Lecture Notes on Multiple-source shortest path, and the Floyd-Warshall Algorithm

Jim Newton

September 22, 2026

1 Introduction

In the previous lectures we have looked at ways to calculate a single-source shortest path in a graph. You've seen techniques for both finding the *length* of such a path and for finding the path itself. We looked at BFS search for finding shortest paths in an unweighted graphs, topological sort for finding shortest paths in a DAG, Bellman-Ford for finding shortest paths in weighted graphs which might have negative cycles, and Dijkstra which is a faster algorithm but cannot support negative edges.

In this lecture we'll look at techniques for multi-source shortest paths. *I.e.*, shortest paths between all pairs of vertices in a graph. The grand result of this lecture is the Floyd-Warshall algorithm in two forms. The first form is for the multi-source shortest path problem. The second form is more general designed to work with arbitrary semirings.

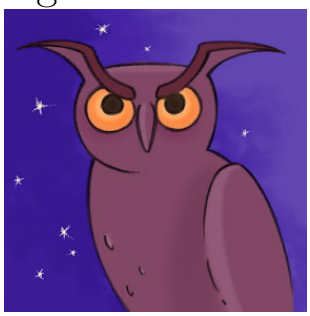
2 Abstract Algebra (Modern Algebra)

In section 4.7 we will generalize matrix multiplication on integers or real numbers to a more general structure called a semiring. Matrix multiplica-

tion involves successive multiply-add operations. As we will see these two operations can be generalized to any two operations in a semiring.

We will not dive right in to semirings. Rather we will introduce semirings in a context of abstract algebra. In this section we want to discuss several abstract algebraic structures. It is not the goal to make you experts in abstract algebra, but rather to introduce the concepts and give an intuition.

`https://mathdoctorbob.org`. I highly recommend two sets of videos on YouTube dealing with abstract algebra. On is by Math Dr. Bob: . This is an entertaining sequence of videos because the math professor seems to also be a body builder, and frankly one wonders how much time this professor Bob has spent power lifting, and how much time he as spent *mathing*.[1] One assumes Dr. Bob's goal is to be the words strongest algebraist.



`https://www.socratica.com`. The second set of videos is by Socratica. On YouTube `https://www.youtube.com/socratica`, find the videos *What is Abstract Algebra? (Modern Algebra)* , *Group Definition (ex-*

[1]Don't be confused, *mathing* is not really a word.

panded) - Abstract Algebra, and *Ring Examples (Abstract Algebra)*.

There are some very vague intuitions to keep in mind

Structure	**Operations supported**
Field	$+$ and $-$ and $\times$ and $\div$
Ring	$+$ and $-$ and $\times$
Semiring	$+$ and $\times$
Group	$(+$ and $-)$ or $(\times$ and $\div)$
Monoid	$+$ or $\times$

The intuition is the following, but we will consider more precise definitions later.

- A *field* is a familiar place where we can perform arithmetic. We have $+$ and $\times$ and we can undo those operations with the exception that we can never divide by 0.

- A *ring* is a place *like* a field, but for which no division is defined. *I.e.*, these are places where multiplication cannot necessarily be undone, but addition and subtraction still behave in the familiar way.

- A *semiring* is a more primitive place than a ring. It is a place where addition and multiplication are defined, but those operations are not necessarily undo-able.

- Yet another type of algebraic structure *group*, which is a set with one operation, which you are free to think of as addition or multiplication. The operation is reversible.

- The most primitive algebraic structure we will consider is the *monoid*. This is a set with one operation; again we can call the operation addition or multiplication, as we like. However, we make no assumption that this operation is reversible.

2.1 The Group

A *group* $(G, \circ, e)$ is a set G having a *well-defined* associative operator, $\circ$, and a neutral element, e. We may think of the operator as addition or multiplication, whichever is more appropriate for the particular set. The following properties hold, by definition, for a group.

1. closure: G is closed under $\circ$
2. associativity: $\forall x, \forall y, \forall z,\ x \circ (y \circ z) = (x \circ y) \circ z$
3. neutral element: $\forall x,\ x \circ e = e \circ x = x$
4. inverse elements: $\forall x \in G$ there is an *inverse* element, denoted $\overline{x}$, with the property that: $x \circ \overline{x} = \overline{x} \circ x = e$.

A group is sometimes denoted $(G, +, 0)$ with an inverse of element x denoted $-x$ to emphasize that it is an additive group, or $(G, \times, 1)$ with an inverse of element x denoted x^{-1} to emphasize that it is a multiplicative group.

As a special case, if the operation is commutative, then the group is called a *commutative group*, or an *Abelian* group.

You have probably already seen groups, even if they were not called such, when you studied vector spaces.

Axiom	Meaning
Associativity of addition	$\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
Commutativity of addition	$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
Identity element of addition	There exists an element $\mathbf{0} \in V$, called the *zero vector*, such that $\mathbf{v} + \mathbf{0} = \mathbf{v}$ for all $\mathbf{v} \in V$.
Inverse elements of addition	For every $\mathbf{v} \in V$, there exists an element $-\mathbf{v} \in V$, called the *additive inverse* of $\mathbf{v}$, such that $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$.
Compatibility of scalar multiplication with field multiplication	$a(b\mathbf{v}) = (ab)\mathbf{v}$ [nb 2]
Identity element of scalar multiplication	$1\mathbf{v} = \mathbf{v}$, where 1 denotes the multiplicative identity in F.
Distributivity of scalar multiplication with respect to vector addition	$a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
Distributivity of scalar multiplication with respect to field addition	$(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

The first 4 properties of a vector space, associativity, commutativity, identity element, and inverse elements specify that a vector spaces is an Abelian group with respect to its addition, $+$, operation. Not all groups are Abelian.

2.2 Examples of Groups

Which of these are groups?

1. $(\mathbb{N}, +, 0)$
2. $(\mathbb{N}, \times, 1)$
3. $(\mathbb{Z}, \times, 1)$
4. $(\mathbb{Z}_p, \times, 1)$
5. $(\mathbb{Z}, +, 0)$

6. $(\mathbb{Q} \setminus \{0\}, \times, 1)$

7. $m \times n$ matrices with addition

8. $n \times n$ matrices with multiplication

To answer the question whether a given set with a given operation forms a group we must verify the group axioms.

Claim 1. $(\mathbb{Z}, +, 0)$ *is an Abelian group.*

Proof. We verify the 4 group axioms, plus we verify commutativity.

1. Closure: Given $a, b \in \mathbb{Z}$, their sum, $a + b \in \mathbb{Z}$.

2. Associativity: Given $a, b, c \in \mathbb{Z}$, their sum, $a + (b + c) = (a + b) + c$.

3. Neutral element: The element 0 is the neutral element, because given $a \in \mathbb{Z}$, we have $0 + a = a + 0 = a$.

4. Inverse elements: Given $a \in \mathbb{Z}$, there exists $-a \in Z$ such that $-a + a = a + -a = 0$.

Therefore we know that $(\mathbb{Z}, +, 0)$ is a group. Now take $a, b \in \mathbb{Z}$, $a + b = b + a$ so addition is commutative. Thus $(\mathbb{Z}, +, 0)$ is an Abelian group. □

There is some confusion as to whether $\mathbb{N}$ includes 0 or not.

In either case $(\mathbb{N}, \times, 1)$ fails to be a group. If you attempt to prove $(\mathbb{N}, +, 1)$ is a group you could show that the set is closed, the operation is associative, and that 1 is the neutral element. However, elements of $\mathbb{N}$

don't have multiplicative inverses in $\mathbb{N}$. For a counter example, consider the element $42 \in \mathbb{N}$, there exists no whole number $a \in \mathbb{N}$ such that $a \times 42 = 42 \times a = 1$. Therefore $(\mathbb{N}, \times, 1)$ fails to be a group.

2.3 The Monoid

A *monoid* $(M, \circ, e)$ is a set E having an associative operator, $\circ$, and a neutral element, e. We can think of the operation as being multiplication or addition, which ever is more evocative for the particular set.

- closure: M is closed under $\circ$.
- associativity: $\forall x, \forall y, \forall z,\ x \circ (y \circ z) = (x \circ y) \circ z$
- neutral element: $\forall x,\ x \circ e = e \circ x = x$

Remember: Every group is a monoid, but not every monoid is a group. Think of a monoid as a group *perhaps* without inverses.

Examples:

1. Positive integers under multiplication,
2. A rational language, Σ^* under concatenation with ε, the empty word, begin the neutral element.
3. Unary functions under composition

Claim 2. $(\mathbb{Z}, \times, 1)$ *is a monoid*

Proof.

- *Closure:* Let $a, b \in \mathbb{Z}$, we have $a \times b \in \mathbb{Z}$.
- *Associativity:* Let $a, b, c \in \mathbb{Z}$, we have $a \times (b \times c) = (a \times b) \times c$.
- *Identity:* Let $a \in \mathbb{Z}$, we have $1 \times a = a \times 1 = a$.

□

2.4 The `fold` operation

The `fold` operation is available in many programming languages, functional and otherwise. The name of the `fold` function and its syntax is also different depending on the language.

Let $(M, \circ, e)$ be a monoid, and $S = \{m_1, m_2, ..., m_n\}$, $m_i \in M$, then

$$fold(S) = \begin{cases} e & \text{if } n = 0 \\ m_1 & \text{if } n = 1 \\ m_1 \circ fold(S \setminus \{m_1\}) & \text{if } n > 1 \end{cases}$$

The `fold` operation extends a binary operation to a multiple arity operation. For example, if you can add two elements, then `fold` allows you to add arbitrarily many (finitely many) elements. If the set is a monoid, then there is a neutral element, so it makes since to ask what is the *sum* of one or of zero elements. We can easily convince ourselves that the sum of 1 element is that element. *E.g.*, the sum of the set $\{3\}$ is 3. And the *sum* of zero integers is 0. Likewise the *product* of zero integers is 1, because 1 is the neutral element of the multiplicative monoid $(\mathbb{Z}, \times, 1)$.

`fold`$(S, +, 0)$	$\sum_{i=i}^{n} m_i$	$m_1 + m_2 + m_3 + ... + m_n$
`fold`$(S, \cup, \emptyset)$	$\bigcup_{i=i}^{n} m_i$	$m_1 \cup m_2 \cup m_3 \cup ... \cup m_n$
`fold`$(S, \times, 1)$	$\prod_{i=i}^{n} m_i$	$m_1 \times m_2 \times m_3 \times ... \times m_n$
`fold`$(S, \wedge, \top)$	$\bigwedge_{i=i}^{n} m_i$	$m_1 \wedge m_2 \wedge m_3 \wedge ... \wedge m_n$
`fold`$(S, strcat, "")$		`strcat`$(m_1, m_2, m_3, ..., m_n)$
`fold`$(S, compose, id)$		$x \mapsto m_n(...m_3(m_2(m_1(x))))$

2.5 The `fold` operation in Python

Use `functools.reduce` with named function or `lambda` function.

```
In [5]: import functools

        functools.reduce(lambda a,b: a+b, [1,2,3,4,5], 0)
Out[5]: 15

In [6]: functools.reduce(lambda a,b: a*b, [1,2,3,4,5], 1)
Out[6]: 120

In [13]: def plus(a,b):
             return a + b

         functools.reduce(plus, [1,2,3,4,5], 1)
Out[13]: 16
```

The function `reduce` was removed from Python 3.

2.6 The Ring

A *ring* is an object $(R, \oplus, 0, \otimes, 1)$, for which

- $(R, \oplus, 0)$ is a Abelian group,
- $(R, \otimes, 1)$ is an monoid, and
- $\otimes$ is distributive across $\oplus$.

Remember that $\otimes$ is not necessarily commutative; nevertheless to be a ring, the distributive law must hold in both directions. *I.e.*, for $a, b, z \in R$, both of the following are true.

$$z \otimes (a \oplus b) = (z \otimes a) \oplus (z \otimes b)$$
$$(a \oplus b) \otimes z = (a \otimes z) \oplus (b \otimes z)$$

In this case, 0 necessarily annihilates w.r.t. $\otimes$. By this we mean that the additive inverse 0 has the property that $0 \times a = a \times 0 = 0$ for all $a \in R$. To be clear the *additive* identity annihilates with respect to the *multiplicative* operation.

This annihilation is not an axiom but rather is a consequence of the axioms, as seen in Theorem 3.

Theorem 3 (Left 0-annihilation)**.** *If R is a ring, then $0 \otimes a = 0$ for every $a \in R$.*

Proof. Let $(R,\ \oplus, 0,\ \otimes, 1)$ be a ring, and let $a \in R$.

$$\begin{aligned} 0 \otimes a &= (0 \oplus 0) \otimes a && \text{because } 0 \oplus x = x\ \forall x \in R \\ & && \text{in particular for } x = 0 \\ &= (0 \otimes a) \oplus (0 \otimes a) && \text{by the distributive law} \end{aligned}$$

Now, since $0 \otimes a \in R$, there must exist an additive inverse, call it ζ, such that $(0 \otimes a) \oplus \zeta = 0$. We can also say that

$$\begin{aligned} (0 \otimes a) \oplus \zeta &= ((0 \otimes a) \oplus (0 \otimes a)) \oplus \zeta && \text{because } \oplus \text{ is well defined} \\ &= (0 \otimes a) \oplus ((0 \otimes a) \oplus \zeta) && \text{by associativity} \\ \underbrace{(0 \otimes a) \oplus \zeta}_{=\,0} &= (0 \otimes a) \oplus \underbrace{(0 \otimes a) \oplus \zeta}_{=\,0} \end{aligned}$$

But since, $(0 \otimes a) \oplus \zeta = 0$, this means $0 = 0 \otimes a$. □

Theorem 4 (Right 0-annihilation)**.** *If R is a ring, then $a \otimes 0 = 0$ for every $a \in R$.*

Proof. The proof follows the same pattern a Theorem 3. □

We can use claims proven earlier to show that $(\mathbb{Z},\ +, 0,\ \times, 1)$ is a ring.

Claim 5. *$(\mathbb{Z},\ +, 0,\ \times, 1)$ is a ring.*

Proof. $(\mathbb{Z},\ +, 0)$ is an Abelian group by Claim 1. $(\mathbb{Z},\ +, 0)$ is a monoid by Claim 2.

Now, take $a, b, z \in \mathbb{Z}$. We are sure that $z \times (a + b) = z \times a + z \times b$ and also that $(a+b) \times z = a \times z + b \times z$. Voilà, the distributive law holds.

Therefore, $(\mathbb{Z},\ +, 0,\ \times, 1)$ is a ring. □

As another example of a ring, consider the set of 3×3 matrices with rational entries, denoted $\mathbb{M}_{3,3}(\mathbb{Q})$.

Claim 6. *$(\mathbb{M}_{3,3}(\mathbb{Q}),\ +, 0_{3,3},\ \times, I)$ is a ring, where $0_{3,3}$ represents the 3×3 zero matrix $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, I represents the 3×3 identity matrix, $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $+$ and $\times$ are matrix addition and multiplication respectively.*

Proof. First, prove that $(\mathbb{M}_{3,3}(\mathbb{Q}), +, 0_{3,3})$ is an Abelian group. Second, prove that $(\mathbb{M}_{3,3}(\mathbb{Q}), \times, I)$ is a monoid. Finally, prove the distributive property. The details are left as an exercise for the student. □

Note that something is missing from the integer example (Claim 5) and the matrix example (Claim 6). In neither case does the multiplication have inverses. *Rings are not required to have multiplicative inverses. I.e.*, there are not necessarily multiplicative inverses of integers, although two particular integers, 1 and -1, do have inverses. Furthermore, there are not necessarily multiplicative inverses of 3×3 matrices, although some 3×3 matrices are invertible, and even sometimes have integer components. For example:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Recall that we say these kinds of *orthonormal* matrices when looking at graph isomorphisms. To show that graph G is isomorphic to H it suffices to find an orthonormal matrix, P_{GH} such that

$$P_{GH} \times A_G \times {P_{GH}}^{-1} = A_H$$

where A_G and A_H are the adjacency matrices of G and H respectively. P_{GH} is a matrix with exactly one 1 in each row and each column. The inverse is easy to calculate, because ${P_{GH}}^{-1} = {P_{GH}}^T$, *i.e.* its inverse is the same as its transpose.

2.7 The Semiring

A *semiring* is an object $(S,\ \oplus, 0,\ \otimes, 1)$, for which

- $(S, \oplus, 0)$ is a commutative monoid,
- $(S, \otimes, 1)$ is a monoid,
- $\otimes$ is distributive across $\oplus$,

$$z \otimes (a \oplus b) = (z \otimes a) \oplus (z \otimes b)$$
$$(a \oplus b) \otimes z = (a \otimes z) \oplus (b \otimes z)$$

- 0 annihilates w.r.t. $\otimes$ (i.e. $\forall x \in S, x \otimes 0 = 0 \otimes x = 0$)

Recall that in the case of a ring, the 0-annihilation property followed as a consequence of the axioms (see Theorem 3). However, to prove Theorem 3, we used the fact that $0 \otimes a$ has an additive inverse. This existence was guaranteed, because the underlying set was a group. In the case of a semiring, the underlying set may not be a group, thus additive inverses are not guaranteed. For this reason 0-annihilation is added as a semiring axiom.

Every ring is a semiring, but not every semiring is a ring. Think of a semiring as a ring *perhaps* without additive inverses. Don't be confused about the inverses. A ring has additive inverses but perhaps not multiplicative inverses. A semiring may lack both.

2.8 Examples of Semirings

Which of these semi-rings are rings?

1. $(\mathbb{N},\ +, 0,\ \times, 1)$
2. $(\mathbb{Z},\ +, 0,\ \times, 1)$
3. $(\mathbb{B},\ \vee, \bot,\ \wedge, \top)$
4. $(\mathbb{Q},\ +, 0,\ \times, 1)$
5. Polynomials with integer coefficients ($+$ or $\times$)
6. $(\mathbb{Z} \cup \{\infty\},\ \min, \infty,\ +, 0)$ The *tropical semiring.*
7. $(\mathbb{N} \cup \{\infty\},\ \max, 0,\ \min, \infty)$

We will make important use of the tropical semiring in Section 4.8. Here, in Claim 9, we prove that it is indeed a semiring.

Claim 7. $(\mathbb{Z} \cup \{\infty\}, \min, \infty)$ *is a commutative monoid.*

Proof. For each monoid axiom, the proof follows from an exhaustive case analysis.

- *Closure:* Let $a, b \in \mathbb{Z} \cup \{\infty\}$.
  - Case 1. If $a = b$, then $\min\{a, b\} = \min\{a, a\} = a \in \mathbb{Z} \cup \{\infty\}$.
  - Case 2. If $a \neq b$ and neither $a = \infty$ nor $b = \infty$, then either $\min\{a, b\} = a \in \mathbb{Z} \cup \{\infty\}$ or $\min\{a, b\} = b \in \mathbb{Z} \cup \{\infty\}$.
  - Case 3. If $a \neq b$ and $\infty \in \{a, b\}$, W.L.O.G. (without loss of generality) assume $a = \infty$. $\min\{a, b\} = b \in \mathbb{Z} \subset \mathbb{Z} \cup \{\infty\}$.
- *Associativity:* Let $a, b, c \in \mathbb{Z} \cup \{\infty\}$.

- Case 1. $a \leq b \leq c$

$$\min\{a, \min\{b, c\}\} = \min\{a, b\} = a$$
$$\min\{\min\{a, b\}, c\} = \min\{a, c\} = a$$

- Case 2. $a \leq c \leq b$

$$\min\{a, \min\{b, c\}\} = \min\{a, c\} = a$$
$$\min\{\min\{a, b\}, c\} = \min\{a, c\} = a$$

- Case 3. $b \leq a \leq c$

$$\min\{a, \min\{b, c\}\} = \min\{a, b\} = b$$
$$\min\{\min\{a, b\}, c\} = \min\{b, c\} = b$$

- Case 4. $b \leq c \leq a$

$$\min\{a, \min\{b, c\}\} = \min\{a, b\} = b$$
$$\min\{\min\{a, b\}, c\} = \min\{b, c\} = b$$

- Case 5. $c \leq a \leq b$

$$\min\{a, \min\{b, c\}\} = \min\{a, c\} = c$$
$$\min\{\min\{a, b\}, c\} = \min\{a, c\} = c$$

- Case 6. $c \leq b \leq a$

$$\min\{a, \min\{b, c\}\} = \min\{a, c\} = c$$
$$\min\{\min\{a, b\}, c\} = \min\{b, c\} = c$$

- *Identity:* Let $a \in \mathbb{Z} \cup \{\infty\}$
  - Case 1: If $a = \infty$, then $\min\{a, \infty\} = \infty = \min\{\infty, a\}$
  - Case 2: If $a \neq \infty$, $\min\{a, \infty\} = a = \min\{\infty, a\}$

- *Commutativity:* Let $a, b \in \mathbb{Z} \cup \{\infty\}$.
  - Case 1: If $a = b = \infty$, then $\min\{a, b\} = \min\{\infty, \infty\} = \min\{b, a\}$.
  - Case 2: If $\infty \notin \{a, b\}$, then $a, b \in \mathbb{Z}$, so $\min\{a, b\} = \min\{b, a\}$.
  - Case 3: If $a \neq b$ and $\infty \in \{a, b\}$, then W.L.O.G. assume $a = \infty$. So, $\min\{a, b\} = \min\{\infty, b\} = \infty = \min\{b, \infty\} = \min\{b, a\}$.

□

Claim 8. $(\mathbb{Z} \cup \{\infty\}, +, 0)$ *is a commutative monoid.*

Proof. For each monoid axiom, the proof follows from an exhaustive case analysis.

- *Closure:* Let $a, b \in$.
  - Case 1: $a, b \in \mathbb{Z}$, then $a + b \in \mathbb{Z} \subset \mathbb{Z} \cup \{\infty\}$
  - Case 2: $a = b = \infty$, then $a + b = \infty \in \mathbb{Z} \cup \{\infty\}$
  - Case 3: $a \neq b$ and $\infty \in \{a, b\}$, then W.L.O.G. assume $a = \infty$ and $b \in \mathbb{Z}$. So $a + b = \infty + b = \infty = b + \infty = b + a$.
- *Associativity:* Let $a, b, c \in \mathbb{Z} \cup \{\infty\}$.
  - Case 1: If $a, b, c \in \mathbb{Z}$, then $(a + b) + c = a + (b + c)$
  - Case 2: If $\infty \in \{a, b, c\}$, then either $a + b = \infty$ or $c = \infty$ so $(a + b) + c = \infty$. Also either $a = \infty$ or $b + c = \infty$, so $a + (b + c) = \infty$. In either case $(a + b) + c = a + (b + c)$.
- *Identity:* Let $a \in \mathbb{Z} \cup \{\infty\}$
  - Case 1: If $a \in \mathbb{Z}$, then $0 + a = a + 0 = a$

- Case 2: If $a = \infty$, then $0 + a = 0 + \infty = \infty = \infty + 0 = a + 0$.

- *Commutative:* Let $a, b \in \mathbb{Z} \cup \{\infty\}$.
  - Case 1: If $a, b \in \mathbb{Z}$, then $a + b = b + a$.
  - Case 2: If $a \in \cup\{\infty\}$ and $b = \infty$, then $a + b = a + \infty = \infty = b + \infty = b + a$.
  - Case 3: If $b \in \cup\{\infty\}$ and $a = \infty$, then $a + b = \infty + b = \infty = \infty + a = b + a$.

□

In Claim 9, we argue that the tropical semiring is actually a semiring. Careful! There is a subtle and confusing point to understand. The thing that is confusing about the tropical semiring is that the $\otimes$ of the semiring corresponds to $+$ of $\mathbb{Z}$, and the multiplicative identity 0 of the semiring corresponds to the additive identity of $\mathbb{Z}$. Remember the following:

Semiring	**Symbol**	**Usage**
addition, $\oplus$	min	$\min\{a, b\}$
additive identity	∞	$\min a, \infty = a$
multiplication, $\otimes$	$+$	$a + b$
multiplicative identity	0	$a + 0 = a$

The fact that semiring multiplication uses the $+$ operator and multiplication uses the min function means the distributive and annihilation axioms are the following.

$$\begin{aligned} \min\{a, b\} + z &= \min\{a + z, b + z\} \\ z + \min\{a, b\} &= \min\{z + a, z + b\} \\ \infty + z &= \infty \\ z + \infty &= \infty \end{aligned}$$

Claim 9. $(\mathbb{Z} \cup \{\infty\},\ \min, \infty,\ +, 0)$ *is a semiring.*

Proof. $(\mathbb{Z} \cup \{\infty\}, \min, \infty)$ is a commutative monoid by Claim 7. $(\mathbb{Z} \cup \{\infty\}, +, 0)$ is a monoid by Claim 8. We must show the distributive law and 0-annihilation, or in this case ∞-annihilation; *i.e.*, that the additive inverse annihilates under semiring multiplication.

Distributive law: Let $a, b, z \in \mathbb{Z} \cup \{\infty\}$, we must show

$$\min\{a, b\} + z = \min\{a + z, b + z\}$$

and

$$z + \min\{a, b\} = \min\{z + a, z + b\},$$

but since $+$ in each case is addition in a commutative monoid (by Claim 8), we know that

$$\min\{a, b\} + z = z + \min\{a, b\} \text{ and } \min\{a + z, b + z\} = \min\{z + a, z + b\}.$$

It, thus, suffices simply to show

$$\min\{a, b\} + z = \min\{a + z, b + z\}.$$

- Case 1: If $z = \infty$, then $\min\{a, b\} + z = \infty = \min\{\infty, \infty\} = \min\{a + z, b + z\}$.
- Case 2: If $\infty \in \{a, b\}$, W.L.O.G. assume $a = \infty$ and $b \in \mathbb{Z}$, then $\min\{a, b\} + z = b + z = \min\{\infty, b + z\} = \min\{a + z, b + z\}$.
- Case 3: If $\infty \notin \{a, b\}$ and $a \leq b$, then $a + z \leq b + z$.
  Thus $\min\{a, b\} + z = a + z = \min\{a + z, b + z\}$.
- Case 4: If $\infty \notin \{a, b\}$ and $b \leq a$, then $b + z \leq a + z$.
  Thus $\min\{a, b\} + z = b + z = \min\{a + z, b + z\}$.

Annihilation: Let $a \in \mathbb{Z} \cup \{\infty\}$, we must show

$$\infty + z = z + \infty = \infty$$

- Case 1: If $x \in \mathbb{Z}$, then $\infty + z = \infty$, and $z + \infty = \infty$.

- Case 2: If $x = \infty$, then $\infty + z = \underbrace{\infty + \infty}_{= \infty} = z + \infty$

□

2.9 Many exotic algebraic structures

There are many other structures which are less and more general the ones we've seen above.

- Field
- Vector Space (Group of vectors, and Field of scalars)
- Module (Group of vectors, and Ring of scalars)
- Integral Domain
- Unique Factorization Domain
- Semigroup
- Inverse semigroup

The list goes on and on and on ...

I recommend the YouTube video: *Lambda World 2018 - Opening Keynote by Edward Kmett.* This is a nice video of *weird* algebraic structures in computer science. The speaker expresses operations in the inner workings of an IDE as an *inverse-semigroup*, which is a structure which does not have proper inverses, but rather has objects which sometimes behave on the left or the right as inverses. *I.e.*, some operations are invertible and some or not.

3 Fast Exponentiation

The goal of this section is to motivate a technique for computing a fast exponentiation in a monoid. We begin by showing the intuition using small integers. Thereafter, we will generalize the technique to any monoid.

3.1 Classic Exponentiation

$$a^b = \underbrace{a \times a \times \cdots \times a}_{b-1 \text{ multiplications}}$$

The naïve algorithm for calculating a^b uses a loop of $\Theta(b)$ multiplications. This can be done with a call to `fold` such as:

```
In [19]: import functools

         def simple_power(a,b):
             return functools.reduce(lambda x,_:x*a, range(b),1)
```

```
In [20]: simple_power(2,5)

Out[20]: 32
```

```
In [21]: simple_power(1.01,10)

Out[21]: 1.1046221254112045
```

Or it can be implemented as shown here.

```
In [22]: def simple_power(a,b):
             answer = 1
             for i in range(b):
                 answer = answer * a
             return answer
```

```
In [23]: simple_power(2,5)

Out[23]: 32
```

```
In [24]: simple_power(1.01,10)

Out[24]: 1.1046221254112045
```

3.2 Introduction to Fast Exponentiation

How can we calculate a number such as 3^{20}? We could use a simple loop or `fold` as illustrated in Section 3.1. But can we do better?

Well, since $3^{20} = 3^{2\times 10} = (3^2)^{10}$ one might suspect that we can save work by first calculating $3^2 = 9$ and thereafter calculating 9^{10}. And now we can use the same technique to efficiently compute 9^{10}. This turns out to be a good strategy, but there are some details to be careful about. 3^{20} simplifies to 9^{10}, because the initial exponent, 20, is even. What if the initial exponent is odd?

How can we calculate a number such as 3^{21}? Can we use a similar technique? Yes! We can, if we notice that $3^{21} = 3^{20+1} = 3^{20} \times 3^1 = 3^{20} \times 3$. We already know how to efficiently compute 3^{20} from the previous paragraph. Thus, we can compute 3^{20} and then multiply the result by 3.

The generalization of this algorithm shown in Section 3.3 defines a recursive procedure which takes a slightly different action (as in the example of 3^{20} vs 3^{21}) depending on whether the given exponent is odd or even.

Notation Let $\overline{d_{n-1} \ldots d_1 d_0}$ denote the base 2 representation of a number.

For example $19 = \overline{10011}$.

Exponentiation Can we calculate a^b knowing that $b = \overline{d_{n-1} \ldots d_0}$?

$$\begin{aligned} a^b &= a^{\overline{d_{n-1} \ldots d_0}} \\ &= a^{2\times \overline{d_{n-1} \ldots d_2 d_1}} \times a^{d_0} \\ &= (a \times a)^{\overline{d_{n-1} \ldots d_2 d_1}} \times a^{d_0} \end{aligned}$$

For example:

$$\begin{aligned} 5^{19} &= 5^{2\times 9+1} \\ &= (5^2)^9 \times 5^1 \\ &= 5^{\overline{10011_2}} \\ &= 25^{\overline{1001_2}} \times 5^1 \end{aligned}$$

We calculate recursively:

$$\begin{aligned} 25^{\overline{1001}} &= 625^{\overline{100}} \times 25^1 \\ 625^{\overline{100}} &= 390625^{\overline{10}} \times 625^0 \\ 390625^{\overline{10}} &= 152587890625^{\overline{1}} \times 390625^0 \\ 5^{19} &= 152587890625 \times 25 \times 5 \end{aligned}$$

$$\begin{aligned} 5^{10011_2} &= \underbrace{25^{1001_2}}\times 5^1 \\ &\quad \underbrace{625^{100_2}}\times 25^1 \\ &\quad \underbrace{390625^{10_2}}_{152587890625^1\times 390625^0}\times 625^0 \\ &= \underbrace{25^{1001_2}}\times 5^1 \\ &\quad \underbrace{625^{100_2}}\times 25^1 \\ &\quad \underbrace{390625^{10_2}}_{152587890625^1\times \underbrace{390625^0}_{1}}\times \underbrace{625^0}_{1} \\ 5^{19} &= \underbrace{25^7}\times 5 \\ &\quad \underbrace{625^4}\times 25 \\ &\quad \underbrace{390625^2}_{152587890625} \\ &= (152587890625 \times 25) \times 5 \end{aligned}$$

How many multiplications were necessary?

$$
\begin{aligned}
t_1 &= 5 \times 5 = 25 \\
t_2 &= t_1 \times t_1 = 625 \\
t_3 &= t_2 \times t_2 = 390625 \\
t_4 &= t_3 \times t_3 = 152587890625 \\
5^{19} &= t_4 \times t_1 \times 5 \\
&= 152587890625 \times 25 \times 5 \\
&= 19073486328125
\end{aligned}
$$

3.3 Fast Exponentiation Algorithm

$$
power(a, b) = \begin{cases} 1 & \text{if } b = 0 \\ power(a \times a, b/2) & \text{if } b \text{ is even} \\ power(a \times a, \lfloor b/2 \rfloor) \times a & \text{if } b \text{ is odd} \end{cases}
$$

FAST-POWER-INTEGER(a, b)
1 if $b = 0$
2 return 1
3 if $b = 1$
4 return a
5 if $odd(b)$
6 return $a \times$FAST-POWER-INTEGER$(a \times a, \lfloor \frac{b}{2} \rfloor)$
7 else
8 return FAST-POWER-INTEGER$(a \times a, \frac{b}{2})$

Note that lines 3 and 4 are not really necessary, but in some cases give a slight performance boost. For example when multiplying matrices, recurring down to $b = 0$ means constructing an identity matrix, simply to multiply it by the given matrix. We can omit this final step. It is not crucial for correctness.

In fact, we need two $\times$'s per 1 bit and one $\times$ per 0 bit within the binary representation of b. That is $\Theta(\log b)$ $\times$'s.

3.4 Fast Exponentiation in a Monoid

In this section we generalize the function, FAST-POWER-INTEGER, shown in Section 3.3.

Definition 3.1 (Exponentiation (positive powers))**.** Let $(E, \circ, e)$ be a monoid. For $n \in \mathbb{N}$ and $x \in M$ we define *exponentiation* as follows:

$$x^n = \begin{cases} e & \text{if } n = 0 \\ x^{n-1} \circ x & \text{otherwise} \end{cases}$$

We can use the fast exponentiation algorithm to calculate x^n in $\Theta(\log n)$ operations. That M is a monoid is important because of associativity. For we wish to calculate: $x^4 = \underbrace{(x \circ x)}_{x^2} \circ \underbrace{(x \circ x)}_{x^2}$, instead of $x^4 = ((x \circ x) \circ x) \circ x$ as in the definition.

It is worth noting that we do require the monoid be commutative. Nevertheless, we can be sure that

$$x^n = x \circ x^{n-1} = x^{n-1} \circ x\,.$$

This fact does not necessarily come from commutativity as one might naïvely guess, but rather from associativity. We omit a proof and just show a motivating example.

$$
\begin{aligned}
x^4 &= x^3 \circ x && \text{by Definition 3.1} \\
&= (x^2 \circ x) \circ x && \text{by Definition 3.1} \\
&= ((x \circ x) \circ x) \circ x && \text{by Definition 3.1} \\
&= x \circ (x \circ (x \circ x)) && \text{by associativity} \\
&= x \circ (x \circ x^2) && \text{by Definition 3.1} \\
&= x \circ x^3 && \text{by Definition 3.1}
\end{aligned}
$$

The following algorithm computes any whole number, p, power of a given element, m, of a monoid. The function accepts the monoid operation and the identity as function parameters. Of course the way you represent programmatically will be particular to the programming language you are using.

```
    FAST-POWER-MONOID(m, p, ∘, identity)
1   if p = 0
2       return identity
3   if p = 1
4       return m
5   if odd(p) then
6       return m∘FAST-POWER-MONOID(m, p − 1, ∘, identity)
7   else
8       return FAST-POWER-MONOID(m ∘ m, p/2, ∘, identity)
```

On line 6, we multiply m on the left, rather than the right. In fact we could multiply on either the left or the right, as we've argued above that $x^n = x \circ x^{n-1} = x^{n-1} \circ x$.

```
In [37]: def fast_power_monoid(m, p, op, identity):
             if p == 0:
                 return identity
             if p == 1:
                 return m
             if p % 2 == 1:
                 return op(m, fast_power_monoid(m, p-1, op, identity))
             return fast_power_monoid(op(m,m), p//2, op, identity)

         def integer_times(a,b):
             return a*b

In [41]: fast_power_monoid(2,5,integer_times, 1) # 2^5
Out[41]: 32

In [42]: fast_power_monoid(3,21,integer_times,1) # 3^21
Out[42]: 10460353203

In [43]: fast_power_monoid(5,19,integer_times,1) #5^19
Out[43]: 19073486328125
```

3.5 Fast Exponentiation in a Group

We may also define negative powers if M is a group. Note that a monoid is not sufficient. For example x^{-3} is the inverse of x^3.

Definition 3.2 (Exponentiation (negative powers))**.** Let $(G, \circ, e)$ be a group. For $n \in \mathbb{Z}$ and $x \in G$, we define

$$x^n = \begin{cases} e & \text{if } n = 0 \\ x^{n-1} \circ x & \text{if } n > 0 \\ \overline{x^{-n}} & \text{if } n < 0 \end{cases}$$

The $n < 0$ case of Definition 3.2 may be computed in either of two ways: $\overline{x^{-n}} = (\overline{x})^n$, leading to the two alternative implementations. The first, `fast_power_group_v1`, requires the inverse of the element being

exponentiated, the second, `fast_power_group_v2`, requires a function to invert any element be given.

In [72]:
```
def fast_power_group_v1(m, p, op, identity, m_inverse):
    if p < 0:
        return fast_power_monoid(m_inverse, -p, op, identity)
    else:
        return fast_power_monoid(m, p, op, identity)

def real_times(a,b):
    return a*b
```

In [73]:
```
fast_power_group_v1(2,5,integer_times, 1, 0.5) # 2^5
```

Out[73]: 32

In [74]:
```
fast_power_group_v1(2,-5,integer_times, 1, 0.5) # 2^-5
```

Out[74]: 0.03125

In [75]:
```
fast_power_group_v1(0.5,-5,integer_times, 1, 2) # 0.5^-5
```

Out[75]: 32

In [76]:
```
def fast_power_group_v2(m, p, op, identity, invert):
    if p < 0:
        return invert(fast_power_monoid(m, -p, op, identity))
    else:
        return fast_power_monoid(m, p, op, identity)

def real_times(a,b):
    return a*b

def invert_real(a):
    return 1/a
```

In [77]:
```
fast_power_group_v2(2,5,integer_times, 1, invert_real) # 2^5
```

Out[77]: 32

In [78]:
```
fast_power_group_v2(2,-5,integer_times, 1, invert_real) # 2^-5
```

Out[78]: 0.03125

In [79]:
```
fast_power_group_v2(0.5,-5,integer_times, 1, invert_real) # 0.5^-5
```

Out[79]: 32.0

3.6 Exponentiation of Matrices

Question Let A be an $n \times n$ matrix, and let b be a positive integer.

How many operations are necessary to calculate A^b ?

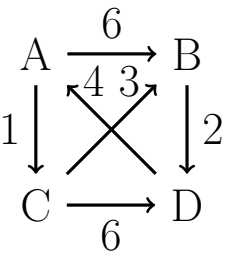
Answer We need $\Theta(\log b)$ matrix multiplications to calculate the power. And if each matrix multiplication requires n^3 scalar multiplications, then A^b needs $\Theta(n^3 \log b)$ multiplications.

Note It is possible to multiply two matrices with less than n^3 operations. For example
`https://en.wikipedia.org/wiki/Strassen_algorithm`
The Strassen algorithm uses $\Theta(n^{log_2(7)})$ operations (which is better). In this case the complexity is better.

4 Multiple-source Shortest Paths

Consider a directed graph, in which the edges are weighted representing distances or travel times.

G :



$$M_G = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Question: What is the shortest path from x to y?
I.e., for all x and y.

4.1 Version with $\Theta(n^4)$

The first approach, uses dynamic programming by calculating shortest paths of length n assuming the shortest paths of length $n-1$ are known.

$$\overbrace{\underbrace{x \to v_1 \to v_2 \to \cdots \to v_{k-1} \to v_k}_{\text{shortest path } x \to v_k} \to y}^{\text{shortest path } x \to y}$$

We have optimal substructure, if whenever $x \to v_1 \to v_2 \to \cdots \to v_{k-1} \to v_k \to y$ is a shortest path between x and y, then $x \to v_1 \to v_2 \to \cdots \to v_{k-1} \to v_k$ is a shortest path between x and v_k.

Let D_k denote the $n \times n$ matrix with components $D_k[x, y]$ being the length of the shortest path from $x \to y$ with *at most* k edges.

4.2 Using dynamic programming

Let D_k denote the $n \times n$ matrix with components $D_k[x, y]$ being the length of the minimum path from $x \to y$ with *at most* k edges.

$$D_k[x, y] = \begin{cases} M[x, y] & \text{if } k = 1 \\ \min\limits_z (D_{k-1}[x, z] + D_1[z, y]) & \text{otherwise} \end{cases}$$
$$= \begin{cases} M[x, y] & \text{if } k = 1 \\ \min\limits_z (D_{k-1}[x, z] + M[z, y]) & \text{otherwise} \end{cases}$$

D_{n-1} contains the shortest path lengths, because the longest possible path has $n - 1$ edges.

$$D_k[x, y] = \begin{cases} M[x, y] & \text{if } k = 1 \\ \min\limits_{z \in V} (D_{k-1}[x, z] + M[z, y]) & \text{otherwise} \end{cases}$$

- To obtain D_{n-1}, we iteratively calculate the sequence $D_2, D_3, ..., D_{n-1}$.

- To calculate D_k, we need only M and D_{k-1}.
- We can simply retain D_k and D_{k-1} and optimize space.
- We use two $n \times n$ arrays rather than $n - 1$ arrays.
- We swap D_k for D_{k+1} on each iteration.

4.3 Algorithm

SLOW-SHORTEST-PATH(M)

```
1  n ← size(M)
2  D_new ← M
3  for k ← 2 to n − 1
4       D_old ← D_new
5       for x ← 1 to n
6            for y ← 1 to n
7                 D_new[x, y] ← min_{z=1}^{n} D_old[x, z] + M[z, y]
8  return D_new
```

This algorithm, sadly, has complexity $\Theta(n^4)$.

4.4 Application and computation

In this section we examine one step of the computation outlined above.

$$M = \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & \infty & \infty & 0 \end{bmatrix} \quad D_2 = \begin{bmatrix} 0 & 4 & 1 & 7 \\ 6 & 0 & \infty & 2 \\ 10 & 3 & 0 & 5 \\ 4 & 10 & 5 & 0 \end{bmatrix}$$

$$
\begin{aligned}
D_3 &= D_2 \otimes M \\
&= \begin{bmatrix} 0 & 4 & 1 & 7 \\ 6 & 0 & \infty & 2 \\ 10 & 3 & 0 & 5 \\ 4 & 10 & 5 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & \infty & \infty & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 4 & 1 & 6 \\ 6 & 0 & 7 & 2 \\ 9 & 3 & 0 & 5 \\ 4 & 8 & 5 & 0 \end{bmatrix}
\end{aligned}
$$

To compute $D_3[i, j]$ we use the formula

$$D_3[i, j] = \min_k (D_2[i, k] + M[k, j]) \,.$$

In particular:

$$
\begin{aligned}
D_3[4, 2] &= \min_k (D_2[4, k] + M[k, 2]) \\
&= \min\{6 + 4, 0 + 10, 3 + 5, \infty + 0\} \\
&= 8
\end{aligned}
$$

If we compare this computation to the traditional matrix multiplication formula $D^3_{i,j} = \sum_k D^2_{i,k} \times D^1_{k,j}$, we see that to compute D_3 we do a *sort of* matrix product, $D_2 \times M$, but replacing $(+, \times)$ with $(\min, +)$. We are calculating M^k, but with different operations, *i.e.*, a different semiring.

4.5 Matrices

If $(E, \oplus, e, \otimes, f)$ is a semiring and $n \in \mathbb{N}$, then the set of $n \times n$ square matrices over E, $(\mathbb{M}_{n\times n}(E), +, 0_{n\times n}, \times, I_{n\times n})$, is a semiring. If

$a, b \in \mathbb{M}_{n\times n}(E)$, then multiplication, addition, and their respective neutral elements are defined as follows: $c = a \times b$, $d = a + b$, $I_{n\times n}$, and $O_{n\times n}$ are defined as follows:

$$
\begin{aligned}
c_{i,j} &= \bigoplus_{k=1}^{n} (a_{i,k} \otimes b_{j,k}) && \textit{row} \times \textit{column} \\
d_{i,j} &= a_{i,j} \oplus b_{i,j} && \textit{element} + \textit{element} \\
I_{i,j} &= \begin{cases} f & \text{if } i = j \\ e & \text{otherwise} \end{cases} && \text{Identity matrix} \\
O_{i,j} &= e && \text{Zero matrix}.
\end{aligned}
$$

This means $(M_{n\times n}(E), \times, I_{n\times n})$ is a monoid. Thus, we can compute M^n using fast exponentiation.

4.6 Powers of matrices and particular semirings

There are also other applications of matrix multiplication using a semiring for the matrix components. Some examples are:

$S = (\mathbb{N}, +, \times, 0, 1)$ The matrices having values $\{0, 1\}$ represent the unweighted graphs. Their powers (values in $\mathbb{N}$) count the k-step walks between the vertices.

$S = (\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$ (called *tropical semiring*)
These matrices, $M_n(S)$, represent maps with distances (possibly negative). Their successive powers consider longer and longer sequences. This is our case.

$S = (\mathbb{N} \cup \{\infty\}, \max, \min, 0, \infty)$ These matrices, $M_n(S)$, represent flow maps, with flows limited to a single path.

$S = (\mathbb{B}, \vee, \wedge, \bot, \top)$ ... A matrix represents an unweighted graph. The n'th power is the transitive closure.

4.7 Version with $\Theta(n^3 \log n)$

In Section 4.3 we developed a single-source shortest path algorithm whose time complexity was regrettably $\Theta(n^4)$. In this section we exploit the FAST-POWER-MONOID function developed in Section 3.4 to improve the time complexity of the multi-source shortest path computation from $\Theta(n^4)$ to $\Theta(n^3 \log n)$.

We will now look at a faster way for computing the shortest distance. Recall that we wish to compute

$$D_{n-1} = M^{n-1}$$

with elements of M coming from semiring $(\mathbb{Z} \cup \{\infty\},\ \min, \infty,\ +, 0)$.

Rather than using SLOW-SHORTEST-PATH(M) from Section 4.3 whose complexity is $\Theta(n^4)$, we will use FAST-SHORTEST-PATH (Section 4.8) which is based on MATRIX-MULTIPLY-TROPICAL($M, n-1$) (complexity $\Theta(n^3 \log n)$). MATRIX-MULTIPLY-TROPICAL uses the FAST-POWER-MONOID function (Section 3.4) to efficiently compute $M^{2^{\lceil \log(n-1) \rceil}}$.

To compute

$$D_{n-1} = M^{n-1}, .$$

Special case:

$$A^b = A^{n-1}\ \forall b \geq n-1\,.$$

This is true for semiring $(\mathbb{Z} \cup \{\infty\},\ \min, \infty,\ +, 0)$, and when the graph contains no negative cycles.

Thus it is not necessary to calculate exactly A^{n-1}. Any higher power of A suffices.

FAST-POWER-TROPICAL can quickly compute the next power of 2: FAST-POWER-TROPICAL($M, 2^{\lceil \log(n-1) \rceil}$).

4.8 Algorithm with complexity $\Theta(n^3 \log n)$

```
    FAST-SHORTEST-PATH(M)
1   return FAST-POWER-MONOID(
2         M,
3         2^⌈log(n−1)⌉,
4         MATRIX-MULTIPLY-TROPICAL,
5         COMPUTE-IDENTITY(n)
6     )
```

```
    MATRIX-MULTIPLY-TROPICAL(M_1, M_2)
1   for x ← 1 to n
2         for y ← 1 to n
3               P[x, y] ← min_{z=1}^{n} M_1[x, z] + M_2[z, y]
4   return P
```

The careful reader will notice that the exponent, $2^{\lceil \log(n-1) \rceil}$, passed to FAST-POWER-MONOID is never zero, so the $b = 0 \rightarrow$ *return Identity* within FAST-POWER-MONOID is never used. Thus this FAST-SHORTEST-PATH can be further optimized to avoid computing the $n \times n$ identity matrix.

Better yet, the function can be efficiently in-lined, because b is either 1 or a power of two (even). The *odd* branch is never taken. Thus $M_1 = M_2$.

```
    FAST-SHORTEST-PATH(M)
1   n ← size(M)
2   D ← M
3   for k ← 2 to ⌈log(n − 1)⌉
4       for x ← 1 to n
5           for y ← 1 to n
6               D_new[x, y] ← min_{z=1}^{n} D[x, z] + D[z, y]
                                                     (was M[z,y])
7       D ← D_new
8   return D
```

4.9 How do we find the path?

For each pair (x, y) we note the predecessor y, denoted $\Pi[x, y]$.

The best path is therefore $x \to \Pi[x, \Pi[x, \cdots \Pi[x, \Pi[x, y]]]] \to \cdots \to \Pi[x, \Pi[x, y]] \to \Pi[x, y] \to y$.

```
    FAST-SHORTEST-PATH(M)
 1  n ← size(M)
 2  D ← M
 3  for x ← 1 to n
 4   for y ← 1 to n
 5    Π[x, y] ← x
 6  for k ← 2 to ⌈log(n − 1)⌉
 7   for x ← 1 to n
 8    for y ← 1 to n
 9     m ← ∞
10     for z ← 1 to n
11        if D[x, z] + D[z, y] < m
12           m ← D[x, z] + D[z, y]
13           Π[x, y] ← z
14     D_new[x, y] ← m
15   D ← D_new
16  return D, Π
```

5 Floyd-Warshall

The Floyd-Warshall algorithm using a slightly different approach to dynamic programming than was used in Section 4.1. Rather than computing the shortest paths of k edges as a function of the shortest paths of $k-1$ edges, the Floyd-Warshall calculates the shortest paths using edges $\{e_1, e_2, ..., e_k\}$ as a function of the shortest paths using edges $\{e_1, e_2, ..., e_{k-1}\}$. This has the effect of reducing the time complexity from $\Theta(n^3 \log n)$ to $\Theta(n^3)$

5.1 Version with $\Theta(n^3)$

Imagine the vertices of a graph, numbered from 1 to n and let $D_k[x, y]$ be the shortest distance between x and y but which only passes through vertices $1 \ldots k$.

$$D_0[x, y] = M[x, y]$$
$$D_k[x, y] = \min(\underbrace{D_{k-1}[x, k] + D_{k-1}[k, y]}_{\text{passing explicitly through } k}, \underbrace{D_{k-1}[x, y]}_{\text{avoiding } k})$$

This suggests an algorithm with complexity $\Theta(n^3)$ in time and $\Theta(n^2)$ in space.

5.2 The Floyd-Warshall algorithm

In this section we see the Floyd-Warshall algorithm for the Tropical semiring. It is assumed that the given matrix, M, has already been initialized with weights of a graph in positions $M_{i,j}$ corresponding to the weight of the graph edge $i \to j$. Additionally the matrix has 0's on the main diagonal, because 0 is the neutral element of the tropical semiring multiplication, which corresponds to integer addition. Finally, all other entries in the matrix are initialized to ∞, the neutral element for the tropical semiring addition, min.

```
FLOYD-WARSHALL(M)
1  n ← size(M)
2  D_new ← M
3  for k ← 1 to n
4     D_old ← D_new
5     for x ← 1 to n
6        for y ← 1 to n
7           D_new[x, y] ← min{D_old[x, k] + D_old[k, y], D_old[x, y]}
8  return D_new
```

5.3 The Floyd-Warshall algorithm for Semiring

In this section, we see a generalization of the Floyd-Warshall algorithm shown in Section 5.2. Now we replace the max and + operations with generic $\oplus$ and $\otimes$ operators of the semiring: $(R,\ \oplus, e,\ \otimes, f)$

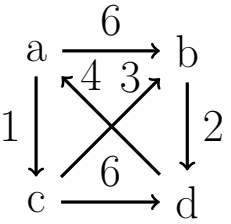
FLOYD-WARSHALL-SEMIRING(M)
1 $n \leftarrow \text{size}(M)$
2 $D_{new} \leftarrow M$
3 for $k \leftarrow 1$ to n
4 $\quad D_{old} \leftarrow D_{new}$
5 $\quad$ for $x \leftarrow 1$ to n
6 $\quad\quad$ for $y \leftarrow 1$ to n
7 $\quad\quad\quad D_{new}[x,y] \leftarrow \big(D_{old}[x,k] \otimes D_{old}[k,y]\big) \oplus D_{old}[x,y]$
8 return D_{new}

Take special care when initializing M.

- **Graph weighted edges:** If the weight of the graph edge $i \to j$ is w, then $M_{i,j} \leftarrow w$.
- **Main diagonal:** The elements of the main diagonal are all initialized to the multiplicative neutral element of the semiring. $M_{i,i} \leftarrow f$ for $i = 1..n$.
- **Otherwise:** In all other cases $M_{i,j} \leftarrow e$.

5.4 Application

G :



Semiring: *i.e.*, $\underbrace{(\mathbb{Z} \cup \{\infty\},\ \min, \infty,\ +, 0)}_{(S,\ \oplus,0,\ \otimes,1)}$

Initialize $D_\emptyset$ using weights of G,
1(multiplicative identity) of semiring on the main diagonal, and
0 (additive identity) of semiring elsewhere.

$$D_\emptyset = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & \infty & \infty & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix}$$

Calculate D_a via vertex a.

$$D_\emptyset = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & \infty & \infty & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix} \qquad D_a = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & 10 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix}$$

$$D_a[d, b] = \min\{D_\emptyset[d, a] + D_\emptyset[a, b], D_\emptyset[d, b]\} = \min\{4 + 6, \infty\} = 10$$
$$D_a[d, c] = \min\{D_\emptyset[d, a] + D_\emptyset[a, c], D_\emptyset[d, c]\} = \min\{4 + 1, \infty\} = 5$$

Calculate D_b via vertex b

$$D_a = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 6 \\ 4 & 10 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix} \qquad D_b = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & 8 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 5 \\ 4 & 10 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix}$$

$$D_b[a, d] = \min\{D_a[a, b] + D_a[b, d], D_a[a, d]\} = \min\{6 + 2, \infty\} = 8$$
$$D_b[c, d] = \min\{D_a[c, b] + D_a[b, d], D_a[c, d]\} = \min\{3 + 2, 6\} = 5$$

Calculate D_c via vertex c.

$$D_b = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 6 & 1 & 8 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 5 \\ 4 & 10 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix} \qquad D_c = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 4 & 1 & 6 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 5 \\ 4 & 8 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix}$$

$$D_c[a,d] = \min\{D_b[a,c] + D_b[c,d], D_b[d,b]\} = \min\{1+5, 10\} = 6$$
$$D_c[d,b] = \min\{D_b[d,c] + D_b[c,b], D_b[d,b]\} = \min\{5+3, 10\} = 8$$

Calculate D_d via vertex d.

$$D_c = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 4 & 1 & 6 \\ \infty & 0 & \infty & 2 \\ \infty & 3 & 0 & 5 \\ 4 & 8 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix} \qquad D_d = \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 4 & 1 & 6 \\ 6 & 0 & 7 & 2 \\ 9 & 3 & 0 & 5 \\ 4 & 8 & 5 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \end{matrix}$$

$$D_d[b,a] = \min\{D_c[b,d] + D_c[d,a], D_c[b,a]\} = \min\{2+4, \infty\} = 6$$
$$D_d[b,c] = \min\{D_c[b,d] + D_c[d,c], D_c[b,c]\} = \min\{2+5, \infty\} = 7$$
$$D_d[c,a] = \min\{D_c[c,d] + D_c[d,a], D_c[c,a]\} = \min\{5+4, \infty\} = 9$$

5.5 Recovering the path?

Like in the previous algorithm, let $\Pi[x,y]$ denote predecessor of y in the shortest path from x to y.

```
    FLOYD-WARSHALL(M)
 1  n ← size(M)
 2  D ← M
 3  for x ← 1 to n
 4   for y ← 1 to n
 5    Π[x, y] ← x
 6  for k ← 1 to n
 7   for x ← 1 to n
 8    for y ← 1 to n
 9     if D[x, k] + D[k, y] < D[x, y]
10      D_new[x, y] ← D[x, k] + D[k, y]
11      Π[x, y] ← Π[k, y]
12     else
13      D_new[x, y] ← D[x, y]
14   D ← D_new
15  return D, Π
```