6.890 Lecture 5 Yuster-Zwick Distance Oracle; Node Weighted APSP
Scribe: Vaggos Chatziafratis
Edited by: Yinzhan Xu **Date:** September 22, 2021

Last time we gave an $\tilde{O}(\sqrt{M}n^{(3+\omega)/2})$ time algorithm for APSP in directed graphs with edge weights in $\{-M,\dots,M\}$.

Today we will give Zwick's APSP algorithm and will then extend it to give Yuster-Zwick's Distance Oracle.

Last time we defined the $(\min,+)$-product of $n \times n$ matrices:

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

We stated this theorem:

Theorem 0.1. *Given two $n \times n$ matrices A, B over $\{-M,\dots,M\} \cup \{\infty\}$, $A \star B$ can be computed in $\tilde{O}(Mn^\omega)$ time.*

1 Zwick's Algorithm

We will show the following.

Theorem 1.1. *All-Pairs Shortest Paths (APSP) on n-node directed graphs, where the edge weights are integers in $\{-M,\dots,M\}$ can be solved in $\tilde{O}(M^{1/(4-\omega)}n^{2+1/(4-\omega)})$ time by a randomized algorithm, where the distances are correct with high probability.*

There is a way to derandomize Zwick's algorithm, so that the same bounds can be obtained by a deterministic algorithm.

Similar to the algorithm from the previous lecture, Zwick's Algorithm handles paths that use at least k nodes, and paths that use fewer than k nodes separately. For long paths, the running time is the same as the previous algorithm, which is $\tilde{O}(Mn^\omega + \frac{n^3}{k})$ time. Zwick's Algorithm improves on the short paths part.

In order to handle shortest paths of length less than k, we combine fast computations of $(\min,+)$ products with the idea of a hitting set argument.

Recall that for every u, v we have picked a representative shortest path $P_{u,v}$, and $\ell(u,v)$ is the number of nodes in $P_{u,v}$.

Proposition 1. *Let G be a directed graph, where edge weights are integers in $\{-M,\dots,M\}$, and k be a fixed parameter. We can compute for every pair of vertices u, v, an estimate $\tilde{d}(u,v)$ such that for all u, v, $\tilde{d}(u,v) \geq d(u,v)$ and with high probability, $\tilde{d}(u,v) = d(u,v)$ for every pair (u,v) where $\ell(u,v) \leq k$. The algorithm runs in time*

$$\tilde{O}\left(k^{3-\omega}Mn^\omega\right).$$

Proof. We will have $\lceil \log_{3/2} k \rceil$ stages. Let V_j be the set of pairs of vertices (u,v) such that $\ell(u,v) \in ((3/2)^{j-1}, (3/2)^j]$, and let $V_{\leq j}$ denote $\cup_{i=1}^j V_i$. In stage j, we will compute $d(u,v)$ for every $(u,v) \in V_{\leq j}$. More specifically, we will compute a matrix D_j such that with high probability,

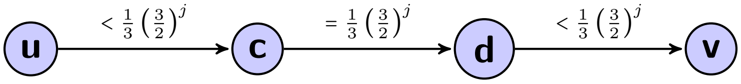
$$D_j(x,y) \begin{cases} = d(x,y) & \text{if } (x,y) \in V_{\leq j} \\ \geq d(x,y) & \text{otherwise.} \end{cases}$$

Note that D_2 can easily be obtained from the edge weights of G.

One could easily obtain a valid D_j from D_{j-1} by simply computing $D_{j-1} \star D_{j-1}$. However, it won't give the running time we desire. Instead, we will take advantage of the hitting set lemma.

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

Here's a figure showing the middle third as the portion of $P_{u,v}$ between nodes c and d:


$< \frac{1}{3}\left(\frac{3}{2}\right)^j$
$= \frac{1}{3}\left(\frac{3}{2}\right)^j$
$< \frac{1}{3}\left(\frac{3}{2}\right)^j$
u
c
d
v

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

$$d(u,v) = \min_{s \in S_j} \{D_{j-1}(u,s) + D_{j-1}(s,v)\}.$$

Thus we can compute $D_j(u,v)$ via

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

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

Breaking X and Y into square blocks of side-length approximately $n/(3/2)^j$, so that there are approximately $(3/2)^j$ blocks in X and Y. We can use the $(\min, +)$-products of all $(3/2)^{2j}$ pairs of blocks to easily recover $X \star Y$. By Theorem 0.1, since D_j has entries in $\{-(3/2)^j M, \ldots, (3/2)^j M\} \cup \{\infty\}$, this takes time

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

Summing over the $\lceil \log_{3/2} k \rceil$ stages, we get a running time of

$$\tilde{O}\left(n^{\omega} M \sum_{j:(3/2)^j < k} ((3/2)^j)^{3-\omega} \right) = \tilde{O}\left(k^{3-\omega} M n^{\omega} \right).$$

□

We are now in a position to complete the proof of Zwick's Theorem. Indeed, combining the long distance algorithm and Proposition 1 and optimizing for k at $k = \frac{n^{(3-\omega)/(4-\omega)}}{M^{1/(4-\omega)}}$, we get a total running time of $\tilde{O}(M^{1/(4-\omega)} n^{2+1/(4-\omega)})$. Observe that both the algorithm for long paths and for short paths compute either the correct distances or overestimate for distances between pairs of nodes; thus minimizing the outputted distances of the two, one can obtain the exact $d(u,v)$ for all $u, v \in V$.

2 Yuster-Zwick Distance Oracle

Our goal here is to construct a "distance oracle", faster than computing APSP. The distance oracle is a graph data structure that can answer distance queries for any given pair of vertices, in some small amount of time. This query time is not necessarily constant.

The theorem we prove is as follows.

Theorem 2.1 (Yuster, Zwick '05)**.** *Let G be a directed graph with edge weights in $\{-M,\dots,M\}$. Then in $\tilde{O}(Mn^\omega)$ time, we can compute an $n\times n$ matrix D such that for every $u,v\in V$,* ***with high probability****:*

$$(D\star D)[u,v]=d(u,v),$$

where $d(u,v)$ is the minimum weight of a shortest path from i to j using n hops. (If the graph does not have negative cycles, then $d(\cdot,\cdot)$ are just the shortest paths distances.)

Recall that the fastest algorithm for APSP in directed graphs requires $\tilde{O}(\sqrt{M}n^{2.5})$ time even if $\omega=2$, whereas the algorithm in the theorem obtains a much faster runtime of $\tilde{O}(Mn^\omega)$, at the cost that now to extract any particular distance, one might need to spend $O(n)$ time.

Note that the theorem above does not imply a faster APSP algorithm, because D may have large entries, making computing $D\star D$ expensive. However, if one only cares about a small number q of distances, one can extract them from D in $O(qn)$ time. (A straightforward way to extract q distances would require q iterations of Dijkstra's algorithm, which would give a runtime of $\Omega(qn^2)$.)

Thus, for *single source shortest paths* (SSSP) we have the following corollary:

Corollary 2.1. *Let $G=(V,E)$ be a directed graph with edge weights in $\{-M,\dots,M\}$ and no negative cycles. Let $s\in V$. Then single-source shortest paths from s can be computed in $\tilde{O}(Mn^\omega)$ time.*

Proof. By Theorem 2.1, we can compute an $n\times n$ matrix D such that $D\star D$ is the correct all-pairs shortest-paths matrix, in $\tilde{O}(Mn^\omega)$ time.

Then for all $v\in V$, we know that:

$$d(s,v)=\min_k\{D[s,k]+D[k,v]\}.$$

Computing this for all $v\in V$ only takes $O(n^2)$ time. Since $\omega\geq 2$, this entire computation is in $\tilde{O}(Mn^\omega)$ time. □

Similarly, we can show that detecting negative cycles is fast since any negative cycle contains a simple cycle of negative weight, and thus corresponds to a path from i to i for some i of length $\leq n$ and negative weight.

Corollary 2.2. *Let G be a directed graph with edge weights in $\{-M,M\}$. Then negative cycle detection can be computed in $\tilde{O}(Mn^\omega)$ time.*

We use the following notations for the proof of the main theorem.

Notation: For $u,v\in V$, let $P_{u,v}$ be an arbitrary shortest path from u to v using at most n hops, and let $\ell(u,v)$ be the number of hops on $P_{u,v}$. Suppose that A is an $n\times n$ matrix and that $S,T\subseteq\{1,\dots,n\}$. Then $A[S,T]$ is the submatrix of A consisting of rows indexed by S and columns indexed by T.

The main algorithm again uses randomness and the hitting set lemma but now we do not take freshly random samples every time, instead we take each successive random sample B_{j+1} to be a random sample from the previous random sample B_j.

Proof of Theorem 2.1.

We claim that Algorithm 1 is our desired algorithm.

Running Time: In iteration j, we multiply an $n\times\tilde{O}\left(\frac{n}{(3/2)^{j-1}}\right)$ matrix by a $\tilde{O}\left(\frac{n}{(3/2)^{j-1}}\right)\times\tilde{O}\left(\frac{n}{(3/2)^{j}}\right)$ matrix, where all entries are at most $(3/2)^jM$ in absolute value (we will show iteration j only needs to consider paths with at most $(3/2)^j$ nodes).

Algorithm 1: YZ(A)

```
$A$ is the weighted adjacency matrix;
Set $D \leftarrow A$;
Set $B_0 \leftarrow V$;
for $j = 1, \ldots, \log_{3/2} n$ do
    Let $D'$ be $D$ but with all entries larger than $M(3/2)^j$ replaced by $\infty$;
    Choose $B_j$ to be a random subset of $B_{j-1}$ of size $\frac{c \cdot n}{(3/2)^j} \log n$;
    Compute $D_j \leftarrow D'[V, B_{j-1}] \star D'[B_{j-1}, B_j]$;
    Compute $\overline{D}_j \leftarrow D'[B_j, B_{j-1}] \star D'[B_{j-1}, V]$;
    foreach $u \in V, b \in B_j$ do
        Set $D[u,b] = \min(D[u,b], D_j[u,b])$;
        Set $D[b,u] = \min(D[b,u], \overline{D}_j[b,u])$;
return D;
```

We can split the matrices to blocks of size around $\frac{n}{(3/2)^j}$ by $\frac{n}{(3/2)^j}$, and compute the Min-Plus product between every pair of blocks. Hence the runtime for iteration j is $\tilde{O}\left(M(3/2)^j(3/2)^j(\frac{n}{(3/2)^j})^\omega\right) = \tilde{O}\left(\frac{Mn^\omega}{((3/2)^j)\omega-2}\right)$. Over all iterations, the running time is, asymptotically, ignoring polylog factors,

$$Mn^\omega \sum_j ((3/2)^{\omega-2})^j \leq \tilde{O}(Mn^\omega).$$

If $\omega > 2$, one of the log factors in the $\tilde{O}$ can be omitted.

Correctness: We will prove the correctness by proving two claims.

Claim 1: For all $j = 0, \ldots, \log_{3/2} n$, $v \in V$, $b \in B_j$, if $\ell(v,b) < (3/2)^j$ then w.h.p. after iteration j, $D[v,b] = d(v,b)$

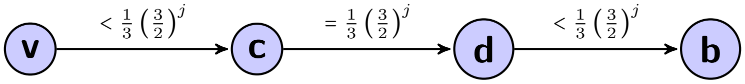
Proof of Claim 1: We will prove it via induction. The base case ($j = 0, \ell(v,b) < (3/2)^0 = 1$) is trivial, since the distance is for one-hop paths is exactly the adjacency matrix. Now, assume the inductive hypothesis is true for $j-1$, that is we have stored correctly $D[u,b] = d[v,b]$ if the shortest path (v,b) has length $\ell(v,b) < (3/2)^{j-1}$. We will show correctness for j. Consider some $v \in V$ and $b \in B_j$. We consider two possible cases depending on how far node b is from v.

Case I: $\ell(v,b) < (3/2)^{j-1}$ (b is near)

But then $b \in B_j \subset B_{j-1}$. By our inductive hypothesis, $D[v,b] = d(v,b)$ w.h.p.!

Case II: $\ell(v,b) \in [(3/2)^{j-1}, (3/2)^j)$ (b is far)

We will need to use our "middle third" technique from Zwick's algorithm.


v
c
d
b
$< \frac{1}{3}\left(\frac{3}{2}\right)^j$
$= \frac{1}{3}\left(\frac{3}{2}\right)^j$
$< \frac{1}{3}\left(\frac{3}{2}\right)^j$

We can choose $c, d \in V$ on the path $P_{v,b}$ such that:

$$\ell(v,c) < \frac{1}{3}\left(\frac{3}{2}\right)^j$$

$$\ell(d,b) < \frac{1}{3}\left(\frac{3}{2}\right)^j$$

$$\ell(c,d) = \frac{1}{3}\left(\frac{3}{2}\right)^j < \left(\frac{3}{2}\right)^{j-1}$$

By a hitting set argument, if c is a large enough constant, $B_{j-1} \cap$ "middle third" $\neq \varnothing$ (w.h.p. depending on c) since $|B_{j-1}| = c\frac{n}{(3/2)^{j-1}} \log n$.

Let x in $B_{j-1} \cap$ "middle third". Then $\ell(v,x) \leq \ell(v,c) + \ell(c,d) < \frac{2}{3}(\frac{3}{2})^j = (\frac{3}{2})^{j-1}$. Since $x \in B_{j-1}$, by induction $D[v,x] = d(v,x)$ w.h.p. at iteration j. By a similar argument we get that w.h.p. $D[x,b] = d(x,b)$ at iteration j (at the beginning of iteration j).

Hence after this iteration, $D[v,b] \leq D[v,x] + D[x,b] = d(v,b)$.

As a small technical note, we will need to actually remove entries larger than $(3/2)^j M$ from D before multiplying, but they are not needed.

Claim 2: For all $u, v \in V$, w.h.p. $(D \star D)[u,v] = d(u,v)$.

Proof of Claim 2: Fix $u, v \in V$, and let j be such that $\ell(u,v) \in ((3/2)^{j-1}, (3/2)^j]$. Look at a shortest path between u and v. Its middle third hence has a length of $(1/3)(3/2)^j$. Then w.h.p. B_j hits this path at some $x \in V$ such that $\ell(u,x), \ell(x,v) < (3/2)^{j-1}$. By Claim 1, $D(u,x) = d(u,x)$ and $D(x,b) = d(x,b)$. Hence:

$$d(u,v) \leq (D \star D)[u,v] \leq min_{x \in B_{j-1}} D(u,x) + D(x,v) \leq d(u,v).$$

This completes the proof.

□

References

[1] Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *FOCS*, pages 389–396, 2005.