1 The Distance Product

Last time we defined the distance product of $n \times n$ matrices:

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

Theorem 1.1. *Given two $n \times n$ matrices A, B over $\{-M, M\}$, $A \star B$ can be computed in $\tilde{O}(Mn^\omega)$ time.*

2 Oracle for All-Pairs Shortest Paths

Theorem 2.1 (Yuster, Zwick '05)**.** *Let G be a directed graph with edge weights in $\{-M, M\}$ and no negative cycles. Then in $\tilde{O}(Mn^\omega)$ time, we can compute an $n \times n$ matrix D such that for every $u, v \in V$, w.h.p.:*

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

Note that this does not immediately imply a fast APSP algorithm, because D may have large entries, making computing $D \star D$ expensive.

Corollary 2.1. *Let $G = (V, E)$ be a directed graph with edge weights in $\{-M, M\}$ and no negative cycles. Let $s \in V$. Then single-source shortest path 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 $O(Mn^\omega)$ 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 $O(Mn^\omega)$ 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 $\leq n$.

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 now prove our main theorem:

Proof of Theorem 2.1. Let $\ell(u,v)$ be the number of nodes on a shortest u to v path. Additionally, for notational convenience, suppose that A is an $n \times n$ matrix and that $S, T \subseteq \{1, \ldots, n\}$. Then $A[S,T]$ is the submatrix of A consisting of rows indexed by S and columns indexed by T.

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 $(3/2)^j M$ (we will show iteration j only needs to consider paths with at most $(3/2)^j$ nodes).

Algorithm 1: YZ(A)

A is a 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 ∞;
  Choose B_j to be a random subset of B_{j-1} of size $S_j = \frac{c \cdot n}{(3/2)^j} \log n$;
  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;

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, $D[v, b] = d(v, b)$

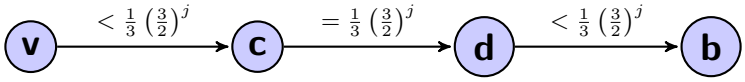
Proof of Claim 1: We will prove it via induction. The base case ($j = 0$) is trivial, since the distance is for one-hop paths is exactly the adjacency matrix. Now, assume the inductive hypothesis is true for $j - 1$. Consider some $v \in V$ and $b \in B_j$. We consider two possible cases:

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

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

We will need to use our "middle third" technique.



We can choose $c, d \in V$ 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 $|B_{j-1}| = c\frac{n}{(3/2)^j} \log n$.

Let x in $B_{j-1} \cap$ "middle third". Then $\ell(v,x) \leq \ell(v,c) + \ell(c,d) \leq \frac{2}{3}(\frac{3}{2})^j = (\frac{3}{2})^{j-1}$. 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.

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 $(1/3)(3/2)^j$.

But then w.h.p. B_j hits this path at some $x \in V$ such that $\ell(u,x), \ell(x,v) \leq (3/2)^{j-1}$. 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. □

3 Node-Weighted All-Pairs Shortest Paths

Here we prove a theorem by Chan [Cha10].

Theorem 3.1. *APSP with node weights can be computed in $O(n^{\frac{9+\omega}{4}})$ or $O(n^{2.84})$ time.*

The idea is to compute long paths ($> s$ hops) via a hitting set argument and running multiple calls to Dijkstra's algorithm, in a running time of $\tilde{O}(\frac{n^3}{s})$. Then, handle short paths ($\leq s$ hops) in $O(sn^{\frac{3+\omega}{2}})$ time via a specialized matrix multiplication.

Let G be a directed graph with node weights $w : V \to Z$. Suppose we just wanted to compute distances over paths of length two.

Let A be the *unweighted* adjacency matrix. Notice that $d_2(u,v) = w(u) + w(v) + \min\{w(j) \mid A[u,j] = A[j,v] = 1\}$.

Suppose we made two copies of A, and sorted one's columns by $w(j)$ in nondecreasing order, and the others rows by $w(j)$ in nondecreasing order.

Then it would suffice to compute $\min\{j \mid A[i,j] = A[j,k] = 1\}$, or the "minimum witnesses" matrix product. We use an algorithm provided by Kowaluk and Lingas [KL05]:

Lemma 3.1 (Kowaluk, Lingas '05)**.** *Minimum witnesses of A, B ($n \times n$ matrices) is in $O(n^{2.616})$ or $O(n^{2+\frac{1}{4-\omega}})$ time.*

Note that this algorithm has been improved on by Czumaj, Kowaluk, and Lingas [CKL07].

Proof. Let p be some parameter that we will choose later. Bucket A by columns into buckets of size p. Bucket B by rows into buckets of size p.

For every bucket $b \in \{1, \ldots, \frac{n}{p}\}$, compute $A_b \cdot B_b$ (boolean matrix product). This takes $O((\frac{n}{p})^2 p^\omega)$ time each, or $O(n^2 p^{\omega-2})$ time each. But there are $\frac{n}{p}$ of these, so this takes $O(\frac{n^3}{p^{3-\omega}})$ time total.

Then for all $i, j \in \{1, \ldots, \frac{n}{p}\}$, do the following. Let b_{ij} be the smallest b such that $(A_b \cdot B_b)[i,j] = 1$. Hence we can just try all the choices of k in bucket b_{ij}, and return the smallest k such that $A_b[i,k]B_b[k,j] = 1$. This is just n^2 exhaustive searches, so this step runs in $O(n^2 p)$ time.

Setting these equal and balancing, we get that we should set $p = n^{\frac{1}{4-\omega}}$ to make the overall time $O(n^{2+\frac{1}{4-\omega}})$. □

How can we compute distances for paths that are longer than two hops? For each $\ell \leq s$, we want to compute D_ℓ such that:

$$D_\ell[u,v] = d(u,v) - w(u) - w(v) \text{ if } \ell(u,v) = \ell$$
$$D_\ell[u,v] = \min_{j \in N(u)} \{w(j) + D_{\ell-1}[j,v]\}$$

This gives rise to a new matrix product! Suppose we are given $D_{\ell-1}$. Let $\overline{D}_{\ell-1}[u,v] = w(u) + D_{\ell-1}[u,v]$. Then we are interested in $(A \odot \overline{D}_{\ell-1})[u,v] = \min\{\overline{D}_{\ell-1}[j,v] \mid A[u,j] = 1\}$.

We can compute this product as follows. Again, let p be a parameter that we will choose later. Sort the columns of $\overline{D}_{\ell-1}$, using $O(n^2 \log n)$ time. Then partition each column into blocks of length p.

Let $D_b[u,v] = 1$ if $\overline{D}_{\ell-1}[u,v]$ is between the $\left(b\frac{n}{p}\right)^{th}$ and the $\left((b+1)\frac{n}{p}\right)^{th}$ element of column v.

Compute the boolean matrix product of A and D_b for all b. Notice that $(A \cdot D_b)[u,v] = 1$ iff there exists an x such that $A[u,x] = 1$ and $\overline{D}_{\ell-1}[x,v]$ is among the b^{th} block of p elements in the sorted order of the v^{th} column. We can finish via an exhaustive search, trying all j such that $D_{\ell-1}[j,v]$ is in the b^{th} block of column v.

This takes $O(\frac{n}{p}n^\omega)$ time for multiplications, and $O(n^2 p)$ time for the exhaustive search. This yields $O(n^{\frac{3+\omega}{2}})$ time after balancing. However, we need to do this s times.

The overall runtime is hence $O(n^{\frac{3+\omega}{2}}s + n^3/s)$, which becomes $O(n^{\frac{9+\omega}{4}})$ time after balancing.

References

[Cha10] T.M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010.

[CKL07] Artur Czumaj, Mirosław Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1):37–46, 2007.

[KL05] Miroslaw Kowaluk and Andrzej Lingas. Lca queries in directed acyclic graphs. In *Automata, Languages and Programming*, pages 241–248. Springer, 2005.

[YZ05] Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *FOCS*, pages 389–396, 2005.