Games, graphs, and machines

Asilata Bapat

August 8, 2023

Contents

Contents **2**

1 Some foundations **3**
- 1.1 Sets . . . 3
- 1.2 Relations . . . 4
- 1.3 Functions . . . 4
- 1.4 Graphs . . . 4
- 1.5 Properties of relations . . . 6

2 Equivalence relations **8**
- 2.1 Modular arithmetic . . . 9

3 Graphs **11**
- 3.1 Overview . . . 11
- 3.2 Adjacency matrix . . . 12
- 3.3 Graph colouring . . . 16
- 3.4 **TODO** Hamiltonian paths and circuits . . . 16

4 TODO Combinatorial games **17**
- 4.1 Easy examples . . . 17
- 4.2 Strategic and Grundy labelling . . . 17
- 4.3 Nim . . . 17

5 TODO Matrix games **18**
- 5.1 Matrices . . . 18

1 Some foundations

We begin by briefly introducing some language to talk about the objects we will encounter in this course. We will revisit this foundational material several times throughout the course in several contexts.

1.1 Sets

Informally, a *set* is an unordered collection of objects with no repetitions. This is the most basic object usually used to discuss almost every construction in mathematics. If T is a set and x is any object, we have the following dichotomy[*]: either x is an element of T, denoted $x \in T$, or x is not an element of T, denoted $x \notin T$. Two sets are equal if and only if they have the same elements. That is, every element of the first set is an element of the second set, and vice versa.

The Zermelo–Fraenkel axioms[†] can be used to develop this theory more formally, but we will not go into the details in this course.

Sets are often denoted by capital letters such as S, T, and potential elements as small letters x, y[‡]. If we are listing all the elements of a set, we put them in curly braces, for example $\{1, 2, 3, 4\}$. We can also specify a set by taking all elements of another set that satisfy a particular property, for example $\{x \in \mathbb{N} \mid x \text{ is even}\}$.

A set S is a *subset* of a set T, denoted $S \subset T$, if every element of S is also an element of T. A set U is a *superset* of a set T, denoted $U \supset T$, if every element of T is also an element of U. There is a unique set that contains no elements. It is called *the empty set* and is denoted $\emptyset$. The empty set is vacuously[§] a subset of every set.

Here are some things we can do with sets.

Unions The union of S and T, denoted $S \cup T$, is the set such that each element of $S \cup T$ is either an element of S or of T, or both.

Intersections The intersection of S and T, denoted $S \cap T$, is the set such that each element of $S \cap T$ is both an element of S and an element of T.

Power set The power set of S, denoted $\mathscr{P}(S)$, is the set whose elements are all the subsets of S.

Cartesian products The Cartesian product of S and T, denoted $S \times T$, is the set whose elements are *ordered pairs* (x, y), where x runs over all the elements of S, and y runs over all the elements of T. Note that if one of the two sets is empty, then the Cartesian product is also empty.

Example 1.

1. $\{1,2\} \cup \{2,3\} = \{1,2,3\}$.
2. $\{1,2\} \cap \{2,3\} = \{2\}$.

Example 2.

1. $\mathscr{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$.
2. $\{1,2\} \times \{2,3\} = \{(1,2), (1,3), (2,2), (2,3)\}$.
3. $\{1,2\} \times \emptyset = \emptyset$.

---

[*] A situation in which exactly one of two possible options is true.

[†] Historical remarks and something about ZFC?

[‡] This is just a convention. In fact, sets are often elements of other sets, so there is no clear distinction between sets and potential elements.

[§] We say that a statement of type "if ... then ...", or equivalently "for every ... we have ..." is *vacuously true* if nothing satisfies the "if" or "for every" condition.

1.2 Relations

Informally, a relation is a specification that links objects of one set and objects of another set. If x is related to y under a relation R, we say that the ordered pair (x, y) satisfies R. For example, we may consider a relation called `is-factor-of`, on pairs of natural numbers, which specifies that (x, y) satisfies `is-factor-of` if and only if x is a factor of y. In this case, $(1, 3)$, $(3, 27)$, $(4, 24)$ are all examples of ordered pairs that satisfy the relation `is-factor-of`[¶].

To model this mathematically, we formally define a relation as a subset $R \subset S \times T$, where S and T are two sets. In this case, the elements of R are precisely the ordered pairs that we think of as satisfying the relation R. In the previous example, we have $S = T = \mathbb{N}$. If we want R to model the relation `is-factor-of`, then we take R to be the subset of $\mathbb{N} \times \mathbb{N}$ consisting of exactly the pairs (x, y) where x is a factor of y.

As in the previous example, we often want S and T to be the same set. In this case, we say that a subset $R \subset S \times S$ is a (binary) relation on S.

1.3 Functions

Informally, a function is a rule that can be used to find the output value given a certain input value. This can be formally expressed using relations, as follows. Let $R \subset S \times T$ be a relation. We say that R is a function if whenever $(s, t) \in R$ and $(s, u) \in R$, we have $t = u$. In other words, any first coordinate has at most one possible second coordinate. In this case, we often write $t = R(s)$ or often $t = f(s)$. We also have the following definitions.

Example 3.

1. *The relation* $\{(a, b) \in \mathbb{N} \times \mathbb{N} \mid a + b \text{ is even } \}$ *is not a function because, for example,* $(2, 4)$ *and* $(2, 0)$ *are both in it.*
2. *The relation* $\{(a, b) \in \mathbb{N} \times \mathbb{N} \mid b = a^2\}$ *is a function.*

Domain The *domain* of this function is the set

$$\{x \in S \mid (x, y) \in R \text{ for some } y \in T\}.$$

Codomain (or range) The *codomain* of this function is the set

$$\{y \in T \mid (x, y) \in R \text{ for some } x \in S\}.$$

If S' is the domain and T' is the range, we usually say that f is a function from S' to T', written $f \colon S' \longrightarrow T'$.

1.4 Graphs

Graphs provide an extremely useful way to organise information about relations. For the moment we use them as powerful visual aids, but we will see later that graphs also lend themselves well to computational tools.

A *directed graph* consists of a *vertex set* V and an *edge set* E. We require that the edge set E is a relation on V, that is, $E \subset V \times V$. We will write

---

[¶] In English, we might read one of these as "3 is a factor of 27".

This is a binary relation because we are looking at a subset of the product of two copies of S. An n-ary relation on S would just be a subset of the product of n copies of S.

this graph as (V, E). Visually, we draw the vertices as nodes and an edge (v, w) as an arrow from v to w.

We think of *undirected graph* as a directed graph with the extra property that the edge relation E is symmetric. That is, $(v, w) \in E$ if and only if $(w, v) \in E$. In this case, we draw the vertices as nodes, and we draw a single segment joining v and w for every corresponding pair of edges (v, w) and (w, v).

Figure 1.1: A directed and an undirected graph

1.4.1 Representing a relation on a set as a graph

Note that the definition of a graph is very similar to the definition of a relation on a single set — in fact, a directed graph is just another way of looking at a relation on a set. More precisely, let R be a relation on a set S. Then we can construct a directed graph whose vertex set is S and whose edge set is R. This point of view is useful in certain situations, as we will see later.

1.4.2 The adjacency matrix of a graph

Recall that a *matrix* is a rectangular array, usually filled with numbers. An $m \times n$ matrix M has m rows (numbered 1 through m) and n columns (numbered 1) through n). The entry in the ith row and jth column is denoted M_{ij}.

It is extremely useful to encode the data of a graph into a matrix, called an *adjacency matrix*. Suppose (V, E) is a graph[**]. Choose an ordering on the elements of V, say the ordered tuple $(v_1, \ldots, v_n)$. We construct the adjacency matrix as an $n \times n$ matrix A, such that

$$A_{ij} = \begin{cases} 1, & (i, j) \in E, \\ 0, & (i, j) \notin E \end{cases}.$$

The adjacency matrix is a matrix that only contains the elements 0 and 1. It encodes the entire information contained in the original graph, in a way that is highly adapted to calculations — we will see more of this soon.

Note that changing the ordering on the elements of V produces a different-looking adjacency matrix. It is related to the original adjacency matrix by a serious of simultaneous swaps of corresponding row and column numbers. For example, the adjacency matrix given by the ordering $(v_2, v_1, v_3, \ldots, v_n)$ can be obtained from A by swapping rows 1 and 2 and also swapping columns 1 and 2.

Example 4. *Let (V, E) be the directed graph shown in Figure 1.1, with the ordering on the vertices chosen to be (a, b, c). Then the adjacency matrix is*

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

Now if we reorder the vertices as (c, b, a), the adjacency matrix becomes

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

[**] For simplicity we usually consider finite sets V when we construct adjacency matrices but in general V may be infinite.

1.5 Properties of relations

Sometimes, relations (on a single set) satisfy further special properties. Here are some common ones. Remember that a relation R is simply a subset of $S \times S$ for some set S. So the following properties are about R as a whole, as a subset of $S \times S$.

Example 5.

1. *The relation*

   $$R = \{(a, b) \in \mathbb{N} \times \mathbb{N} \mid a \text{ divides } b\}$$

   is reflexive, anti-symmetric, and transitive.

2. *The relation*

   $$R = \{(a, b) \in \mathbb{N} \times \mathbb{N} \mid a + b \text{ is odd}\}$$

   is symmetric but not reflexive or transitive.

Reflexivity A relation R is *reflexive* if $(x, x) \in R$ for each $x \in S$.

Symmetry A relation R is *symmetric* if whenever we have $(x, y) \in R$, we also have $(y, x) \in R$.

Anti-symmetry A relation R is *anti-symmetric* if having both $(x, y) \in R$ and $(y, x) \in R$ implies that $x = y$.

Transitivity A relation R is *transitive* if whenever $(x, y) \in R$ and $(y, z) \in R$, we also have $(x, z) \in R$.

Note that the properties of being *symmetric* and *anti-symmetric* are almost but not quite complementary to each other: if a relation is both symmetric and anti-symmetric, it means that only pairs of the form (x, x) can be in the relation[††]. However, not all pairs of this form have to satisfy the relation (i.e. the relation need not be reflexive).

The adjacency matrix can be helpful in order to read off properties about the relation. For example, since a reflexive relation has all possible pairs (x, x) in it, all diagonal entries A_{ii} of the adjacency matrix must equal 1, and conversely if $A_{ii} = 1$ for each i, then the relation is reflexive.

Similarly, a relation is symmetric if $A_{ij} = A_{ji}$ for each i, j. That is, if the adjacency matrix is symmetric. A relation is anti-symmetric if whenever $i \neq j$ and $A_{ij} = 1$, we have $A_{ji} = 0$.

What does it mean in terms of the adjacency matrix if a relation is transitive? The answer to this question is slightly more complicated, and we will get back to it later.

1.5.1 Closures of relations

If S is any set, then the entire cartesian product $S \times S$ is itself a relation on S. Note that certain properties are true for $S \times S$: for example, of the four properties discussed in the previous section, $S \times S$ has reflexivity, symmetry, and transitivity.

If R is any relation on S, it makes sense to ask about the *reflexive closure* (resp. symmetric or transitive closure) of R. In the following discussion we'll talk about the reflexive closure, but you can use the same definition for symmetric and transitive closures respectively.

Informally, we'd like the reflexive closure of R to be the smallest relation on S that contains R, and which is reflexive. If R is already reflexive, then it is its own reflexive closure. Otherwise, the reflexive closure will contain some more elements. But what does *smallest* mean in the above context[‡‡]? To make this precise, we give the following definition.

[††] Convince yourself of this from the definitions!

[‡‡] If S is a finite set, then we can say that that smallest means the one with the least number of elements, but we give a general definition because we don't want to be restricted to this case.

Definition 6. *A reflexive (resp. symmetric, transitive) closure of R is a set $\overline{R}$ with the following properties.*

1. $R \subset \overline{R} \subset S \times S$.
2. $\overline{R}$ *is reflexive (resp. symmetric, transitive).*
3. *If T is a subset of $S \times S$ such that $R \subset T \subsetneq \overline{R}$, then T is not reflexive (resp. symmetric, transitive).*

It can be shown that reflexive (resp. symmetric, transitive) closures always exist, and that they are unique[§§]. We won't prove this formally, but instead we will just produce a construction of each.

Let us first tackle the reflexive closure. To make a relation reflexive, we need to add in all pairs of the form $\{(x, x)\}$, where $x \in S$. So you can convince yourself that the reflexive closure is simply the set $R \cup \{(x, x) \mid x \in S\}$: not only is this new relation reflexive, but also if you take away any pair that is not already an element of R, you get something non-reflexive. In terms of adjacency matrices, the reflexive closure is the relation corresponding to the matrix obtained by changing all diagonal entries of the original adjacency matrix to 1.

Similarly, the *symmetric closure* of R is obtained by adding the flipped pair $\{(b, a)\}$ for every pair $(a, b) \in R$. This is the same thing as taking $R \cup \{(a, b) \mid (b, a) \in R\}$. In terms of the adjacency matrix, we obtain this by symmetrising the adjacency matrix[¶¶]: whenever $A_{ij} = 1$, we also set $A_{ji} = 1$.

Once again, it is not so easy to describe how to construct the *transitive closure* of a relation R, but it can be done by developing some techniques for working with adjacency matrices. We will revisit this later once we have those techniques.

[§§] Think about when it makes sense to ask for the closure of a relation with respect to a property, and when you can expect it to exist uniquely. For example, it doesn't really make sense to ask for the anti-symmetric closure of a relation. Do you see why?

[¶¶] This is the same as taking $\frac{1}{2}(A + A^t)$. Do you see why?

2 Equivalence relations

Recall that a relation R on a set S is just a subset of the product $S \times S$. We take a short tour through the theory of equivalence relations, which are extremely important in constructing all sorts of mathematical structures.

Definition 7. *A* equivalence relation *is one that is reflexive, symmetric, and transitive.*

Example 8. *label:ex:parity Let R be the relation on $\mathbb{Z}$ defined as*

$$R = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid a - b \text{ is even}\}.$$

Usually, if we have an equivalence relation R on a set S, we say that $x \sim_R y$ if (x, y) is in R. If the context is clear, we will simply say $x \sim y$. The most important application is that having an equivalence relation on a set allows us to treat an object x as "being equivalent" to an object y if $x \sim y$: the equivalence relation gives us a new way of identifying various objects. We will capture this identification with the notion of *equivalence classes**.

Definition 9. *Let R be an equivalence relation on a set S. For any $x \in S$, the* equivalence class *of x, denoted $[x]$, is the subset of S defined as follows:*

$$[x] = \{y \in S \mid x \sim_R y\}.$$

In **??**, $a \sim b$ if and only if they have the same parity, so there are two equivalence classes of R on $\mathbb{Z}$, namely [0] and [1]. Note that [0] is the same as [2] or [−6], and [1] is the same as [−55] or [7], but it's traditional to use the smallest non-negative values, which are [0] and [1].

The special properties that an equivalence relation satisfies guarantees the following proposition.

Proposition 10. *Let R be an equivalence relation on a set S.*

1. *Every element of S belongs to at least one equivalence class (its own!).*
2. *If $x, y \in S$ such that $y \in [x]$, then $[x] = [y]$. In other words, the set of equivalence classes of an equivalence relation partitions*† *the set S into disjoint subsets whose union is S.*

Proof. Let x be any element of S. First note that $x \in [x]$ by reflexivity, which proves the first statement. To prove the second statement, suppose that $x, y \in S$ such that $y \in [x]$. To show that $[x] = [y]$, we need to show that for every $z \in S$, we have $z \in [x]$ if and only if $z \in [y]$.

Recall that $y \in [x]$ means that $x \sim_R y$. If $z \in [y]$, then we have $z \in [x]$ by transitivity: $x \sim_R y$ and $y \sim_R z$ implies $x \sim_R z$. On the other hand, since we know that $y \in [x]$, we also have $x \in [y]$ by symmetry, and then by the previous argument we see that if $z \in [x]$ then $z \in [y]$ by transitivity. The proof is now complete. □

s a *representative*

Often we can uncover new structures by working with the set of equivalence classes rather than the original set S, and it can even give rise to new structures. An important example of this technique is modular arithmetic.

2.1 Modular arithmetic

As an important application of equivalence classes, we briefly study modular arithmetic. First recall the relation from cref:ex:parity. We can observe that in the integers, the sum of two numbers is always even. The sum of an even with an odd is odd, and the sum of two odd numbers is always odd. But the set of even numbers has another name: $[0]$, and the set of odd numbers is also called $[1]$ with respect to this relation.

So we can express the above statements by writing down the following statements instead.

1. Whenever $a \in [0]$ and $b \in [0]$, we have $a + b \in [0]$.
2. Whenever $a \in [0]$ and $b \in [1]$, we have $a + b \in [1]$.
3. Whenever $a \in [1]$ and $b \in [0]$, we have $a + b \in [1]$.
4. Whenever $a \in [1]$ and $b \in [1]$, we have $a + b \in [0]$.

Let us instead express this by defining a *new addition operation* on the set[‡] $\{[0], [1]\}$. We will simply define this addition using the four properties above, which can be written more concisely as

$$[a] + [b] := [a + b] \text{ for each } a, b \in \mathbb{Z}.$$

Because we know the properties we stated above about even/odd addition, we have effectively proven that it actually doesn't matter whic representative we take for each equivalence class. This is the idea behind modular arithmetic.

Exercise 11. *Check that $\sim_d$ is an equivalence relation.*

More generally, fix a *modulus* $d \in \mathbb{N}$. We say that $x \sim_d y$ if $x - y$ is divisible by d, which is also written as $d \mid x - y$. More traditionally, we write $x \equiv y \pmod{d}$. Note that if $x \sim_d y$, then there is some integer $m \in \mathbb{Z}$ such that $x - y = md$.

In this case, we have equivalence classes $[0], [1], \ldots, [d - 1]$. Note that $[d] = [0]$ again. But if $0 \le e, f < d$, how do we know for sure that $[e] \neq [f]$ when $e \neq f$? We know this by Euclid's algorithm, which guarantees that for every integer n and positive integer d, we can write a *unique* equation

$$n = qd + r, \quad 0 \le r < d.$$

---

[*] The idea is that we can treat all elements of one equivalence class as being interchangeable in some sense.

[†] If $S = S_1 \cup \cdots \cup S_n$, we say that it is a *partition* if $S_i \cap S_j = \emptyset$ for $i \neq j$. In this case we write $S = S_1 \sqcup \cdots \sqcup S_n$, or more concisely, $S = \bigsqcup_{i=1}^{n} S_i$.

[‡] Note that this set is *not* equal to $\mathbb{Z}$! It is also not equal to the set $\{0, 1\}$. Instead this is a set with two elements, which are themselves subsets of $\mathbb{Z}$.

In our case, suppose that $e \geq f$. Since $0 \leq e - f < d$, the equation for $e-f$ has to be $e-f = 0 \cdot d + (e-f)$. On the other hand if $[e] = [f]$ then we also have a valid equation that looks like $e - f = m \cdot d + 0$ for some m. Matching up the two, we see that $m = 0$ and $e = f$ is the only possibility.

Having established this, we now know that we have exactly d different equivalence classes, namely $[0], [1], \dots, [d-1]$. Of course these can be represented by different integers. For example, $[1] = \{\dots, 1-2d, 1-d, 1, 1+d, 1+2d, \dots\}$, so any of these elements would do as a representative of $[1]$. We will write $\mathbb{Z}/d\mathbb{Z} = \{[0], \dots, [d-1]\}$ to be the set of equivalence classes in this case.

Once again we define a *new addition operation*, this time on $\mathbb{Z}/d\mathbb{Z}$. The definition is the same: for any $[a], [b] \in \mathbb{Z}/d\mathbb{Z}$, set

$$[a] + [b] := [a + b].$$

We now have to check whether this is *well-defined*[§] Suppose that $[p] = [a]$ and $[q] = [b]$. Then $p - a = md$ and $q - b = nd$ for some integers m, n. Adding these, we see that $(p+q) - (a+b) = (m+n)d$, and so $[p+q] = [a+b]$. Indeed, our operation is well-defined! This is called modular addition.

Notice that this has properties similar to the addition in the integers, with some key differences. For example, we have the following.

similarity $[0] + [a] = [a] + [0] = [a]$

similarity $[a] + [b] = [b] + [a]$

difference! $[a] + [a] + \cdots + [a]$ can equal $[0]$ even if $[a] \neq 0$. For example, $[1] + [1] + [1] = [0]$ when $d = 3$.

What about multiplication? Can we define a modular multiplication? Let us try. We will attempt to define a multiplication operation by saying that

$$[a] \cdot [b] \text{ should b } [ab].$$

Exercise 12. *What are some similarities and differences between modular multiplication and usual integer multiplication?*

Again, we must check that this is well-defined. Suppose that $[p] = [a]$ and $[q] = [b]$. Then $p - a = md$ and $q - b = nd$ for some integers m, n. Note that $pq - aq = mqd$ and $aq - ab = nad$. Adding these, we see that $pq - ab = (mq + na)d$, so $[pq] = [ab]$, and this multiplication is well-defined! This is called modular multiplication.

[§] This means that if $[p] = [a]$ and $[q] = [b]$, do we have $[p + q] = [a + b]$? If not, we don't have a good definition because it depends on the specific representative we had chosen!

3 Graphs

3.1 Overview

Let us recall the definitions. A (directed) graph consists of a vertex set V and an edge set $E \subset V \times V$. If $(a, b) \in E$, we also write $a \rightarrow b$ as a directed edge. Typically we consider finite vertex sets when we work with concrete examples. An *undirected* graph is one in which the edge relation is symmetric: $(a, b) \in E$ if and only if $(b, a) \in E$. In this case, we often group the two flipped ordered pairs $\{(a, b), (b, a)\}$ and think of it as a *single* undirected edge $a - b$. Note that in this case if $a = b$, then the set $\{(a, b), (b, a)\}$ just becomes $\{(a, a)\}$, so we don't get a double loop.

Usually we consider *simple* graphs, that is, those where we disallow multiple edges and parallel loops.

3.1.1 **TODO** Draw some pictures?

3.1.2 Some natural questions

Graphs are a natural tool used to model various kinds of networks. This includes, for example, road/rail/flight networks, electrical/water flow networks, the "Facebook friend" graph, links between webpages, etc. Sometimes, these networks can be enhanced by adding "edge weights", which can be used, for example, to represent the distance between the two corresponding vertices, or in the context of flows, the "capacity" of an edge[*]. There are some very natural questions that one can ask about graphs: either practical ones that come up in many of the above contexts, or more theoretical ones. Here is a sample list, by no means exhaustive.

1. Is there a route from point A to point B?
2. How long is the route, and what is the shortest path?
3. How many routes are there? How long are they?
4. How much water/current/etc can flow through the network when at full capacity?
5. Is there a good way to figure out natural "clusters" in the graph? For example, how does Facebook know whom to suggest to you as a potential friend?
6. Can you find an unbroken path along the edges of the graph that goes through each vertex exactly once? (This is the *Hamiltonian path* problem.)
7. Can you find an unbroken path along the edges of the graph that goes through each edge exactly once? (This is the *Eulerian path* problem.)

---

[*] In a "normal" graph, we usually take each edge to have weight 1.

8. What is the shortest circuit (path that comes back to the starting point) that visits each vertex exactly once?
9. Is the graph *planar*? That is, can you draw the graph on a plane without crossing any of the edges?

3.2 Adjacency matrix

Recall the definition of an *adjacency matrix* of a graph. Given a graph (V, E), first we order the set V into a tuple $(v_1, \ldots, v_n)$. Then we create an $n \times n$ matrix A such that $A_{ij} = 1$ if $i \longrightarrow j$ in the graph, and $A_{ij} = 0$ otherwise. In this section we will see how studying adjacency matrices of graphs helps us make progress towards some of the questions above.

3.2.1 Matrix products

Example 13. *Suppose that*

$$A = \begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 & -2 \\ 2 & 3 & 4 \end{pmatrix}$$

Then

$$AB = \begin{pmatrix} 4 & 7 & 6 \\ -2 & -3 & -4 \end{pmatrix}.$$

First we recall matrix products. If A is an $m \times n$ matrix and B is an $n \times p$ matrix, then we can construct a product matrix AB, defined as follows:

$$(AB)_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{in}B_{nj} = \sum_{k=1}^{n} A_{ik}B_{kj}.$$

3.2.2 Powers of the adjacency matrix

Figure 3.1: A directed graph

Consider the example directed graph shown in cref:fig:adjmatrix. The adjacency matrix and its square are

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Note that $A^k = 0$ for all $k > 2$. From the graph and from the matrix, we see that the only nonzero entry in A^2 is the entry at position $(1, 5)$, which equals 3. It arises as the sum $1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1$, which itself records all the possible compositions of two edges such that the composed path goes from 1 to 5. As in the picture, there are exactly three possibilities, and so the answer is 3.

This is a general phenomenon, and we have the following result.

Proposition 14. *label:prop:adj-power Let A be the adjacency matrix of a simple directed graph (V, E). Suppose that the vertices are ordered as $(v_1, \ldots, v_n)$. Then the entry in the (i, j)th position of the kth power A^k of A counts the number of paths of length k from the vertex v_i to the vertex v_j.*

Proof. We proceed by induction. Indeed for $k = 1$, from the definition of the adjacency matrix, the (i, j)th entry equals 1 if and only if there is an edge from i to j in the graph. Now assume that we know the result for some $k > 0$, and we prove it for $k + 1$.

Let $B = A^k$, so that we can write $A^{k+1} = B \cdot A$. We calculate the (i, j)th entry of A^{k+1} as follows.

By the definition of matrix product, we know that this entry is the following sum:

$$(A^{k+1})_{i,j} = B_{i,1} \cdot A_{1,j} + B_{i,2} \cdot A_{2,j} + \cdots + B_{i,n} \cdot A_{n,j}.$$

For each number $1 \leq \ell \leq n$, we know that $B_{i,\ell}$ is the number of paths of length k from v_i to v_ℓ, and $A_{\ell,j}$ is the number of edges from v_ℓ to v_j. All together, the product $B_{i,\ell}A_{\ell,j}$ equals the number of paths of length $k+1$ from v_i to v_j that travel through the vertex v_ℓ. Since we add over all possible vertices v_ℓ, the result (which is the (i, j)th entry of A^{k+1}) is the total number of paths of length $k+1$ from v_i to v_j. □

We can also use the adjacency matrix to answer questions about connectedness of graphs. Suppose we want to know whether there is a path (of any length) from a vertex v_i to a vertex v_j. The previous proposition tells us that to find paths of a given length k, we need to look at entries of A^k. So as long as we find a positive entry in the (i, j)th spot of some power of A, we know that we have found a path. In other words, we can look at the (i, j)th entry of a sum $A + A^2 + \cdots$, and stop once we find a positive entry.

But how do we know when to stop adding? To answer this question, let us analyse the shortest possible path from some v_i to some v_j, under the assumption that there is at least one path.

Proposition 15. *If v_i and v_j are vertices in the graph such that there is at least one path from v_i to v_j, then the length of the shortest path from v_i to v_j cannot be more than n. Further, if $v_i \neq v_j$, then the length of the shortest path from v_i to v_j cannot be more than $n - 1$.*

3.2.3 The Boolean product and transitive closures

In this subsection and the next, we study a couple of variant products on the adjacency matrix, that let us compute different things about our graphs. The first variant is the *Boolean product*, which will be used to compute transitive closures.

First we define the following binary operations on the set $\{0, 1\}$. That is, we define the following functions $\{0, 1\} \times \{0, 1\} \to \{0, 1\}$.

Boolean addition This is also known as "OR" or "$\vee$", and is defined as follows:

$$0 \vee 0 = 0, \quad 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1.$$

Boolean multiplication This is also known as "AND" or "$\wedge$", and is defined as follows[†]:

$$1 \wedge 1 = 1, \quad 0 \wedge 1 = 1 \wedge 0 = 0 \wedge 0 = 0.$$

[†] Note that Boolean multiplication coincides with the usual multiplication operation restricted to the set $\{0, 1\}$.

The Boolean matrix product is then defined on matrices with entries in the set $\{0, 1\}$, and also outputs a matrix with entries in the same set $\{0, 1\}$. To define the Boolean matrix product, we use $\vee$ instead of $+$, and $\wedge$ instead of $\times$" respectively, as follows. Let A be an $m \times n$ matrix and B be an $n \times k$ matrix, both with entries in the set $\{0, 1\}$. Then the Boolean product $A * B$ is defined as follows (entry-wise):

$$\begin{aligned}(A * B)_{i,j} &= (A_{i1} \wedge B_{1j}) \vee (A_{i2} \wedge B_{2j}) \vee \cdots \vee (A_{in} \wedge B_{nj}) \\ &= \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}.\end{aligned}$$

Now let A be the adjacency matrix of a graph. Then the (i, j)th entry of the Boolean square of A equals 1 if and only if there exists a path of length two from i to j in the graph. This is because the (i, j)th entry is a Boolean sum ($\vee$) of several entries, and the ℓth such entry equals 1 if and only if there is an edge from i to ℓ and also an edge from ℓ to j. The Boolean sum of all of these equals 1 if and only if at least one of the entries is equal to 1, which is true if and only if there is some path of length two from i to j. Extending this reasoning to a k-fold product, we obtain the following result. The proof is similar to that of cref:prop:adj-power and so we omit it.

Proposition 16. *Let A be the adjacency matrix of a simple directed graph (V, E). Suppose that the vertices are ordered as $(v_1, \ldots, v_n)$. Then the entry in the (i, j)th position of the kth Boolean power A^{*k} of A equals* 1 *if there is a path of length k from the vertex v_i to the vertex v_j, and equals* 0 *otherwise.*

3.2.4 Weighted graphs and weighted adjacency matrices

Now suppose that $G = (V, E)$ is a *weighted graph*. This means that each edge has an associated *weight*, which is usually a non-negative real number. Mathematically, we can write this as a function $w \colon E \longrightarrow \mathbb{R}$, sending each edge to a real number. In practical applications, graphs often have edge weights, for example the length of a road or the cost of going through a toll bridge, and weighted graphs are models of these situations. We would like to use adjacency matrices to compute the weight of the least-cost (that is, smallest weight) path between any pair of vertices. We can achieve this by writing down a *weighted adjacency matrix*, and by computing a new product on it. The weighted adjacency matrix simply lists the weight of each edge. The diagonal entries are all 0 because one can get from any vertex to itself with zero cost (by not moving). All entries (i, j) where (i, j) is not an edge are set to ∞[‡].

Definition 17. *Let $G = (V, E)$ be a directed graph with weight function $w \colon E \longrightarrow \mathbb{R}$. Suppose that the vertices are ordered as $(v_1, \ldots, v_n)$. The weighted adjacency matrix of G is an $n \times n$ matrix W, defined as follows:*

$$W_{ij} = \begin{cases} 0, & \textit{if } i = j, \\ w((i, j)), & \textit{if } (i, j) \in E, \\ \infty, & \textit{otherwise}. \end{cases}$$

[‡] We use the symbol ∞ as a placeholder for an extremely large number: for any real number r in our calculations, we will set $r + \infty = \infty$ and $\min\{r, \infty\} = r$.

Example 18. *Consider the weighted graph shown below.*

Its weighted adjacency matrix is

$$W = \begin{pmatrix} 0 & 5 & 2 & \infty \\ \infty & 0 & \infty & 3 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

.

Note that this adjacency matrix is set up in a way such that the (i, j)th entry shows the minimum-cost path of length at most 1 (that is, either one edge or no edge, in the case that $i = j$) from i to j. To find the minimum-cost path of length at most 2 from i to j, we need to iterate over all possible intermediate steps $i \longrightarrow \ell \longrightarrow j$, add the edge weights of $i \longrightarrow \ell$ and $\ell \longrightarrow k$, and then take the minimum. This operation is extremely similar to the standard matrix product, except that instead of multiplying the (i, ℓ)th entry with the (ℓ, j)th entry we are adding them, and instead of adding over all possibilities we are taking the minimum over all possibilities. We define this "min-plus" matrix product as follows.

Definition 19. *Let A be an $m \times n$ matrix and B be an $n \times k$ matrix, such that the entries of A and B are either real numbers or ∞. The "min-plus" product of A and B, denoted $A \odot B$, is defined as follows (entry-wise):*

$$(A \odot B)_{i,j} = \min\{(A_{i1} + B_{1j}), (A_{i2} + B_{2j}), \ldots, (A_{in} + B_{nj})\}.$$

Now let W be the weighted adjacency matrix of a weighted graph. Note that the (i, j)th entry of $W \odot W$ is precisely the weight of the minimum-weight path from i to j that has at most two edges. Generalising this, we have the following proposition. The proof is similar to that of cref:prop:adj-power, and is omitted.

Proposition 21. *Let W be the weighted adjacency matrix of a weighted graph with n vertices.*

1. *The (i, j)th entry of $W^{\odot k}$ is the weight of the minimum-weight path from i to j that has at most k edges.*
2. *If all the edge weights are non-negative, then the (i, j)th entry of $W^{\odot(n-1)}$ is the weight of the minimum-weight path (with any number of edges) from i to j.*

Example 20. *For the graph in Example 18, the second and third min-plus powers of the weighted adjacency matrix are:*

$$W^{\odot 2} = \begin{pmatrix} 0 & 3 & 2 & 8 \\ \infty & 0 & \infty & 3 \\ \infty & 1 & 0 & 4 \\ \infty & \infty & \infty & 0 \end{pmatrix},$$

and

$$W^{\odot 3} = \begin{pmatrix} 0 & 3 & 2 & 6 \\ \infty & 0 & \infty & 3 \\ \infty & 1 & 0 & 4 \\ \infty & \infty & \infty & 0 \end{pmatrix}.$$

Indeed, the entries of the min-plus cube give the minimum weights of possible paths between any pairs of vertices in the graph.

3.2.5 The technique of repeated squaring

This section is an aside. We discuss the method of *repeated squaring* to quickly find powers of a matrix (or indeed, to quickly find powers in general). This method works for any associative product operation, including the standard matrix product, the Boolean matrix product, and the min-plus matrix product. For concreteness, we discuss it for the standard matrix product.

Let A be a square matrix. The naive method to compute a power of A, for example A^8, would be to multiply A serially with itself 8 times. This consist of 7 matrix product operations. However, there is a quicker method: if we first find and save A^2, then we can multiply that with itself to obtain and save A^4, and finally multiply that with itself to get A^8. In total, that corresponds to only 3 matrix product operations! This is considerably faster than serial multiplication.

But what if we don't have an even number, or a power of two as the power we need to compute? Suppose we are trying to compute A^n where

n is not necessarily a power of two. In this case, we simply square the matrix repeatedly, saving the results, until we reach a power less than or equal to n. Then we write n as a sum of distinct powers of two[§], and then multiply together the corresponding powers of A to get the final result. Here is an example.

Example 22. *Suppose that* $n = 19$. *In this case, we remember* $M_0 = A$, $M_1 = A^2$, $M_2 = M_1^2 = A^4$, $M_3 = A^8$, *and* $M_4 = A^{16}$. *Finally, note that* $19 = 16 + 2 + 1 = 2^4 + 2^1 + 2^0$, *and so*

$$A^{19} = M_4 \cdot M_1 \cdot M_0.$$

This process corresponds to a total of 6 *matrix product operations (four squarings and two multiplications), as opposed to the* 18 *product operations required for serial multiplication.*

3.3 Graph colouring

3.3.1 TODO The four-colour problem

3.3.2 TODO The chromatic function

3.4 TODO Hamiltonian paths and circuits

[§] Writing a positive integer n as the sum of distinct powers of two is also called *binary writing*. There are several ways to obtain it. For example, we can follow the following recursive algorithm: if n is even, we write it as $2m$, and if n is odd, we write it as $2m + 1$. Repeating the process on the m obtained until we reach 1, we obtain an expression which expands to a sum of distinct powers of two. For example,

$$\begin{aligned} 7 &= 2(3) + 1 = 2(2(1) + 1) + 1 \\ &= 4 + 2 + 1. \end{aligned}$$

4 TODO Combinatorial games

We begin the course with some games. The theory of games is a rich subject that can be used to model problems in logic, computer science, economics, and social science, depending on the rules you impose on your games. We will focus on *impartial combinatorial games*.

An impartial combinatorial game is usually played with two players and satisfies the following conditions.

1. There is a (usually finite) set of possible *game states*.
2. There are rules that describe the possible moves from a given game state to other game states.
3. The game is *impartial*, which means that the rules to go from one game state to the next do not depend on which player is about to make the move[*].
4. The players alternate making moves to move from one game state to the next.
5. The first player to be unable to make a move loses the game[†].
6. There is complete information (the entire game state is known to both players at all times).
7. There are no chance moves.

4.1 Easy examples

4.2 Strategic and Grundy labelling

4.3 Nim

---

[*] Contrast this to a game such as chess, in which one player may only move the white pieces and the other player may only move the black pieces.

[†] This is called *normal play*. In the variant called *misère play*, the first player unable to make a move wins the game.

5 **TODO** Matrix games

5.1 Matrices