Universiteit
Leiden
The Netherlands

Finding Shortest Paths in Parallel

Romke Bak

Supervisors:
Dr. Alfons W. Laarman
Dr. Rudy van Vliet

Bachelor Thesis

Opleiding Informatica

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

17/7/2020

Abstract

This thesis presents parallel algorithms for two extensions of the all-pairs shortest path problem (APSP) in directed graphs: the strong APSP problem (SAPSP), which asks to compute a shortest path for each pair of vertices in a directed unweighted graph, and the all shortest paths problem (ASPP), which asks to compute for each pair (p, q) of vertices an acyclic subgraph in which each path from p to q is a shortest path in the original graph and each shortest path from p to q in the original graph corresponds to a path in the subgraph. We describe a solution to both problems using an algebraic framework and derive the parallel algorithms in a similar way as was done for the APSP problem. The algorithms use $\mathcal{O}(n^4)$ resp. $\mathcal{O}(n^5)$ processors and run in $\mathcal{O}(\log^3 n)$ resp. $\mathcal{O}(\log^2 n)$ time.

Contents

1 Introduction

Graphs are fundamental data structures. They are used to model a wide variety of systems, such as computer networks, social networks, roads and the world wide web. As the amount of available storage grows exponentially (see Figure 1), we can expect that the size of graphs computers need to work with also roughly grows exponentially. To keep answering fundamental questions concerning these ever-growing graphs, we must use modern computers. As we see in section 2.2, modern computers make extensive use of parallelism. To efficiently utilize parallelism, new solutions to existing problems must be found. For example, J. L. Träff and C. D. Zaroliagis give a parallel algorithm [TZ00] for the single-source shortest path problem on planar directed graphs.

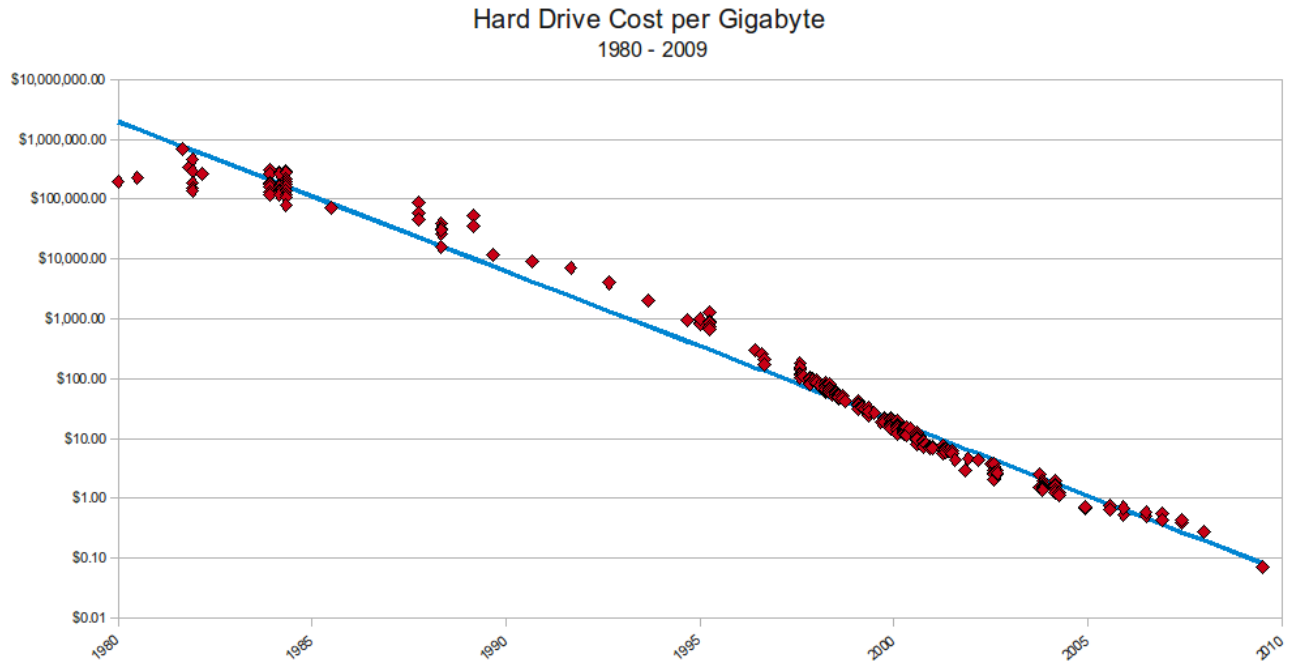

Figure 1: Price per gigabyte storage has been exponentially decreasing over time
Source: mkomo.com/cost-per-gigabyte

One of the most important questions concerning graphs is how to find a shortest path between any two nodes. However, there may exist multiple shortest paths between the same nodes. There are situations where knowledge of which possible shortest paths exist is desirable. E.g when a shortest path is subject to specific requirements. This might occur in network packet routing [RBW07] where network nodes might get overloaded and therefore temporarily unavailable. Knowledge of which possible shortest paths exist might help the sender of a packet choose a shortest path such that in the event that the packet encounters an unavailable node, the likelihood of another path being available and as short as the original (shortest) path is as high as possible.

This motivates the research question:

- How can we compute a single shortest path between every pair of vertices in a directed unweighted graph (SAPSP) and can we determine all shortest paths that exist (ASP), while efficiently utilising parallelism?

This leads us to the following sub-questions:

- Which resources do these calculations require?
- What kind of data structures do these calculations require?
- How do we prove the correctness of a given solution?

In this thesis, we contribute two algorithms concerning directed graphs that effectively make use of parallelism. The first algorithm is a solution to the strong all-pairs shortest path problem. This is the problem of finding for each pair of vertices (p, q) a shortest path from p to q. The second algorithm solves the more complex problem of computing all shortest paths, which we call the all shortest paths problem (ASPP):
for each pair of vertices (p, q), a data structure is determined in which all possible shortest paths from p to q are encoded. From this data structure we can construct an acyclic subgraph in which all paths from p to q are exactly all shortest paths from p to q in the original graph. Both algorithms are in Nick's Class (see paragraph 2.2).

In section 2.1, we introduce how to express resource requirements for algorithms and in 2.2 how to formulate what 'efficiently parallelizable' means. We then introduce the mathematical structure of rings and semirings in 2.3. In 2.4 we give a concrete example of a semiring: the tropical semiring. We show that this structure can be used as entries in a matrix in an algorithm to compute shortest distances between vertices in a directed unweighted graph. We also prove the correctness of this algorithm and show that it is efficiently parallelizable.

In section 3.1, we extend the tropical semiring and show that we can use this extension in an algorithm that returns for all pairs of vertices a shortest path. Then, in section 3.2, we give another extension of the tropical numbers in which we can encode multiple possible paths and use this structure in an algorithm that returns for all pairs of vertices an encoding of all shortest paths. We prove the correctness of this algorithm and show that it is efficiently parallelizable. We then show how we can extract from a single structure containing all shortest paths between a pair of vertices a subgraph in which the paths between these vertices are exactly all shortest paths in the original graph.

2 Background

2.1 Complexity of Algorithms

The complexity of an algorithm refers to the amount of resources that are required to run the algorithm. For example, the time complexity refers to the amount of time that is required to run an algorithm. Complexity is usually expressed in terms of the size of the input for an algorithm. However, we are often only interested in the asymptotic behavior of its resource requirements. This motivates Definition 2.1 for the 'Big $\mathcal{O}$ notation':

Definition 2.1. Big $\mathcal{O}$ notation
Let f and g be two functions where f expresses an amount of computational resources required in terms of the size of the input for an algorithm. We say that f is in $\mathcal{O}(g)$ if there exist an $m > 0$ and $n' > 0$ such that $f(n) \leq mg(n)$ for all $n > n'$. This means that the asymptotic resource requirements for the algorithm are bound by the asymptotic behavior of g.

2.2 Parallel Computing and Nick's Class

Moore's law [Moo65] refers to the exponential growth in number of transistors per integrated chip. As we can see in Figure 2, up to around the year 2005, this increase in number of transistors has been translated in an increasing performance per thread. From 2005 and forward, performance per thread has been plateauing. In contrast, the number of cores per chip has picked up an exponential growth. This means that computational power still has been increasing, as computational power is proportional to the performance per thread times the number of threads that can be executed in parallel (the number of cores). However, sequentially designed algorithms cannot make use of multiple cores. This leads to a bottleneck of how large the data size can be on which these algorithms operate. To circumvent this bottleneck, new algorithms need to be designed that are expressed in terms of computations that can be performed in parallel.
In order to reason over the complexity of parallel algorithms, we not only need to take into account what their run time complexity is, but also the complexity of the number of processors they need. The complexity of the computational resources required is then the product of these two complexity classes. For example, if an algorithm needs 5 processors and takes 7 seconds to finish, it needs a total of $5 \times 7 = 35$ CPU seconds.
Typically, we try to find parallel algorithms for problems in the polynomial complexity class (P), in other words problems that can be solved in $\mathcal{O}(n^k)$ time, for some positive constant k. The goal of parallelization of a problem in P is to decrease the run time complexity as much as possible, ideally to a subpolynomial complexity, meaning that for any arbitrarily small $\epsilon > 0$, the time complexity is in $\mathcal{O}(n^\epsilon)$. Intuitively we can think of this as 'near constant' run time. An example of such a time complexity is $\mathcal{O}(\log n)$. This inherently means using a polynomial number of processors, since a polynomial amount of computational resources (= no. of processors $\times$ run time) is still required.[1]

[1] Otherwise, a sequential computer emulating a parallel computer by time sharing would yield a (sequential) subpolynomial solution to a problem which requires polynomial time. This is a contradiction.

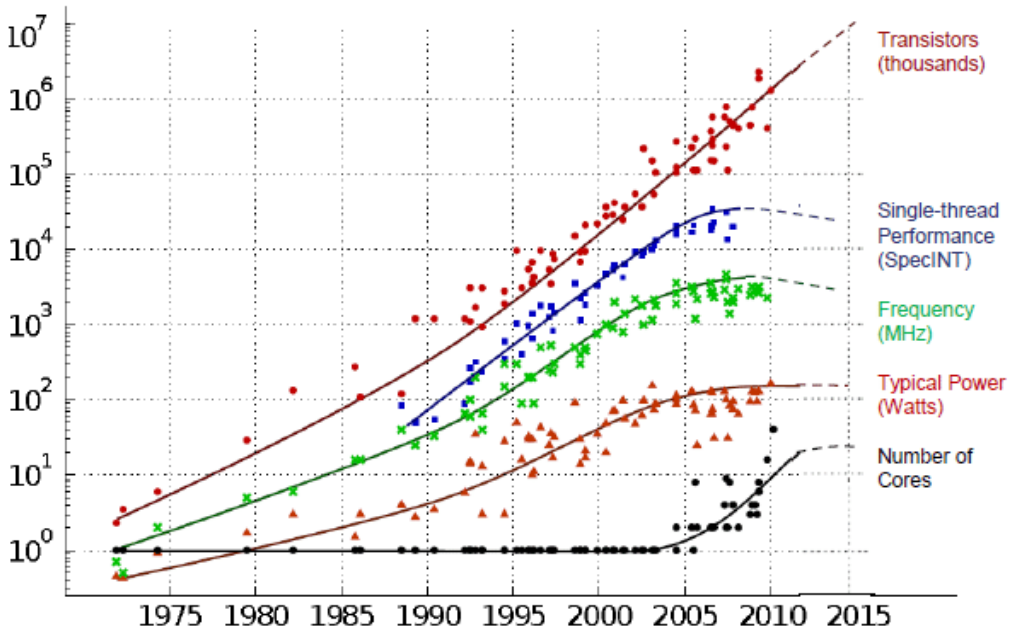

Figure 2: Computational power keeps growing exponentially, but it is only being put in more cores.[Mic19]

This leads to Definition 2.2 of what it means for an algorithm to be efficiently parallelizable:

Definition 2.2. Nick's Class
An algorithm is considered efficiently parallelizable if it runs in $\mathcal{O}(\log^c n)$ (polylogarithmic) time for some constant $c > 0$, using $\mathcal{O}(n^k)$ (polynomial) number of processors for some constant $k > 0$. The set of problems that can be solved by such an algorithm is called Nick's Class (NC).

Nick's Class is named after Nick Pippenger. This term was coined by Stephen Cook[Coo85], who also did extensive research on the topic of parallelism.

2.3 Rings, Semirings and Matrices over Semirings

Matrices are well known mathematical objects. Usually they contain integer, real or complex valued entries and matrix multiplication is then defined in terms of multiplying and adding the individual entries. However, confining matrices to contain only integers, reals or complex numbers is too restrictive for our purpose of computing shortest paths: we will use matrices containing tropical numbers (Definition 2.6), paths (Definition 3.2) and tuples of sets of edges (Definition 3.6). These kinds of matrices will form the basis for the algorithms we propose. As long as there is a notion of how to add and multiply the entries of a matrix, we can use the definition of matrix multiplication. This motivates Definition 2.3 of what we mean by 'entries that can be added and multiplied'.

Definition 2.3. Ring
A ring is a triple $(R, \oplus, \otimes)$, where R is a set of elements and $\oplus$ and $\otimes$ are functions of type $\oplus, \otimes : R \times R \to R$ (they take two elements of R and return a new element of R) such that the following identities hold:

$\forall x, y, z \in R :$

$x \oplus y = y \oplus x$	commutativity
$(x \oplus y) \oplus z = x \oplus (y \oplus z)$	associativity
$(x \otimes y) \otimes z = x \otimes (y \otimes z)$	associativity
$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$	(left) distributivity
$(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$	(right) distributivity
$\exists 0 \in R : \forall x \in R : x \oplus 0 = x$	existence of neutral element
$\exists 1 \in R : \forall x \in R : x \otimes 1 = 1 \otimes x = x$	
$\forall x \in R : \exists x' \in R : x \oplus x' = 0$	existence of additive negatives

The function $\oplus$ is referred to as addition, and $\otimes$ as multiplication. We often refer simply to R as the ring (instead of referring to $(R, \oplus, \otimes)$) when the operations $\oplus$ and $\otimes$ are implicitly given. See the course notes [Ste17] for more information on rings.

It is easy to check that the triples $(\mathbb{Z}, +, \times)$, $(\mathbb{R}, +, \times)$ and $(\mathbb{C}, +, \times)$, with $\mathbb{Z}, \mathbb{R}$ and $\mathbb{C}$ respectively the integers, reals and complex numbers, are rings.
Now, we can define in 2.4 what it means to have a matrix where the entries are not necessarily integer, real or complex valued. In this thesis, we are only interested in square matrices, and will therefore use the terms 'square matrix' and 'matrix' interchangeably.

Definition 2.4. (Square) Matrices over Rings
Define the set $\mathrm{Mat}_n(R)$ of $n \times n$ (square) matrices over a ring $(R, \oplus, \otimes)$ to be the set of all square $n \times n$ grids where each entry contains an element of the ring R. I.e. for a matrix $M \in \mathrm{Mat}_n(R)$, we have $M_{ij} \in R$ for all $1 \leq i, j \leq n$.

Define addition and multiplication for matrices $P, Q \in \mathrm{Mat}_n(R)$ analogue to 'normal' matrices:

$$(P+Q)_{ij} = P_{ij} \oplus Q_{ij}$$

and

$$(P \times Q)_{ij} = \oplus_{k=1}^{n}(P_{ik} \otimes Q_{kj})$$

It turns out that the set of matrices over a ring in itself also form a ring, as formally stated in lemma 1.

Lemma 1. The triple $(\mathrm{Mat}_n(R), +, \times)$ is a ring.

The proof for this follows directly from the definitions and will therefore not be fully given. However, its neutral elements will be given in lemma 2 and it will be proven that $\mathrm{Mat}_n(R)$ is associative in its multiplication in lemma 3. The rest of the requirements for a ring should then be easy to check. With associativity, we conclude in corollary 3.1 that we can perform exponentiation.

Lemma 2. With $0, 1 \in R$ the neutral elements for respectively addition and multiplication in R, the neutral elements in $\mathrm{Mat}_n(R)$ are:

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad \text{neutral for multiplication}$$

$$Z_{ij} = 0 \qquad \text{neutral for addition}$$

I is called the identity matrix and Z the zero matrix.

Proof. With $M \in \mathrm{Mat}_n(R)$ any matrix:

$$(M+Z)_{ij} = M_{ij} \oplus 0 = M_{ij}$$

$$\begin{aligned} (M \times I)_{ij} &= \oplus_{k=1}^{n}(M_{ik} \otimes I_{kj}) \\ &= M_{ij} \otimes I_{jj} && \text{For all } k \neq j, \text{ we know } I_{kj} = 0 \\ &= M_{ij} \otimes 1 \\ &= M_{ij} \end{aligned}$$

□

Lemma 3. Let R be a ring. For any three elements $A, B, C \in \mathrm{Mat}_n(R)$, we have $(A \times B) \times C = A \times (B \times C)$. This proves associativity of $\mathrm{Mat}_n(R)$.

Proof.

$$
\begin{aligned}
(A \times (B \times C))_{ij} &= \bigoplus_{k=1}^{n} A_{ik} \otimes (B \times C)_{kj} \\
&= \bigoplus_{k=1}^{n} A_{ik} \otimes \left(\bigoplus_{l=1}^{n} B_{kl} \otimes C_{lj} \right) \\
&= \bigoplus_{k=1}^{n} \bigoplus_{l=1}^{n} A_{ik} \otimes B_{kl} \otimes C_{lj} && \text{distributivity} \\
&= \bigoplus_{l=1}^{n} \bigoplus_{k=1}^{n} A_{ik} \otimes B_{kl} \otimes C_{lj} && \text{commutativity} \\
&= \bigoplus_{l=1}^{n} \left(\bigoplus_{k=1}^{n} A_{ik} \otimes B_{kl} \right) \otimes C_{lj} && \text{distributivity} \\
&= \bigoplus_{l=1}^{n} (A \times B)_{il} \otimes C_{lj} \\
&= ((A \times B) \times C)_{ij}
\end{aligned}
$$

□

Corollary 3.1. For any matrix $M \in \mathrm{Mat}_n(R)$ with R an arbitrary ring, the exponent M^p is well defined for all $p \in \mathbb{Z}_{\geq 1}$. If we define $M^0 = I$ with

$$
I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}
$$

where $0, 1 \in R$ are the neutral elements for respectively addition and multiplication, the exponent M^p is well defined for all $p \in \mathbb{Z}_{\geq 0}$.

Proof. The expansion of the expression $M^p = M \times \cdots \times M$ does not specify the placement of brackets. Given associativity, the placement of brackets has no impact on the result, and therefore M^p is well defined. □

Matrices over rings have been used to solve many graph theory problems, such as computing the transitive closure [FM71] for every vertex, and the all-pairs bottleneck path problem [VWY09], in which we interpret the weights of edges of a weighted graph as maximum capacities and want to find the maximum capacity between each pair of vertices.
As we shall see shortly, we sometimes work with structures that don't have additive inverses. These structures are called semirings, as given in Definition 2.5 and we conclude in corollary 3.2 that semirings also have a well defined exponentiation.

Definition 2.5. Semiring.
A semiring is a ring without the requirement that each element has an additive inverse. Semirings are sometimes called rigs, a ri*n*g without *n*egatives.

Corollary 3.2. In the proof for lemma 3, the existence of additive negatives is not used. We conclude that for any matrix $M \in \mathrm{Mat}_n(R)$ with R an arbitrary semiring, the exponent M^p is well defined for all $p \in \mathbb{Z}_{\geq 0}$.

2.4 Tropical Numbers and Shortest Distances

One problem that can be solved in NC is the problem of finding the shortest distance between two vertices in a directed graph. This problem is solved by iteratively squaring a matrix with entries over the tropical semiring T. The tropical semiring T is a concrete example of a semiring, popularized by Imre Simon [Sim94] and is given in Definition 2.6:

Definition 2.6. Tropical Semiring
Define the tropical semiring $(T, \oplus, \otimes)$ to be $T = \mathbb{Z}_{\geq 0} \cup \{\infty\}$ and define for all $p, q \in T$ addition and multiplication as follows:

$$p \oplus q = \min(p, q) = \begin{cases} p & p \leq q \\ q & p > q \end{cases}$$

where $x < \infty$ for all $x \in \mathbb{Z}_{\geq 0}$

$$p \otimes q = \begin{cases} p + q & p \neq \infty \text{ and } q \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

The tropical semiring is also called the tropical numbers.

We can trivially check that this is a semiring with ∞ the neutral element for addition and 0 the neutral element for multiplication. (Note that conventionally, '1' is used as the symbol for the neutral element for multiplication. In the case of the tropical numbers, the actual integer $0 \in \mathbb{Z}$ is the neutral element for multiplication as we have $p \otimes 0 = p$ for all $p \in T$.) A matrix over T (Definition 2.7) can be used to compute the shortest distance between any two vertices in a directed graph.

Definition 2.7. Tropical Matrix for a Directed Graph
For a directed unweighted graph $G = (V, E)$, with $V = \{1, \cdots, n\}$, $E \subseteq V \times V$ and T the tropical numbers, define the tropical matrix $M \in \mathrm{Mat}_n(T)$ for G to be

$$M_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j \text{ and } (i, j) \in E \\ \infty & \text{otherwise} \end{cases}$$

with $i, j \in \{1, \cdots, n\}$.

As we will need to express the shortest distance between two nodes as a mathematical value, we give the function sd in Definition 2.8:

Definition 2.8. With $\mathrm{sd} : V \times V \to T$ and $i, j \in V$, define $\mathrm{sd}(i, j)$ to be the shortest distance from i to j, or ∞ if there exists no path. Note that for all $v \in V$ we have $\mathrm{sd}(v, v) = 0$ (the shortest distance from v to itself is always equal to 0).

We can interpret the matrix M from Definition 2.7 as follows: for any pair of vertices $i, j \in V$, we have $M_{ij} = \mathrm{sd}(i, j)$ if $\mathrm{sd}(i, j) \leq 1$ and $M_{ij} = \infty$ if $\mathrm{sd}(i, j) > 1$. In other words, M contains exactly all shortest distances of length 1 or shorter. In lemma 4, we show that we can increase this number by exponentiation of M and conclude in corollary 4.1 that we can find all shortest distances.

Lemma 4. Let M be the tropical matrix for a graph G. For all $p \in \mathbb{Z}_{\geq 1}$ and $i, j \in V$ we have $(M^p)_{ij} = \mathrm{sd}(i,j)$ if $\mathrm{sd}(i,j) \leq p$ and $(M^p)_{ij} = \infty$ if $\mathrm{sd}(i,j) > p$. In other words, M^p contains exactly all shortest distances of length p or shorter.

Proof. We prove this by induction on p. We know that the lemma holds for $p = 1$. Assume M^{p-1} contains exactly all shortest distances of length $p-1$ or shorter. For any entry $(M^p)_{ij}$, we have:

$$\begin{aligned}(M^p)_{ij} &= (M^{p-1} \times M)_{ij} \\ &= \bigoplus_{k=1}^{n} (M^{p-1})_{ik} \otimes M_{kj} \\ &= \min\{(M^{p-1})_{ik} \otimes M_{kj} \mid k \in \{1, \cdots, n\}\}\end{aligned}$$

If $\mathrm{sd}(i,j) \leq p-1$ then for $k = j$ we have $(M^{p-1})_{ik} \otimes M_{kj} = (M^{p-1})_{ij} + M_{jj} = (M^{p-1})_{ij} = \mathrm{sd}(i,j)$. There is no k for which this value is lower, because if such a vertex k existed, we would have $(M^{p-1})_{ik} + M_{kj} < (M^{p-1})_{ij}$ and therefore $\mathrm{sd}(i,k) + \mathrm{sd}(k,j) < \mathrm{sd}(i,j)$, which is a contradiction. We conclude that the lowest value of the set $\{(M^{p-1})_{ik} \otimes M_{kj} \mid k \in \{1, \cdots, n\}\}$ will be for $k = j$ and we have $(M^p)_{ij} = (M^{p-1})_{ij} = \mathrm{sd}(i,j)$.

If $\mathrm{sd}(i,j) = p$, there exists a shortest path of length p from i to j. Let l be the one but last vertex of such a path. The shortest distance from i to l must then be $p-1$ and the shortest distance from l to j must be 1. By our induction assumption for M^{p-1}, we have $p-1 = \mathrm{sd}(i,l) = (M^{p-1})_{il}$ and $1 = \mathrm{sd}(l,j) = M_{lj}$. For $k = l$ we have $(M^{p-1})_{ik} + M_{kj} = (p-1) + 1 = p = \mathrm{sd}(i,j)$. There is no k for which this value is lower, because that would imply that $(M^{p-1})_{ik} + M_{kj} = \mathrm{sd}(i,k) + \mathrm{sd}(k,j) < \mathrm{sd}(i,j)$. Therefore, the lowest value of the set $\{(M^{p-1})_{ik} \otimes M_{kj} \mid k \in \{1, \cdots, n\}\}$ will be for $k = l$ and we have $(M^p)_{ij} = (M^{p-1})_{il} + M_{lj} = (p-1) + 1 = p = \mathrm{sd}(i,j)$

If $\mathrm{sd}(i,j) > p$, assume for some k that $(M^{p-1})_{ik} + M_{kj} < \infty$. This implies that $\mathrm{sd}(i,k) = (M^{p-1})_{ik} \leq p-1$ and $\mathrm{sd}(k,j) = M_{kj} \leq 1$ and therefore $\mathrm{sd}(i,k) + \mathrm{sd}(k,j) \leq (p-1) + 1 = p < \mathrm{sd}(i,j)$. We know that the inequality $\mathrm{sd}(i,k) + \mathrm{sd}(k,j) < \mathrm{sd}(i,j)$ is a contradiction, so for all k, we have $(M^{p-1})_{ik} \otimes M_{kj} = \infty$ and therefore $(M^p)_{ij} = \infty$. □

Corollary 4.1. As no shortest distance can be more than $n-1$, for the matrix M^q with $q \geq n-1$ we have $(M^q)_{ij} = \mathrm{sd}(i,j)$ for all $i, j \in V$.

We now have a neat way to solve the all-pairs shortest path problem. Tropical matrix exponentiation is a well known trick to compute shortest distances [MS85]. It was first described by A. Shimbel [Shi53], even though the concept of tropical numbers was only later formalised. The algorithm for finding shortest distances using tropical matrix exponentiation is very similar to the Floyd-Warshall algorithm [Flo62] for solving the APSP problem. The next step, as presented in lemma 5 and its corollary 5.1, is to show that there exists an algorithm in Nick's Class (Definition 2.2) to compute these computations

Lemma 5. Assume that for a ring R, addition resp. multiplication can be computed in $\mathcal{O}(t_\oplus)$ resp. $\mathcal{O}(t_\otimes)$ time if allowed $\mathcal{O}(p_\oplus)$ resp. $\mathcal{O}(p_\otimes)$ processors.
With $M \in \mathrm{Mat}_n(R)$, the matrix M^2 can be computed in $\mathcal{O}(t_\oplus \log n + t_\otimes)$ time if allowed $\mathcal{O}(n^3(p_\oplus + p_\otimes))$ processors.
The matrix M^q with q a power of 2 can be computed in $\mathcal{O}((t_\oplus \log n + t_\otimes) \log q)$ time if allowed $\mathcal{O}(n^3(p_\oplus + p_\otimes))$ processors.

Proof. To compute a single entry $(M^2)_{ij} = \bigoplus_{k=1}^{n} M_{ik} \otimes M_{kj}$, we compute $M_{ik} \otimes M_{kj}$ for all k in parallel in $\mathcal{O}(t_\otimes)$ time with $\mathcal{O}(np_\otimes)$ processors. This leaves us with a list $s = [s_1, \cdots, s_n]$ (where $s_k = M_{ik} \otimes M_{kj}$) which we want to sum together. We can halve the size of this list by computing $s' = [s_{2i-1} \oplus s_{2i} \mid 1 \leq i \leq n/2]$ in $\mathcal{O}(t_\oplus)$ time using $\mathcal{O}(np_\oplus)$ processors. Performing this halving $\log n$ times, we end up with a the value for M_{ij} in $\mathcal{O}(t_\oplus \log n + t_\otimes)$ time if allowed $\mathcal{O}(\max(np_\oplus, np_\otimes)) = \mathcal{O}(n(p_\oplus + p_\otimes))$ processors. Doing this in parallel for all n^2 entries in M^2, we need a total of $\mathcal{O}(t_\oplus \log n + t_\otimes)$ time if allowed $\mathcal{O}(n^3(p_\oplus + p_\otimes))$ processors.
We can then compute M^q by repeatedly squaring M. If we do this, we get the matrices
$M, M^2, M^4, M^8, M^{16}, \cdots$. Therefore, we need to square M a total of $\log q$ times with itself to get to M^q. This leaves us with $\mathcal{O}((t_\oplus \log n + t_\otimes) \log q)$ time if allowed $\mathcal{O}(n^3(p_\oplus + p_\otimes))$ processors. □

Corollary 5.1. The matrix $M^{n-1} = M^q$, with M the tropical matrix for some graph and $q = 2^{\mathrm{ceil}(\log(n-1))}$, can be computed in $\mathcal{O}(\log^2 n)$ time, using $\mathcal{O}(n^3)$ processors, assuming normal addition and taking the minimum of two integers can be done on one processor in constant time. As $\mathcal{O}(q) = \mathcal{O}(n)$, this follows trivially from lemma 5. This proves that finding shortest distances is in Nick's Class.

3 Parallel shortest path finding

3.1 A solution to the Strong All-pairs Shortest Path Problem

Here, we give a solution to the strong all-pairs shortest path problem, as in Definition 3.1. This problem is solved by extending the tropical numbers to a semiring $(S, \oplus, \otimes)$ where intuitively (some of) the elements of S are shortest paths.

Definition 3.1. Strong All-pairs Shortest Path Problem
Given a directed graph $G = (V, E)$, find for all pairs of vertices $v, w \in V$ a shortest path from v to w, or ∞ if no path exists from v to w.

To solve this problem, we will represent a path as a string of edges. With this notion, we give the shortest path semiring in Definition 3.2:

Definition 3.2. Shortest Path Semiring of a Graph
Given a graph $G = (V, E)$, interpret E as an alphabet and let E^* be the Kleene closure, the set of all strings over E. Define the shortest path semiring $(S, \oplus, \otimes)$ for G as:

$$S = \{\infty\} \cup E^*$$

with $a, b \in S$:

$$a \oplus b = \text{shortlex}(a, b)$$

$$a \otimes b = \begin{cases} \text{concat}(a, b) & a \neq \infty \text{ and } b \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

The function shortlex(a, b) returns the 'smallest' of its arguments a and b where $a < b$ if a is a shorter string than b, or if they are the same length and a is lexicographically smaller than b. We also define: $\forall a \in E^* : a < \infty$. This equips S with a well defined addition.
Concatenation is as usual: with $a = a[1] \cdots a[l], b = b[1] \cdots b[m]$ we have

$$(a \otimes b)[i] = \begin{cases} a[i] & 1 \leq i \leq l \\ b[i-l] & l < i \leq l + m \end{cases}$$

S forms a semiring with ∞ as the neutral element for addition and ϵ the neutral element for multiplication. Note that the elements of S are strings of edges and do not necessarily form valid paths in the graph, i.e. the target and source vertex of two consecutive edges are not guaranteed to be the same. This 'problem' will vanish in lemma 8.
We see that this is structurally similar to the tropical numbers. We formalize this by giving a homomorphism from S to the tropical numbers in lemma 6. Intuitively, a homomorphism (Definition 3.3) is a function which 'preserves structure'.

Definition 3.3. Homomorphism
For two semirings $(R, \oplus_R, \otimes_R)$ and $(R', \oplus_{R'}, \otimes_{R'})$, a (semiring) homomorphism is a function $f : R \to R'$ such that:

$$f(1_R) = 1_{R'} \qquad \text{preservation of neutral element for multiplication}$$
$$f(x \oplus_R y) = f(x) \oplus_{R'} f(y) \qquad \text{distributivity over addition}$$
$$f(x \otimes_R y) = f(x) \otimes_{R'} f(y) \qquad \text{distributivity over multiplication}$$

Lemma 6. The function $f : S \to T$ with S the shortest path semiring and T the tropical numbers given by $p \mapsto \text{len}(p)$ for all paths $p \in S \setminus \{\infty\}$ and $\infty \mapsto \infty$ is a homomorphism.

Proof. The function f preserves the neutral element for multiplication: $f(\epsilon) = \text{len}(\epsilon) = 0$. It distributes over addition and multiplication:

$$\begin{aligned}
&\text{with } p, q \in S \setminus \{\infty\} \\
f(p \oplus_S q) &= \text{len}(p \oplus_S q) \\
&= \min\{\text{len}(p), \text{len}(q)\} \\
&= \min\{f(p), f(q)\} \\
&= f(p) \oplus_T f(q) \\
\\
f(p \oplus_S \infty) &= f(p) = f(p) \oplus_T \infty \\
&= f(p) \oplus_T f(\infty) \\
\\
f(p \otimes_S q) &= \text{len}(\text{concat}(p, q)) \\
&= \text{len}(p) + \text{len}(q) = \text{len}(p) \otimes_T \text{len}(q) \\
&= f(p) \otimes_T f(q) \\
\\
f(p \otimes_S \infty) &= f(\infty) = \infty \\
&= \text{len}(p) \otimes_T \infty \\
&= f(p) \otimes_T f(\infty)
\end{aligned}$$

This proves f to be a homomorphism. □

Just like the tropical matrix for a directed graph contains exactly all shortest distances of length 1 or shorter, we give the shortest path matrix for a directed graph in Definition 3.4 such that it contains for each pair of vertices a shortest path iff the shortest distance between the pair of vertices is 1 or shorter.

Definition 3.4. Shortest Path Matrix for a Directed Graph
For a graph $G = (V, E)$, with $V = \{1, \cdots, n\}$, $E \subseteq V \times V$ and S the shortest path semiring of G, define the shortest path matrix $M \in \text{Mat}_n(S)$ for G to be

$$M_{ij} = \begin{cases} \epsilon & i = j \\ (i, j) & i \neq j \text{ and } (i, j) \in E \\ \infty & i \neq j \text{ and } (i, j) \notin E \end{cases}$$

The shortest path from any vertex to itself is the empty path ϵ of length 0 and the shortest path between any two distinct vertexes i, j connected by an edge is the (singled edge) path (i, j) of length 1. Therefore, for each pair $i, j \in V$ we have $M_{ij} = \text{sd}(i, j)$ if $\text{sd}(i, j) \leq 1$ and $M_{ij} = \infty$ if $\text{sd}(i, j) > 1$.
This matrix can be used to compute shortest paths, as we see in lemma 8, and its corollary 8.1. To prove this lemma, we use lemma 7 which states that if we apply f componentwise on the entries of a matrix, we get a matrix homomorphism.

Lemma 7. The homomorphism $f : S \to T$ with S the shortest path semiring and T the tropical numbers as given in lemma 6 can be lifted to a homomorphism between matrix rings $f' : \mathrm{Mat}_n(S) \to \mathrm{Mat}_n(T)$ by applying f componentwise on all entries of $M \in \mathrm{Mat}_n(S)$: $f'(M)_{ij} = f(M_{ij})$.

Proof. We show that f' preserves the neutral element (the identity matrix) and distributes over addition and multiplication:

$$I_{ij} = \begin{cases} \epsilon & i = j \\ \infty & i \neq j \end{cases} \qquad I \text{ is the identity matrix in } \mathrm{Mat}_n(S)$$

$$f'(I)_{ij} = \begin{cases} f(\epsilon) = \mathrm{len}(\epsilon) = 0 & i = j \\ f(\infty) = \infty & i \neq j \end{cases} \qquad f'(I) \text{ is the identity matrix in } \mathrm{Mat}_n(T)$$

$$\begin{aligned} f'(P+Q)_{ij} &= f((P+Q)_{ij}) = f(P_{ij} \oplus_S Q_{ij}) \\ &= f(P_{ij}) \oplus_T f(Q_{ij}) \\ &= f'(P)_{ij} \oplus_T f'(Q)_{ij} \\ &= (f'(P) + f'(Q))_{ij} \end{aligned}$$

$$\begin{aligned} f'(P \times Q)_{ij} &= f((P \times Q)_{ij}) \\ &= f\left(\bigoplus_{k=1}^{n} P_{ik} \otimes Q_{kj}\right) \\ &= \bigoplus_{k=1}^{n} f(P_{ik}) \otimes f(Q_{kj}) && f \text{ distributes because it is a homomorphism} \\ &= \bigoplus_{k=1}^{n} f'(P)_{ik} \otimes f'(Q)_{kj} \\ &= (f'(P) \times f'(Q))_{ij} \end{aligned}$$

□

Lemma 8. Let M be the shortest path matrix for a graph G. Then for all $p \in \mathbb{Z}_{\geq 1}$ and all $i, j \in V$ the entry $(M^p)_{ij}$ contains a shortest path from i to j if $\mathrm{sd}(i,j) \leq p$ and $(M^p)_{ij} = \infty$ if $\mathrm{sd}(i,j) > p$.

Proof. We need to prove that:

- all entries $(M^p)_{ij} \neq \infty$ contain a valid path from i to j
- for all $i, j \in V$ we have $f((M^p)_{ij}) = \mathrm{sd}(i,j)$ if $\mathrm{sd}(i,j) \leq p$ and $(M^p)_{ij} = \infty$ if $\mathrm{sd}(i,j) > p$ (with f from lemma 6)

This combined gives us that if $\mathrm{sd}(i,j) \leq p$ we have $f((M^p)_{ij}) = \mathrm{sd}(i,j) \neq \infty$ and therefore $(M^p)_{ij} \neq \infty$ contains a valid path of length $\mathrm{sd}(i,j)$.
We show that all entries $(M^p)_{ij} \neq \infty$ contain a valid path from i to j where the target vertex of each edge is connected by the source vertex of the following edge is by induction on p. We observe

that for $p = 1$, the matrix $M^p = M$ does not contain invalid paths. We assume M^{p-1} does not contain invalid paths. An entry $(M^{p-1} \times M)_{ij} = (M^p)_{ij} \neq \infty$ will by definition be a concatenation of the valid path $(M^{p-1})_{ik}$ from i to k with the valid path M_{kj} from k to j for some k, which in turn is also a valid path (M_{kj} will be either the empty path ϵ if $k = j$ or just the single edge path (k, j) if $k \neq j$).
Then, we show that for all $i, j \in V$ we have $f((M^p)_{ij}) = \mathrm{sd}(i, j)$ if $\mathrm{sd}(i, j) \leq p$ and $f((M^p)_{ij}) = \infty$ (equivalent with $(M^p)_{ij} = \infty$) if $\mathrm{sd}(i, j) > p$. This is trivial if we use the homomorphism $f' : \mathrm{Mat}_n(S) \to \mathrm{Mat}_n(T)$ as defined in lemma 7. We have: $f((M^p)_{ij}) = f'(M^p)_{ij} = (f'(M)^p)_{ij}$. We know that $f'(M)$ is the tropical matrix of G. Therefore, by lemma 4, $f((M^p)_{ij}) = (f'(M)^p)_{ij} = \mathrm{sd}(i, j)$ if $\mathrm{sd}(i, j) \leq p$ and $f((M^p)_{ij}) = (f'(M)^p)_{ij} = \infty$ if $\mathrm{sd}(i, j) > p$. □

Corollary 8.1. $(M^{n-1})_{ij}$ contains a shortest path from i to j, or ∞ if there is no path from i to j.

We show that this solution is in Nick's Class in lemma 9.

Lemma 9. The matrix M^{n-1} can be computed in $\mathcal{O}(\log^3 n)$ time with $\mathcal{O}(n^4)$ processors, proving that this gives an algorithm in Nick's Class which solves the strong all-pairs shortest path problem.

Proof. If two elements $a = a[1] \cdots a[l], b = b[1] \cdots b[m] \in S$ that appear as entries in M^r for some $r \geq 1$ are valid paths, their lengths are at most n. Therefore they can be multiplied in constant time with $\mathcal{O}(n)$ processors by computing

$$(a \otimes b)[i] = \begin{cases} a[i] & 1 \leq i \leq l \\ b[i - l] & l < i \leq l + m \end{cases}$$

for all $1 \leq i \leq l + m \leq 2n$ in parallel. If $a = \infty$ or $b = \infty$, multiplication takes constant time and processors.
Now for addition. In the event that $l \neq m$, $a = \infty$ or $b = \infty$, the result of $a \oplus b$ is easily determined in constant time with $\mathcal{O}(n)$ processors (the result still needs to be written somewhere, which takes $\mathcal{O}(n)$ processors). If $l = m$, we need to determine which of a and b is lexicographically first. The ordering on E can be defined as follows: with $(v_s, v_t), (w_s, w_t) \in E$ we have $(v_s, v_t) < (w_s, w_t)$ iff $v_s < w_s$ or ($v_s = w_s$ and $v_t < w_t$). The lexicographical ordering on $a, b \in E^*$ is defined as: $a < b$ iff $a[i] < b[i]$ and $a[j] = b[j]$ for some i and all $j < i$. We compute $s = [\mathrm{comp}(a[i], b[i]) \mid 1 \leq i \leq l = m]$ where $\mathrm{comp} : E \times E \to \{LT, EQ, GT\}$ returns whether a is less than, equal or greater than $b[i]$. This is done in constant time with $\mathcal{O}(n)$ processors. We then remove all leading EQ values from s in $\mathcal{O}(\log n)$ time with $\mathcal{O}(n)$ processors by repeatedly halving s until the $s[0] \in \{LT, GT\}$ with $s' = [t(s[2i], s[2i + 1]) \mid 1 \leq i \leq n/2]$ where:

$$t(x, y) = \begin{cases} EQ & \text{if } x = y = EQ \\ LT & \text{if } x = LT \text{ or } x = EQ \text{ and } y = LT \\ GT & \text{otherwise} \end{cases}$$

$s[0]$ now tells us which of a or b is lower, or $a = b$ if $s[0] = EQ$. We conclude that addition takes $\mathcal{O}(\log n)$ time with $\mathcal{O}(n)$ processors.
With lemma 5, we now know that $M^{n-1} = M^q$ with $q = 2^{\mathrm{ceil}(\log(n-1))}$ can be computed in $\mathcal{O}((t_\oplus \log n + t_\otimes) \log q) = \mathcal{O}(\log^3 n)$ time with $\mathcal{O}(n^3(p_\oplus + p_\otimes)) = \mathcal{O}(n^4)$ processors. □

3.2 A solution to the All Shortest Paths Problem

We saw how we can construct a single shortest path between any pair of vertices. However, there may exist more than one shortest path. Generally, there may exist up to $\mathcal{O}(2^n)$ shortest paths for a single pair of vertices. This motivates the all shortest paths problem, as given in Definition 3.5.

Definition 3.5. All Shortest Paths Problem
Given a directed graph $G = (V, E)$, find for each pair of vertices $i, j \in V$ that are connected via a path for each step $1 \leq l \leq \text{sd}(i, j)$ the set $s[l] \subseteq E$ of exactly all edges that may appear as the l-th step of a shortest path from i to j.
I.e. for each path-connected pair of vertices $i, j \in V$, find the tuple $s = (s[1], \cdots, s[\text{sd}(i, j)])$ (of length $\text{sd}(i, j)$) such that for all $1 \leq l \leq \text{sd}(i, j)$ we have:

$$s[l] = \{e_l \in E \mid e_1 \cdots e_{\text{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\}$$

We see that the information as requested in the all shortest paths problem is an element $s \in (\mathcal{P}(E))^*$, where $\mathcal{P}$ is the powerset operator and * the Kleene closure. I.e. s is a string over sets of edges. We formalise this notion in Definition 3.6. We interpret s as follows: assume $s = (\{(1, 2), (1, 3)\}, \{(2, 4), (3, 4)\})$. This would mean that for each shortest path from vertex 1 to vertex 4, the first edge of that path is either edge $(1, 2)$ or $(1, 3)$ and the second edge either $(2, 4)$ or $(3, 4)$. If we combine this with the fact that for all edges in a valid path, the target of each edge must be the same as the source of the next edge, we find all possible paths from vertex 1 to vertex 4: $(1, 2)(2, 4)$ and $(1, 3)(3, 4)$.

Definition 3.6. All Shortest Paths Semiring
For a graph $G = (V, E)$, define the all shortest paths semiring $(A, \oplus, \otimes)$ as:

$$A = \{\infty\} \cup (\mathcal{P}(E))^*$$

$$\begin{aligned} &\text{with } a, b \in A \setminus \{\infty\} \\ &\qquad a = (a[1], a[2], \cdots, a[l]) \\ &\qquad b = (b[1], b[2], \cdots, b[m]) \end{aligned}$$

$$\infty \oplus \infty = \infty$$

$$a \oplus \infty = \infty \oplus a = a$$

$$(a \oplus b)[i] = \begin{cases} a[i] \cup b[i] & l = m \\ a[i] & l < m \\ b[i] & l > m \end{cases}$$

$$a \otimes b = \begin{cases} \text{concat}(a, b) & a \neq \infty \text{ and } b \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

We use the all shortest paths matrix as given in Definition 3.7 to find partial solutions to the all shortest paths problem in lemma 10 and a solution for each pair of vertices in its corollary 10.1.

Definition 3.7. All Shortest Paths Matrix
Define the all shortest paths matrix $M \in \mathrm{Mat}_n(A)$ for a graph $G = (V, E)$ and A the all shortest paths semiring for G as follows:

$$M_{ij} = \begin{cases} \epsilon & i = j \\ (\{(i,j)\}) & i \neq j \text{ and } (i,j) \in E \\ \infty & (i,j) \notin E \end{cases}$$

Lemma 10. Let M be the all shortest paths matrix for a graph G. Then for all $p \in \mathbb{Z}_{\geq 1}$ and all $i, j \in V$ the matrix entry $(M^p)_{ij}$ is a solution to the all shortest paths problem for the pair of vertices i, j if $\mathrm{sd}(i,j) \leq p$ and $(M^p)_{ij} = \infty$ if $\mathrm{sd}(i,j) > p$.

Proof. Let the function $g : A \to T$ with T the tropical numbers be defined by $s \mapsto \mathrm{len}(s)$ for all $s \in A \setminus \{\infty\}$ and $\infty \mapsto \infty$. g is a homomorphism. The proof for this is identical to the proof for lemma 6. Define $g' : \mathrm{Mat}_n(A) \to \mathrm{Mat}_n(T)$ by $g'(M)_{ij} = g(M_{ij})$. This is also a homomorphism. The proof for this is identical to the proof for lemma 7.

To prove the lemma, we need to prove that:

- $(M^p)_{ij}$ has the correct length, i.e. $g((M^p)_{ij}) = \mathrm{sd}(i,j)$ if $\mathrm{sd}(i,j) \leq p$ and $g((M^p)_{ij}) = \infty$ if $\mathrm{sd}(i,j) > p$
- if $\mathrm{sd}(i,j) \leq p$, then for all $1 \leq l \leq \mathrm{sd}(i,j)$ we have $(M^p)_{ij}[l] = \{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\}$

We show that for all $i, j \in V$ we have $g((M^p)_{ij}) = \mathrm{sd}(i,j)$ if $\mathrm{sd}(i,j) \leq p$ and $g((M^p)_{ij}) = \infty$ if $\mathrm{sd}(i,j) > p$. This is trivial if we use the homomorphism g'. We have: $g((M^p)_{ij}) = g'(M^p)_{ij} = (g'(M)^p)_{ij}$. We know that $g'(M)$ is the tropical matrix for G. Therefore, by lemma 4, $g((M^p)_{ij}) = (g'(M)^p)_{ij} = \mathrm{sd}(i,j)$ if $\mathrm{sd}(i,j) \leq p$ and $g((M^p)_{ij}) = (g'(M)^p)_{ij} = \infty$ if $\mathrm{sd}(i,j) > p$.

We prove the second point by induction on p. This is trivially true for $M = M^1$ if we look at Definition 3.7. Assume the lemma to be true for M^{p-1}. We look at the definition for $(M^p)_{ij}$

$$(M^p)_{ij} = \bigoplus_{k=1}^{n} (M^{p-1})_{ik} \otimes M_{kj}$$

1. Assume $\mathrm{sd}(i,j) \leq p$. This implies $g((M^p)_{ij}) = \mathrm{sd}(i,j)$, and therefore we know that $(M^p)_{ij}$ is the summation of $(M^{p-1})_{ik} \otimes M_{kj}$ for all $1 \leq k \leq n$ with $g((M^{p-1})_{ik} \otimes M_{kj}) = \mathrm{sd}(i,j)$ (all k

for which g is larger get 'filtered' by definition of $\oplus$):

$$(M^p)_{ij} = \bigoplus_{\substack{k=1 \\ g((M^{p-1})_{ik} \otimes M_{kj}) = \mathrm{sd}(i,j)}}^{n} (M^{p-1})_{ik} \otimes M_{kj}$$

addition of elements of the same length is componentwise union

therefore, for all $1 \leq l \leq \mathrm{sd}(i,j)$, we have:

$$(M^p)_{ij}[l] = \bigcup_{\substack{k=1 \\ g((M^{p-1})_{ik} \otimes M_{kj}) = \mathrm{sd}(i,j)}}^{n} \left((M^{p-1})_{ik} \otimes M_{kj}\right)[l]$$

For any edge $e \in (M^p)_{ij}[l]$, we know by the equation above that there exists some k such that e is an edge of either a shortest path from i to k or k to j (and that there exists a path from i to k as well as from k to j). A shortest path form i to k concatenated with a shortest path from k to j is once again a shortest path, as we have $\mathrm{sd}(i,k) + \mathrm{sd}(k,j) = g((M^{p-1})_{ik} \otimes M_{kj}) = \mathrm{sd}(i,j)$. Therefore, e is an edge of some shortest path from i to j and we have:

$$(M^p)_{ij}[l] \subseteq \{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\}$$

2. If $\mathrm{sd}(i,j) \leq p-1$, choose $k = j$, and we have for all $1 \leq l \leq \mathrm{sd}(i,j)$:

$$\begin{aligned}
&\{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\} \\
=&(M^{p-1})_{ij})[l] \\
=&((M^{p-1})_{ij} \otimes M_{jj})[l] \\
=&((M^{p-1})_{ik} \otimes M_{kj})[l] \\
\subseteq&(M^p)_{ij}[l]
\end{aligned}$$

3. If $\mathrm{sd}(i,j) = p$, then for all $1 \leq l \leq \mathrm{sd}(i,j)$, let some $e \in \{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\}$ be given. Choose any shortest path $e_1 \cdots e_{\mathrm{sd}(i,j)}$ from i to j such that $e = e_l$. The shortest path $e_1 \cdots e_{\mathrm{sd}(i,j)}$ is a concatenation of the shortest path $e_1 \cdots e_{\mathrm{sd}(i,j)-1}$ to some k with $e_{\mathrm{sd}(i,j)} = (k,j)$. We therefore have:

$$\begin{aligned}
&e \in ((M^{p-1})_{ik} \otimes M_{kj})[l] \\
&e \in (M^p)_{ij}[l]
\end{aligned}$$

and we conclude:

$$\{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\} \subseteq (M^p)_{ij}[l]$$

If we combine 1, 2 and 3, we see that if $\mathrm{sd}(i,j) \leq p$, then for all $1 \leq l \leq \mathrm{sd}(i,j)$ we have $(M^p)_{ij}[l] = \{e_l \in E \mid e_1 \cdots e_{\mathrm{sd}(i,j)} \in E^* \text{ a shortest path from } i \text{ to } j\}$

□

Corollary 10.1. The set of edges $(M^{n-1})_{ij}[l]$ are exactly all the edges that appear in the l-th step on a shortest path from i to j. Therefore, $(M^{n-1})_{ij}$ is a solution to the all shortest paths problem.

We show in lemma 11 that the solution in corollary 10.1 is in Nick's Class.

Lemma 11. The matrix M^{n-1} can be computed in $\mathcal{O}(\log^2 n)$ time with $\mathcal{O}(n^5)$ processors.

Proof. For an element $a = a[1] \cdots a[l] \in S \setminus \{\infty\}$ that appears as an entry in M^r for some $r \geq 1$, we know that all $a[i]$ are pairwise disjoint because no edge can appear in two different positions in two shortest paths. Therefore, to accurately represent a, it is enough to only remember for each edge in which $a[i]$ it is contained (in contrast with remembering for each i and each $e \in E$ whether $e \in a[i]$ holds) and what the total length of a is. We represent a as a mapping $h_a : E \to \mathbb{Z}_{\geq 1} \cup \{\infty\}$ such that for all edges $e \in E$: $h_a(e) = i$ if $e \in a[i]$ and $h_a(e) = \infty$ if $e \notin a[i]$ for all i, along with the length $\text{len}(a) = l$. For an entry $a = \infty$, we denote $\text{len}(a) = \infty$.
To perform the addition $c = a \oplus b$ for two elements $a, b \in A$ represented by $(h_a, \text{len}(a))$ and $(h_b, \text{len}(b))$, we first check if $\text{len}(a) < \text{len}(b)$ or $\text{len}(b) = \infty$. If that is the case, we return $(h_a, \text{len}(a))$ as the result. If $\text{len}(a) > \text{len}(b)$ or $\text{len}(a) = \infty$ we return $(h_b, \text{len}(b))$. If $\text{len}(a) = \text{len}(b) \neq \infty$, we compute:

$$h_c(e) = \begin{cases} h_a(e) & \text{if } h_b(e) = \infty \\ h_b(e) & \text{otherwise} \end{cases}$$

for all $e \in E$ in parallel and set $\text{len}(c) = \text{len}(a)$. This runs in constant time and because $|E| \leq |V \times V| = n^2$, this takes $\mathcal{O}(n^2)$ processors.
To perform the product $c = a \otimes b$, we can directly return ∞ if either $\text{len}(a) = \infty$ or $\text{len}(b) = \infty$. Otherwise, we compute:

$$h_c(e) = \begin{cases} h_a(e) & \text{if } h_a(e) \neq \infty \\ \text{len}(a) + h_b(e) & \text{if } h_b(e) \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

for all $e \in E$ in parallel and set $\text{len}(c) = \text{len}(a) + \text{len}(b)$. This runs in constant time and takes $\mathcal{O}(n^2)$ processors.
With lemma 5, we prove that $M^{n-1} = M^q$ with $q = 2^{\text{ceil}(\log(n-1))}$ can be computed in $\mathcal{O}((t_\oplus \log n + t_\otimes) \log q) = \mathcal{O}(\log^2 n)$ time with $\mathcal{O}(n^3(p_\oplus + p_\otimes)) = \mathcal{O}(n^3(n^2 + n^2)) = \mathcal{O}(n^5)$ processors. □

In Definition 3.8, we construct a subgraph for a given pair of vertices in which all paths from one vertex to the other are exactly all shortest paths. This is proven in lemma 12.

Definition 3.8. All Shortest Paths Subgraph
Let a pair $i, j \in V$ of vertices be given. Define the all shortest paths subgraph $G'_{ij} = (V', E')$ as the subgraph of G with $E' \subseteq E$ limited to all the edges that are in some set of $(M^n)_{ij}$, and $V' \subseteq V$ all the vertices that are connected to an edge in E'.

Lemma 12. Let a pair $i, j \in V$ of vertices be given. The all shortest paths subgraph G'_{ij} is an acyclic subgraph in which each path from i to j is a shortest path in G and each shortest path from i to j in G is a path in G'.

Proof. Note that because no vertex can occur in two different positions in two different shortest paths from i to j in G, we know that for every vertex $v \in V'$ all its outgoing edges in E' are contained in $(M^n)_{ij}[k]$ for some $1 \leq k \leq \mathrm{sd}(i,j)$ and all its incoming edges are contained in $(M^n)_{ij}[k-1]$ for the same k.

With $(v_1, v_2)(v_2, v_3)(v_3, v_4) \cdots (v_l, v_{l+1})$ any path in G' (where (v_k, v_{k+1}) is an edge with source vertex v_k and target v_{k+1} and l is the length) we prove by contradiction that this path cannot contain cycles. Assume $v_p = v_q$ for some $1 \leq p < q \leq l+1$. The outgoing edge (v_p, v_{p+1}) from v_p is contained in $(M^n)_{ij}[k]$ for some k. This means that the next edge (v_{p+1}, v_{p+2}) is contained in $(M^n)_{ij}[k+1]$ and so forth. This leads us to the fact that the edge (v_{q-1}, v_q) is contained in $(M^n)_{ij}[k+(q-1)-p]$ and since $(v_{q-1}, v_q) = (v_{q-1}, v_p)$ it must also be contained in $(M^n)_{ij}[k-1]$. Because every edge in E' is contained in exactly one entry from $(M^n)_{ij}$, and with $p < q$ we know $k-1 \neq k+(q-1)-p$, the existence of a path with a cycle in G' is a contradiction.

For any path $(v_1, v_2)(v_2, v_3)(v_3, v_4) \cdots (v_l, v_{l+1})$ in G' of length l with $v_1 = i$ and $v_{l+1} = j$ (a path from i to j) we prove that $l = \mathrm{sd}(i,j)$ (proving that it is a shortest path): it is obvious that $l \geq \mathrm{sd}(i,j)$. We assume $l > \mathrm{sd}(i,j)$. We know that the edge (v_1, v_2) is contained in $(M^n)_{ij}[k]$ for some $1 \leq k \leq \mathrm{sd}(i,j)$ and therefore the edge (v_l, v_{l+1}) is contained in $(M^n)_{ij}[k+l-1]$. Because $1 \leq k$ and $\mathrm{sd}(i,j) < l$ we have $k+l-1 > \mathrm{sd}(i,j)$. This is a contradiction because M_{ij} is only defined for the indices $1, \cdots, \mathrm{sd}(i,j)$.

For any shortest path p from i to j in G, all its vertices and edges are contained in G' and therefore p is a path in G'.

□

4 Conclusions

Based on an existing solution for the all-pairs shortest path problem, we have given algorithms for two extensions of this problem. One algorithm solves the strong all-pairs shortest path problem, which asks to compute a shortest path for each pair of vertices in a directed graph. The other algorithm, when given two vertices i, j in a directed graph, constructs a subgraph which contains exactly all shortest paths from i to j. The first algorithm runs in $\mathcal{O}(\log^3 n)$ time and requires $\mathcal{O}(n^4)$ processors, the second algorithm runs in $\mathcal{O}(\log^2 n)$ time and requires $\mathcal{O}(n^5)$ processors, proving that these problems are in Nick's Class. We have seen that these algorithms utilise matrix exponentiation with matrices whose entries are data structures, instead of normal numbers. Furthermore, we have given a proof of correctness of these algorithms.

References

[Coo85] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. 1985.

[Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, June 1962.

[FM71] M. J. Fischer and A. R. Meyer. Boolean matrix multiplication and transitive closure. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 129–131, 1971.

[Mic19] Dr. George Michelogiannakis. Computation and communication in a post moore's law era, 2019.

[Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. 1965.

[MS85] Diane Maclagan and Bernd Sturmfels. Introduction to tropical geometry. 1985.

[RBW07] Mario Strasser Rainer Baumann, Simon Heimlicher and Andreas Weibel. A survey on routing metrics. 2007.

[Shi53] Alfonso Shimbel. Structural parameters of communication networks. *The bulletin of mathematical biophysics*, 15(4):501–507, Dec 1953.

[Sim94] Imre Simon. On semigroups of matrices over the tropical semiring. 1994.

[Ste17] P. Stephenhagen. Algebra ii. *Available at* http://websites.math.leidenuniv.nl/algebra/algebra2.pdf, 2017.

[TZ00] Jesper L. Träff and Christos D. Zaroliagis. A simple parallel algorithm for the single-source shortest path problem on planar digraphs. *Journal of Parallel and Distributed Computing*, 60(9):1103 – 1124, 2000.

[VWY09] Virginia Vassilevska, R. Ryan Williams, and Raphael Yuster. All pairs bottleneck paths and max-min matrix products in truly subcubic time. *Theory of Computing*, 5(9):173–189, 2009.