The goal of this week is to show the following Theorem of Zwick:

Theorem 0.1. *All-Pairs Shortest Paths (APSP) on unweighted directed graphs can be solved in $\widetilde{O}(n^{2+1/(4-\omega)})$ time, where ω is the matrix multiplication exponent.*

Zwick's algorithm computes distances between close nodes (nodes u, v with $d(u, v) < P$) and far-away nodes (nodes (u, v) with $d(u, v) \geq P$) separately.

Both cases, however, take advantage of the Hitting Set Lemma, which can be proven using standard probability arguments.

Lemma 0.1. *Suppose we have poly(n) sets $S_1, \ldots, S_k$ of $\{1, \ldots, n\}$, each of size $\geq L$. Then a random sample $S \subseteq \{1, \ldots, n\}$ with $|S| = c(n/L \lg n)$ for a sufficiently large constant c hits all S_i in at least one element with high probability.*

We will start by addressing nodes of distance P or more away from each other.

Proposition 1. *Let $G = (V, E)$ be an unweighted directed graph. Fix a parameter P. In $\widetilde{O}(n^3/P)$ time, one can compute $d(u, v)$ for every u, v with $d(u, v) \geq P$.*

Proof. For every pair u, v of nodes at least P apart in V, let $S_{u,v}$ be a set containing the nodes in some shortest path from u to v. Pick a hitting set S of size $\Theta(n/P \lg n)$ so that S hits every $S_{u,v}$ (with high probability).

For each $s \in S$, compute $d(s, v)$ for all $v \in V$ using breadth-first-search in $O(n^2)$ time. Similarly, for each $s \in S$, also compute $d(v, s)$ for all $v \in V$ using breadth-first-search (on G with its edges inverted) in $O(n^2)$ time.

Since S hits each $S_{u,v}$ for every u, v with $d(u, v) \geq P$, it follows that for some $s \in S$ we have $d(u, v) = d(u, s) + d(s, v)$. Thus in $O(n^2|S|)$ time we can compute

$$d'(u, v) = \min_{s \in S} d(u, s) + d(s, v),$$

which is the correct distance for all u, v at least P apart. □

In order to handle shortest paths of length less than P, we introduce the *distance product*.

Definition 0.1. *Let A, B be $n \times n$ matrices. Define the distance product by*

$$(A \star B)[i, j] = \min_k \{A(i, k) + B(k, j)\}.$$

Although we will not prove it, a theorem of Fisher, Mayer, et al. states that if $A \star B$ can be computed in $T(n)$ time, then APSP in weighted graphs can be done in $O(T(n))$ time, and vice-versa.

It turns out that distance products can be computed relatively quickly.

Theorem 0.2. *if A, B are $n \times n$ matrices with entries in $\{-M, \ldots, M\}$, then $A \star B$ can be computed in $\widetilde{O}(Mn^\omega)$ time.*

Proof. Define matrices A' and B' with entries

$$A'[i, j] = (n + 1)^{M - A(i,j)}$$
$$B'[i, j] = (n + 1)^{M - B(i,j)}$$

Computing the integer product of A' and B' we obtain C' with entries

$$C'[i,j] = \sum_k (n+1)^{2M-(A(i,k)+B(k,j))}.$$

For a given i, j, we can then compute $(A \star B)[i,j] = \min_k A(i,k) + B(k,j)$, which we will refer to as L, as follows.

Observe that $(n+1)^{2M-L} \leq C'[i,j]$ because $(n+1)^{2M-L}$ is a summand in $C'[i,j]$. At the same time, $C'[i,j] \leq (n+1)^{2M-L} \cdot n$ because $(n+1)^{2M-L}$ is the largest summand in $C'[i,j]$ and $C'[i,j]$ has only n summands. Therefore, we can set L to be the smallest integer such that $C'[i,j] \geq (n+1)^{2M-L}$.

Note that we are dealing with integers having $O(M \lg n)$ bits in C', for which operations take $\widetilde{\mathrm{O}}(M)$ time. Bearing this to mind, it is straightforward to see that the above method computes $A \star B$ in $\widetilde{\mathrm{O}}(Mn^\omega)$ time. □

By combining fast computations of distance products with the idea of a hitting set, we can now obtain a fast algorithm for computing distances between close-together nodes.

Proposition 2. *Let $G = (V, E)$ be an unweighted directed graph, and P be a fixed parameter. We can compute $d(u,v)$ for pairs of nodes less than P apart in time*

$$\widetilde{O}\left(n^\omega P^{3-\omega}\right).$$

Proof. We will have $\lceil \lg_{3/2} P \rceil$ stages. Let V_j be the set of pairs of vertices (u,v) such that $d(u,v) \in [(3/2)^{j-1}, (3/2)^j)$, and let $V_{\leq j}$ denote $\cup_{i=1}^{j} V_i$. In stage j, we will compute every $d(u,v)$ for every $(u,v) \in V_j$. More specifically, we will compute a matrix D_j such that for all $(x,y) \in V_{\leq j}$, $D_j[x,y] = d(x,y)$; and $D_j[x,y] = \inf$ for all $(x,y) \notin V_{\leq j}$. Note that D_1 can easily be obtained from the adjacency matrix of G.

One could easily obtain a valid D_j from D_{j-1} by simply computing and cleaning up $D_{j-1} \star D_{j-1}$. However, we cannot afford to compute $n \times n$ matrix distance products. Instead, we will take advantage of hitting sets.

For every $(u,v) \in V_j$, consider a shortest path $P_{u,v}$ from u to v. The *middle third* of $P_{u,v}$ is a set of $\lfloor (3/2)^{j-1} \rfloor$ nodes appearing consecutively in $P_{u,v}$ such that at most $(3/2)^{j-1}$ nodes precede them, and at most $(3/2)^{j-1}$ follow them.

At stage j, take a random $S_j \subseteq V$ with $|S_j| \in \Theta(\frac{n}{(3/2)^{j-1}} \lg n)$ so that for all $(u,v) \in V_j$ with high probability V hits a node $s_{u,v}$ in the middle third of $P_{u,v}$. Observe that because $s_{u,v}$ is in the middle third of $P_{u,v}$, we get that $(u, s_{u,v}), (s_{u,v}, v) \in D_{\leq j-1}$.

It follows that with high probability, for all $(u,v) \in V_j$,

$$d(u,v) = \min_{s \in S_j} D_j[u,s] + D_j[s,v].$$

Thus we can compute $D_j(u,v)$ to be

$$\min(D_{j-1}[u,v], \min_{s \in S_j} D_j[u,s] + D_j[s,v]).$$

This is easy to do in n^2 time once we have already computed each $\min_{s \in S} D_j[u,s] + D_j[s,v]$, which can be obtained by computing the product $X \star Y$ where X contains the columns in D_{j-1} corresponding with elements of S_j, and Y contains the rows in D_{j-1} corresponding with elements of S_j. In other words, by selecting a hitting set S_j, we are able to use the distance product of matrices much smaller than D_{j-1} in order to compute D_j.

Breaking X and Y into square (or smaller) blocks of side-length approximately $(3/2)^j$, we can use the distance products of all $(3/2)^{2j}$ pairs of blocks to easily recover $X \star Y$. By theorem 0.2, since D_j has entries in $\{0, \ldots, (3/2)^j\}$ (as well as entries with value inf which Theorem 0.2 can easily be adapted to handle) this takes time

$$\widetilde{\mathrm{O}}\left((3/2)^{2j}\left(\frac{n}{(3/2)^j}\right)^\omega (3/2)^j\right) = \widetilde{\mathrm{O}}\left(n^\omega ((3/2)^{3-\omega})^j\right).$$

Summing over the $\lceil \lg_{3/2} P \rceil$ stages, we get time

$$\widetilde{\mathrm{O}}\left(n^{\omega} \sum_{j:(3/2)^j < P} ((3/2)^j)^{3-\omega}\right) \leq \widetilde{\mathrm{O}}\left(n^{\omega} P^{3-\omega}\right).$$

□

We are now in a position to complete the proof of Zwick's Theorem. Indeed, combining Proposition 1 and Proposition 2 and optimizing for P (at $P = n^{(3-\omega)/(4-\omega)}$), we get total runtime of $\widetilde{\mathrm{O}}(n^{2+1/(4-\omega)})$. Observe that both the algorithm from Proposition 1 and from Proposition 2 compute either correct distances or overestimates for distances between pairs of nodes (that are correct for far nodes); thus minimizing the outputted distances of the two, one can obtain the exact $d(u, v)$ for all $u, v \in V$.