1 The Min-Sum Product

Last time, we asked the question of calculating $APSP_n$ in $o(n^3)$ time. We showed that $APSP_n$ can be solved in time $O(\text{MSP}_n \log n)$ where MSP_n is the problem of solving the Min-Sum Product. We will also prove in the homework that $APSP_n = O(MSP_n + n^2)$. This means that a natural way to attack the All Pairs Shortest Paths problem is to attack the Min-Sum Product. Indeed, a state-of-the-art result from Ryan Williams shows that MSP_n can be solved in time $O(n^3/2^{c\sqrt{\log n}})$[Wil13], which gives us a better algorithm for $APSP_n$. In this lecture we will outline and discuss this result.

As a reminder, the Min-Sum Product of A and B, where A is an $n \times \ell$ matrix and B is an $\ell \times n$ matrix, is defined to be the $\ell \times \ell$ matrix denoted $(A \odot B)$ such that for all $1 \leq i, j \leq n$,

$$(A \odot B)_{ij} = \min_{k=1}^{\ell} (A_{ik} + B_{kj}) \tag{5.1}$$

2 Ryan Williams' Algorithm for APSP/MSP

In 2014, Ryan Williams proved the following breakthrough result for MSP under the assumption that the elements of the matrix are bounded by some constant M. As discussed, this gives a state-of-the-art result for APSP.

Theorem 5.1. *There is a randomized algorithm that solves MSP_n for matrices with entries in $\{0, 1, ...M\} \bigcup \{\infty\}$ that runs in time $O(n^3/2^{c\sqrt{\log n}} \cdot \text{polylog}M)$*[1]

This is an interesting result, because we can think of $n^3/2^{c\sqrt{\log n}}$ as being "between" $n^3/\log n^c$ and n^{3-c}:

$$\left(\frac{n^3}{\log n^c} = \frac{n^3}{2^{\log\log n}}\right) \ll \frac{n^3}{c\sqrt{\log n}} \ll \left(\frac{n^3}{2^{c \log n}} = n^{3-c}\right) \tag{5.2}$$

The *big* question is if MSP_n can be solved in time $O(n^{3-c})$ for some $c > 0$, but this is in some sense the next best thing.

Because the proof is more simple but still retains the main ideas of the original result, we will prove the following weaker result instead:

Theorem 5.2. *For some $\varepsilon > 0$, there is a randomized algorithm that solves $MSP_{n\times n}$ for matrices with elements in $\{0, 1, ..., M\}$ that runs in time $O((n^3/2^{(\log n)^\varepsilon}) \cdot \text{polylog}n)$*

3 Splitting MSPs Into Smaller MSPs

The algorithm for computing $MSP_{n\times n}$ works by cleverly splitting the Min-Sum Product of two $n \times n$ matrices into smaller, easier to solve instances of Min-Sum Product. Observe that if A and B are two $n \times n$ matrices, we can decompose $A \odot B$ as shown in Figure 1.1, where each A_i consists of ℓ columns of A and each B_i consists of ℓ rows of B:

[1] $O(\text{polylog}n)$ is defined as $O((\log n)^c)$ for some constant c

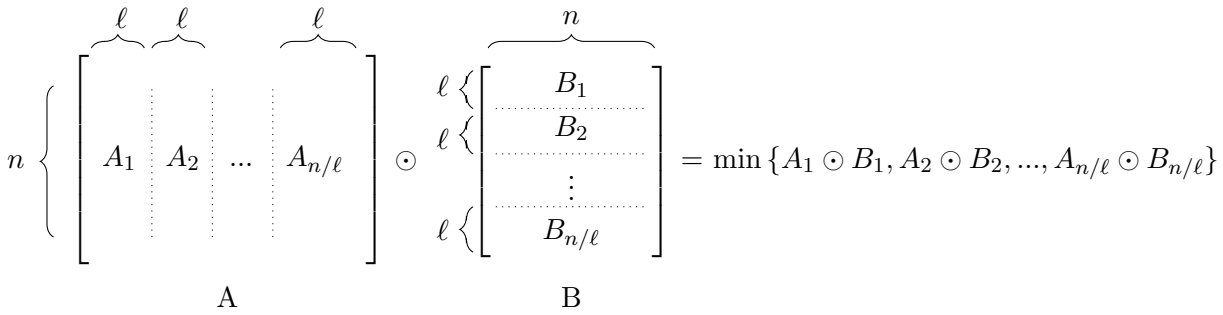

Figure 5.1: Decomposing Min-Sum Product

We can convince ourselves that this decomposition is correct since if $C = A \odot B$ then

$$\begin{aligned} C_{ij} &= \min_{k=1}^{n}\{A_{ik} + B_{kj}\} \\ &= \min\{\min_{k=1}^{\ell}\{A_{ik} + B_{kj}\}, \min_{k=\ell+1}^{2\ell}\{A_{ik}, B_{kj}\}, ..., \min_{k=n-\ell}^{n}\{A_{ik}, B_{kj}\}\} \\ &= \min\{(A_1 \odot B_1)_{ij}, (A_2 \odot B_2)_{ij}, ..., (A_{n/\ell} \odot B_{n/\ell})_{ij}\} \end{aligned} \tag{5.3}$$

We want to choose ℓ such that $\text{MSP}_{(n\times\ell)\odot(\ell\times n)}$ takes $O(n^2)$ time. Then the algorithm above would take time $O(n^3/\ell)$. This is because each individual product $A_i \odot B_i$ takes time $O(n^2)$, and there are n/ℓ such products. The final step is to take the componentwise min over the n/ℓ matrices $A_1 \odot B_1, A_2 \odot B_2, ..., A_{n/\ell} \odot B_{n/\ell}$, each of which have n^2 elements, and this takes time $O(n^3/\ell)$ as well. Hence the total time would be $O(n^3/\ell)$.

If $\ell = 1$, then this doesn't yield an improvement, so we want ℓ to be sufficiently large. In fact, 5.3 argues that this holds for $\ell = 2^{(\log n)^\varepsilon}/\log M$ for some $\varepsilon > 0$ (if we allow randomization).

Lemma 5.3. *There exists an $\varepsilon > 0$ such that computing $MSP_{(n\times\ell)\times(\ell\times n)}$ for $\ell = 2^{(\log n)^\varepsilon}/\log M$ can be done in time*

$$O\Big(\frac{n^2 \cdot (\log M)^{O(1)} \cdot (\log n)^2}{\ell}\Big).$$

One should see how this implies the main result (5.2)[2]. In order to prove 5.3, we need the following crucial result by Don Coppersmith, which gives an efficient algorithm for multiplying these "thin" matrices. Note that this is for normal matrix multiplication, not the Min-Sum Product.

Theorem 5.4 (Coppersmith)**.** *Multiplying $(n \times n^{0.1}) \times (n^{0.1} \times n)$ matrices over any field can be done using $O(n^2 \log^2(n))$ arithmetic operations.*

For those interested in the proof of this theorem, Ryan Williams explains it in detail in Appendix C of his paper.

4 Polynomials To Calculate Min-Sum Product

Observe that if $C = A \odot B$, where A is a $n \times \ell$ matrix and B is a $\ell \times n$ matrix, then for all $1 \le i, j \le n$,

$$C_{ij} = \min_{k=1}^{\ell}\{(A_{ik} + B_{kj})\} = \vec{a_i} \odot \vec{b_j}$$

[2]Note that $\ell \gg (\log n)^2$, and so ℓ "kills" the $(\log n)^2$ term in the numerator.

where a_i is the ith row of A and b_i is the jth row of B. This is the fundamental operation we are worried about. Specifically, given two vectors $\vec{u}$ and $\vec{v}$ of length ℓ, we want to quickly calculate

$$\vec{u} \odot \vec{v} = \min_{k=1}^{\ell} \{\vec{u}(k) + \vec{v}(k)\}.$$

We can think of $\vec{u}$ as both an ℓ-length vector, as well as a bitstring of length $\ell \log M$. In order to calculate perform this operation, we will construct polynomials on the individual bits of the bitstrings of $\vec{u}$ and $\vec{v}$ to calculate the bits in the result of $\vec{u} \odot \vec{v}$.

Suppose we had the following (slightly false) lemma:

Lemma 5.5 (False!!!)**.** *Suppose $C(\vec{u}, \vec{v})$ computes the the ith bit of $\min_{k=1}^{\ell} \{\vec{u}(k) + \vec{v}(k)\}$. Then there exists an algorithm that outputs a (multivariate) polynomial*

$$p(x_1, x_2, ..., x_{\ell \log M}, y_1, y_2, ..., y_{\ell \log M})$$

over the field $\mathbb{F}_2$[3] *such that for all $\vec{u}, \vec{v}$, we have*

$$p(\vec{u}, \vec{v}) = C(\vec{u}, \vec{v}).$$

In addition, the degree of p is less than $\log (\ell \log M)^{O(1)}$.

Since p is a polynomial over $\mathbb{F}_2$, it is also a multilinear polynomial, meaning that every term consists of the product of some subset of the variables (no exponents above one appear in the polynomial).

The value in this lemma is that the polynomial p has the extremely useful property that it has low degree. In general, low-degree polynomials are our friends. Indeed, this property is crucial to the result we are working towards. Using this property, we can derive the result:

Lemma 5.6. *If $C(\vec{u}, \vec{v})$ computes the ith bit of $\min_{k=1}^{\ell} \{\vec{u}(k) + \vec{v}(k)\}$, we can compute $C(\vec{x}, \vec{y})$ for all $\vec{x}, \vec{y} \in \{a_1, a_2, ..., a_n\} \times \{b_1, b_2, ..., b_n\}$ in time $\widetilde{O}(n^2)$*[4]

This implies 5.3 because we can just apply 5.6 multiple times to find the value of each of the $O(\log M)$ bits of each element of $A_i \odot B_i$.

5 Evaluating the Polynomials Efficiently

Now we will prove 5.6 by showing that we can evaluate the polynomial p that we got from 5.5 at $\vec{x}, \vec{y} \in \{a_1, a_2, ..., a_n\} \times \{b_1, b_2, ..., b_n\}$ in time $\widetilde{O}(n^2)$. This can be done at one by multiplying two specially-designed matrices. Since p is a polynomial over the field $\mathbb{F}_2$, it is also a multilinear polynomial. In addition, it's degree is at most D. As a result, we can write $p(\vec{x}, \vec{y})$ in the form

$$p(\vec{x}, \vec{y}) = \sum_{\substack{S,T \\ |S|+|T| \leq D}} \alpha_{ST} \vec{x}^S \vec{y}^T$$

where $\vec{u}^S$ denotes the monomial

$$\prod_{i \in S} u(i)$$

and the α_{ST} are coefficients (either 0 or 1).

[3] $\mathbb{F}_2$ is the field consisting of the elements $\{0, 1\}$ with addition and multiplication modulo 2 as operations

[4] $\widetilde{O}(f(n))$ is defined to be $O(f(n) \cdot \operatorname{poly} \log n)$. In other words, it hides the log terms.

Let K be the set of $\langle S,T\rangle$ such that $\alpha_{ST}=1$, and fix an order on it. We can also think of K as the set of terms of p. We let A' be the $n\times|K|$ matrix such that $A'_{ij}=\vec{a_i}^{S_j}$ where $\langle S_j,T_j\rangle$ is the jth elemenet of K. Similarly, let B' be the $|K|\times n$ matrix such that $B'_{ij}=\vec{b_j}^{T_i}$. We observe that $A'B'$ is a $n\times n$ matrix, such that for all $1\le ij\le n$,

$$(A'B')_{ij}=\sum_{\substack{\langle S,T\rangle\in K\\ \alpha_{ST}=0}}\alpha_{ST}\vec{a_i}^S\vec{b_j}^T=p(\vec{a_i},\vec{b_j}).$$

If we can show that A' and B' are both $n\times n^{0.1}$ matrices, we are golden, since we can just use Coppersmith's algorithm to multiply them and get our answer. Let $N=2\ell\log M$ be the number of variables of p. Hence the number of terms of p is at most

$$\binom{N}{1}+\binom{N}{2}+...\binom{N}{D}\le N^{O(D)} \tag{5.4}$$

We want to show that $N^{O(D)}$ is at most $n^{0.1}$, so we choose ε sufficiently large such that the following calculation goes through:

$$\begin{aligned}
\ell &\le \frac{2^{(0.1\log n)^\varepsilon}}{\log M} &&\iff\\
\ell\log M &\le 2^{(0.1\log n)^\varepsilon} &&\iff\\
\log(\ell\log M) &\le (0.1\log n)^\varepsilon &&\iff\\
O(\log(\ell\log M))^{O(1)} &\le 0.1\log n &&\iff\\
O(\log(\ell\log M))^{O(1)}\cdot\log(2\ell\log M) &\le 0.1\log n &&\iff\\
O(D)\log N &\le 0.1\log n &&\iff\\
N^{O(D)} &\le n^{0.1}
\end{aligned} \tag{5.5}$$

This means that we can use the Coppersmith matrix multiplication algorithm to compute $p(\vec{x},\vec{y})$ for all $\vec{x},\vec{y}\in\{a_1,a_2,...,a_n\}\times\{b_1,b_2,...,b_n\}$ in time $\widetilde{O}(n^2)$. This gives us 5.6. All that remains is to fix 5.5 slightly, and then prove it, and the proof of 5.2 will be complete.

6 Proving Lemma 5.5

We assumed 5.5, but it is false. Yet it is not that false. We just have to add a little randomization. Here is the corrected statement of 5.5:

Lemma 5.7. *Suppose $C(\vec{u},\vec{v})$ computes the the ith bit of $\min_{k=1}^{\ell}\{u(k)+v(k)\}$. Then there exists a* randomized *algorithm that outputs a (multivariate) polynomial*

$$p(x_1,x_2,...,x_{\ell\log M},y_1,y_2,...,y_{\ell\log M})$$

over the field $\mathbb{F}_2$ such that $Pr[p(\vec{u},\vec{v})=C(\vec{u},\vec{v})]\ge 99\%$.

The only difference now is that for each i,j, the polynomial correctly computes $C(a_i,b_j)$ with probability at least 99%. If we repeat this experiment $O(\log n)$ times, and output the componentwise majority of $\{C^1_{ij},C^2_{ij},...,C^{\log n}_{ij}\}$, the resulting matrix is completely correct with probability at least $1-1/\mathrm{poly}(n)$.

We will now outline a proof sketch for 5.7 by expressing $C(\vec{u}, \vec{v})$ as a low-depth circuit, and then converting that circuit into a polynomial (using a result that we will prove in Homework 2, Problem 3).

Suppose $C(\vec{x}, \vec{y})$ calculates the ith bit of $\min_{k=1}^{\ell}\{u(k) + v(k)\}$. The proof takes the following two steps:

1. Write $C(\vec{x}, \vec{y})$ as a circuit with size $O(n)^{O(1)}$, constant depth, and unbounded fanin. By constant depth, we mean that there is a constant c such that all paths from the inputs to the circuit to the output pass at most c logic gates. By unbounded fanin, we mean that logic gates such as OR can take any number of inputs.

   Without loss of generality, by De Morgan's laws, we can say that this circuit is made up just of OR gates and NOT gates, because any AND gates can be expressed in terms of OR and NOT gates, with only a constant increase in depth.

   The fact that it can be represented by such a circuit appears as Lemma 2 in [Wil13], and is proven in Appendix A of the same paper.

2. In Problem 3 of Homework 2, we will show how to convert our circuit into a polynomial. Specifically, any circuit with depth $O(1)$ and size s can be represented probabilistically by a polynomial over $\mathbb{F}_2$ of degree $O(\log s)^{O(1)}$.

   This is where the randomization comes in. If an OR gate has a large number of inputs, we would end up with a polynomial with too large of a degree. However, as we will see on the homework, with some randomization, we can reduce the degree substantially. The details to this part are left in the homework.

References

[Wil13] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *CoRR*, abs/1312.6680, 2013. 1, 1