All-Pairs Shortest Paths with Matrix Multiplication

Chandler Burfield

March 15, 2013

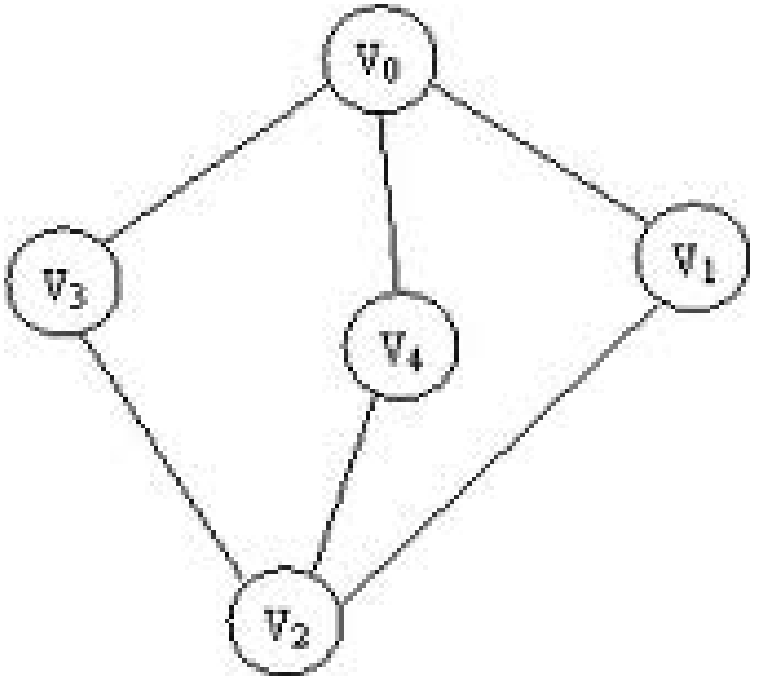
Outline

1. Seidel's algorithm
2. Faster Matrix Mulitplicaltion
   - Strassen's Algorithm
   - Speeding up Seidel's Algorithm
3. More Advanced APSP Algorithms that Use Matrix Multiplication

All-Pairs Shortest Paths

Problem

To find the shortest path between all vertices $v \in V$ for a graph $G = (V, E)$.



Definitions

A: the input to Seidel's algorithm

Let A be the adjacency matrix, an n x n boolean matrix where a 1 represents an edge between node i and node j in the graph G.

D: the output of Seidel's algorithm

Let D be the distance matrix, an n x n integer matrix with d_{ij} representing the length of the shortest path from vertex i to vertex j in the graph G.

Seidel's Algorithm

Seidel(A)

Let $Z = A \cdot A$

Let B be an n x n 0-1 matrix,

where $b_{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } (a_{ij} = 1 \text{ or } z_{ij} > 0) \\ 0 & \text{otherwise} \end{cases}$

IF $\forall i, j,\ i \neq j\ b_{ij} = 1$ then

Return D = 2B-A

Let T = Seidel(B)

Let $X = T \cdot A$

Return n x n matrix D,

where $d_{ij} = \begin{cases} 2t_{ij} & \text{if } x_{ij} \geq t_{ij} \cdot degree(j) \\ 2t_{ij} - 1 & \text{if } x_{ij} < t_{ij} \cdot degree(j) \end{cases}$

Function

Seidel's algorithm is an APSP algorithm that can be used on unweighted, undirected graphs.

Analysis of Seidel's Algorithm

Seidel(A)

Let $Z = A \cdot A$

Let B be an n x n 0-1 matrix,

where $b_{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } (a_{ij} = 1 \text{ or } z_{ij} > 0) \\ 0 & \text{otherwise} \end{cases}$

IF $\forall i, j,\ i \neq j\ b_{ij} = 1$ then

Return D = 2B-A

Let T = Seidel(B)

Let $X = T \cdot A$

Return n x n matrix D,

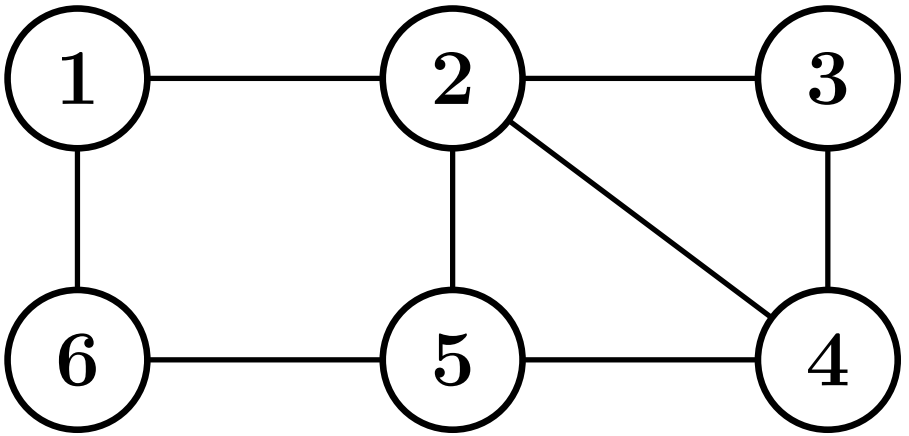
where $d_{ij} = \begin{cases} 2t_{ij} & \text{if } x_{ij} \geq t_{ij} \cdot degree(j) \\ 2t_{ij} - 1 & \text{if } x_{ij} < t_{ij} \cdot degree(j) \end{cases}$

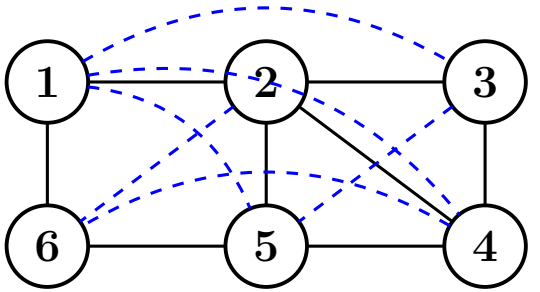
Runtime

$O(M(n)logn)$

where M(n) denotes the time necessary to multiply two n x n integer matricies

Example for Seidel's Algorithm





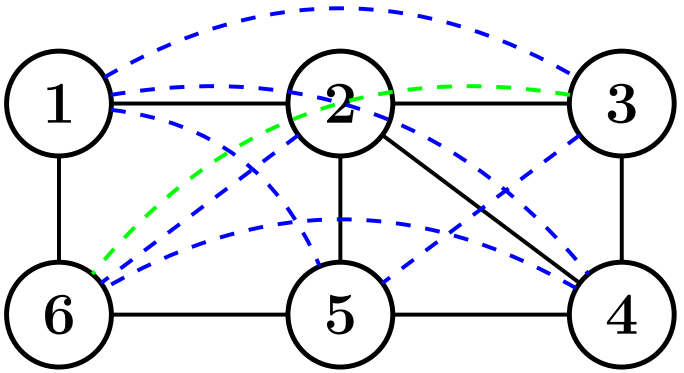
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Example for Seidel's Algorithm Continued

$$Z = AA = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 & 1 & 2 & 0 \\ 0 & 4 & 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 & 2 & 0 \\ 1 & 2 & 1 & 3 & 1 & 1 \\ 2 & 1 & 2 & 1 & 3 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Example for Seidel's Algorithm Continued



$$Z_2 = A_2A_2 = BB = \begin{pmatrix} 5 & 4 & 3 & 4 & 4 & 3 \\ 4 & 5 & 3 & 4 & 4 & 3 \\ 3 & 3 & 4 & 3 & 3 & 4 \\ 4 & 4 & 3 & 5 & 4 & 3 \\ 4 & 4 & 3 & 4 & 5 & 3 \\ 3 & 3 & 4 & 3 & 3 & 4 \end{pmatrix} \qquad B_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Example for Seidel's Algorithm Continued

$$T = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 \end{pmatrix}$$

$$X = AT = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 2 & 3 & 3 & 1 \\ 1 & 4 & 1 & 2 & 2 & 2 \\ 3 & 3 & 2 & 2 & 4 & 2 \\ 2 & 3 & 1 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 & 3 & 1 \\ 1 & 5 & 2 & 4 & 2 & 2 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 & 1 & 2 & 2 & 2 & 1 \\ 1 & 0 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 1 & 0 & 1 \\ 1 & 2 & 3 & 2 & 1 & 0 \end{pmatrix}$$

Correctness of Seidel's Algorithm

Lemma 1:
If the distance d_{ij} is even then the resulting distance is $2t_{ij}$ and $2t_{ij}$-1 otherwise.

Lemma 2:
Let i and j be a pair of distinct vertices in G. For any neighbor k of j, we have d_{ij} -1 $\leq d_{ik} \leq d_{ij} + 1$. More over, there exists a neighbor k of j with $d_{ik} = d_{ij} + 1$

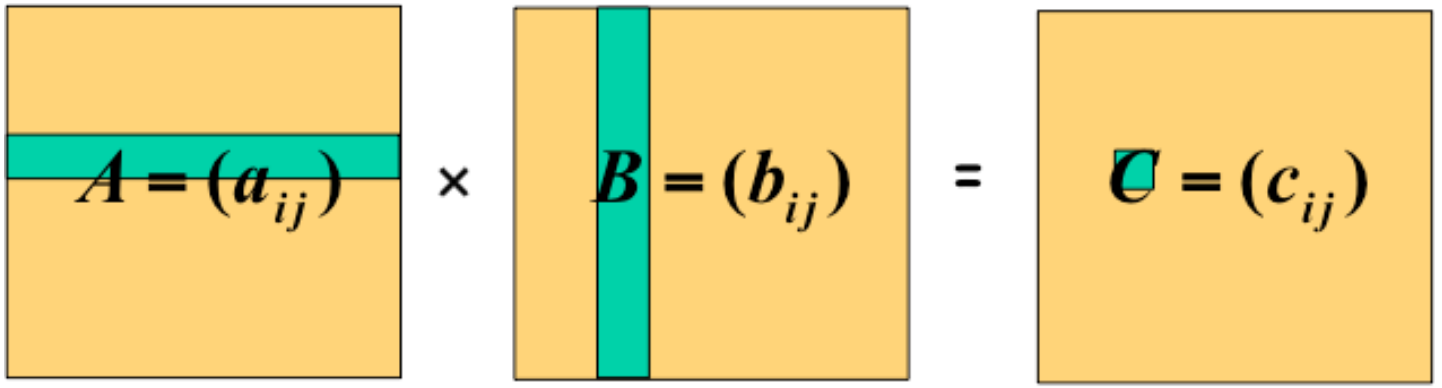
Lemma 3:
If d_{ij} is even then $t_{ik} \geq t_{ij}$ for all neighbors of k and j
If d_{ij} is odd then $t_{ik} \leq t_{ij}$ for all neighbors of k of j,
$t_{ik} < t_{ij}$ for at least one neighbor k of j

Lemma 4:
Let $\Gamma(j)$ be the set of neighbors of node j, then
d_{ij} is even iff $\sum_{k \in \Gamma(j)} t_{ik} \geq t_{ij} \cdot degree(j)$
d_{ij} is odd iff $\sum_{k \in \Gamma(j)} t_{ik} < t_{ij} \cdot degree(j)$

Square Matrix Multiplication

- Naively square matrix multiplication takes $O(n^3)$ time.
- Strassen's algorithm can preform it in $O(n^{2.807})$
- The fastest algorithm works in $O(n^{2.373})$

Naive Matrix Multiplication Algorithm



$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

This can be done in $O(n^3)$ time.

Naive Matrix Multiplication for 2x2 Matricies

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

8 multiplications

4 additions

Analysis of Algorithm

Recurrence Relation: $T(n) = 8T(n/2) + O(n^2)$

Runtime: $O(n^{log8/log2}) = O(n^3)$

Strassen's Algorithm for 2x2 Matrices

2 x 2 Matrices:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22})B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

7 multiplications

18 additions/subtractions

Strassen's Algorithm

Strassen for n x n Matrices

We can think of an n x n matrix as a 2 x 2 matrix whose elements are $\frac{n}{2}$ x $\frac{n}{2}$ matrices and apply the 2x2 algorithm recursively.

Analysis of Algorithm

Recurrence Relation:
$T(n) = 7T(n/2) + O(n^2)$

Runtime:
$O(n^{log7/log2}) = O(n^{2.807})$

Fast Matrix Multiplication and Seidel's Algorithm

- Recall that Seidel's Algorithm runs in $O(M(n)logn)$.
- Using Strassen's algorithm instead of the naive algorithm, we decrease the runtime from $O(n^3 logn)$ to $O(n^{2.81} logn))$.
- If we used the fastest algorithm available to date, we could achieve a runtime of $O(n^{2.373} logn)$.

Other APSP Algorithms Involving Matrix Multiplication

Consider integer edge weights on the set $\{0,1,..,M\}$.

- APSP in weighted, undirected graphs: Shoshan-Zwick '99, $O(Mn^{2.38})$
- APSP in weighted, directed graphs: Zwick '98, $O(M^{0.68}n^{2.58})$

APSP Algorithms with Integer Weights {0, 1, 2,..., M} that Use Matrix Multiplication

Graph	**Algorithm**	**Runtime**
Unweighted and Undirected	Seidel '95	$n^{2.38}$
Weighted and Undirected	Shoshan-Zwick '99	$Mn^{2.38}$
Weighted and Directed	Zwick '98	$M^{0.68}n^{2.58}$