

- For any two boolean matrices  $A_{n \times n}$  and  $B_{n \times n}$ , we call  $k$  a *witness* of tuple  $(i, j)$  whenever  $A_{ik} = B_{kj} = 1$ . A matrix  $W_{n \times n}$  is called a *witness matrix* of boolean matrices  $A_{n \times n}$  and  $B_{n \times n}$  whenever  $W_{ij}$  stores a witness corresponding to tuple  $(i, j)$  for every  $1 \leq i, j \leq n$ . The witness matrix has many applications, including efficiently computing all-pairs shortest paths in unweighted undirected graphs and transitive closure of directed graphs.
- The  $ij$ -th entry of  $AB$ , denoted by  $[AB]_{ij}$ , has the number of witnesses of tuple  $(i, j)$ .

First, we devise an algorithm to find the witness of those  $(i, j)$ -entries of  $W$  for which there is exactly one witness; that is, for such  $(i, j)$  tuple, there is only one  $k$  such that  $A_{ik} = B_{kj} = 1$ . Later, with the help of random sampling, this algorithm is extended to find every entry of  $W$ .

- Lemma 1: Let  $A'_{n \times n}$  be a matrix with  $A'_{ij} = jA_{ij}$  for every  $1 \leq i, j \leq n$ . For any  $(i, j)$ , if  $[AB]_{ij} = 1$ , then  $[A'B]_{ij}$  has the unique witness of  $(i, j)$ .

Proof: If  $[AB]_{ij} = 1$ , then there exists a  $k$  such that  $A_{ik} = 1$  and  $B_{kj} = 1$ . That is,  $k$  is the witness for  $[AB]_{ij}$  being equal to 1, and for every  $k' \neq k$ , either  $A_{ik'} = 0$  or  $A_{k'j} = 0$ . Hence,  $[A'B]_{ij} = \sum_{k=1}^n A'_{ik} B_{kj} = k$ .

- Using this observation, the following deterministic algorithm finds the correct witness for every  $(i, j)$  that has a unique witness:
  - (a) for every  $1 \leq i, j \leq n$
  - (b)  $A'_{ij} = jA_{ij}$
  - (c) compute  $A'B$  and  $AB$
  - (d) for every  $1 \leq i, j \leq n$
  - (e) if  $[AB]_{ij}$  is equal to 1 then  $W_{ij} = [A'B]_{ij}$  else  $W_{ij} = 0$

- Let  $w$  be the number of witnesses for any tuple  $(i, j)$ . We show that a random sample  $R \subseteq [n]$ , with  $\frac{n}{2} \leq w|R| \leq n$ , is very likely to have a witness for  $(i, j)$  when  $w \geq 2$ .

Next, we define matrices  $A^R$  and  $B^R$ . For every  $k \in [1, n]$ , if  $k \notin R$ , set both the  $k^{\text{th}}$  column of  $A^R$  and the  $k^{\text{th}}$  row of  $B^R$  as null vectors; on the other hand, if  $k \in R$ , then  $A^R_{ik} = kA_{ik}$  for every  $i$ , and  $B^R_{kj} = B_{kj}$ .

Observation: If  $[AB]_{ij} = 1$  with witness  $k$  belonging to  $R$ , then  $[A^R B^R]_{ij} = k$ .

Theorem 1: For any  $(i, j)$ , given  $[AB]_{ij} = w \geq 2$ , the probability  $[A^R B^R]_{ij} = 1$  is at least  $\frac{1}{2e}$ .

Proof: This probability is

$$\begin{aligned}
 &= \frac{\binom{w}{1} \binom{n-w}{|R|-1}}{\binom{n}{|R|}} \\
 &= \frac{w|R|}{n} \left( \prod_{j=0}^{w-2} \frac{n-|R|-j}{n-1-j} \right) \\
 &\geq \frac{w|R|}{n} \left( \prod_{j=0}^{w-2} \frac{n-|R|-j-(w-j-1)}{n-1-j-(w-j-1)} \right)
 \end{aligned}$$

$$\begin{aligned}
&= \frac{w|R|}{n} \left( \prod_{j=0}^{w-2} \frac{n-w-(|R|-1)}{n-w} \right) \\
&= \frac{w|R|}{n} \left( 1 - \frac{|R|-1}{n-w} \right)^{w-1} \\
&\geq \frac{1}{2} \left( 1 - \frac{1}{w} \right)^{w-1} \quad \left( \text{since } \frac{n}{2} \leq w|R| \leq n, \frac{w|R|}{n} \geq \frac{1}{2}, \text{ and } \frac{|R|-1}{n-w} = \frac{|R|-1}{w(\frac{n}{w}-1)} \leq \frac{1}{w} \right) \\
&\geq \frac{1}{2e}. \quad (\text{since } w \geq 2)
\end{aligned}$$

- For any entry  $[AB]_{ij}$ , if there are many witnesses, then having a small  $|R|$  helps. On the other hand, if  $[AB]_{ij}$  has a small number of witnesses, then having a large  $|R|$  would obviously help. Hence, every size in  $S = \{1, 2, \dots, \frac{n}{2}\}$  is tried for  $|R|$ . Specifically, for  $w = [AB]_{ij}$ , there exists a  $|R| \in S$  satisfying  $\frac{n}{2} \leq w|R| \leq n$ . Besides, as argued below, trying each of these values for  $|R|$  suffice to ensure there will only be a few entries of  $W$  left to be computed via brute-force.

For every  $d \in S$ , repeatedly sampling  $R$  of size  $d$ , independently and uniformly at random from  $[n]$  for  $O(\lg n)$  times, further reduces the probability an entry of  $W$  left empty. That is, from Theorem 1, the probability none of these  $R$  vectors lead to a unique witness for  $(i, j)$ -th entry is at most  $(1 - \frac{1}{2e})^{O(\lg n)}$ , for any  $(i, j)$ . Of course, witnesses for some of the entries may not be found via this clever idea; these missing witnesses are found via brute-force.

- $C \leftarrow AB$ ; initialize  $W$  to null matrix
  - for every  $d \in S$ 
    - repeat for  $c \cdot (\lg n)$  times //value of  $c$  to be fixed later
    - choose a subset  $R \subseteq [n]$  of size  $d$ , independently and uniformly at random
    - construct  $A^R, B^R$ , and  $A^{R \bmod}$ , where  $[A^{R \bmod}]_{ij}$  is  $j[A^R]_{ij}$  for every  $i, j$
    - $C^R \leftarrow A^R B^R; Z \leftarrow A^{R \bmod} B^R$
    - for every  $1 \leq i, j \leq n$
    - if  $C_{ij} > 0$  and  $C^R = 1$  then  $W_{ij} \leftarrow Z_{ij}$
    - for every  $(i, j)$ , if  $C_{ij} > 0$  and  $W_{ij} = 0$ , find a witness of  $(i, j)$  by brute-force
- For any  $C_{ij}$ , there exists a  $d \in S$  such that  $\frac{n}{2} \leq C_{ij} \cdot d \leq n$ . From the above description, probability that a random choice of  $R$  does not have a unique witness for  $W_{ij}$  is at most  $(1 - \frac{1}{2e})$ . Hence, probability  $W_{ij}$  not found after  $c \cdot \lg n$  iterations is at most  $(1 - \frac{1}{2e})^{c \lg n}$ . For having the error probability polynomially small, upper bounding  $(1 - \frac{1}{2e})^{c \lg n}$  with  $\frac{1}{n}$ , leads to 3.77 being a lower bound on  $c$ .

Since the probability an entry of  $W$  not found after  $c \cdot \lg n$  iterations is at most  $\frac{1}{n}$ , by the time algorithm reaches step (i), the expected number of witnesses remaining to be found is  $n$ . Since each entry of  $W$  can be determined in  $O(n)$  time by brute-force, step (i) takes  $O(n^2)$  expected time.

Step (f) takes  $O(MM(n))$  time, where  $MM(n)$  denotes time to multiply two  $n \times n$  matrices, and this step gets executed  $O((\lg n)^2)$  times. Steps (g)-(h) take  $O(n^2)$  time and they get executed  $O((\lg n)^2)$  times. Since  $MM(n)$  is  $\Omega(n^2)$ , this algorithm takes  $O(MM(n)(\lg n)^2)$  expected time.

References:

R. Motwani and P. Raghavan, Randomized Algorithms. Cambridge University Press, 1995.