D ← A · (B ∧ R)                          ▷ Matrix multiplication over the integers
        L' ← {(i, j) ∈ L | D_ij ≤ c}
        Find witnesses for the entries in L'      ▷ And remove L' from L
        R ← good matrix
    end for
end while
```

The matrix R is defined to be *good* if two conditions hold:

- The sum of the entries of D in L is at most $3/4$ of what it was in the previous iteration, so it drops by at least $3/4$ each iteration. This guarantees that $D = \mathbf{0}$ after the $\lceil 1 + 3 \log_{4/3} n \rceil$ iterations.
- The fraction of entries of D in L that go from a value bigger than c to 0 it at most α.

[1] $\tilde{\mathcal{O}}(f(n)) := \mathcal{O}(f(n)(\log f(n))^{\mathcal{O}(1)})$

The way this works is similar to the idea of BPWM: we still select a subset of entries to multiply and hope we will get entries small enough to find witnesses for them. It is just that we can find witnesses for entries between 1 and c, instead of entries that are exactly 1.

The steps that are not immediately clear in the algorithm is how to actually find the witnesses for the suitable entries in L' and how to generate the good matrix R. It turns out that we can find a suitable R in $\tilde{\mathcal{O}}(n^\omega)$.

First we look at what happens when we remove a random set of entries from R in each step:

Lemma 1. *The choice $R \leftarrow R \wedge S$ where S is a uniformly random boolean matrix is good with probability at least $1/6$.*

Proof. The lemma will follow from these claims:

Claim 2. *The sum of entries of D in L goes down by at least $3/4$ with probability at least $1/3$.*

Proof. The expected factor X the sum of entries of D in L shrinks by is $1/2$ (as is true of any subset of entries).

We have $\mathrm{P}\left[X < \frac{3}{4}\right] = 1 - \mathrm{P}\left[X \geq \frac{3}{4}\right] \geq 1 - \mathrm{E}[X]/\frac{3}{4} = 1 - \frac{2}{3} = \frac{1}{3}$ where we used Markov's inequality. ■

Claim 3. *The probability that a fixed entry of D which is at least c drops down to 0 is at most $1/2^c$.*

Proof. There are at least c 1-entries in the corresponding column of $B \wedge R$ which have a 1 in the respective entry of the row of A. All these entries have to become zero, which happens with probability at most $1/2^c$. ■

Claim 4. *The probability that more than a fraction α of the entries of D in L goes down from at least c to 0 is at most $\frac{1}{2^c}\frac{1}{\alpha} = \frac{1}{8}$.*

Proof. The expected value of that fraction is obviously less than $1/2^c$ by Claim 2. Applying Markov's inequality again, we immediately obtain the desired result. ■

Combining these results, the probability of the sum of entries of D in L going down by $3/4$ *and* at most α fraction of the values going from at least c to 0 is at least $1/3 - 1/8 > 1/6$. The subtraction is motivated in the fact that the events are not independent, and in the worst case all the outcomes where more than α fraction of the values go from at least c to 0 are contained in those where the sum of entries goes down by $3/4$. □

It turns out we can switch the uniform distribution for S to a low-size probability space that retains many of the aspects of the uniform distribution, but is small enough that we can iterate over all possible outputs in polylogarithmic time.

Such a probability space is the c-wise ϵ-dependent random sample space as described in [3] where $\epsilon = \frac{1}{2^{c+1}}$ will fit our use case.

A distribution is c-wise ϵ-dependant if for every subset of size $\leq c$ of the variables, the probability density function constrained on it deviates at most ϵ in the L_1-Norm.

We will not go into detail into the construction of such a sample space, because it is quite complicated.

Instead, we only use the fact that there exists a construction for such a sample space that only takes $\mathcal{O}(c + \log\log n + \log\frac{1}{\epsilon})$ random bits to sample. It follows that the size of the sample space is only $\mathcal{O}((\log n)^{\mathcal{O}(1)})$.

We now want to prove that the probability of finding a good matrix by choosing S from this c-wise ϵ-dependent sample space is larger than zero, so it is guaranteed that we find a suitable matrix when looking at all possible outputs of the sample space.

Lemma 2. *The choice $R \leftarrow R \wedge S$ where S is a matrix of n^2 c-wise ϵ-dependant random variables is good with probability at least $1/12 - 2\epsilon$.*

Proof.

Claim 5. *The sum of entries of D in L goes down by at least $3/4$ with probability at least $1/3 - 2\epsilon$.*

Proof. The expected factor X the sum of entries of D in L shrinks by is $\leq 1/2 + \epsilon$. This is obvious since X is $\frac{1}{n^2}$ times the sum of n^2 Bernoulli variables. Since summing commutes with the expected value, it is also less than the largest of the expected values of any of these variables, which is at most $1/2 + \epsilon$.

Applying Markov's inequality again: $\mathrm{P}\left[X < \frac{3}{4}\right] = 1 - \mathrm{P}\left[X \geq \frac{3}{4}\right] \geq 1 - \mathrm{E}[X]/\frac{3}{4} = 1 - \frac{4}{3}\,\mathrm{E}[X] \geq 1/3 - 2\epsilon$ ■

Claim 6. *The probability that a fixed entry of D which is at least c drops down to 0 is at most $1/2^c + \epsilon$.*

Proof. This is obvious, because our c-wise ϵ-dependent distribution deviates at most ϵ from the uniform one when constrained on a subset of size $\leq c$, in particular the event that some c entries of S that contribute to D all drop to 0. ■

Claim 7. *The probability that more than a fraction α of the entries of D drop from at least c to 0 is at most $1/4$.*

Proof. The number of entries dropping from c to 0 is the sum of n Bernoulli variables with $p \leq 1/2^c + \epsilon$ each, so the expected fraction Y of "thrown" entries is also $\leq 1/2^c + \epsilon$.

We apply Markov's law once again: $\mathrm{P}[Y \geq \alpha] \leq \mathrm{E}[Y]/\alpha \leq (\frac{2}{2^c})/\alpha = \frac{1}{4}$. ■

Thus, analogously to the case of the uniform distribution, we have obtained $\mathrm{P}[R \text{ is "good"}] \geq \frac{1}{3} - 2\epsilon - \frac{1}{4} = \frac{1}{12} - 2\epsilon$. □

This probability is greater than zero for $\epsilon < \frac{1}{24}$. This is always the case, as $c \geq 9$, so $\epsilon = \frac{1}{2^{c+1}} \leq \frac{1}{2^{10}} < \frac{1}{24}$.

We now know that we still have a chance of finding a "good" matrix even when sampling from our low-size sample space. We only make use of this result indirectly, as it implies that in principle we can and will obtain a suitable S for some inputs.

As noted before, our sample space only needs $\mathcal{O}(c + \log\log n + \log\frac{1}{\epsilon})$ random bits to produce a result. This means that the size of the sample space (the set of all possible results) is only $\mathcal{O}(2^c\ \log n\ 2^{c+1})^{\mathcal{O}(1)}$, which is polylogarithmic in n.

Checking if a matrix is "good" requires only $\mathcal{O}(n^\omega + n^2)$ time, so we find a good matrix R in $\tilde{\mathcal{O}}(n^\omega)$.

The only part that we have not covered is actually finding the witnesses for L'. This turns out to be doable in much the same way as finding R.

Lemma 3. *By iterating over all possible "random" input bits of a c-wise ϵ-dependant random sample space D with $\epsilon < \frac{1}{2^c}$, every subset S of size $m \leq c$ of the random variables will assume all 2^m possible values.*

Proof. Suppose not, i.e. there exist some subset $S = \{x_{i_1}, \ldots, x_{i_m}\}$ of the variables that doesn't assume the values $y_{i_1}, \ldots, y_{i_m}$.

Then obviously $P_D[x_{i_1} = y_{i_1}, \ldots, x_{i_m} = y_{i_m}] = 0$, while the uniform distribution has a probability of $\frac{1}{2^m} \geq \frac{1}{2^c}$ of generating these values. This however violates the c-wise ϵ-dependence of D, because the probability function on a subset should differ at most $\epsilon < \frac{1}{2^c}$ from the uniform distribution. □

This guarantees that every entry of L' drops to 1 for some of the matrices S, because for every entry of value at most c, all but one of the contributing entries will be filtered out by S. In fact, every contributor will be the only one left in S for some S, as per the above lemma.

Checking this for a fixed S only takes $\mathcal{O}(n^\omega + n^2)$ time. We can do this at the same time as searching for a "good" R, where we also calculate the same matrix $A \cdot (B \wedge R_{old} \wedge S)$, so we only have an overhead of $\mathcal{O}(n^2)$.

In each iteration of the inner loop, at most α fraction of the entries greater than c of D will vanish (i.e. their witnesses cannot be found). So after the inner loop completes, we will have found the witnesses for at least $(1-\alpha)^{\lceil 1+3\log_{4/3} n \rceil}$ fraction of L.

Claim 8. $(1-\alpha)^{\lceil 1+3\log_{3/3} n \rceil} > 0.85$

Proof. We have $\alpha = \frac{8}{2^c} \leq \frac{8}{2^{\log\log n + 9}} = \frac{1}{64 \log n}$. Because $3/\log_2(4/3) < 8$, we have $\lceil 1 + 3\log_{4/3} n \rceil \leq 2 + 8\log_2 n$.

Thus we have an lower bound of $(1-\alpha)^{2+8\log_2 n} = (1-\alpha)^2((1-\alpha)^{\log_2 n})^8 \geq (1-\frac{1}{64})^2((1-\frac{1}{64\log n})^{\log n})^8 \geq (1-\frac{1}{64})^{10} > 0.85$.

The last step is because $(1-\frac{a}{x})^x$ is monotonically growing (as base and exponent both are) and starts at $1-a$ for $x = 1$. □

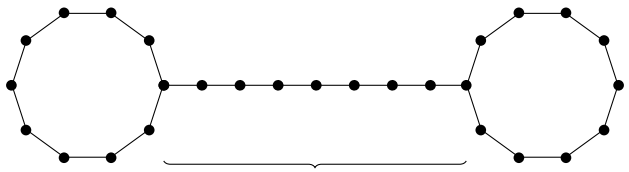
Thus, we find witnesses for at least 85% of the entries L, so we only need to execute the outer loop at most $\log_{0.85} \frac{1}{n^2}$ times, which is $\mathcal{O}(\log n)$.

Because finding a "good" matrix takes $\tilde{\mathcal{O}}(n^\omega)$ and each of the loops is executed at most $\mathcal{O}(\log n)$ times, the entire algorithm only takes $\tilde{\mathcal{O}}(n^\omega)$ time.

3 Applications

The algorithm for the boolean product witness problem is most useful for fast algorithms for all-pair-style graph problems, e.g. the all-pair shortest paths problem.

In general, algorithms that provide some paths (e.g. shortest ones) between all vertices cannot be faster than $O(n^3)$ because the output for some graphs is $\Theta(n^3)$. This is made clear by the following graph:



It is clear that the distance between any pair of nodes from both the left and the right subgraph is $\Theta(n)$, because any path must pass through the $\frac{n}{3}$ connecting nodes. There are $\frac{n}{3}$ nodes in the left and $\frac{n}{3}$ nodes in the right subgraph, so there are $\frac{n^2}{9}$ pairs with shortest paths of length at least $\frac{n}{3}$, so the output is at least of size $\frac{n^3}{27}$.

Instead, we consider these problems in a form where we only need to provide a *witness* or *successor* between all vertices, which is the first vertex on the path. In most cases, this is equivalent, because a subpath is also a solution for its end points. The path can be extracted quickly by following the witnesses.

3.1 Witnesses for transitive closure of a directed graph

Consider a directed unweighted graph given by its adjacency matrix A. The adjacency matrix of its transitive closure can be easily computed as $T = A^\infty = A^n$, which only takes $\mathcal{O}(n^\omega \log n)$. But can we also produce witnesses for the connected vertices such that we can construct a path from all vertices to all reachable vertices?

An initial idea is to compute the witnesses for $T = A \cdot T$, which we can do in $\tilde{\mathcal{O}}(n^\omega)$ using our algorithm. This accurately gives us the next vertex on a path. But it is not enough to reconstruct the full paths, because such a path can contain a cycle and it is possible that we would just go in a loop.

We could also just use the shortest paths in A, but that can only be done in $\mathcal{O}(n^{(\omega+3)/2})$ for (dense) directed graphs.

Instead, we transform the graph into one that doesn't contain cycles. Find all the strongly connected components in $G = (V, E)$ and build a new graph $G' = (V', E')$. Each vertex v'_i of G' is a strongly connected component in G with the vertices $v_{i1}, \ldots, v_{ir_i}$, and an edge is placed if there was a edge between two vertices of the components, i.e. $E' = \{(v'_i, v'_j) \mid \exists (v_{ix}, v_{jy}) \in E\}$. The edge (v_{ix}, v_{jy}) is called the *associated* edge to (v'_i, v'_j).

The graph G' doesn't contain any cycles, so our naive algorithm gives a correct result for it. If we can now find the witnesses within each strongly connected component, we can join the results to get the final witnesses.

We can calculate the witnesses $\hat{W}$ within a strongly connected graph as follows:

Run a BFS from any vertex v_0, giving a BFS tree T. For any edge (u, v) in T, we set $\hat{W}(u, w)$ to v for any descendant w of v. These are obviously correct witnesses.

Now run a BFS from v_0 on the same graph but with reversed edges. When visiting a node u coming from v, then set $\hat{W}(u, w)$ to v for all w if it isn't set yet.

This can be thought of as follows:

For the witness for u, v: if v is a descendant of u in T, then we have the witness from the first (forward) BFS.

If not, then we go up one level in the second tree and try again. In the worst case, we go all the way up to v_0, but we have witnesses for all targets for v_0 because we started the first BFS there.

We have the witnesses $\hat{W}$ within each SCC and W' for G' to combine.

To calculate the witness from v_{ik_1} to v_{jk_2}:

If $i = j$, i.e. the vertices are in the same strongly connected component, then we alread have found the witness in $\hat{W}$.

Otherwise, we need to traverse the components. Let $k = W'(v'_i, v'_j)$ be the next SCC on our path. By definition of the witnesses, we know that $(v'_i, v'_k) \in V'$, so there exists some associated edge $(v_{ix}, v_{jy}) \in V$. We then set

$$W(v_{ik_1}, v_{jk_2}) = \begin{cases} v_{ky} & k_1 = x \\ \hat{W}(v_{ik_1}, v_{ix}) & \text{otherwise} \end{cases}$$

This means that if we are at the associated edge to the next SCC, we will take it. Otherwise, we move towards "our" vertex of the edge.

The algorithm takes $\mathcal{O}(n^2)$ for calculating the strongly connected components, $\tilde{\mathcal{O}}(n^\omega)$ for the witnesses between them and $\mathcal{O}(n^2)$ for merging the solutions, so $\tilde{\mathcal{O}}(n^\omega)$ in total.

3.2 APSP for undirected unweighted graph

Seidel produced a subcubic algorithm that produces these witnesses from the adjacency matrix and the distance matrix of a graph [2]. The principle is that for the first vertex on a shortest path, it is a necessary and sufficient condition that the distance to the target decreases by exactly one.

So given the adjacency matrix A and the distance matrix D of our graph, for the path between i and j we want to find some successor k such that $A_{ik} = 1$ and $D_{kj} = D_{ij} - 1$.

We can determine the witness for all pairs i, j by computing the witness matrices $W^{(d)}$ for the products $A \cdot B^{(d)}$ where $B^{(d)}_{ij} = 1 \Leftrightarrow D_{ij} = d - 1$. Then $S_{ij} = W^{(D_{ij})}_{ij}$.

The problem with this approach is that we need to solve the witness problem n times, which already takes $\Omega(n^3)$ time. The realization needed is that because the distances the distances to a target j for the neighbours of i can only differ by at most 1, it suffices to find a k such that $A_{ik} = 1$ and $D_{kj} \equiv D_{ij} - 1 \pmod 3$.

Thus we redefine the witness matrices $W^{(d)}$ for the same product, but with $B^{(d)}_{ij} = 1 \Leftrightarrow D_{ij} \equiv d - 1 \pmod 3$. Now we have $S_{ij} = W^{(D_{ij} \bmod 3)}_{ij}$ and only need to calculate the witnesses for the three products with $B^{(0)}$, $B^{(1)}$, $B^{(2)}$.

Seidel also found a $\mathcal{O}(n^\omega \log n)$ algorithm for computing the distance matrix D given A.

Originally, the use of BPWM was proposed, giving an running time of $\mathcal{O}(n^\omega \log^2 n)$ only in the expected case. Using our deterministic algorithm for the boolean product witness problem we can achieve almost the same complexity in *all* cases, only replacing the $\log^2 n$ with some polylogarithmic factor.

References

[1] Noga Alon et al. "Witnesses for boolean matrix multiplication and for shortest paths". In: *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*. IEEE. 1992, pp. 417–426.

[2] Raimund Seidel. "On the all-pairs-shortest-path problem". In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. ACM. 1992, pp. 745–749.

[3] J Naor and M Naor. "Small-bias probability spaces: efficient constructions and applications". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM. 1990, pp. 213–223.

[4] Volker Strassen. "Gaussian elimination is not optimal". In: *Numerische mathematik* 13.4 (1969), pp. 354–356.

[5] Don Coppersmith and Shmuel Winograd. "Matrix multiplication via arithmetic progressions". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 1–6.

[6] François Le Gall. "Powers of tensors and fast matrix multiplication". In: *Proceedings of the 39th international symposium on symbolic and algebraic computation*. ACM. 2014, pp. 296–303.