Notes for Topological Data Analysis I (AMAT 583)

Håvard Bakke Bjerkevik, Barbara Giunti, Michael Lesnick

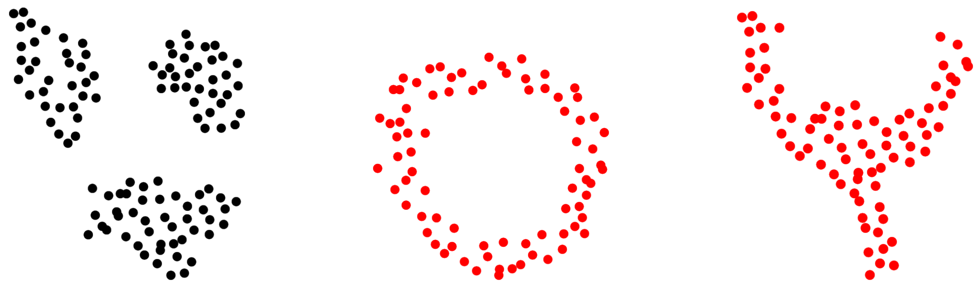
May 6, 2024

About these notes These are in-progress course notes for several sections of AMAT 583 being taught at UAlbany in Spring 2024 by the authors. The notes are being written jointly by the authors, using previous years' AMAT 583 course materials by one of the authors (Lesnick) as the starting point for some sections. Feedback and on these notes is most welcome (including any typo catches), and can be shared with any of the authors.

Contents

1 Introduction

Topological data analysis (TDA) is the branch of data science whose goal is to apply *topology* to study the *shape* of data, e.g., geometric features like clusters, loops, and tendrils:

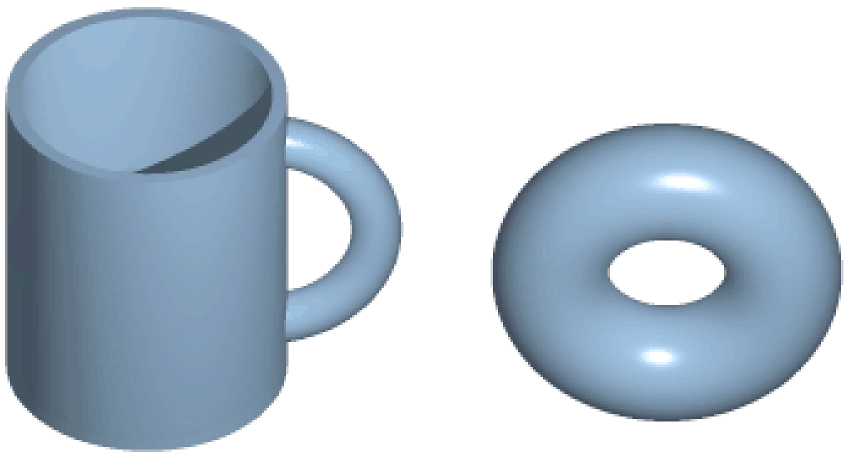


In the last 20 years, TDA has become a very active area of data science, with many applications to the sciences and a rich theory. This course will provide an introduction to TDA, with a focus on conceptual and algorithmic foundations, and connections to important related concepts in data science. Along the way, we will also devote substantial time to related topics such as elementary graph theory, algorithmic complexity, graph algorithms, and clustering. In this sense, the term "TDA" is interpreted broadly in this course.

1.1 What is Topology?

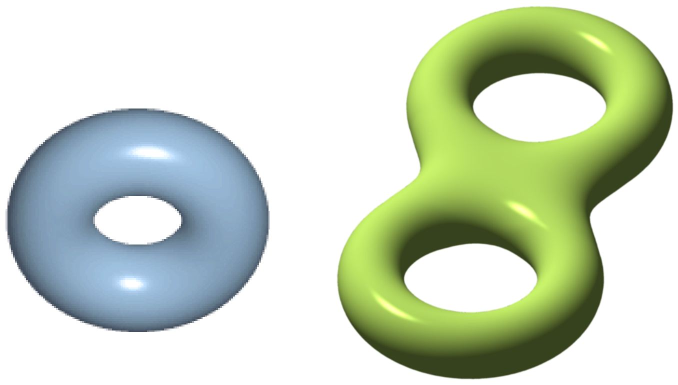
Topology is one of the major subfields of mathematics, and plays a central role in most of modern pure mathematics. Topology also has many applications, e.g., to physics, biology (DNA structure), economics, data science/machine learning. It is a very active area of research, and has been for more than a century.

Informally, topology studies the properties of shapes that are preserved under *continuous deformations*, e.g., bending, twisting, stretching (but not tearing, puncturing, or gluing). These are the kinds of deformations you could do to a rubber sheet without damaging it; as such, topology is sometimes called "rubber sheet geometry." For example, a coffee mug made of very stretch rubber could be continuously deformed into a donut:



See `https://en.wikipedia.org/wiki/Topology#/media/File:Mug_and_Torus_morph.gif` for an animation of this.

On the other hand, a donut cannot be continuously deformed into a two-holed donut:

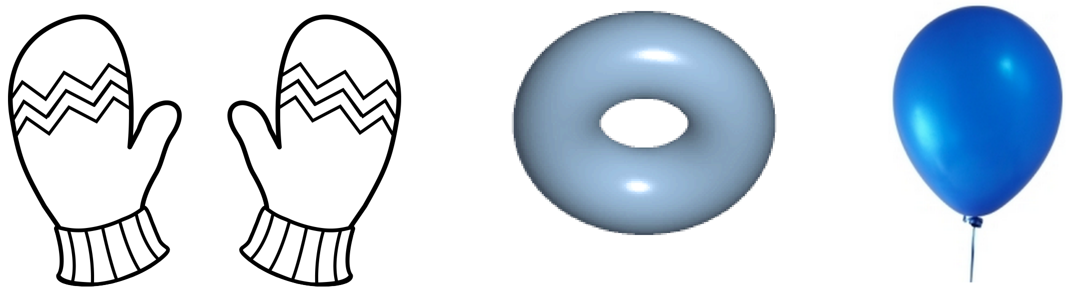


(This is not immediately clear, but can be proven with the help of tools we will introduce in this course.) This hints at a major theme in topology: The primary example of a property preserved by continuous deformation is the presence/number of *holes*.

In topology, we in fact define i-dimensional holes for $i \geq 0$.

- 0-D holes are connected components.
- 1-D holes in 3-D objects are "holes you can see through."
- 2-D holes in 3-D objects are hollow spaces.

For example, topologists would say that the pair of the pair of mittens below has two 0-D holes; the donut has one 0-D hole and one 1-D hole; and the balloon has one 0-D hole and one 2-D hole.



Holes of dimension 3 and higher cannot appear in 3-dimensional object, but can appear in higher-dimensional objects. (Thus, they are not so easy to visualize, but can be formally defined and studied much as lower-dimensional holes.)

Of course, many fundamental geometric properties of shapes are *not* preserved by continuous deformation, e.g., lengths, angles, and corners/edges. These properties are important in mathematics, but are *systematically disregarded* in topology.

Subareas of Topology There are several distinct but closely related subareas of topology, and it is useful for the beginning student to be aware of this:

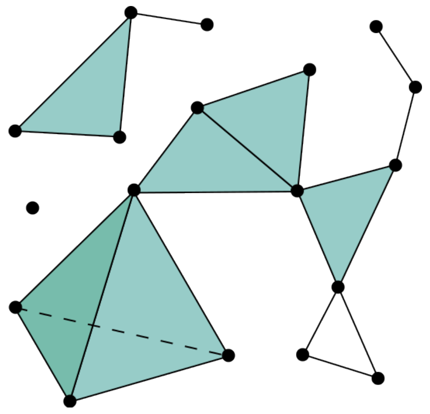
1. *Point-set topology* (also called general topology) is one area. This concerns foundational technical concepts needed advanced mathematics, such as analysis (i.e., advanced caluclus).
2. *Algebraic topology* studies holes in shapes using algebra (in particular, linear algebra and vector spaces).
3. *Manifold topology* studies topological properties of curves, surfaces and their higher-dimensional generalizations, and the way that their topology interacts with geometric structure (like lengths and angles).
4. *Computational topology* concerns efficient algorithms for big topology computations, such as computing the number of holes in a large shape. Since TDA requires such calculations, computational topology and TDA are closely related. As the field of TDA has grown in recent years, so has the area of computational topology.

The material of this course is most closely aligned with algebraic topology and computational topology. We will mostly avoid point-set topology and manifold topolgy, though we may discuss aspects of these areas in passing.

1.2 Graphs and Simplicial complexes

To develop computational topology, one needs a formal model of a geometric object which can be easily stored and manipulated on a computer. In TDA, the most popular such model is the *simplicial complex.*

Informally, a simplicial complex is a shape built out of vertices, edges, triangles, tetrahedra, and their higher dimensional analogues, as in the following picture



We mention a very important special case: a simplicial complex built from only vertices and edges is called a *graph.* Graphs play an extremely important role in data science, computer science, and mathematics. Graphs will play a major role in this course, and we will devote significant time to basic graph theory, including algorithmic aspects.

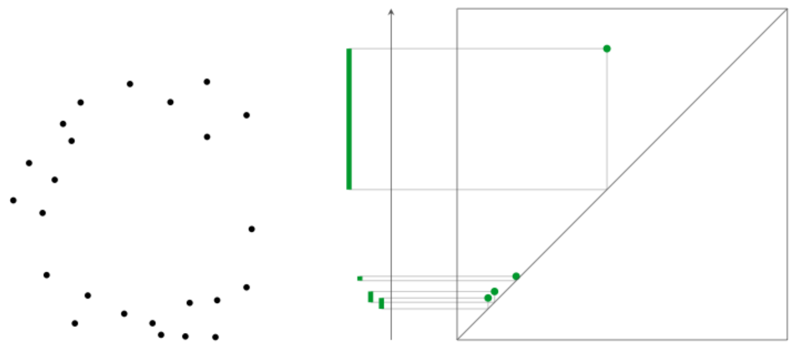
1.3 Homology and Persistent Homology

In topology, we study holes using a construction called *homology*, which is defined in terms of linear algebra. In the second semester of this course, we will study homology from computational perspective.[1]

Homology leads naturally to a TDA tool called *persistent homology.* Persistent Homology provides invariants of data called *barcodes.* A barcode is a collection of intervals $[b, d)$ in $\mathbb{R}$. Intuitively, each interval represents a shape feature of the data (e.g., cluster, or loop); the interval length is a measure of *size* of the feature.

For example, the barcode of the point cloud below has two intervals. The large interval corresponds to the loop in the data. The small interval is regarded as *topological noise*; it could be removed by a small perturbation of the data.

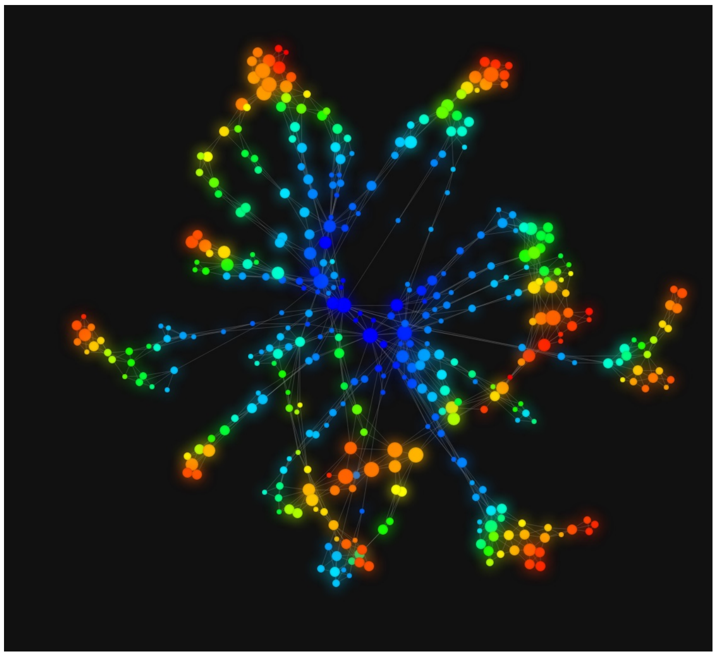
[1] In fact, a second construction, the *homotopy groups* (including the fundamental group) are also used in topology to study holes. Homotopy groups are theoretically important, but are harder to compute, and so will not be explicitly discussed in our course.



As we will see, barcodes can be used to study the shape of a data set (exploratory data analysis), and also to define features of complex data (e.g. biomolecules) for supervised learning.

Mapper Aside from persistent homology, the best known and most popular TDA tool is *Mapper*, introduced by Gunnar Carlsson, Gurjeet Singh, Facundo Mémoli in 2007. This is core idea behind the TDA/data analysis company Ayasdi, recently accuired by Symphony AI. Mapper is a sort of *stratified clustering*, which represents a data set as a network. This will be explained later.

For now, to pique your curiosity, here is an example of a data visualization created with Mapper:



1.4 The Strengths and Limitations of TDA

How useful is TDA? This is a natural question for a student to ask. Here is an attempt at an answer:

There are dozens or hundreds of papers on scientific applications of TDA (see `http://donut.topology.rocks`). The best of these tell very nice scientific stories that would have been difficult to tell with other tools, e.g., in neuroscience, materials science, genetics, and sensor networks. There have been two apparently successful commercial companies devoted to TDA. The popularity of TDA (e.g., among neuroscientists and machine learning specialists) has grown steadily over the past two decades.

To us, to particularly attractive features of TDA are:

- *Flexibility*: It offers a flexible pipeline for studying many different kinds of data, and different aspects of the shape of data.
- *Understandability*: TDA has an elegant, well developed theory, built upon decades of progress in topology.

That said, TDA is not magic; it can require care and expertise to use effectively in practice. In many settings, computational cost of the methods can be a significant limitation.

It is worth emphasizing that the tools and ideas of TDA are still very much under development. As tools get better, we can expect to see new successes.

It also useful to bear in mind that TDA is a *tool*. As with other tools, success in an application often has as much to do with the expertise and knowledge of the user, and the effort expended, as it does with the tool.

Finally, we mention that TDA is just one data science tool among many. For a beginning data science student, what is most important is to develop a solid conceptual understanding of data science; this is more important than learning any one tool. In designing this course, we have aimed to emphasize fundamental topics that will be useful even beyond TDA.

That said, we believe that your study of TDA in this course will deepen your understanding of data science and introduce you to a natural and elegant set of tools for studying the shape of data.

2 Sets and functions

We will assume little in way of mathematical background to start, and begin with the very basics.

2.1 Sets

For our purposes, the following informal definition of a set will suffice:

Definition 2.1. A *set* is a collection of distinct elements.

Example 2.2. $S_1 = \{A, B\}$ is a set. This is the set consisting of the two letters A and B.

Example 2.3. $S_2 = \{0, 1, 2\}$ is a set, consisting of the three numbers 0, 1, and 2.

Example 2.4. $\{0, 1, a, b\}$ is also a set. This example shows that a set is allowed to have elements of different types.

As shown above, we can specify a set by listing its elements. When we do that, we put the elements in curly braces. The order in which we write the elements in the curly braces does not matter, so for instance $\{A, B\}$ and $\{B, A\}$ are considered to be the same set. Sets do not allow for repeated elements, and by convention, any repeated elements in the notation for a set are ignored, so that $\{A, A, B, A, B, B\} = \{A, B\}$.

Example 2.5. Here are some examples of infinite sets:

- The natural numbers $\{0, 1, 2, \ldots\}$, denoted $\mathbb{N}$.
- The integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$, denoted $\mathbb{Z}$.

For a finite set S, we use the notation $|S|$ for the number of (different) elements of S, sometimes called the *cardinality* of S. For instance, $|\{A, A, B, A, B, B\}| = |\{A, B\}| = 2$.[2]

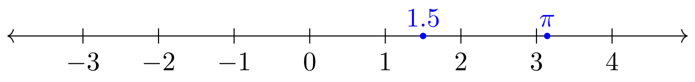
[2]For infinite sets, the notion of cardinality becomes subtle – as an example, $\mathbb{Z}$ and $\mathbb{N}$ have the same cardinality, while $\mathbb{R}$ is larger. Though interesting, the cardinality of infinite sets will not be relevant to us.

Example 2.6 (Real numbers)**.** Another important example of an infinite set is the real numbers, denoted $\mathbb{R}$. There are several equivalent ways to define the real numbers, but the following seems most friendly: A real number is a decimal number with a (possibly) infinite number of digits to the right of the decimal. For example, each of the following are real numbers:

- 7 (because we can write this as 7.0)
- 1.2
- 3.833... (the dots indicates that the 3's go on forever)
- $\pi = 3.1415\ldots$ (in π the digits to the right of the decimal do not repeat)

The one important caveat here is that if a real number ends in an infinite sequence of 9's, we regard this as the same real number as the one obtained by removing the infinite sequence of nines and increasing the rightmost remaining digit by one. For example, 2.7999... and 2.8 are understood to be the same real number.

We visualize the set real numbers as an infinite line, as shown below, where each point on the line represents a real number:



Example 2.7. $\{\}$ is a set, called the *empty set.* This is the unique set with no elements. It is often denoted by $\emptyset$.

Example 2.8. A set with a single element is called a *singleton set.* For example, $\{2\}$ is a set with one element. Note that this is not the same as the number 2; rather it is *the singleton set with element 2.*

Example 2.9. We can have a set whose elements are themselves sets. For example for sets S_1 and S_2 defined as above, $S_3 = \{S_1, S_2\} = \{\{A, B\}, \{0, 1, 2\}\}$ is a set.

Example 2.10. According to the axioms of formal set theory (which we have not discussed here), a set is not allowed to contain itself. For example,

$$S_1 = \{a, b, S_1\}$$

is not a valid set.

Notation for an Element Contained in a Set If a set S contains an element a, we write $a \in S$. Otherwise, we write $a \notin S$.

Example 2.11. $1 \in \{1, 2, 3\}$, but $4 \notin \{1, 2, 3\}$.

Equality of Sets We say sets S and T are *equal*, and write $S = T$, if the elements of S and T are the exactly same. If S and T are not equal, we write $S \neq T$.

Example 2.12. $\{A, B, C\} = \{B, C, A\}$

Example 2.13. $\{\{A, B\}, \{0, 1, 2\}\} \neq \{A, B, 0, 1, 2\}$. The first set has two elements, each of which is a set, while the second set has five elements.

2.2 Operations on Sets

Subsets For sets S and T, we say S is a *subset* of T, and write $S \subset T$, if every element of S is also an element of T. This is sometimes instead written as $S \subseteq T$.

Example 2.14. It is clear that for any set S, S is a subset of itself. Moreover, the empty set $\emptyset$ is a subset of S. To explain, since the empty set contains no elements, it is vacuously true that every element of $\emptyset$ is an element of S.

Example 2.15. As any integer can be written as a decimal number (with a 0 to the right of the decimal place), we may regard $\mathbb{Z}$ as a subset of $\mathbb{R}$.

We sometimes specify a subset via a property its elements satisfy. For example, consider the set

$$S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

and the subset

$$T = \{0, 2, 4, 6, 8\},$$

i.e., S denotes the set of all digits, and T denotes the set of all even digits. Then T can be specified as the set of integers z such that z is divisible by 2. We write this in symbols as follows:

$$T = \{z \in S \mid z \text{ is divisible by } 2\}.$$

the symbol "|" translates into words as "such that." Alternatively, a colon can be used in place of |, i.e.,

$$T = \{z \in S : z \text{ is divisible by } 2\}.$$

We will now define unions, intersections and complements of sets. These are illustrated schematically in Figure 1.

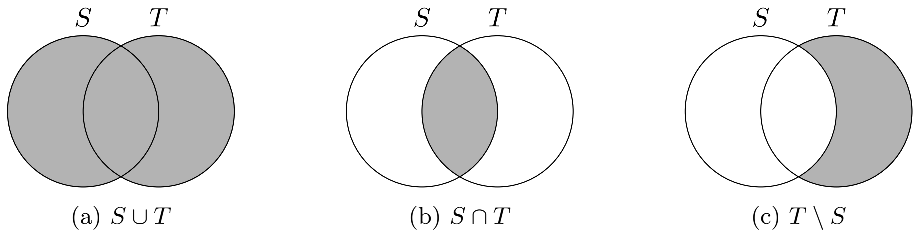


Figure 1: The sets $S \cup T$, $S \cap T$ and $T \setminus S$ shown in grey.

Union The union of two sets S and T, denoted $S \cup T$, is the set consisting of all elements in *either S or T*.

Example 2.16. If $S = \{A, B\}$ and $T = \{B, C\}$, then $S \cup T = \{A, B, C\}$.

Intersection The intersection of two sets S and T, denoted $S \cap T$, is the set consisting of all elements in *both S and T*.

Example 2.17. If $S = \{A, B, C\}$ and $T = \{B, C, D\}$, then $S \cap T = \{B, C\}$.

Complements The set $T \setminus S$ is defined to be the set of elements in T which are not also in S. That is,

$$T \setminus S = \{t \in T \mid t \notin S\}.$$

Example 2.18.

1. If $T = \{A, B, C\}$ and $S = \{A\}$, then $T \setminus S = \{B, C\}$.
2. If $T = \{A, B, C\}$ and $S = \{a\}$, then $T \setminus S = T$.

2.3 Intervals

Intervals are subsets of $\mathbb{R}$ which are particularly important in topology.

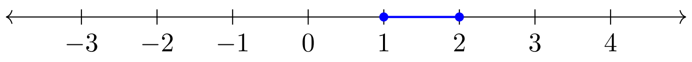
Definition 2.19. An *interval* is a non-empty, connected subset of $\mathbb{R}$.

We have not formally defined connectedness, but intuitively, this means that an interval can consist of only one "piece," not several pieces.

Example 2.20. As illustrated below,

$$\{x \in \mathbb{R} \mid 1 \leq x \leq 2\}$$

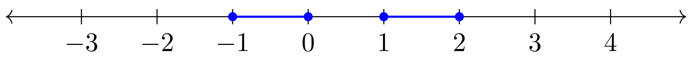
is an interval.



On the other hand,

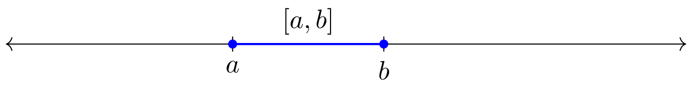
$$\{x \in \mathbb{R} \mid -1 \leq x \leq 0\} \cup \{x \in \mathbb{R} \mid 1 \leq x \leq 2\}$$

is not an interval because it consists of two components.



There are 9 types of intervals, 4 of finite length and 5 of infinite length. We specify them all now. For $a \leq b \in \mathbb{R}$,

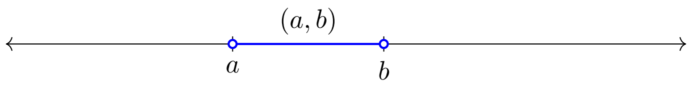
$$[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}.$$



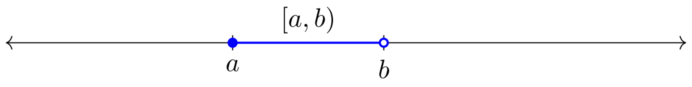
Note that $[a, a] = \{a\}$ for all $a \in \mathbb{R}$. The interval $[0, 1]$ plays a special role in topology (though perhaps less so in this course), and is denoted I.

For $a < b \in \mathbb{R}$,

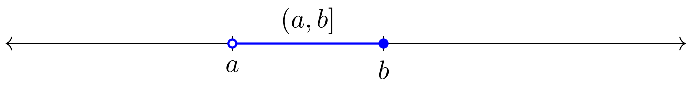
$$(a, b) = \{x \in \mathbb{R} \mid a < x < b\}$$



$$[a, b) = \{x \in \mathbb{R} \mid a \leq x < b\},$$



$$(a, b] = \{x \in \mathbb{R} \mid a < x \leq b\}.$$



For $a \in \mathbb{R}$,

$$\begin{aligned}(a,\infty) &= \{x \in \mathbb{R} \mid a < x < \infty\},\\ [a,\infty) &= \{x \in \mathbb{R} \mid a \leq x < \infty\},\\ (-\infty,a) &= \{x \in \mathbb{R} \mid x < a\},\\ (-\infty,a] &= \{x \in \mathbb{R} \mid x \leq a\}.\end{aligned}$$

There is only one interval of the final type: $(-\infty,\infty) = \mathbb{R}$.

2.4 Cartesian Products

Definition 2.21. A *tuple* is an ordered sequence of elements, and we use the notation $(a_1, a_2, \ldots, a_n)$ for the tuple that contains the elements $a_1, \ldots, a_n$ in that order. A tuple of the form (a_1, a_2) is called a *pair* (or *ordered* pair).

For sets S and T, the Cartesian Product of S and T, denoted $S \times T$, is the set of all ordered pairs (s,t) with $s \in S$ and $t \in T$. In symbols, we write this as follows:

$$S \times T = \{(s,t) \mid s \in S,\ t \in T\}.$$

Example 2.22. For $S = \{1,2\}$ and $T = \{A,B\}$,

$$S \times T = \{(1,A),(1,B),(2,A),(2,B)\}.$$

Example 2.23. By definition, $\mathbb{R} \times \mathbb{R}$ is the set of all ordered pairs of real numbers. $\mathbb{R} \times \mathbb{R}$ is usually written as $\mathbb{R}^2$.

More generally, we can define the *Cartesian product*

$$S_1 \times S_2 \times \cdots \times S_n$$

of $n \geq 1$ sets $S_1, S_2, \ldots, S_n$ to be the set of all ordered lists $(s_1, s_2, \ldots, s_n)$ of length n, where $s_i \in S_i$ for each i. Thus, in symbols, the definition is:

$$S_1 \times S_2 \times \cdots \times S_n = \{(s_1, s_2, \ldots, s_n) \mid s_i \in S_i\}.$$

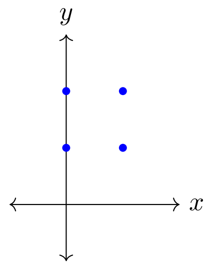
Example 2.24. As a set, n-dimensional Euclidean space $\mathbb{R}^n$ is the Cartesian product of n copies of the real numbers $\mathbb{R}$.

Proposition 2.25. *Given sets $S_i, \ldots, S_n$ and $T_i, \ldots, T_n$, if $S_i \subset T_i$ for each i, then*

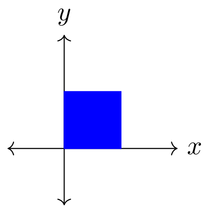
$$S_1 \times S_2 \times \cdots \times S_n \subset T_1 \times T_2 \times \cdots \times T_n.$$

In particular, the proposition tells us that if $A, B \subset \mathbb{R}^2$, then we can think of $A \times B$ as a subset of $\mathbb{R}^2$, as in the following examples:

Example 2.26. $\{0,1\} \times \{1,2\} = \{(0,1),(0,2),(1,1),(1,2)\}$



Example 2.27. $I^2 = \{(x, y) \in \mathbb{R}^2 \mid 0 \leq x, y \leq 1\}$.



2.5 Functions

Here is the usual (informal) definition of a function:

Definition 2.28. Given sets S and T, a *function* f from S to T is a rule which assigns each $s \in S$ exactly one element in T, denoted $f(s)$. We often write f as $f\colon S \to T$ to make clear the choice of S and T. We call S the *domain of* f, and T the *codomain* of f.

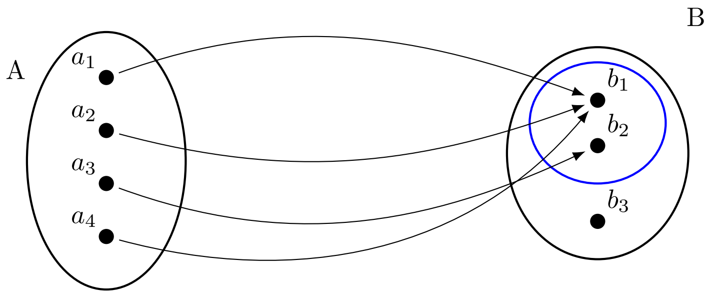


Figure 2: Depiction of function $f : A \to B$, for the sets $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, b_3\}$. The image of f is given by the elements in B in the blue ellipse.

Definition 2.29. The *image* of $f : S \to T$, denoted $\text{im}(f)$, is the subset of T defined as follows:

$$\text{im}(f) = \{y \in T \mid y = f(x) \text{ for some } x \in S\}.$$

Definition 2.30. $f : S \to T$ is said to be

- injective (or 1-1) if $f(x) \neq f(y)$ whenever $x \neq y$,
- surjective (or onto) if $\text{im}(f) = T$,
- bijective if f is is both injective and surjective.

Example 2.31. Let $S = \{1, 2, 3\}$ and $T = \{A, B\}$. Then choosing $f(1) = B$, $f(2) = A$ and $f(3) = B$ is enough to describe f exactly. In this case, $\text{im}(f) = \{A, B\} = T$, and f is surjective, but not injective.

Example 2.32. Let us define a function $f : \mathbb{N} \to \mathbb{N}$. We cannot list the infinitely many elements $f(0), f(1), \ldots$ one by one; instead, we can describe a general rule for determining $f(s)$ given s. For instance, we can define f by letting $f(s) = 2s$ for all $s \in \mathbb{N}$. That is, $f(0) = 0$, $f(1) = 2$, $f(2) = 4$, etc. This function is injective, but not surjective, and its image is $\{s \in \mathbb{N} \mid s \text{ is even}\}$, the set of even natural numbers.

Definition 2.33. Given functions $f : S \to T$ and $g : T \to U$, we say that the *composition of* f *and* g is the function $g \circ f : S \to U$ given by $g \circ f(x) = g(f(x))$.

Definition 2.34. For any set S, the *identity function on S* is the function

$$\mathrm{Id} : S \to S,$$

given by $\mathrm{Id}(s) = s$ for all $s \in S$. We sometimes also write the identity function on S as Id_S.

More generally, for any $S \subset T$, the *inclusion function* $j : S \to T$ is the function given by $j(s) = s$ for all S.

Definition 2.35. Functions $f : S \to T$ and $g : T \to S$ are *inverses* if

$$g \circ f = \mathrm{Id}_{\mathrm{S}} \quad \text{and} \quad f \circ g = \mathrm{Id}_T\,.$$

We say g is an inverse of f (and vice versa). If f has an inverse, we say f is *invertible*, and denote its inverse as f^{-1}.

The straightforward proof of the following result is left as an exercise:

Proposition 2.36. *A function $f : S \to T$ is a bijection if and only if f has an inverse.*

2.6 Relations

Definition 2.37. A *relation* on a set S is a subset of $S \times S$. That is, it contains elements of the form (s, t), where $s, t \in S$. For R be a relation on S, we write sRt if $(s, t) \in R$ and $s \not R t$ if $(s, t) \notin R$.

Definition 2.38. We say that a relation $\leq$ is a *total order* on S if for all $s, t, u \in S$, the following conditions are satisfied.

(Reflexivity) $s \leq s$,

(Transitivity) if $s \leq t$ and $t \leq u$, then $s \leq u$,

(Anti-symmetry) if $s \neq t$ and $s \leq t$, then $t \not\leq s$,

(Strongly connected) for all $s, t \in S$, either $s \leq t$ or $t \leq s$.

Example 2.39. Define a relation $\sim$ on $\{a, b, c, d\}$ by the following table

$$\begin{array}{c|cccc} & \mathrm{a} & \mathrm{b} & \mathrm{c} & \mathrm{d} \\ \hline \mathrm{a} & 1 & 0 & 1 & 0 \\ \mathrm{b} & 0 & 1 & 0 & 1 \\ \mathrm{c} & 1 & 0 & 1 & 0 \\ \mathrm{d} & 0 & 1 & 0 & 1 \end{array},$$

where $x \sim y$ if and only if the entry of the table with row label x and column label y is 1. Reflexivity holds because the diagonal of the table consists of all 1's, and symmetry holds because the table is symmetric (as a matrix).

Example 2.40. If S is $\mathbb{N}$, $\mathbb{Z}$ or $\mathbb{R}$, then $\leq$ defined in the usual way is a total order on S.

Example 2.41. Both "less than" ($<$) and "greater or equal to" ($\geq$) are relations on $\mathbb{Z}$. For example, $4 < 5$, $2 \not< -1$, $3 \geq 3$, and $-3 \not\geq -2$.

Example 2.42. The *power set* $\mathcal{P}(S)$ of a set S is the set consisting of all subsets of S. The subset $\subset$ is a relation on $\mathcal{P}(S)$.

Definition 2.43. Given a total order $\leq$ on a set S, we define the *opposite* total order $\leq^{\mathrm{op}}$ by letting $s \leq^{\mathrm{op}} t$ if and only if $t \leq s$.

Definition 2.44. A relation $\sim$ on a set S is an *equivalence relation* if

(Reflexivity) For all $s \in S$, $s \sim s$;

(Transitivity) For all $s, t, u \in S$, if $s \sim t$ and $t \sim u$, then $s \sim u$;

(Symmetry) For all $s, t \in S$, $s \sim t$ if and only if $t \sim s$.

Note that the first two properties are in common with the notion of total order Definition 2.38, but, crucially, the third is not.

Example 2.45.

1. For every set S, the relation $\sim$ given by $x \sim y$ for all $x, y \in S$ is an equivalence relation;
2. The relation given by $x \sim y$ if and only $x = y$ is also an equivalence relation.

Example 2.46. The relation on $\mathbb{Z}$ given by $a \sim b$ if and only if $a - b$ is even is an equivalence.

- $a - a = 0$ which is even, thus reflexivity is satisfied;
- If $a - b$ is even, and $b - c$ is even, then $a - c = a - b + b - c$ is even as a sum of even numbers is even;
- If $a - b$ is even, then $b - a = -(a - b)$ is also even.

Definition 2.47. Let $\sim$ be an equivalence relation on a set S. For $x \in S$, the *equivalence class of* x, denote by $[x]$ is the set $\{y \in S \mid y \sim x\}$. We also refer to $[x]$ as an *equivalence class of* $\sim$.

Example 2.48. Let $E = \{\text{even integer numbers}\}$ and $O = \{\text{odd integer numbers}\}$. Consider the equivalence relation of Example 2.46. What is the equivalence class of 0 in this case? Take $z \in \mathbb{Z}$, $z - 0$ is even if and only if z is even. Thus, $[0] = E$. Analogously, the equivalence class of 1 is O.

Note that, in the previous example, $[-2] = [34] = [0] = [e]$ for all $e \in E$. This illustrates that different elements can have the same equivalence class. We will develop a fuller understanding of this below.

Proposition 2.49. *For every equivalence relation $\sim$ on a set S, every element in S is contained in exactly one equivalence class.*

Proof. For every $x \in S$, by reflexivity, $x \in [x]$. Suppose then $x \in [z] = \{y \in S \mid y \sim z\}$. Thus, $x \sim z$ and therefore $z \sim x$. For all $y \in [z]$, $y \sim z$ and thus, by transitivity, $y \sim x$ and $y \in [x]$. This shows that $[z] \subset [x]$. A similar argument shows that $[x] \subset [z]$. Therefore, $[x] = [z]$. ◻

Notation 2.50. Let S be a set and $\sim$ an equivalence relation on it. The symbol $S/_\sim$ denotes the equivalences classes of $\sim$ in S.

Example 2.51. Let $\sim$ be the equivalence relation on $\mathbb{Z}$ defined in Example 2.46. Then $\mathbb{Z}/_\sim = \{E, O\}$.

Lemma 2.52. *Let S be a set. For every equivalence relation $\sim$ on S and $x, y \in S$, we have $x \sim y$ if and only if $[x] = [y]$.*

Proof. Assume first $x \sim y$. By transitivity, if $z \sim y$ then $z \sim x$, thus $[y] \subset [x]$, and by analogous argument, $[x] \subset [y]$, proving the claim.

Conversely, assume $[x] = [y]$. By reflexivity, $x \sim x$, $x \in [x] = [y]$, and thus $x \sim y$. ◻

Let P be a set whose elements are all sets. Then we use the notation $\bigcup_{A \in P} A$ for the union of all the elements of P. That is, $x \in \bigcup_{A \in P} A$ if and only if there is a set $A \in P$ with $x \in A$.

Definition 2.53. Let S be a set, and P be a set of nonempty subsets of S. Then P is a *partition* of S if

- $\bigcup_{A\in P} A = S$, and
- for all $A \neq B \in P$, $A \cap B = \emptyset$.

In other words, a partition of S is a collection of non-empty subsets of S such that each element of S is contained in exactly one element of P.

Example 2.54. If $S = \{a, b, c, d\}$, then $P = \{\{a, d\}, \{b\}, \{c\}\}$ is a partition of S.

Remark 2.55. Given a set S with an equivalence relation $\sim$ on S, $S/\!\sim$ forms a partition of S. Conversely, if P is a partition of S, there is an associated equivalence relation $\sim$ on S where $x \sim y$ if and only if x and y belong to the same element of P. For example, the partition of Example 2.54 yields an equivalence relation given by the following table:

	a	b	c	d
a	1	0	0	1
b	0	1	0	0
c	0	0	1	0
d	1	0	0	1

In fact, these constructions define inverse functions between the set of all equivalence relations on S and the set of all partitions on S. By way of these functions, we think of equivalence relations and partitions as being "essentially the same thing."

2.7 Exercises on Sets and Functions

Exercise 2.1. For each pair of sets, say whether the two sets are equal.

1. $\{1, 2, 3\}$, $\{a, b, c\}$
2. $\{1, 2, 3\}$, $\{3, 2, 1\}$
3. $\{1, 2, 3\}$, $\{3, 2, 1, 0\}$

Exercise 2.2. Give an example of a set with three elements, each of which is a set.

Exercise 2.3.

1. How many elements does the set $\{\{1, 2\}\}$ have?
2. Is this set equal to the set $\{1, 2\}$?

Exercise 2.4. Adapting what is done in the notes/lecture for the even integers, write down two different expressions for the set of odd integers.

Exercise 2.5. Recall the definition of power set from Example 2.42

a. What is $\mathcal{P}(\{A, B\})$?

b. How many elements are in $\mathcal{P}(\mathcal{P}(\{A, B\}))$? Justify your answer.

Exercise 2.6. If T is a finite set with n elements,

a. how many elements does $\mathcal{P}(T^2)$ have?

b. how many elements does $(\mathcal{P}(T))^2$ have?

Exercise 2.7. Find $S \cup T$, $S \cap T$, $S \setminus T$ and $T \setminus S$ for

a. $S = \{1, 2, 3\}$ and $T = \{1, 3, 5\}$,

b. $S = \{a, b, c, d\}$ and $T = \{a, d\}$,

c. $S = \mathbb{N}$ and $T = \mathbb{Z}$,

d. $S = \emptyset$ and $T = \{2, 4, 6\}$.

Exercise 2.8. Which of the following sets are empty for all sets S and T?

a. $S \setminus (S \cup T)$,

b. $S \setminus (S \cap T)$,

b. $S \cup \emptyset$,

c. $S \cap \emptyset$,

d. $S \setminus \emptyset$,

e. $\emptyset \setminus S$,

f. $(S \cap \{1\}) \cap (S \cap \{2\})$.

Exercise 2.9. For each of the following subsets of $\mathbb{R}$, state whether the subset is an interval, and if so, express it in interval notation.

a. $\{3\}$

b. $\mathbb{Z}$

c. $\{0, 3\}$

d. $\{x \in \mathbb{R} \mid x < 0\}$

e. $\{x \in \mathbb{R} \mid x \leq 1\}$

f. $\{x \in \mathbb{R} \mid -4 < x \leq 1\}$

g. $\{x \in \mathbb{R} \mid x < 3 \text{ and } x > 0\}$

h. $\{x \in \mathbb{R} \mid x < 3 \text{ or } x > 0\}$

i. $\{x \in \mathbb{R} \mid x < 0 \text{ or } x > 3\}$

Exercise 2.10. Express each of the following intervals in the bracket notation for sets.

a. $[3, \infty)$

b. $(-\infty, 3)$

c. $[3, 5)$

d. $(3, 5)$

e. $[3, 5]$

Exercise 2.11. For each of the following pairs of sets $S, T \subset \mathbb{R}$, sketch the Cartesian product $S \times T$.

a. $S = \{1\}$, $T = \emptyset$

b. $S = \{1\}$, $T = \{2\}$

c. $S = \{1, 2\}$, $T = \{2\}$

d. $S = \{1, 2, 3\}$, $T = \{2\}$

e. $S = \{1\}$, $T = \{1, 2, 3\}$

f. $S = \{1, 2, 3, 4\}$, $T = \{1, 2, 3, 4\}$

g. $S = \{1\}$, $T = \mathbb{R}$

h. $S = [0, 1]$, $T = \{1\}$

i. $S = [0, 1]$, $T = [0, 2]$

Exercise 2.12. Is the function from Figure 2 injective or surjective?

Exercise 2.13. Prove Proposition 2.36.

Exercise 2.14.

a. For $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x - 1$, what is $\mathrm{im}(f)$?

b. For $f : [0, 2) \to \mathbb{R}$ given by $f(x) = x - 1$, what is $\mathrm{im}(f)$?

Exercise 2.15. For each function $f : S \to T$, state whether f in injective, surjective, bijective, or "none."

a.

$$\begin{gathered} S = \{1, 2\}, \qquad T = \{A, B, C\}, \\ f(1) = B,\ f(2) = C. \end{gathered}$$

b.

$$\begin{gathered} S = \{1, 2, 3\}, \qquad T = \{A, B\}, \\ f(1) = B,\ f(2) = A,\ f(3) = B. \end{gathered}$$

c.

$$\begin{gathered} S = T = \mathbb{N}, \\ f(z) = z + 3. \end{gathered}$$

d.

$$\begin{gathered} S = T = \mathbb{N}, \\ f(z) = 2z. \end{gathered}$$

e.

$$\begin{gathered} S = T = \mathbb{R}, \\ f(x) = x^2 - 1. \end{gathered}$$

Exercise 2.16. For each function, give an explicit expression for its image, and say whether the function is an injection, surjection, or bijection. If the function is a bijection, also give its inverse.

a. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^3$

b. $f : \mathbb{R} \to \mathbb{R}^2$, $f(x) = (x, x^3)$ [here, just use the bracket notation to express $\mathrm{im}(f)$.]

c. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^4$

d. $f : \mathbb{R} \to [0, \infty)$, $f(x) = x^4$

e. $f : \mathbb{R} \to [-1, 1]$, $f(x) = \cos x$

f. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = |x|$.

g. $g : S^1 \times I \to \mathbb{R}^2$, $g(x, y) = x$. (Here, it is understood that $x \in S^1$ and $y \in I$.)

h. $h : S^1 \to \mathbb{R}^2$, $h(x, y) = (|x|, |y|)$.

Exercise 2.17.

a. Give a bijection $f : (0, 1] \to [1, \infty)$. What is the inverse of f?

b. Give a bijection $g : (0, 1) \to [0, 1]$.

Exercise 2.18.* Define a total order on $\mathbb{Z}$ that is not the usual $\leq$ or $\leq^{op}$.

Exercise 2.19. A total order on a set S is *well-ordered from below* if any non-empty subset of S has a smallest element (i.e. an element m such that for all $x \neq m$ in the subset, $m \leq x$).

- Is the usual total order on $\mathbb{N}$ well-ordered from below?
- Is the usual total order on $\mathbb{R}$ well-ordered from below?

Exercise 2.20. Describe the equivalence classes of the following equivalence relations:

1. $S = \mathbb{R}^2$, and $a \sim b \in S$ if a and b are at the same Euclidean distance from the origin;
2. $S = \mathbb{R}$, and $a \sim b \in S$ if $b = -a$ or $b = a$;
3. $S = \mathbb{R}$, and $a \sim b \in S$ if $a - b \in \mathbb{Z}$;

Exercise 2.21. Let $|$ be the relation on $\mathbb{Z}$ given by $a|b$ is there exists $c \in \mathbb{Z}$ such that $bc = a$ (i.e., b divides a). Check whether or not $|$ satisfies the axioms of total order and equivalence relation. Would the result change on $\mathbb{Q}$?

Exercise 2.22. Given the set $S = \{1, 2, 3, 4\}$, and suppose that $\sim$ is an equivalence relation on S. You are given information that $1 \sim 2$ and $2 \sim 3$. Show that there are exactly two possibilities for the relation $\sim$, and describe both (i.e., for all a, $b \in S$, say whether or not $a \sim b$).

Exercise 2.23. Let $S = \{a, b, c\}$.

- Is $\{a, b, c\}$ a partition of S?
- Find all partitions of S.

Exercise 2.24. Which of the following relations $\sim$ on $\mathbb{Z}$ are equivalence relations? Explain your answer, and for each relation that is an equivalence relation, give the set $\mathbb{Z}/\sim$ of all equivalence classes.

a. $x \sim y$ iff $x - y$ is divisible by 3. (Note: 0 is considered to be divisible by 3.). HINT: Compare this problem to Example 2.48, recalling that by definition, an even integer is an integer which is divisible by 2.

b. $x \sim y$ iff $x - y = 1$,

c. $x \sim y$ iff $xy \geq 0$.

d. $x \sim y$ iff $x = y$ or $x = -y$.

Exercise 2.25. Which of the following tables define equivalence relation $\sim$ on $\{a, b, c, d\}$? For each, if the table defines an equivalence relation, give the corresponding set of equivalence classes.

(i)

	a	b	c	d
a	1	1	0	1
b	1	1	0	0
c	0	0	1	0
d	1	0	0	1

(ii)

	a	b	c	d
a	1	0	0	1
b	0	1	1	0
c	0	1	1	0
d	1	0	0	1

Exercise 2.26. Suppose we are given an equivalence relation $\sim$ on a set S and elements $a, b, c \in S$ such that such that $a \not\sim b$ and $a \sim c$. Is it true that $b \sim c$? Explain your answer.

Exercise 2.27. Consider the equivalence relation on $\mathbb{R}^2$ given by $(x_1, x_2) \sim (y_1, y_2)$ if and only if $x_1 = y_1$. Give an explicit description of $\mathbb{R}^2/_\sim$.

Exercise 2.28. Suppose we are given an equivalence relation $\sim$ on a set S and elements $a, b, c, d \in S$ such that such that $a \not\sim b$, $b \sim c$, and $c \not\sim d$. Is always true, sometimes true, or never true that $a \sim d$? Explain your answer.

3 Graphs

3.1 Types of graphs

A graph is an object consisting of a set of vertices and a set of edges, where each edge connects two vertices. Formally, we have the following definitions.

Definition 3.1.

(i) An *undirected graph* is a pair $G = (V, E)$, where V is a finite set and E is a set of subsets of V of size two.

(ii) A *directed graph* is a pair $G = (V, E)$, where V is a finite set and E is a set of pairs (a, b) where $a, b \in V$, and $a \neq b$.

In both cases, we call the elements of V *vertices* and the elements of E *edges*.

Let v be a vertex of a graph G, and let e be an edge of G with $v \in e$. Then we say that v and e are *incident* to each other. The *degree* of v, denoted by $\deg(v)$, is the number of edges of G that are incident to e.

Example 3.2. Let $G = (V, E)$, with $V = \{a, b, c, d\}$ and $E = \{\{a, b\}, \{b, c\}\}$. Then G is a undirected graph with four vertices a, b and c, and two edges, one between a and b, and one between b and c. The vertex b has degree 2, a and c both have degree 1, and d has degree 0; see Figure 3a.

Example 3.3. Let $G = (V, E)$, with $V = \{a, b, c\}$ and $E = \{(a, b), (b, c), (a, c), (c, a)\}$. Then G is a directed graph with three vertices a, b and c, and four edges, one from a to b, one from b to c, and one in each direction between b and c; see Figure 3b.

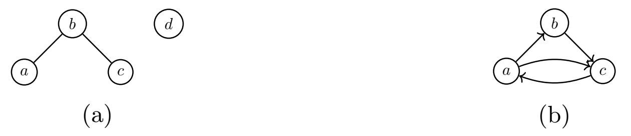


Figure 3: Depiction of the graphs of Example 3.2 (a) and Example 3.3 (b)

Definition 3.4. A graph $G = (V, E)$ is *complete* if for every $u, v \in V$ with $u \neq v$, there exists an edge $\{u, v\}$.

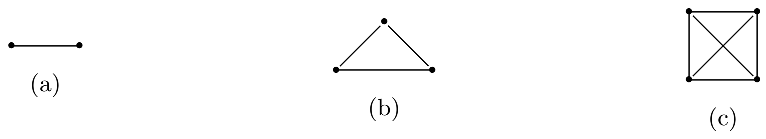


Figure 4: Complete graphs on 2, 3, and 4 vertices respectively.

Note that there is no unique way to draw a graph, even if some choices are clearer than others:

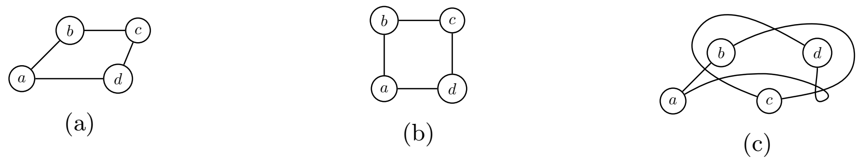


Figure 5: Three depictions of the same graph.

Definition 3.5. A *weighted graph* is a triple (V, E, w), where (V, E) is a (directed or undirected) graph and $w \colon E \to \mathbb{R}$ is a function. We call $w(e)$ the *weight* of an edge $e \in E$.

To distinguish the graphs of Definition 3.1 (where no weight function w is specified) from weighted graphs, we call the former *unweighted*.

Example 3.6. Let G be the graph in Example 3.2 with $w \colon E \to \mathbb{R}$ defined by $w(\{a, b\}) = 2$ and $w(\{b, c\}) = 3$. This makes G into a weighted graph, drawn in Figure 6. The figure illustrates the graph in the standard way, with each edge e labeled with its weight $w(e)$.

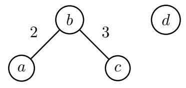


Figure 6: Weighted graph from Example 3.6.

Definition 3.7. An unweighted graph $G = (V, E)$ can be regarded as a weighted one, by taking $w(e) = 1$ for all $e \in E$.

From now on, "graph" will mean "unweighted undirected graph" unless we explicitly mention that the graph is directed or weighted (or both). This kind of graph is sometimes referred to as a *simple* graph, to distinguish it from a more general class of undirected graphs where one allows loops and multiple edges.

Definition 3.8. Let $G = (V, E)$ be a graph, and suppose we have fixed a total order on V. Write $V = \{v_1, \ldots, v_n\}$ respecting this total order. The *adjacency matrix* of G is the matrix M for which $M_{i,j} = 0$ if there is no edge between v_i and v_j, and $M_{i,j} = 1$ if there is an edge.

Weighted and directed graphs can also be described conveniently using matrices. If the graph is directed, then we can define $M_{i,j}$ as zero or one depending on whether there is an edge *from* v_j *to* v_i. Observe that this can make the matrix asymmetric, while the adjacency matrix of an undirected graph is always symmetric. If the (directed or undirected) graph is weighted, then we can let $M_{i,j}$ be the weight of the edge from v_j to v_i (if the edge exists).

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 1 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 0 \\ d & 0 & 0 & 0 & 0 \end{array}$$

(a)

$$\begin{array}{c|ccc} & a & b & c \\ \hline a & 0 & 0 & 1 \\ b & 1 & 0 & 0 \\ c & 1 & 1 & 0 \end{array}$$

(b)

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 1 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 1 \\ d & 1 & 0 & 1 & 0 \end{array}$$

(c)

Figure 7: Adjacency matrices of the graphs Figure 3a, Figure 3a, and Figure 5, respectively.

$$\begin{bmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & 0 & 1 & \dots & 1 \\ \vdots & & \ddots & & \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix}$$

(a)

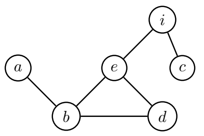
$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 2 & 0 & 0 \\ b & 2 & 0 & 3 & 0 \\ c & 0 & 3 & 0 & 0 \\ d & 0 & 0 & 0 & 0 \end{array}$$

(b)

Figure 8: Adjacency matrices for a complete graph, and of the weighted graph in Example 3.6, respectively.

$$V = \{a, b, c, d, e, i\}$$
$$E = \{\{a, b\}, \{b, e\}, \{d, b\}, \{e, d\}, \{i, e\}, \{c, i\}\}$$

(a)



(b)

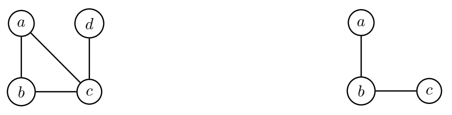
$$\begin{array}{c|cccccc} & a & b & c & d & e & i \\ \hline a & 0 & 1 & 0 & 0 & 0 & 0 \\ b & 1 & 0 & 0 & 1 & 1 & 0 \\ c & 0 & 0 & 0 & 0 & 0 & 1 \\ d & 0 & 1 & 0 & 0 & 1 & 0 \\ e & 0 & 1 & 0 & 1 & 0 & 1 \\ i & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

(c)

Figure 9: Three ways to describe a graph: with a list of vertices and edges, with drawing, and with an adjacency matrix.

Graphs are extremely useful for describing structures and connections in the real world. For instance, one can define a weighted graph where the vertices are cities, and there is a weighted edge between two cities labeled with the length of the shortest road between them. Another widespread example is the one of social media: a graph where the vertices are users on a social media platform. The edges may be directed (for example, $a \to b$ if user a follows user b) or undirected (for example, $a - b$ if a and b are friends). It is easy to store and manipulate graphs on computers. Using algorithms on graphs, data scientists analyze travel options, how information flows through social networks, genetic subtypes of pathogens, and much more.

Definition 3.9. Let $G = (V, E)$ be a graph. A *subgraph* of G is a graph $G' = (V', E')$ such that $V' \subset V$, and $E' \subset E$. If $V' \subset V$, then the *subgraph of* G *induced by* V' is the graph $G' = (V', E')$, where E' is the set of all edges in E between vertices in V'.



(a) (b)

Figure 10: A graph (a) and a subgraph of it (b) which is not induced.

Note that we cannot choose any subsets $V' \subset V$ and $E' \subset E$ and be guaranteed to obtain a subgraph (V', E'), since if E' contains an edge between $u \notin V'$ and v, then (V', E') is not a graph.

Definition 3.10. A *clique* in a graph G is a complete subgraph of G.

Example 3.11. In the graph depicted in Figure 10(a), the vertices a, b, and c form a clique.

3.2 Paths and subgraphs

Definition 3.12. Let u and v be vertices of an undirected graph G. A *walk from u to v* in G is a sequence P of vertices

$$u = v_0, v_1, \dots, v_\ell = v$$

of G, where $\ell \geq 0$ and $\{v_i, v_{i+1}\} \in E$ for $i = 0, \dots, \ell - 1$. If all the vertices $v_0, \dots, v_\ell$ are different, we call P a *path.* If $\ell \geq 3$, $v_0, \dots, v_{\ell-1}$ is a path, and $v_0 = v_\ell$, then we call P a *cycle.*

Note that by putting $\ell = 0$, we get a path of length zero from any vertex to itself.

The above definitions extend to weighted graphs, simply by ignoring the weight function.

Definition 3.13. For $G = (V, E, w)$ a weighted graph, the *length* (or *cost*) of a walk $W = v_0, \dots, v_\ell$ in G, is the quantity

$$\text{length(W)} = \sum_{i=0}^{l-1} w(\{v_i, v_{i+1}\}).$$

In particular, if G is unweighted (i.e., $w(e) = 1$ for all $e \in E$), then the length of this walk is ℓ.

Example 3.14. In Figure 3b, a, b, c, a, c is a walk, writing it as the sequence v_0, v_1, v_2, v_3, v_4, where $v_0 = v_3 = a$, $v_1 = b$, and $v_2 = v_4 = c$. This walk has length 4, and it is neither a path nor a cycle, since $v_0 = v_3$ and $v_2 = v_4$. In the same graph, a, b, c is a path. Note that, again in the same graph, the sequence a, b, c, a, c, a is a walk and not a cycle, even if it starts and ends at the same vertex, because the vertex c is visited twice. Still on the same graph, an example of a cycle is a, b, c, a. Another example of a cycle is a, d, c, b, a in Figure 5.

Definition 3.15. A path P between the vertices v and u in G is called a *shortest path* or *minimal path* if there is no other path P' in G between v and u such that $\text{Length}(P') \leq \text{Length}(P)$

There can be multiple minimal paths. For example, in the graph from Figure 5, both the path a, b, c and a, d, c are minimal.

3.3 Connected components

Definition 3.16. A graph $G = (V, E)$ is *connected* if for all $u, v \in V$, there is a path from u to v in G.

We can split any graph into pieces such that each piece is connected, and there is no edge between different pieces. We call each such piece as a connected component. We can formally define these using terminology from the previous section.

Definition 3.17. Let $G = (V, E)$ be a graph, and let $\sim$ be the equivalence relation on V with $u \sim v$ if and only if there is a path from u to v. A connected component of G is a subgraph induced by an equivalence class of $\sim$.

Example 3.18. Let G be a connected graph. Then G has exactly one connected component, which is G itself.

Example 3.19. Let G be the graph in Figure 6. Then G has two connected components $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$, where

$$\begin{aligned} V_1 &= \{a, b, c\}, \\ E_1 &= \{\{a, b\}, \{b, c\}\}, \\ V_2 &= \{d\}, \\ E_2 &= \emptyset. \end{aligned}$$

C_1 is the subgraph of G induced by V_1, and C_2 is the subgraph of G induced by V_2.

3.4 Trees

Definition 3.20.

(ii) A *forest* is a graph with no cycle.

(i) A *tree* is a connected forest with at least one vertex.

As the names suggest, a forest is a graph made of trees, i.e., each connected components of a forest is a tree. It is possible to show the following theorem:

Theorem 3.21. *Let T be a tree with n vertices. Then T has exactly $n - 1$ edges.*

An intuitive explanation of this theorem is that we need at least $n - 1$ edges to connect all the vertices, but the moment we add more edges than that, we start creating cycles.

Definition 3.22. A *leaf* of a tree T is a vertex v with $\deg(v) = 1$.

Definition 3.23. A tree with a distinguished vertex, called the *root*, is called a *rooted tree.*

Typically, rooted trees are drawn by placing the root at the top, as shown below, where the root is r:

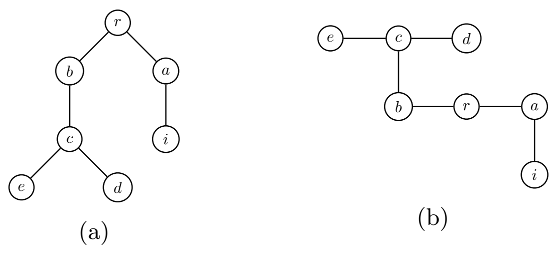


Figure 11: Two drawings of the same graph, on the left the standard way of illustrating a rooted tree.

Example 3.24. Rooted trees are useful to store data on a computer in a way that makes it easy to retrieve information. Consider the example in Figure 12. This tree has the property that for any vertex v, the value of vertices below and to the left of v are less than v, while the value of vertices below and to the right of v are more than v. A tree with this property is called *binary search tree (BST)*, a fundamental data structure in computer science. By storing a set of numbers in a BST, we can determine if a given number n is in the set by starting at the root and walking downwards along the edges, going left or right depending on whether n is smaller than or greater than the number stored at the current vertex. For example to search the tree of Figure 12 for $n = 30$, we start at the root, 37. $30 < 37$, so we go to the left. $30 > 19$ so we next go right. $30 > 29$, so we go right again. Finally we arrive vertex 31. This has no nodes below it, so we report that 31 is not in the set of numbers stored in this BST.

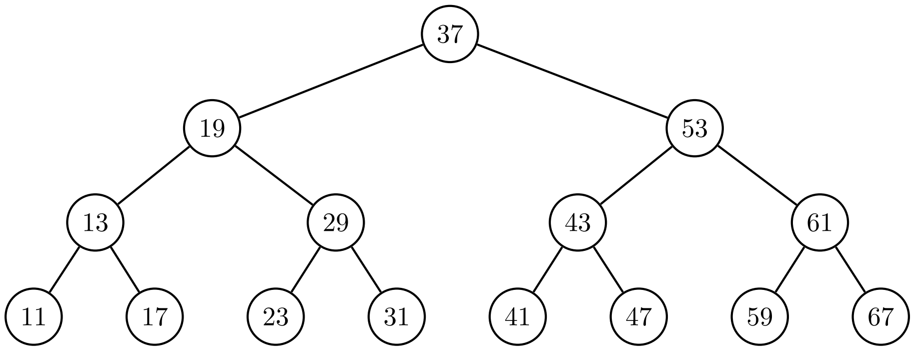


Figure 12: A set of numbers stored in a binary search tree.

Definition 3.25. A *spanning tree* of G of a graph $G = (V, E)$ is a subgraph $T = (V, E')$ of G which is a tree.

Theorem 3.26. *A graph G has a spanning tree if and only if G is connected.*

Definition 3.27. Let G be a connected weighted graph. For any spanning tree $T = (V_T, E_T)$ of G, let

$$w(T) = \sum_{e \in E_T} w(e).$$

If T' is a spanning tree of G such that $w(T') \le w(T)$ for any spanning tree T of G, then T' is a *minimum spanning tree* of G.

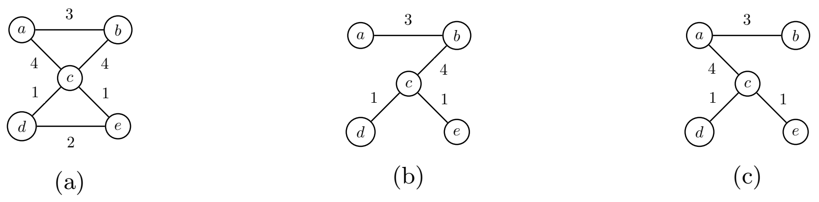


Figure 13: A graph (a) and its two minimum spanning trees (b) and (c).

Example 3.28. Let G be the graph drawn in Figure 13. There are two minimum spanning trees of G. Observe that we cannot obtain a minimum spanning tree by simply removing the two edges with the highest weight, since the resulting graph would not be a tree.

Example 3.29. In the graph in Figure 3a, the vertices a, b, c and d have degree 2, 3, 3 and 4, respectively.

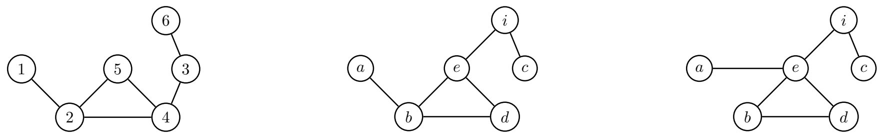
Here are some fun mathematical facts about degree. See if you can prove them yourself.

Theorem 3.30. *Let $G = (V, E)$ be a graph. Then the number of edges of G is equal to $\frac{1}{2} \sum_{v \in V} \deg(v)$.*

Theorem 3.31. *Let G be a graph. The number of vertices with an odd degree is even.*

3.5 Properties of graphs

Definition 3.32. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a bijection $f\colon V_1 \to V_2$ such that $\{v_i, v_j\} \in E_1$ if and only if $\{f(v_i), f(v_j)\} \in E_2$. We write $G_1 \cong G_2$ if G_1 and G_2 are isomorphic, and we say that f is an *isomorphism* between G_1 and G_2.



(a) G_1 (b) G_2 (c) G_3

Figure 14: G_1 and G_2 are isomorphic, but G_3 is not isomorphic to either of the first two.

Many important properties of graphs are *isomorphism invariant*; that is, if G and H are isomorphic graphs and G has such a property, then H has this property too. For instance, the number of edges or vertices is isomorphism invariant, and so are the properties of being a tree, being complete, having a path of a certain length, and just about any meaningful property that a graph can have. Informally, we consider isomorphic graphs G and H to be "essentially the same", or to have the same structure, even if we do not have $G = H$.

Example 3.33. In Figure 14, we have an isomorphism $f\colon G_1 \to G_2$ given by $f(1) = a$, $f(2) = b$, $f(3) = i$, $f(4) = e$, $f(5) = d$, $f(6) = c$. G_3 is not isomorphic to either G_1 or G_2. One way to prove this is to observe that G_3 has a vertex with degree 4, and G_1 and G_2 do not. This is an example of an isomorphism invariant, and how such invariants can be used to prove that certain graphs are not isomorphic.

It is natural to ask if there is an efficient way to check if two graphs are isomorphic. This is actually a famous open question: `https://en.wikipedia.org/wiki/Graph_isomorphism_problem`. When we talk about computational complexity later in the course, we will introduce machinery that lets us discuss what it means for a problem to be efficiently solvable.

Theorem 3.34. *Let $\mathcal{G}$ be a set of graphs. Then isomorphism is an equivalence relation on $\mathcal{G}$.*

The diameter of a graph is another example of an isomorphism invariant:

Notation 3.35. Given a graph $G = (V, E)$, we denote by $p(v, w)$ the length of the shortest path between v and w in G.

Definition 3.36. The *diameter* of a graph $G = (V, E)$ is $\max_{u,v\in V} p(u, v)$. That is, it is the length of the "longest shortest path" in G.

Example 3.37. Any complete graph with at least two vertices has diameter 1, because the shortest path between any pair of vertices is formed by a single edge. In Figure 12, the diameter is 6, which is the shortest path between any leaf node on the left and any leaf node on the right.

Definition 3.38. Let $G = (V, E)$ be a graph with $n \geq 2$ vertices and v a vertex in G. The *closeness centrality* of v is

$$C(v) = \frac{n-1}{\displaystyle\sum_{w\in V} p(v, w)}\,.$$

Intuitively, a node with a large closeness centrality in a graph is a "highly connected" node, or a *hub*. The factor $n - 1$ normalizes the value, so that we can compare the centrality of vertices in different graphs of various sizes.

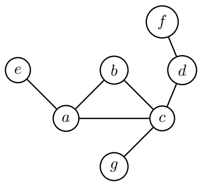


Figure 15: G_1

Example 3.39. Let us compute the closeness centrality of the vertices a, c, and f in the graph depicted in Figure 15. First, we count the length of the shortest path from each of these vertices to all the other vertices:

$$\begin{array}{c|cccccc} & b & c & d & e & f & g \\ a & [1 & 1 & 2 & 1 & 3 & 2] \end{array} \qquad \begin{array}{c|cccccc} & a & b & d & e & f & g \\ c & [1 & 1 & 1 & 2 & 2 & 1] \end{array} \qquad \begin{array}{c|cccccc} & a & b & c & d & e & g \\ f & [3 & 3 & 2 & 1 & 4 & 3] \end{array}$$

Therefore, we have

$$\begin{aligned} C(a) &= \frac{7-1}{1+1+2+1+3+2} = \frac{6}{10} = \frac{3}{5} \\ C(c) &= \frac{7-1}{1+1+1+2+2+1} = \frac{6}{8} = \frac{3}{4} \\ C(f) &= \frac{7-1}{3+3+2+1+4+3} = \frac{6}{16} = \frac{3}{8} \end{aligned}$$

3.6 Exercises on Graphs

Exercise 3.1. Write the graph from Figure 5 using the formal notation of Definition 3.1.

Exercise 3.2. Draw every graph with n vertices up to isomorphism, for $n = 1, 2, 3, 4$. Hint: do not label the vertices.

Exercise 3.3. The *complement* of a graph $G = (E, V)$ is the graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{V} = V$ and edges given by all possible edges over V that are not in E. Draw the complement of the graphs in Figure 3a and Figure 5.

Exercise 3.4. List all the possible induced subgraphs of the graphs in Figure 10.

Exercise 3.5. If 5 people shake hands with each other, how many handshakes take place? What does this have to do with graph theory?

Exercise 3.6. Among a group of 5 people, is it possible for everyone to be friends with exactly 2 of the people in the group (assuming that if A is friends with B, then B is friends with A)? What about 3 of the people in the group?

Hint: for the second question, use Theorem 3.31

Exercise 3.7. Write, using all three notations, a graph with 7 vertices and 10 edges.

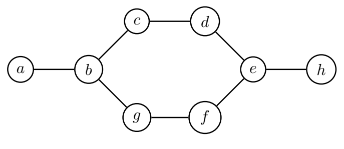
Exercise 3.8. Write, using all three notations, a forest with 8 vertices and 3 connected components.

Exercise 3.9.* How many edges does a complete graph on n vertices have?

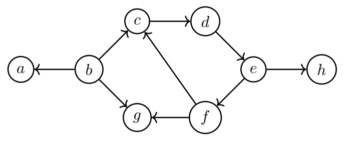
Exercise 3.10.* Prove Theorem 3.26.

Exercise 3.11.* Prove Theorem 3.30.

Exercise 3.12.* Prove Theorem 3.31.



(a)



(b)

Figure 16: Depiction of the graphs of Exercise 3.14 (a) and Exercise 3.15 (b).

Exercise 3.13. Prove that the relation on the set of vertices V of a graph defined in Definition 3.17 is an equivalence relation, and thus the connected components of a graph are equivalence classes.

Exercise 3.14. Let G be the undirected graph drawn in Figure 16 (a).

a. Find a path of maximal length in G. What is the length of the path you found?

b. Find a cycle in G.

c. Find a spanning tree for G.

d. How many spanning trees does G have?

Exercise 3.15. Let G be the directed graph drawn in Figure 16 (b).

a. Find a path of maximal length in G. What is the length of the path you found?

b. Find a cycle in G.

c. Find all the paths from b to g in G.

Exercise 3.16.* Show that if there is a walk from u to $v \neq u$ in a graph G, then there is a path from v to u.

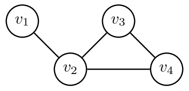


Figure 17

Exercise 3.17. Let G be the graph in Figure 17. What is the number of walks of length exactly 3 in G? What is the number of walks of length at most 3 in G? What is the number of walks of length 3 from v_1 to v_3? From v_3 to itself?

Exercise 3.18.* Show Theorem 3.21.

Exercise 3.19.

a) Draw four trees with 6 vertices each. One with (exactly) 5 leaves, one with 4, one with 3, and one with 2.

b)* Prove that it is impossible for a tree with $n \geq 2$ vertices to have more than $n - 1$ or less than 2 leaves.

Exercise 3.20. Find the diameter of the graph in Figure 16a and the closeness centrality of a, b and c. Do the same for Figure 14a and the vertices 4 and 6.

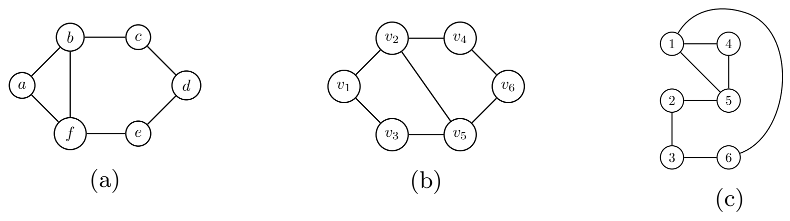


Figure 18: Depiction of the graphs of Exercise 3.21.

Exercise 3.21. For each pair of graphs in Figure 18, explicitly write an isomorphism or show why they are not isomorphic.

Exercise 3.22. For each of the following graphs $G = (V, E)$, (i) sketch the graph, (ii) give an explicit expression for each connected component, and (iii) say whether the graph is a tree or forest.

a. $V = \{a, b, c, d, e\}$, $E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}\}$.

b. $V = \{a, b, c, d, e\}$, $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$,

c. $V = \{a, b, c, d, e, f\}$, $E = \{\{a, b\}, \{b, c\}, \{c, a\}, \{d, e\}, \{e, f\}\}$,

d. $V = \{a, b, c, d, e, f\}$, $E = \{\{a, b\}, \{b, c\}, \{c, a\}, \{d, e\}, \{e, f\}, \{f, a\}\}$,

e. $V = \{a, b, c\}$, $E = \{\}$.

4 Metric Spaces

4.1 Definitions and Basic Examples

There are many settings in data science, computer science, and mathematics where one wishes to measure the *distance* between a pair of mathematical objects. *Metric spaces* are the standard formalism for this:

Definition 4.1 (Metric space)**.** A *metric* on a set S is a function $d : S \times S \to [0, \infty)$ satisfying the following three properties:

(i) $d(x, y) = 0$ if and only if $x = y$,

(ii) (symmetry) $d(x, y) = d(y, x)$ for all $x, y \in S$,

(iii) (triangle inequality) $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in S$.

A *metric space* is a pair (S, d), where S is a set and d is a metric on S.

The first two properties are very natural. The third, the triangle inequality, might be a bit more mysterious on first encounter. It can be seen as a sort of stability principle for metrics. Exercise 4.1 gives a precise statement of the stability of a metric, which is an immediate consequence of the triangle inequality.

Definition 4.2. The most important and most familiar metric is the *Euclidean distance* d_2 on $\mathbb{R}^n$, given by

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

This has a simple geometric interpretation: For $x \neq y$, $d_2(x, y)$ is the length of the line segment in $\mathbb{R}^n$ connecting x and y. The triangle inequality has a corresponding geometric interpretation: In the case that x, y, and z are all distinct and not collinear (i.e. there exists no line intersecting all three points), the triangle inequality is the statement that the length of any side of a triangle is always less than the sum of the lengths of the other two sides. This is where the name "triangle inequality" comes from.

Definition 4.3. Extending the above definition, for $p \in [1, \infty)$, we define a metric d_p on $\mathbb{R}^n$ by

$$d_p(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}.$$

In particular, $d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$. The metric d_1 is often called the *Manhattan distance*, or the *taxicab metric*. We also define a metric d_∞ on $\mathbb{R}^n$ by

$$d_\infty(x, y) = \max_{1 \le i \le n} |x_i - y_i|.$$

It can be shown that for any $x, y \in \mathbb{R}^n$, we have

$$d_\infty(x, y) = \lim_{p \to \infty} d_p(x, y),$$

which explains the choice of notation. Note that when $n = 1$, all of these metrics are are equal.

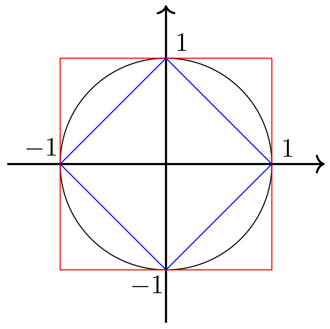


Figure 19: The points with distance 1 to the origin, with respect to different metrics on $\mathbb{R}^2$: The case of d_1 is illustrated in blue, d_2 in black, and d_∞ in red.

Remark 4.4. If $M = (S, d)$ is a metric space, we often abuse notation/language slightly and conflate M with its underlying set S. This convention sometimes allows us to simplify notation. For example,

- "an element of M" means "an element of S,"
- a "subset of M" means "a subset of S,"
- if $M = (S, d)$ and $M' = (S', d')$ are metric spaces, a function $f : M \to M'$ is understood to be a function $f : S \to S'$.
- $M = (S, d)$ is called *finite* if S is finite.

Definition 4.5 (Subspaces of metric spaces)**.** If $M = (S, d)$ is a metric space and $S' \subset S$, then there is a metric d' on S' given by $d'(x, y) = d(x, y)$ for all $x, y \in S'$. Thus, we can always regard a subset of a metric space as a metric space. We call (S', d') a *subspace* of M, and call d' the metric *induced by d*.

We sometimes use the same notation for d and the induced metric d'.

Example 4.6. Let $M = (\mathbb{R}^2, d_2)$ and let $S' = S^1$, i.e., S' is the unit circle. Then, letting d' be the induced metric of Definition 4.5, $d'((1,0),(0,1)) = \sqrt{2}$.[3]

We are particularly interested in Definition 4.5 in the case that the subset S' is finite, because real-world data is often modeled as a finite subspace of a metric space.

Remark 4.7 (Matrix representation of finite metric spaces)**.** Let $M = (S, d)$ be a finite metric space. Ordering the points of S arbitrarily, let us write the i^{th} point of S as s_i. We can then represent d as a matrix D, where $D_{ij} = d(s_i, s_j)$. D is symmetric and has zeros on the diagonal. Such matrices are the input to many data science algorithms, e.g., for clustering.

Example 4.8. For S any set, we can define a metric on S by

$$d(x,y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

The first two properties of a metric are clear. The triangle inequality follows from a simple case analysis.

We leave the proof of the following proposition as an exercise.

Proposition 4.9. *Let $G = (V, E, w)$ be a weighted connected graph such that $w(e) > 0$ for all e and $\operatorname{im}(w)$ is finite. Define*

$$d \colon V \times V \to [0, \infty)$$

by taking $d(u, v)$ to be the length of a shortest path from u to v. Then d is a metric on V.

The metric of Proposition 4.9 is called the *shortest path metric on G.* Note that the condition that $\operatorname{im}(w)$ is finite in Proposition 4.9 is satisfied if either E is finite or G is unweighted.

Proposition 4.9 gives a way of constructing a metric space from a graph. We next give a way of constructing a graph from a metric space:

Definition 4.10. Given $r \geq 0$ and a metric space (S, d), we define the r-neighborhood graph $N(S)_r = (S, E_r)$ by letting $\{x, y\} \in E_r$ if $x \neq y$ and $d(x, y) \leq r$.

4.2 Edit Distance

We now describe a metric, the *edit distance*, arising naturally in bioinformatics and computer science. Let S be a finite set, and let V denote the set of all finite sequences of elements of S, including the empty sequence. We will call the elements of S *characters* and the elements of V *words.* For example, in genetics applications, we take $S = \{A, C, G, T\}$, which stand for the nucleotides adenine, cytosine, guanine, and thymine, the four nucleotides which are the building blocks of DNA. Then V is the set of all possible DNA sequences.

Definition 4.11. We define three kinds of *elementary operations* on a word in V:

- *replace* one character with another at a single position,
- *remove* one character at a single position;
- *insert* one character at any position.

Let $G = (V, E)$ be the graph where E consists of pairs $\{v, w\}$ such that v and w differ by a single elementary operation. The *edit distance* on V is the shortest path metric on $G = (V, E)$.

[3]Some readers might have expected that $d'((1,0)(0,1)) = \frac{\pi}{2}$, the length of the shortest arc connecting $(1,0)$ and $(0,1)$ in S^1. There is a different metric m on S^1, the *intrinsic metric*, for which $d(x,y)$ is the length of the shortest arc in S^1 connecting x and y. The intrinsic metric of a subspace of a metric space can be defined in much more generality, but some technical conditions are required. We will not discuss intrinsic metrics further here.

Example 4.12. As an example, let us compute the pairwise distances between the following three sequences:

$x =$AATTACA

$y =$GATTACA

$z =$ATTA

We have $d_e(x, y) = 1$, since it is enough to change the first elements in both strings. On the other hand, $d_e(x, z) = d_e(y, z) = 3$, because we need to add 3 elements to z to obtain either x or y.

Example 4.13. Let us find all the words that have distance one from the word AB over the set $\{A, B, C\}$. We can get BB, CB, AA or AC by replacing a letter. We can get A or B by removing a letter. By inserting a letter, we can get AAB, ABA, BAB, ABB, CAB, ACB or ABC. Since all of these were obtained from AB with a single elementary operation, they all have distance one from AB.

Note that the edit distance takes only integer values.

4.3 Wasserstein Distance

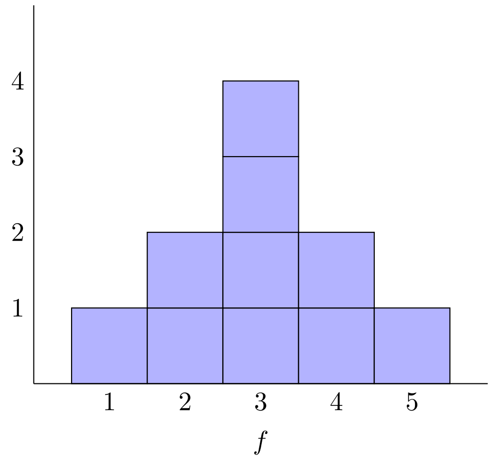
We next consider another metric, the *Wasserstein distance*, which plays a major role in data science. The Wasserstein distance is sometimes called the *Earth mover's distance*, where in this case "Earth" is synonymous with "dirt". We'll introduce the Wasserstein distance in a way that emphasizes the interpretation of this distance in terms of transporting piles of dirt. To minimize the technicalities, we introduce the definition only in a very restrictive setting; we comment on generalizations in Remark 4.23.

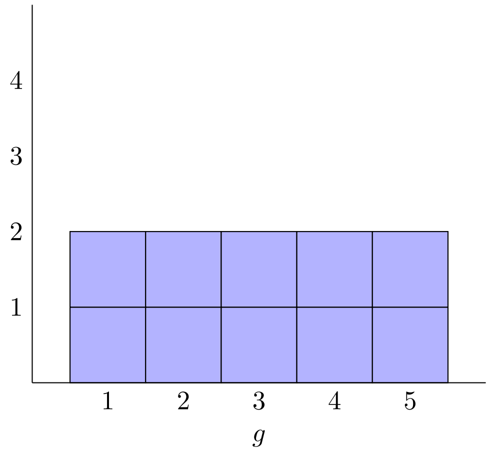
For m a positive integer, let $[m] = \{1, \dots, m\}$. We will think of $[m]$ as a set of locations, each of which stores a pile of dirt, and a function $f\colon [m] \to [0, \infty)$ as specifying the amount of dirt at each location; that is, $f(x) = c$ means c units of dirt are stored at location x.

Example 4.14. Let $f\colon [5] \to [0, \infty)$ be given by

$$f(1) = f(5) = 1,\ f(2) = f(4) = 2,\ f(3) = 4.$$

Let $g\colon [5] \to [0, \infty)$ be given by $g(x) = 2$ for all $x \in [5]$. We interpret f and g as piles of dirt, as illustrated below.





Now suppose we are given functions $f, g\colon [m] \to [0, \infty)$ with $\sum_{i=1}^{m} f(i) = \sum_{i=1}^{m} g(i)$. Informally, the main question underlying the definition of Wasserstein distance is this: How much work do we need to do in order to transform the arrangement of dirt f into the arrangement of dirt g? Here, we assume that moving one unit of dirt one step requires one unit of work. (This notion of work neglects any cost associated to the vertical movement of dirt, e.g., if dirt is piled high in one location and needs to be lowered in order to move it.)

Example 4.15. Letting $f, g\colon [5] \to [0,\infty)$ be as in Example 4.14, we can transform f into g using four units of work. There are multiple ways to do this. For instance, we can move one unit of dirt from 3 to 1, and one unit from 3 to 5. Let us call this *plan 1.* Alternatively, we can move one unit from 3 to 2, one unit from 2 to 1, one unit from 3 to 4, and one unit from 4 to 5. We call this *plan 2.* It can be checked that there is no way to transform f into g with fewer than four units of work.

To formalize the notion of transforming f into g, we introduce the following definition:

Definition 4.16. A *transport plan* from f to g is a function $\Gamma\colon [m]\times[m] \to [0,\infty)$ such that for all $i, j \in [m]$,

$$f(x) = \sum_{i=1}^{m} \Gamma(x,i) \qquad g(x) = \sum_{i=1}^{m} \Gamma(i,x).$$

The interpretation of Definition 4.16 is as follows:

- $\Gamma(x,y)$ is the amount of dirt we transport from x to y.
- The first equality tells us that the total amount of dirt moved **from** location x is $f(x)$.
- The second equality says that the total amount of dirt moved **to** location x is $g(x)$.

Example 4.17. Plans 1 and 2 from Example 4.14 can be formalized as transportation plans Γ_1 and Γ_2, respectively, where $\Gamma_1(x,y)$ is (x,y)-entry of the following matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

and $\Gamma_2(x,y)$ is (x,y)-entry of the following matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The summing condition in Definition 4.16 can be interpreted as saying that summing the rows and columns, respectively, of these transport matrices should give f and g. That is, the sum of the elements of row i is $f(i)$, and the sum of the elements of column i is $g(i)$[4].

Let $T(f,g)$ denote the set of all transport plans from f to g.

Definition 4.18. For $\Gamma \in T(f,g)$, let

$$\text{work}(\Gamma) = \sum_{i=1}^{m}\sum_{j=1}^{m} \Gamma(i,j)|i-j|.$$

[4]Using this observation, one can treat the search for a valid transport plan as a sudoku-like game where one has to fill in numbers in a matrix with restrictions given by the sums of the rows and columns.

The Wasserstein distance[5] between f and g, denoted $W(f,g)$, is given by

$$W(f,g) = \inf_{\Gamma \in T(f,g)} \text{work}(\Gamma).$$

We interpret the term $\Gamma(i,j)|i-j|$ in the definition of work(Γ) as the amount of work required by transport plan Γ to move dirt from position i to position j.

Example 4.19. One readily checks that for Γ_1 and Γ_2 the transport plans of Example 4.17, we have work(Γ_1) = work$(\Gamma_2) = 4$.

We leave the proof of the following as an exercise:

Proposition 4.20. *For any fixed positive integer m, and $c > 0$, Definition 4.18 defines a metric on the set of functions of the form $f\colon [m] \to [0,\infty)$ with $\sum_{i=1}^m f(i) = c$.*

Remark 4.21. The Wasserstein distance can be be computed in practice via a standard optimization technique called *linear programming.* This can be expensive for very large problem instances; to lower the computational cost, one can compute an approximation of the Wasserstein distance.

Example 4.22. It can be illuminating to compare the Wasserstein distance to the Euclidean and Manhattan distances. First note that a function $f\colon [m] \to [0,\infty)$ can be naturally identified with a vector v_f in $\mathbb{R}^m$. For example, the functions f and g of Example 4.14 are identified with the vectors $v_f = (1,2,4,2,1)$ and $v_g = (2,2,2,2,2)$ in $\mathbb{R}^5$, respectively. We have $d_2(v_f, v_g) = \sqrt{6}$ and $d_1(v_f, v_g) = 4$.

A more instructive example is the following: define $f : [m] \to [0,\infty)$ by

$$f(x) = \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if } x = m \\ 0 & \text{otherwise.} \end{cases}.$$

Then $d_2(v_f, v_g) = \sqrt{2}$, an $d_1(v_f, v_g) = 2$. But $W(f,g) = m-1$. This highlights an important point: The Wasserstein distance is sensitive to the distance between points of $[m]$, whereas d_2 and d_1 are insensitive to this.

Remark 4.23. The definition of the Wasserstein distance given in Definition 4.18 can be generalized in several directions. For example, the set $[m]$ can be replaced with $\mathbb{R}^m$; one then replaces the sum in the definition of transport plan with an integral. This yields a continuous version of the Wasserstein distance, suitable for comparing probability density functions.

4.4 Exercises on Metric spaces

Exercise 4.1 (Stability of metrics)**.** Suppose that (S,d) is a metric space and $x, x', y, y' \in S$. Use the triangle inequality to show that if $d(x,x') \leq \epsilon$ and $d(y,y') \leq \epsilon$, then $|d(x,y) - d(x',y')| \leq 2\epsilon$.

Exercise 4.2. Prove Proposition 4.9.

Exercise 4.3. Suppose we define d in the same way as in Proposition 4.9, but G is disconnected. Why is d not a metric on G?

Exercise 4.4. Show that Example 4.8 indeed defines a metric.

[5]It is common to define a family of Wasserstein distances, with each $p \in [1,\infty) \cup \{\infty\}$ giving a metric W_p. In that setting, the distance we define is W_1.

Exercise 4.5. Which of the following are metric spaces?

- $(\mathbb{N}, d)$, where $d(a, b) = a - b$.
- $(\{*\}, d)$, where $d(*, *) = 0$.
- $(\mathbb{R}, d)$, where $d(x, y) = (x - y)^2$.

Exercise 4.6. Let $S = \{(0, 0), (1, 0), (0, 1), (1, 1), (4, 0)\} \subseteq \mathbb{R}^2$, and let d be the Euclidean metric on $\mathbb{R}^2$. Draw $N(S)_\epsilon$ for $\epsilon = 0, 1, 2, 3$. How many connected components does $N(S)_\epsilon$ have for each of these choices of ϵ?

Exercise 4.7. Let $M = (S, d)$ be a metric space with at least two points. For $c \in \mathbb{R}$, define $cd \colon S \times S \to [0, \infty)$ by $(cd)(x, y) = c(d(x, y))$ for all $x, y \in S$. That is, every distance is multiplied by c. For which c is (S, cd) a metric?

Exercise 4.8. Let $M = (S, d)$ be a metric space. For $a, b \in M$, a point $m \in M$ is a *midpoint* of a and b if $d(a, m) = d(m, b) = \frac{1}{2}d(a, b)$.

(i) Find an M such that for all $a \neq b \in M$, a and b do not have a midpoint.

(ii) Find an M such that for all $a, b \in M$, a and b have exactly one midpoint.

(iii) Find an M such that for some $a, b \in M$, a and b have more than one midpoint.

Exercise 4.9.[*] Recall the definition of an isomorphism between graphs (Definition 3.32). How would you define an isomorphism between metric spaces?

Exercise 4.10. Prove that d_∞ is a metric on $\mathbb{R}^n$. [Hint: To show the triangle inequality, use the triangle inequality for the ordinary Euclidean distance on $\mathbb{R}$.]

Exercise 4.11. Check that the edit distance (Definition 4.11) is a metric. [NOTE: You can do this directly, or by proving Exercise 4.2, which is a generalization.]

Exercise 4.12. For each of the following functions $d : \mathbb{R} \times \mathbb{R} \to [0, \infty)$, say whether d is a metric. Justify your answer.

a. $d(x, y) = |x| + |y|$.

b. $d(x, y) = \frac{|x|}{1+|y|}$.

c. $d(x, y) = |x - y|^3$.

Exercise 4.13. For each of the following functions $d : \mathbb{R}^2 \times \mathbb{R}^2 \to [0, \infty)$, say whether d is a metric. Justify your answer.

a. $d(x, y) = |x_2 - y_2|$,

b. $d(x, y) = |x_1 - y_1|^3 + |x_2 - y_2|^3$,

c. $d(x, y) = d_1(x, y)d_2(x, y)$,

d. $d(x, y) = 2|x_1 - y_1| + 3|x_2 - y_2|$.

Exercise 4.14. Give an example of two different metrics on $\mathbb{R}^2$ whose restrictions to the x-axis are equal.

Exercise 4.15. Compute the edit distance between the following pairs of words:

a. $x = AAAA$, $y = AA$

b. $x = AAAA$, $y = AAAT$

c. $x = GTAA$, $y = TAAG$.

d. $x = GGGG$, $y = TT$.

Exercise 4.16. Give three *different* examples of a metric on $\mathbb{R}$. HINT: d_2, d_1, and d_∞ are all equal on $\mathbb{R}$.

Exercise 4.17. Give an example of a function $d : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ which satisfies the second and third properties of a metric, but not the first.

Exercise 4.18. Define $f, g : \{1, 2, 3\} \to [0, \infty)$ by

$$f(1) = 1,\ f(2) = 1,\ f(3) = 1,$$

$$g(1) = 2,\ g(2) = 0,\ g(3) = 1,$$

What is the Wasserstein distance $W(f, g)$?

Exercise 4.19. Define $f, g : \{1, 2, 3\} \to [0, \infty)$ by

$$f(1) = 1,\ f(2) = 1,\ f(3) = 1,$$

$$g(1) = 3,\ g(2) = 0,\ g(3) = 0,$$

What is the Wasserstein distance $W(f, g)$?

Exercise 4.20.* For a fixed positive integer m, let S denote the set of functions

$$f : [m] \to [0, \infty)$$

such that $\sum_{i=1}^m f(i) = 1$. Let $f, g, h \in S$. Show that if $\Gamma_1 \in T(f, g)$ and $\Gamma_2 \in T(g, h)$, then $\Gamma_3 \in T(f, h)$, where

$$\Gamma_3(i, j) = \sum_{\substack{k \in [m] \\ g(k) \neq 0}} \frac{\Gamma_1(i, k)\Gamma_2(k, j)}{g(k)}.$$

Exercise 4.21.* Using Exercise 4.20*, prove that the Wasserstein distance is a metric on S.

5 Clustering

5.1 Introductory Remarks

Clustering is one of the oldest and most important problems in data science. We begin a rough, informal description of the problem:

> **The Clustering Problem: Given a finite metric space X find a partition P of X such that points of X in the same element of the partition are close, while points in different elements of the partition are far apart. We call elements of P *clusters.***

We will see shortly that this description of clustering is problematic, in some ways. Yet it aligns well with some approaches to clustering, and is a good place to begin our study of this topic.

Example 5.1. Intuitively, there are two clusters in Figure 20, colored red and blue respectively.

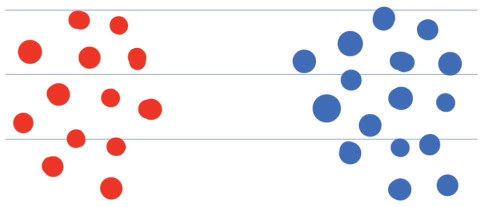


Figure 20: Two clusters

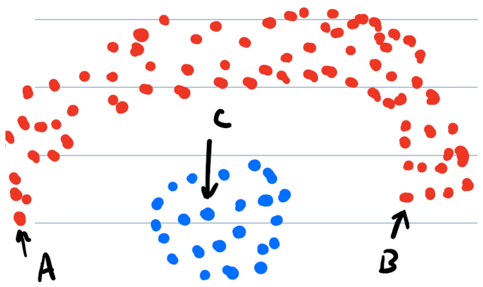


Figure 21: Two clusters, red and blue, with points in the red one that are closer to the points in the blue one than between each other.

Example 5.2. Again, intuition suggests that there are two clusters in Figure 21, but the points A and B are further from each other than they are from C, even though it seems A and B should be clustered together, while C should be in a different cluster.

Example 5.3. In Figure 22, the cluster structure is *hierarchical*; it is ambiguous as to whether there are two clusters or five clusters. In practice, one often wishes to perform clustering in a way that is sensitive to such structure.

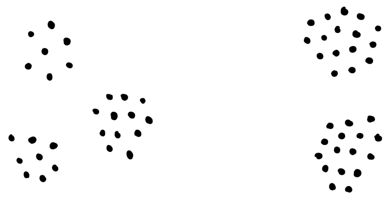


Figure 22: Hierarchical cluster structure: Are there two clusters of five clusters?

Example 5.2 and Examples 5.2 and 5.3 point to deficits with the informal definition of clustering given above. Indeed, devising a good definition of clustering is problematic, and there is no universally agreed-upon definition. Instead, there are many proposals for different definitions, leading to many different approaches to clustering with different strengths and weaknesses. In this section, we will discuss a few of these many approaches.

It is important to remember that while the above examples illustrate the clustering problem for data in $\mathbb{R}^2$, we often are most interested in clustering data in $\mathbb{R}^n$ for n very large, or metric data that does not embed in $\mathbb{R}^n$ at all. Such data cannot be visualized directly, which creates a need for methods that allow us to study the shape of the data without seeing that shape. Clustering is one of the main approaches to this.

5.2 Motivating Examples

Previous sections of the notes already hint at some real-world settings where we might wish to cluster data:

- Pathogen subtyping: Suppose we have a set S of genomes of a virus, like HIV or influenza. We can regard S as a metric space using the edit distance (Definition 4.11) or a variant thereof. Clustering S allows us to understand the subtype structure of the virus population. This biologists to approach the clinical treatment of the virus via *divide-and-conqueror approach*, i.e., by developing separate treatments for different virus subtypes.
- Recall that we can model a social network with symmetric connections (like Facebook or Linked-in) as an undirected graph $G = (V, E)$ where V is the set of users and edges are connections between users. Regarding V as a metric space with the shortest path metric, we can cluster V in order to find natural subgroups of users. One natural application is to advertising; the cluster structure of the network can guide how ads are matched to users. For example, if the network has three distinct clusters, it might make sense for an advertiser to create three separate ad campaigns for the same product, each targeting a different cluster.
- A third class of examples where clustering is useful comes from oncology. Like the first example, this also concerns genetics. The kind of data considered here is *gene expression data.* For example, we may have a data set $X = \{x^1, \dots, x^{300}\} \subseteq \mathbb{R}^{24000}$, arising as follows:
  - There are 300 human subjects with breast cancer.
  - Each x^i represents a tissue sample from one patient.
  - We consider the level of expression of 24000 genes in each tissue sample.
  - Writing $x^i = (x^i_1, \dots, x^i_{24000})$, x^i_j is the level of expression of gene j in tissue sample i.

  Gene expression levels are measured using *RNA sequencing*, a standard and highly successful technology. In principle, clusters in X correspond to cancer subtypes. In practice, such clustering is a messy business. When working with such high-dimensional data, the noise in the data is much stronger than the signal relevant cancer, so one often first processes the data so as to only retain a few dozen genes most relevant to cancer. There are many ways of doing such preprocessing, and no universally accepted standard.

5.3 Formal specification of the input and output of the clustering problem

The input of the clustering problem is:

(1) A finite set of points $X = \{x_1, \dots, x_n\} \subseteq \mathbb{R}^m$, or more generally, a finite metric space $X = (X, d)$.

(2) Some clustering methods (e.g., k-means clustering, discussed below) also require the number of clusters to be given as input.

The output of the clustering problem is either

(1) A partition of X, or of a subset of X, OR

(2) A hierarchical partition (which we will define later).

5.4 k-means clustering

The first clustering method we will consider is k-means clustering, arguably the most common clustering method. This takes as input finite set $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$, and a number of clusters $k \in \{1, \dots, n\}$. It outputs a partition of X of size k. For any $Y \subset \mathbb{R}^n$ finite, we let $\overline{Y}$ denote the mean of Y, i.e., $\overline{Y} = \dfrac{1}{|Y|} \sum_{y \in Y} y$.

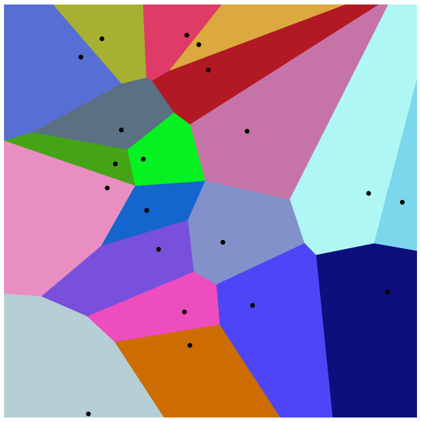


Figure 23: The Voronoi diagram of points in $\mathbb{R}^2$.

Definition 5.4. For a partition $C = \{C_1, \ldots, C_k\}$ of X, let

$$\text{Cost}(C) = \sum_{i=1}^{k} \sum_{x \in C_i} d_2(x, \mu_i)^2,$$

where

$$\mu_i = \overline{C_i}.$$

An *(optimal) k-means clustering* of X is a partition $C^* = \{C_1^*, \ldots, C_k^*\}$ which minimizes $\text{Cost}(C)$.

Note that

$$\text{Cost}(C) = \sum_{x \in X} d_2(x, \mu_x)^2,$$

where μ_x is the mean of the cluster containing x, i.e., $\mu_x = \mu_i$, where $x \in C_i$. Thus, C^* minimizes the sum of squared distances from each point to the mean of the cluster the point belongs to.

We note the following simple property of C^*:

Proposition 5.5. *For $x \in C_i^*$, $d_2(x, \mu_i) \leq d_2(x, \mu_j)$ for all j.*

Proof. If to the contrary, $d_2(x, \mu_i) > d_2(x, \mu_j)$, then by reassigning x to C_j^* we obtain a new partition with smaller cost than C^*, contradicting that C^* is the partition of minimum cost. □

To interpret the above proposition geometrically, it is useful to introduce the formalism of *Voronoi cells*, a fundamental notion in computer science and mathematics.

Definition 5.6 (Voronoi Diagram)**.** For $U \subset \mathbb{R}^n$ finite and $u \in U$, the *Voronoi cell* of u is the set

$$V(u) := \{y \in \mathbb{R}^n \mid \|y - u\| \leq \|y - v\| \text{ for all } v \in U\}.$$

The Voronoi cells of points in U determine a decomposition of $\mathbb{R}^n$ which is called the *Voronoi diagram of U*. See Section 5.4 for an illustration.

In the following, we use Definition 5.6 in the case $U = \{\mu_1, \ldots, \mu_k\}$.

Proposition 5.7. *For each i, $C_i^* \subset V(\mu_i)$.*

Proof. This follows immediately from Proposition 5.5 and the definition of a Voronoi cell. □

Corollary 5.8. *If C and C' are clusters in a k-means clustering of a set $X \subseteq \mathbb{R}^2$, then there is a straight line separating C from C_2.*

This result is illustrated in Figure 24.

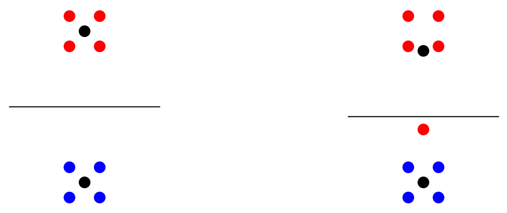


Figure 24: Two different clusterings of eight points, colored red and blue. The means of each cluster are shown in black. In (a), we have an optimal 2-means clustering, and each cluster lies within the corresponding Voronoi cell. In (b), the clusters do not respect the Voronoi regions, so this cannot be an optimal 2-means clustering.

5.4.1 Lloyd's Algorithm

It turns out that computing C^* can be computationally very expensive. So in practice, we usually compute compute a partition C via a computationally fast approach which tries to minimize $\mathrm{Cost}(C)$, but is not guaranteed to find C^*. We now present a classical algorithm for this called *Lloyd's algorithm* in pseudocode[6]

Algorithm 1 Lloyd's Algorithm

```
1: Choose an initial set of cluster centers $\{\mu_1, \cdots, \mu_k\} \subset X$, e.g., at random.
2: repeat
3:   for $i = 1$ to $k$ do
4:     $C_i \leftarrow X \cap \mathrm{Vor}(\mu_i)$
5:   for $i = 1$ to $k$ do
6:     $\mu_i \leftarrow \overline{C_i}$
7: until No $\mu_i$ changes between successive iteration of the outer loop.
8: return  $C = \{C_1, \ldots, C_k\}$
```

Remark 5.9. The version of Lloyd's algorithm written above tacitly assumes that $x \in X$ is never equidistant to two or more cluster centers. If that does happen, then in lines 4-5, Lloyd's algorithm must assign x only to C_i, where i the smallest value such that $x \in \mathrm{Vor}(\mu_i)$.

Example 5.10. Consider the metric space $X = (X, d_X)$, where $X = \{(0,0), (1,0), (0,1), (-2,-2), (-2,-3)\}$, and $d_X = d_1$, i.e., d_X is the restriction of the Manhattan distance. Assume $k = 2$ and set the initial values of $\mu_1 = (0,0)$, $\mu_2 = (2,2)$. Running Lloyd's the algorithm yields

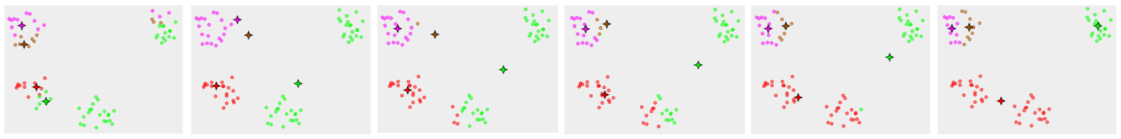
$$C = \{(0,0), (1,0), (0,1)\}, \{(-2,-2), (-2,-3)\}\}.$$

In fact, one gets the same partition C if $\mu_1 = (1,0)$, $\mu_2 = (0,0)$. (You can check this yourself by running Lloyd's algorithm for both initial conditions.)

Example 5.11. This animation from Wikipedia illustrates Lloyd's algorithm on a medium-sized data set in $\mathbb{R}^2$:
`https://en.wikipedia.org/wiki/K-means_clustering#/media/File:K-means_convergence.gif`

[6]As this is our first instance of pseudocode in this notes, we remind the reader that pseudocode is a specification of an algorithm that is optimized for a reader to understand the essential ideas of the algorithm. It is not code that can be directly run in a programming language, and certain low level details needed for an implementation may be omitted. For example, in Lloyd's algorithm below, some detail about how to implement the outer loop is omitted. In practice, this would usually be implemented with a while loop, and an additional variables would be introduced to check whether any μ_i has changed.

Example 5.12. The following figure, also from Wikipedia, illustrates that Lloyd's algorithm does not always find the partition C that minimizes $\text{Cost}(C)$:



Here $k = 4$. Here, one sees visually that the data has four clear clusters, and the partition minimizing the cost is the one that divides the data into these clusters. Yet with the initial choice of the μ_i shown, Lloyd's algorithm groups two of the clusters together (red) and splits one of the two clusters (pink and brown).

5.4.2 Understanding Lloyd's Algorithm

We have just seen in Example 5.12 that K-means does not necessarily find an optimal k-means clustering. The reader may wonder, then, why we are justified in using Lloyd's algorithm at all. The main conceptual justification for Lloyd's algorithm is that until the algorithm converges, *each step of Lloyd's algorithm decreases* $\text{Cost}(C)$.

To understand this, it is helpful to rewrite the cost function to make its dependence on $U = \{\mu_1, \ldots, \mu_k\}$ explicit:

$$\text{Cost}(C, U) = \sum_{i=1}^{k} \sum_{x \in C_i} d_2(x, \mu_i)^2,$$

Note that the above expression is well defined for any partition $C = \{C_1, \ldots, C_k\}$ and any values of $U \subset \mathbb{R}^n$, even when $\mu_i \neq \overline{C}_i$.

Assume for simplicity that when we run Lloyd's algorithm, we do not encounter the case where $x \in X$ is equidistant from two cluster centers μ_i and μ_j. When we run the main loop of the algorithm, we alternate between two steps: The first step updates C while leaving U fixed by setting $C_i \leftarrow \text{Vor}(\mu_i) \cap X$. If this step changes any C_i, then it strictly lowers the value of $\text{Cost}(C, U)$, because if x is moved from X_j to X_i, then it means that $d_2(x, \mu_i) < d_2(x, \mu_j)$.

The second step updates U while leaving C fixed, by setting $\mu_i \leftarrow \bar{C}_i$. If this step changes any μ_i then it also strictly lowers the value of $\text{Cost}(C, U)$, this is because of the following proposition:

Proposition 5.13. *For any finite set $Y \subset \mathbb{R}^n$, the unique minimizer of the function*

$$f(z) = \sum_{y \in Y} d_2(y, z)^2$$

is $z = \bar{Y}$.

Sketch of Proof. f is a convex function so, according to a standard result from multivariable calculus, has a unique minimizer, namely the point where the gradient of f is 0. An easy calculation shows that this point is $\overline{Y}$. □

5.4.3 Practical Aspects of k-Means Clustering

Besides being useful for understanding the mathematics underlying k-means clustering, the cost function $\text{Cost}(C)$ is practically useful for clustering, in at least two ways.

First, since the output of Lloyd's algorithm is sensitive to the choice of initial cluster centers, we can run Lloyd's algorithm for a fixed choice of X and k many times, to obtain many partitions of X. Moreover, we can compute the cost of each partition and return the partition with the smallest cost. Doing this alleviates to some extent Lloyd's algorithm sensitivity to initial conditions.

Second, one of the main disadvantages of k-means clustering is that it requires k to chosen in advance. To address this, a common approach, called the *elbow method*, is to perform the clustering for a range of choices of k, and then graph the cost of the partition found as a function of k. Typically, this cost decreases as a function of k, so one looks for *elbows* in this plot, i.e., values of k where the slope of the graph notably decreases. Such values are sometimes considered good candidates for the choice of k. However, this approach has significant limitations: often such plots lack clear elbows, and there is no formal way distinguish elbows from non-elbows.

5.5 Single linkage clustering

Recall the definition of $N(X)_r$ from Definition 4.10. Recalling the correspondence between partitions and equivalence relations from Remark 2.55, for P a partition of a set X and $x, y \in X$, we will write "$x \sim y$ *in* P" when x and y belong to the same element of P.

Definition 5.14. Let (X, d) be a finite metric space. For $r \geq 0$, we define the *single linkage clustering* at r as the partition $SL(X)_r$ of X where $x \sim y$ in P_r if $x, y \in X$ belong to the same component of $N(X)_r$. Equivalently,

$$SL(X)_r = \{X' \subseteq X \mid X' \text{ is the vertex set of a component of } N(X)_r\}.$$

Example 5.15. Let $X = \{1, 3, 4\}$ be equipped with the standard metric. The graphs $N(X)_r$ are illustrated in Figure 25. By taking connected components, we get the clusterings $\{\{1\}, \{3\}, \{4\}\}$ for $r < 1$, $\{\{1\}, \{3, 4\}\}$ for $1 \leq r < 2$, and $\{\{1, 3, 4\}\} = \{X\}$ for $r \geq 2$.

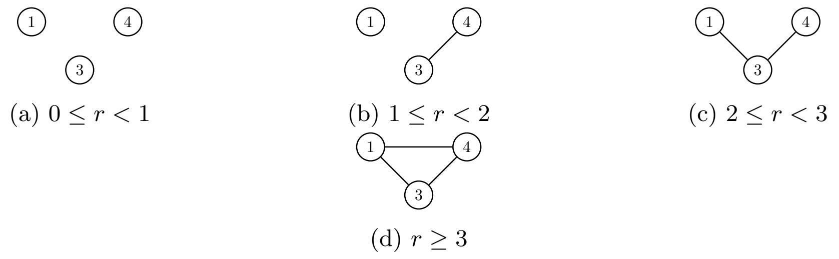

(a) $0 \leq r < 1$ (b) $1 \leq r < 2$ (c) $2 \leq r < 3$ (d) $r \geq 3$

Figure 25: The graphs $N(X)_r$ for different r for X as in Example 5.15.

5.5.1 A Geometric Viewpoint on Single Linkage Clustering

In the case that $X \subset \mathbb{R}^n$, there is a pleasant geometric interpretation of single linkage clustering that is the starting point for some key concepts in TDA.

Definition 5.16. A *filtration* is a collection $F = \{F_r\}_{r \in [0,\infty)}$ of objects (e.g., graphs or subsets of $\mathbb{R}^n$), such that $F_r \subseteq F_s$ for all $r \leq s$.

We already met this concept for graphs, even if we did not mention it explicitly: for every (finite) metric space (X, d), the collections of graphs $\{N(X)_r\}_{r \in [0,\infty)}$ is a filtration (recall the definition of $N(X)_r$ from Definition 4.10). Indeed, it is a collection of graphs on the same set of vertices and with increasingly more edges.

Another example that is very relevant for TDA is the following:

Example 5.17. Let X be a finite subset of $\mathbb{R}^n$. For $r \geq 0$, define the union-of-balls filtration $U(X)$ by

$$U(X)_r = \{y \in \mathbb{R}^n \mid d_2(x, y) \leq r \text{ for some } x \in X\}.$$

Example 5.18. Consider the points set $X \subset \mathbb{R}^2$ depicted on the left in Figure 26. The figure shows $U(X)_r$ for several choices of r. In the context of clustering, we particularly interested in the number of *components* (i.e., distinct pieces) of $U(X)$. (We have not formally defined components in this setting, but this can be done.). $U(X)_r$ consists 11 points, i.e. 11 connected components. In the second step, some of the balls are touching, and $U(X)_r$ has 4 connected components, which becomes three on the next step. We can see that the number of connected components of $U(X)_r$ continues to decrease as r increases, until one has a single connected component (at the second-to-last step). Observe that for a relatively long range of parameters, $U(X)_r$ has three connected components, reflecting the presence of three clusters in X.

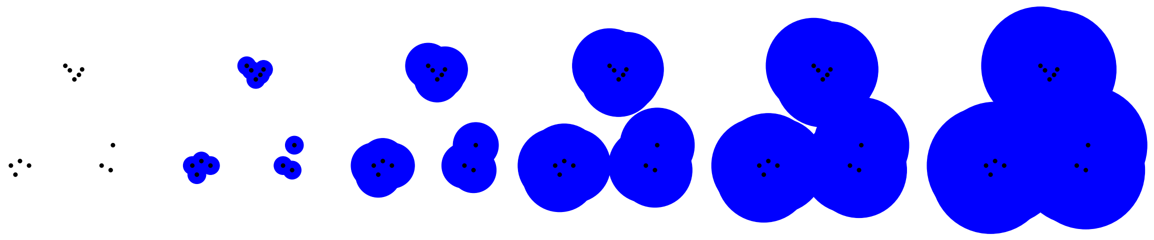


Figure 26: Union of balls of points in $\mathbb{R}^2$

As Example 5.18 illustrates, for suitable data set $X \subset \mathbb{R}^n$ and parameter r, the components of $U(X)_r$ correspond to clusters in X. This suggests a clustering method: Fixing some $r \geq 0$, two points in X belong to the same cluster if they are contained in the same connected component of $U(X)_r$. In fact, this is a special case of the single linkage clustering introduced above: If $X \subset \mathbb{R}^n$ and $d_X = d_2$, then $x, y \in X$ in $SL(X)_r$ if and only if they belong to the same connected component of $U(X)_{\frac{r}{2}}$. But we emphasize that the union-of-balls filtration is defined specifically for a finite subset of $\mathbb{R}^n$, while we have defined single linkage more generally for finite metric spaces.

5.6 Hierarchical clustering

Consider the single linkage clustering built from Figure 26. For different radii r, we have a different number of clusters, so one may ask how do we correctly choose r. As we have seen in Example 5.3, it is also possible that there is no "correct" choice of r, but rather that the useful information is how the clusters evolve for varying radii. To formalize this intuition, we now define hierarchical clustering (recall the definition of total order Definition 2.38):

Definition 5.19. A *hierarchical partition* (or hierarchical clustering) of a set X, denoted by $P = \{P_r\}_{r \in T}$, for a total order T, consists of

(1) A family $\{X_r\}_{r \in T}$ of subsets of X such that $X_r \subseteq X_s$ for all $r \leq s \in T$.

(2) A partition P_r of each X_r such that if $x \sim y$ in P_r, then $x \sim y$ in P_s for all $r \leq s \in T$.

Intuitively, property (1) says that as r increases we are clustering more and more points together, while property (2) says that if x and y are clustered together in some P_r, they stay clustered together for all larger values of r.

Note: often, but not always, we have $X_r = X$ for all r, as in the following key example.

Example 5.20. For a finite metric space X, we obtain a hierarchical partition $SL(X) = \{SL(X)_r\}_{r \in [0,\infty)}$, where $SL(X)_r$ is defined as in Definition 5.14. Here $X_r = X$ for all r, so property (1) of Definition 5.19 is trivially satisfied. Property (2) holds essentially because as $N(X)_r \subset N(X)_s$ for $r \leq s$, so connected components of $N(X)_r$ may merge in $N(X)_s$, but never split apart.

5.6.1 Dendrograms

A nice, pictorial way to describe a hierarchical clustering is via dendrogram. Informally, to construct the dendrogram $D(P)$ of a hierarchical clustering $P = \{P_r\}_{r\in T}$ with $T = \{r_1, \ldots, r_m\}$ a finite totally ordered set, we

1. place a vertex at height r for each set $v \in P_r$, and
2. draw an edge between $v \in P_{r_i}$ and $w \in P_{r_{i+1}}$ if $v \subseteq w$.

Example 5.21. Consider the set $X = \{A, B, C, D, E, F, G\}$ and the totally ordered set $T = \{0, 1, 2, 3, 4, 5, 6\}$ with the collection of partitions:

$$\begin{aligned}
P_0 &= \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{G\}\} \\
P_1 &= \{\{A, B\}, \{C\}, \{D\}, \{E, F\}, \{G\}\} \\
P_2 &= \{\{A, B\}, \{C\}, \{D, E, F\}, \{G\}\} \\
P_3 = P_4 = P_5 &= \{\{A, B, C\}, \{D, E, F, G\}\} \\
P_6 &= \{A, B, C, D, E, F, G\}
\end{aligned}$$

Its dendrogram is depicted in Figure 27.

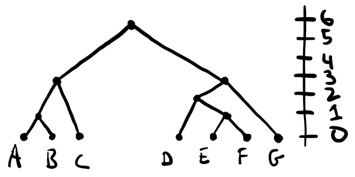


Figure 27: Drawing of the dendrogram from the hierarchical partition of Example 5.21.

Here is a more formal definition of a dendrogram:

Definition 5.22. Let $P = \{P_r\}_{r\in T}$ be a collection of partitions of a finite set X, where $T = \{r_1, \ldots, r_m\}$ is a finite totally ordered set. A *dendrogram* of P consists of a graph $D(P) = (V, E)$, together with a *height* function $h\colon V \to T$, where

$$\begin{aligned}
V &= \{(v, r) \mid r \in T,\, v \in P_r\} \\
E &= \{\{(v, r_i), (w, r_{i+1})\} \mid v \subseteq w\}, \\
h(v, r) &= r.
\end{aligned}$$

We omit the proof of the following proposition:

Proposition 5.23. *For any hierarchical partition P indexed by a finite totally ordered set, $D(P)$ is a forest.*

5.6.2 Single Linkage Dendrograms

We wish to construct a dendrogram from a single linkage clustering $SL(X)$. But we have only defined dendrograms for hierarchical partitions indexed by finite, totally ordered sets. We deal with this as follows:

Definition 5.24. For X a finite metric, define its *critical set* $Cr(X) \subset [0, \infty)$ to be the set of values r such that $SL(X)_q \neq SL(X)_r$ for any $q \in [0, \infty)$ with $q < r$.

Informally, then, $Cr(X)$ is the set of values of r where $SL(X)_r$ changes.

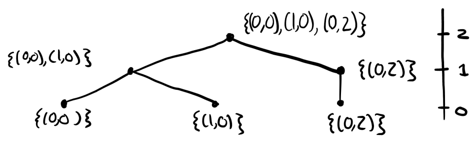
Definition 5.25. Given a hierarchical partition $P = \{P_r\}_{r \in T}$ and $T' \subset T$, the *restriction of P to T'* is the hierarchical partition $P^{T'} := \{P_r\}_{r \in T'}$.

Definition 5.26. For X a finite metric space, we define $D(SL(X))$ to be $D(SL(X)^{Cr(X)})$. In other words, we define the dendrogram of $SL(X)$ to be the dendrogram of the restriction of $SL(X)$ to its set of critical values.

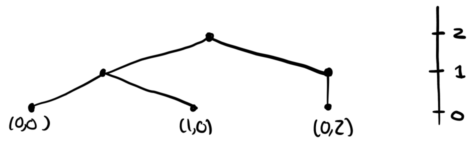
Example 5.27. Let $X = \{(0,0),(1,0),(0,2)\}$ with the Manhattan distance. Then

$$SL(X)_r = \begin{cases} \{\{(0,0)\},\{(1,0)\},\{(0,2)\}\} & \text{for } r \in [0,1) \\ \{\{(0,0),(1,0)\},\{(0,2)\}\} & \text{for } r \in [1,2) \\ \{\{(0,0),(1,0),(0,2)\}\} & \text{for } r \in [2,\infty) \end{cases}$$

$Cr(X) = \{0,1,2\}$. The dendrogram $D(SL(X))$ is shown in Figure 28a, with each vertex labeled by its corresponding set. In fact, the sets corresponding to each vertex of $D(SL(X))$ are determined by the sets at the leaves, by taking unions. Therefore, we can label only the leaves of the dendrogram by their associated sets, without losing any information. Moreover, when the set at a leaf is a singleton, we just write the unique element in the set. This yields a dendrogram with a much simpler labeling, as in Figure 28b.



(a) Dendrogram with all vertices labeled by their corresponding sets



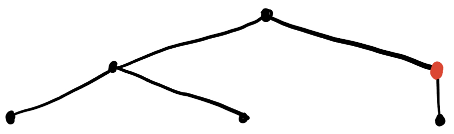
(b) The same dendrogram, using the simpler labeling convention

Figure 28: Plots of the dendrogram of Example 5.27

5.6.3 Simplifying a Dendrogram.

The dendrograms constructed above can be simplified without any loss of information by removing vertices of degree 2. To explain, we first explain the underlying idea for general graphs: suppose the graph $G = (V,E)$ is a forest, and a vertex $v \in V$ of degree 2 is incident to u and w. Since G is a forest, it has no cycles, so $\{u,w\} \notin E$. Thus, we can remove v from V, add $\{u,w\}$ to E, and remove both $\{u,v\}$ and $\{v,w\}$ from E.

Let us now apply this idea to a dendrogram $D(P)$: If v is a vertex of degree 2 in $D(P)$ incident to u and w with $h(u) < h(v) < h(w)$, we remove v and its incident edges, and insert the edge $\{u,w\}$. We will call the dendrogram obtained by performing removing degree 2 vertices in this way the *simplified dendrogram.* In the dendrogram of Example 5.27, it is possible to remove exactly one vertex. We will illustrate this in Figure 29.


(a) Vertex to be removed is marked in red

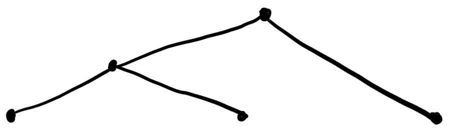
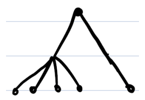

(b) Dendrogram after removal of red vertex

Figure 29: Removal of a vertex with degree 2 in the dendrogram of Example 5.27.

To plot a dendrogram in a way that avoids edge crossings and avoids taking unnecessary space, it is standard to draw the edges in a dendrogram as vertical and horizontal segments with right-angled elbows. This is illustrated in Figures 30b and 31 below:


(a)

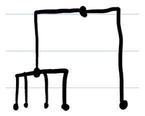

(b)

Figure 30: In (a) we have a simplified dendrogram, and in (b) we draw it with the right angles for the edges.

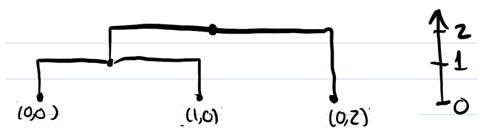


Figure 31: The simplified dendrogram of Example 5.27, plotted with the right angles

5.6.4 Getting a Clustering From a Dendrogram

Informally, to read the clustering P_r from a dendrogram $D(P)$, we can "cut" the dendrogram with a horizontal line at level r; each component below the cut line corresponds to a cluster (see Figure 32). This same cutting idea also makes sense for $r \in [0, \infty)$ and single linkage dendrograms.

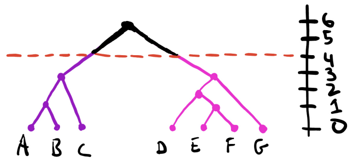


Figure 32: Drawing of the dendrogram of Example 5.21 with a horizontal cut at $r = 4$. The associated clustering is $\{\{A, B, C\}, \{D, E, F, G\}\}$

.

5.6.5 An algorithm to construct a single linkage dendrogram

We now give pseudocode for computing the simplified single linkage dendrogram of a finite metric space. For simplicity, we assume that all non-zero distances between pairs of points are distinct. To understand the algorithm, recall that each vertex in a dendrogram corresponds to a set; our pseudocode identifies each vertex with the corresponding set.

Remark 5.28. For an efficient implementation of this algorithm, we require a method of storing S such that we can quickly check whether the sets $w_x = w_y$ in line 11, and also quickly merge these sets in line 12. This is done via the *union-find data structure*, which we will discuss later.

5.7 Average linkage clustering

Single linkage is a pleasantly simple clustering method, but it is very unstable with respect to outliers. Thus, when dealing with noisy data, alternative approaches are usually preferred. Single linkage is a

Algorithm 2 Simplified Single Linkage Dendrogram Construction

```
Input: A finite matrix space $(X, d)$, represented, e.g., as a distance matrix
Output: Simplified single linkage dendrogram $D$ of $X$
 1: $Y \leftarrow$ list of all pairs $\{x, y\} \subset X$ with $x \neq y$
 2: Sort $Y$ according to distance between the elements of each pair, breaking the ties arbitrarily.
 3: Initialize $S$ as an empty list of sets ($S$ will store the clusters at the current height)
 4: Initialize $D$ as an empty forest
 5: for all $x \in X$ do
 6:    Add $\{x\}$ to $S$
 7:    Add vertex $\{x\}$ to $D$ at height 0
 8: for all $\{x, y\} \in Y$ in increasing order do
 9:    $w_x \leftarrow$ the set in $S$ containing $x$
10:    $w_y \leftarrow$ the set in $S$ containing $y$
11:    if $w_x \neq w_y$ then
12:       Remove $w_x$ and $w_y$ from $S$ and add $v := w_x \cup w_y$ to $S$.
13:       Add vertex $v$ to $D$ at height $d(x, y)$
14:       Add the edges $\{v, w_x\}$, $\{v, w_y\}$ to $D$
```

member of a family of clustering methods called the *agglomerative methods*, which begin with each point as a separate cluster, and then iteratively merge a pairs of clusters which minimizes some cost function. In single linkage, the cost is the minimum distance between two points in different clusters; this viewpoint on single linkage is made clear by Algorithm 2.

Here we consider a different agglomerative method called *average linkage*, which takes the cost to be the average distance between points in different clusters. Other agglomerative methods that we will not discuss here include *complete linkage* (cost is the maximum distance between points in different clusters) and *Ward clustering* (cost is the minimum increase in total within-cluster variance).

We give an algorithmic description of average linkage which directly outputs a (simplified) dendrogram:

Definition 5.29. Let S and T be two finite sets, and d a metric on $S \cup T$. The *average distance* between S and T is

$$\delta(S, T) := \frac{1}{|S|\,|T|} \sum_{s \in S} \sum_{t \in T} d(s, t)$$

Algorithm 3 Average Linkage Dendrogram Construction

```
Input: A finite matrix space $(X, d)$, represented, e.g., as a distance matrix
Output: Simplified average linkage dendrogram $D$ of $X$
 1: Initialize $S$ as an empty list of sets ($S$ will store the clusters at the current height)
 2: Initialize $D$ as an empty forest
 3: for all $x \in X$ do
 4:    Add $\{x\}$ to $S$
 5:    Add vertex $\{x\}$ to $D$ at height 0
 6: while $|S| > 1$ do
 7:    Find $w \neq w' \in S$ such that $\delta(w, w') \leq \delta(u, u')$ for all $u \neq u' \in S$
 8:    Remove $w$ and $w'$ from $S$, and add $v := w \cup w'$ to $S$.
 9:    Add vertex $v$ to $D$ at height $\delta(w, w')$
10:    Add the edges $\{v, w\}$, $\{v, w'\}$ to $D$
```

One can also specify the hierarchical partition $AL(X) = \{AL(X)_r\}_{r \in [0,\infty)}$ provided by average linkage; essentially the different values of the cluster set S arising as the algorithm runs are the different clusterings $AL(X)_r$. But rather than state this formally, we convey the idea with an example:

Example 5.30. Let us consider the average linkage clustering of Example 5.27, again using Manhattan distance. The initial clusters are

$$AL(X)_0 = \{\{(0,0)\}, \{(1,0)\}, \{(0,2)\}\}.$$

At height $r = 1$, the clusters merge, yielding the clustering

$$AL(X)_1 = \{\{(0,0), (1,0)\}, \{(0,2)\}\}.$$

At height 2.5, the clusters merge again, yielding

$$AL(X)_{2.5} = \{\{(0,0), (1,0)\}, \{(0,2)\}\}.$$

Between these height values, the clusters do not change, so in summary, we have

$$AL(X)_r = \begin{cases} \{\{(0,0)\}, \{(1,0)\}, \{(0,2)\}\} & \text{for } r \in [0,1) \\ \{\{(0,0), (1,0)\}, \{(0,2)\}\} & \text{for } r \in [1,2.5) \\ \{\{(0,0), (1,0), (0,2)\}\} & \text{for } r \in [2.5,\infty) \end{cases}$$

5.8 Density-based clustering

For X a finite metric space, the construction of the single linkage dendrogram $D(SL(X))$ we have seen in above can be summarized via the following pipeline:

$$\boxed{\text{Metric space}} \to \boxed{\text{Graph filtration}} \to \boxed{\text{Hierarchical partition}} \to \boxed{\text{Dendrogram}}$$

By a *graph filtration*, we mean a family of graphs $G = \{G_r\}_{r \in T}$ indexed by a totally ordered set such that $G_r \subset G_s$ whenever $r \leq s \in T$; the graph filtration constructed in single linkage clustering *neighborhood graph filtration* $N(X) = \{N(X)_r\}_{r \in [0,\infty)}$.

In fact, this same pipeline can be used to define other hierarchical clustering methods, by changing the construction of the graph filtration considered, while leaving the rest of the pipeline the same (up to minor details). We call such such clustering methods *topological methods.* We consider one simple but fundamental example here, which we call *density-based* hierarchical clustering. (Average linkage, however, is *not* a topological clustering method; it does not fit into this framework!)

To start, we fix a parameter $r > 0$. We then choose a *density function* $\gamma\colon X \to [0,\infty)$ on X. Informally, a density function is a function that has a low value on points in dense regions of the data, and a high value on points in sparse regions of the data. Density functions are a very old and well studied concept in data science, with multiple books devoted to their study. We give three standard examples of density functions:

1. $\gamma(x) = \frac{1}{\deg(x)}$, where $\deg(x)$ is the degree of x in $N(X)_r$ for some $r \geq 0$.
2. the *kernel density function*
$$\gamma(x) = \frac{1}{|X|} \sum_{y \in X} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{d(x,y)^2}{\sigma^2}},$$
where $\sigma > 0$ is a parameter and c is a constant.
3. the *k-nearest neighbors* density function, where $\gamma(x) = d_k(x)$, where $d_k(x)$ denotes the distance of x to its k-th nearest neighbor in X.

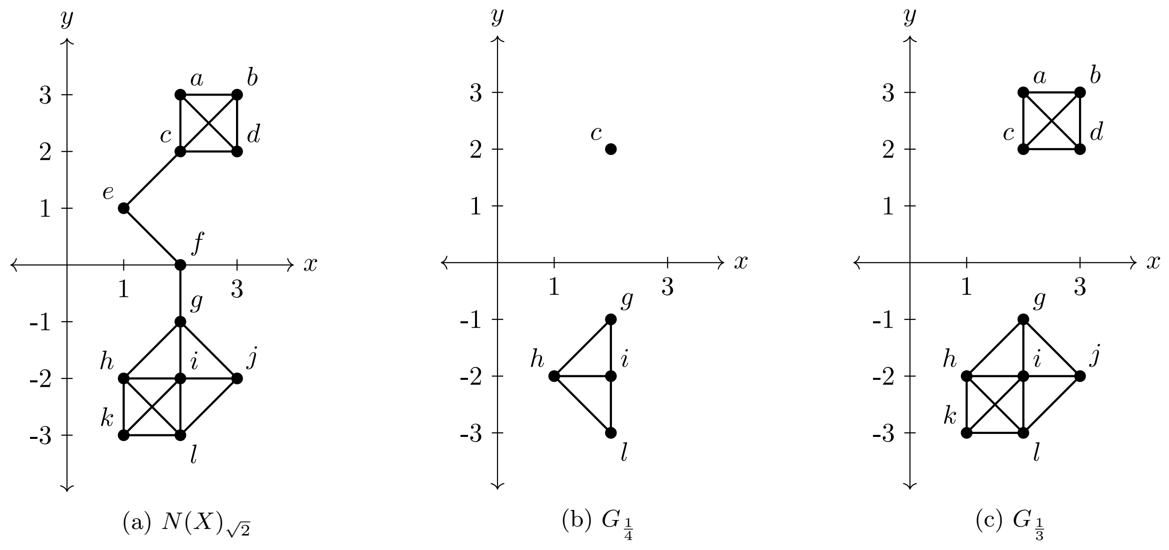


Figure 33: The neighborhood graph $N(X)_{\sqrt{2}}$ of a set X of 12 points in $\mathbb{R}^2$, along with the subgraphs $G_{\frac{1}{3}}$ and $G_{\frac{1}{4}}$, where the density function is $\frac{1}{\deg(x)}$.

Let $T = \operatorname{im}\gamma$, regarded as a totally ordered set. For $t \in T$, let

$$X_t = \{x \in X \mid x \in \gamma(x) \geq t\},$$

and let G_t be the subgraph of $N(X)_r$ induced by X_z (see Definition 3.9). (Concretely, that means G_t has vertex set X_t and all possible edges of $N(X)_r$ between these vertices.) Note that if $t \geq t' \in T$ then $G_t \subset G'_t$. Thus, the graphs G_t form a *graph* filtration $G := \{G_t\}_{t \in T}$.

We obtain a hierarchical partition $DB(X, r)$ from G exactly as we did for single linkage, i.e., by partitioning each X_t according to the path components of G_t. We then construct the dendrogram $D(DB(X, r))$ as in Section 5.6.1. In contrast to the single linkage dendrogram, where all leaves are at height 0, $D(DB(X, r))$ can have leaves at different heights.

Example 5.31. Figure 33a shows the neighborhood graph $N := N(X)_{\sqrt{2}}$ of 12 points in $\mathbb{R}^2$, where the Euclidean metric is used.

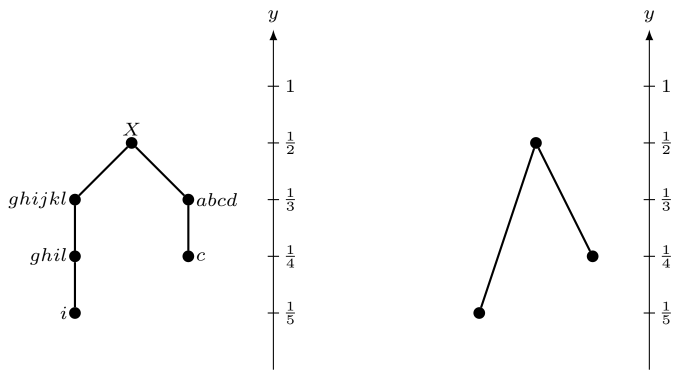
The degree of the nodes of N are as follows:

Degree	Vertices
5	i
4	c, g, h, l
3	a, b, d, j, k
2	e, f

Therefore, using the density function $\gamma(x) = \frac{1}{\deg(x)}$ where $\deg(x)$ is the degree of x in $N(X)_{\sqrt{2}}$, $G_{\frac{1}{5}}$ is the graph with vertex $\{i\}$ and no edges, $G_{\frac{1}{2}} = N$, and $G_{\frac{1}{4}}$, and $G_{\frac{1}{3}}$ are as shown in Figures 33b and 33c. Therefore,

$$DB(X, \sqrt{2})_t = \begin{cases} \{\{i\}\} & \text{for } t = \frac{1}{5}, \\ \{\{g, h, i, l\}, \{c\}\} & \text{for } t = \frac{1}{4}, \\ \{\{g, h, i, j, k, l\}, \{a, b, c, d\}\} & \text{for } t = \frac{1}{3}, \\ \{X\} & \text{for } t = \frac{1}{2}. \end{cases}$$

Thus, the unsimplified and simplified dendrograms of $DB(X, \sqrt{2})$ look as follows:



5.9 Barcodes of dendrograms

In this subsection, we consider dendrograms whose totally ordered set T is a finite subset of $\mathbb{R}$. We construct simplified representation of such a dendrogram called a *barcode*, which plays an essential role in topological data analysis.

Informally, a *multiset* is a set in which elements are allowed to appear more than once. For example $\{a, a, b\}$ is a multiset, where a appears twice and b appears once.

Definition 5.32. A *barcode* is a multiset of intervals in $\mathbb{R}$.

For example, $\{[0, 1), [0, 1), [2, 3)]\}$ is a barcode.

We will give an algorithmic construction of the barcode. To begin, for each component of the dendrogram, we add an extra edge from the top-most vertex of the component to a new vertex with height ∞. Each component of the dendrogram gets its own "vertex at infinity". We call the resulting dendrogram an *augmented dendrogram.* Figure 34 illustrates an augmented dendrogram with one component and its barcode Figure 34. The edge adjacent to the vertex at infinity is represented with an arrow. Note that the leaves of the dendrogram have multiple heights, as can arise from density-based clustering (but not from single or average linkage).

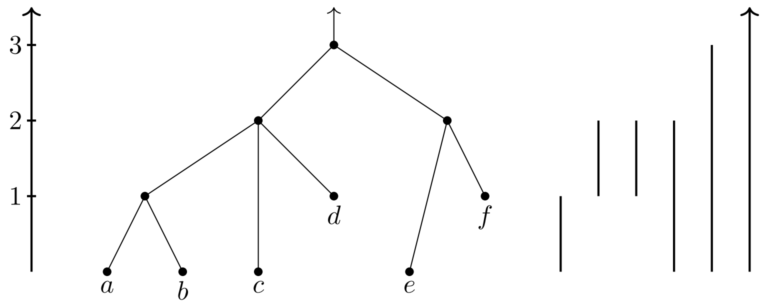


Figure 34: A dendrogram and its barcode $\{[0, 1), [1, 2), [1, 2), [0, 2), [0, 3), [0, \infty)\}$.

The following gives the algorithmic construction of the barcode of a dendrogram

Algorithm 4 Barcode construction

```
Input: Dendrogram $D$
1: $\bar{D} \leftarrow$ augmented dendrogram of $D$
2: $L \leftarrow$ set of leaves of $D$
3: Sort $L$ by increasing height, breaking ties arbitrarily.
4: $\mathcal{B} \leftarrow \emptyset$.
5: for each $v \in L$ in increasing order do
6:     Traverse the upward path in $\bar{D}$ starting from $v$, marking each vertex encountered, until reaching
       a vertex $w$ that is already marked or has height $\infty$
7:     Add the interval $[h(v), h(w))$ to $\mathcal{B}$.
8: Return $\mathcal{B}$.
```

Example 5.33. We construct the barcode of the dendrogram in Figure 34 using the method just described. For a leaf v, let γ_v denote the path from v computed by the algorithm. We sort the leaf nodes by height, e.g., a, b, c, e, d, f.

- γ_a terminates at a vertex of height ∞; so we add the interval $[0, \infty)$ to $\mathcal{B}$.
- γ_b meets γ_a at height 1; we add $[0, 1)$ to $\mathcal{B}$.
- γ_c meets γ_a at height 2; we add $[0, 2)$ to $\mathcal{B}$.
- γ_e meets γ_a at height 3, we add $[0, 3)$ to $\mathcal{B}$.
- γ_d meets γ_a at height 2; we add $[1, 2)$ to $\mathcal{B}$.
- γ_f meets γ_e at height 2; we add $[1, 2)$ to $\mathcal{B}$.

Thus, the barcode is $\{[0, 1), [1, 2), [1, 2), [0, 2), [0, 3), [0, \infty)\}$.

Remark 5.34. We can think of this as a procedure where we cut the dendrogram at vertex w, each time we encounter a vertex w that has previously been marked. Formally, if the upward path starting from vertex v is

$$\gamma_v = v = x_1, \dots, x_n = w,$$

where w has previously been marked, then we remove the edge $\{x_{n-1}, w\}$ from the dendrogram and add a new vertex w', along with an new edge $\{x_{n-1}, w'\}$. This yields a new dendrogram D' where vertices have degree at most 2. Figure 35 illustrates the cut dendrogram D' arising from Example 5.33. Intuitively, by straightening each component of the cut dendrogram, we get the barcode.

Besides providing intuition for barcodes of dendrograms, this cutting construction leads to a useful procedure for obtaining a single clustering from a dendrogram, which we discuss below.

The usual interpretation of the barcode of a dendrogram is as follows: Each interval corresponds to a cluster in the data (at some scale), and loosely speaking, the size of the interval is a measure of importance of the corresponding cluster.

Example 5.35. The barcode arising from Example 5.31 is $\{[5, -\infty), [4, 2)\}$. (Here, because the order on $T = \{5, 4, 3, 2\}$ is the opposite order of the usual one, the intervals are written with the larger coordinate first, and the infinite height is written as $-\infty$ rather than ∞.)

Remark 5.36. Barcodes can also be defined in a different way, via linear algebra, that generalizes beyond the clustering setting to detect higher order structure like loops in data. This is the essential idea of *persistent homology.*

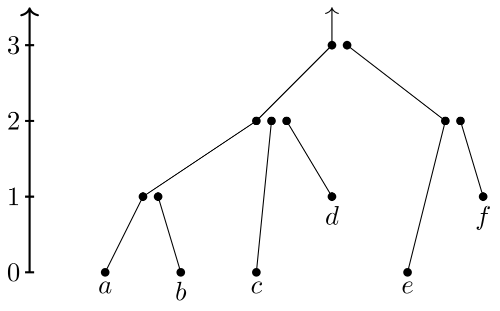


Figure 35: A cut up dendrogram.

5.10 The ToMATo approach to density-based clustering

While dendrograms are useful for visualizing hierarchical clustering, applications often require us to make a single fixed choice of a clustering. When working with single linkage or average linkage, one usually just chooses one of the clusterings in the hierarchical partition, with the help of the dendrogram to guide the choice. As noted earlier, we can interpret this visually as horizontally cutting the dendrogram.

This approach is less effective for dendrograms arising from density-based clustering, because the leaves of such dendrogams have different heights. A density-based hierarchical clustering methodology called ToMATo (short for Topological Mode Analysis Tool), introduced by Chazal et al. in 2009, therefore introduces a different approach to obtaining a single clustering from a dendrogram. The approach is closely related to the construction of barcodes from a dendrogram described in Section 5.9.

To explain the idea, recall that we can think of the barcode construction as a procedure for cutting up a dendrogram D into a new dendrogram D' with vertices of degree at most 2. Noting that each vertex of D is also a vertex of D', this cutting procedure induces a partition $V/\sim$ of the vertex set V of D, where $v \sim w$ if and only if v and w lie on the same component of D'. In fact, $V/\sim$ induces a partition of X, under the mild assumption that every $x \in X$ appears somewhere in hierarchical partition giving rise to D. Recall that for each vertex v of D, there is an associated set (i.e., cluster) $S_v \subset X$. For each $x \in X$, there is a unique vertex $v = v(x)$ of D of minimum height such that $x \in S_v$. We obtain a partition of X by taking $x \sim y$ if and only if $v(x)$ and $v(y)$ lie on the same component of D'. If all leaf nodes have distinct heights, this partition is uniquely determined.

Example 5.37. For D the simplified dendrogram arising from Example 5.31, D' has two components, and the resulting clustering is $\{\{e, f, g, h, i, k, l\}, \{a, b, c, d\}\}$.

Unfortunately, if D has many small branches, the resulting clustering will have too many small clusters, and may therefore miss natural coarse-level cluster structure. To correct for this, one can fix a parameter $\tau > 0$ and consider the following variant of the above cutting procedure: When considering a leaf v whose upward path meets a previously marked vertex w, we cut the dendrogram only if $h(w) - h(v) > \tau$. This yields a new cut dendrogram D^τ, where D^0 is the cut dendrogram D' defined above. Then the procedure above for obtaining a partition of X based on D' extends to give a partition of X based on D^τ, which we call the *τ-cut clustering of X*.

Example 5.38. Let us find the 2-cut clustering (or ToMATo clustering with $\tau = 2$) of the dendrogram D in Figure 34. In Figure 35, the 0-cut of D is shown, but in the 2-cut, we have to be careful to not cut off branches of length 2 or shorter. Specifically, the branches starting at b, c, d and f have lengths $1 - 0 = 1$, $2 - 0 = 2$, $2 - 1 = 1$ and $2 - 1 = 1$, respectively, so these need to be glued back on. But the branch starting at e has length $3 - 0 = 3$, which is more than 2, so this branch remains cut off. This gives the cut dendrogram shown in Figure 36. Observe that the τ-clustering for any $2 \leq \tau < 3$ is exactly the same.

Next, we find the clustering, which is a partition of $\{a, b, c, d, e, f\}$. This is given by the connected components of the tree. We have one component containing a, b, c and d, and one component with e and f. Thus, the 2-clustering is $\{\{a, b, c, d\}, \{e, f\}\}$.

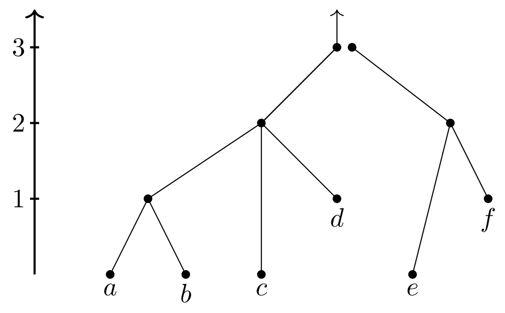


Figure 36: The τ-cut of a dendrogram for $2 \le \tau < 3$.

Example 5.39. For D the simplified dendrogram arising from Example 5.31, the cut dendrogram D^τ is D' for $\tau > 2$, in which case the associated clustering is $\{\{e, f, g, h, i, k, l\}, \{a, b, c, d\}\}$. We have $D^\tau = D$ for $\tau \le 2$, in which case the associated clustering is $\{X\}$.

The question now arises of how to choose τ. For this one, can use the barcode $\mathcal{B}$, which is readily visualized. One hopes to choose τ such that most intervals in $\mathcal{B}$ have length $< \tau$, and the few remaining intervals in $\mathcal{B}$ have length significantly larger than τ. In cases where there is prominent cluster structure and the parameters of a density-based clustering are well chosen, one may be able to find such τ. That said, there will not always be a clear choice τ, as, e.g., when the dendrogram looks as in Example 5.31. The need to choose τ, along with the other parameters for density-based clustering, is one limitation of this approach.

5.11 Exercises on Clustering

Exercise 5.1. Let $X = \{1, 2, 4, 5\} \subseteq \mathbb{R}$ be equipped with the standard metric. Find optimal k-means clustering(s) of X for $k = 1, 2, 3, 4$. Does any of these have several valid answers?

Exercise 5.2. Let $X = \{(0, 0), (1, 0), (0, 2), (1, 2)\} \subseteq \mathbb{R}^2$. Pick what you believe to be the 2-means clustering of X (you do not have to show that it is the correct one) and calculate the sum $\sum_{i=1}^k \sum_{x \in C_i} d_2(x, \mu_i)^2$ as in Definition 5.4 for your choice of clustering.

Exercise 5.3. Use Corollary 5.8 to argue that the clustering in Figure 21 does not form a correct 2-means clustering.

Exercise 5.4. Let $X = \{1, 2, 4, 5, 10\} \subseteq \mathbb{R}$ be equipped with the standard metric. Find the single linkage clustering of X for $r = 0, 1, 3, 7$.

Exercise 5.5. Let $X = \{(0, 0), (1, 0), (0, 2), (1, 2), (4, 6)\} \subseteq \mathbb{R}^2$ be equipped with the Euclidean metric. Find the single linkage clustering of X for $r = 1, 2$, and find the smallest r that gives just one cluster.

Exercise 5.6. Argue that by applying single linkage clustering for an appropriate r, we get the clustering shown in Figure 21.

Exercise 5.7. Let $X = \{AAAA, BAAA, AA, CCCC, CDDD\}$ be equipped with the edit distance d_e. Find the single linkage clustering of (X, d_e) for $r = 1, 2, 3$.

Exercise 5.8. Construct the dendrogram of the collection of partitions from Exercise 5.7.

Exercise 5.9.[*] Prove Proposition 5.23.

Exercise 5.10.[*] Why is the hypothesis $deg(v) = 2$ important in the simplification technique for dendrograms?

Exercise 5.11. Consider the set of all the words in Exercise 4.15 with the edit distance. Build the dendrogram of this metric space using the algorithm in the notes.

Exercise 5.12. Let $X = \{1, 2, 3, 4\}$, and let $P = \{P_0, P_1, P_2, P_3\}$ be the hierarchical partition of X given by

$$\begin{aligned} P_0 &= \{\{1\}, \{2\}, \{3\}, \{4\}\} \\ P_1 &= \{\{2\}, \{1, 3, 4\}\} \\ P_2 &= \{\{2\}, \{1, 3, 4\}\} \\ P_3 &= \{X\} \end{aligned}$$

Draw the dendrogram of P. (Hint: writing the elements of X in the order $1, 2, 3, 4$ at the bottom of the dendrogram might not be a good idea.)

Exercise 5.13. Construct the dendrogram of the collection of partitions from Exercise 5.5.

Exercise 5.14. Sketch the single linkage clustering dendrogram of the set of points in Figure 22. (Hint: It should be possible to read off from your dendrogram that there are many points, and that it is reasonable to split the set of points into either five or two clusters.)

Exercise 5.15.

(a) Find a set of points $X = \{a, b, c, d, e\} \subseteq \mathbb{R}$ (or $\mathbb{R}^2$ if you prefer) whose single linkage clustering dendrogram is the one in Figure 37.

(b) Find a weighted graph on the set $X = \{a, b, c, d, e\}$ of vertices whose single linkage clustering (constructed using the shortest path metric) dendrogram is the one in Figure 37.

In either case, how many connected components does $N(X)_1$ have? How many connected components does $N(X)_6$ have?

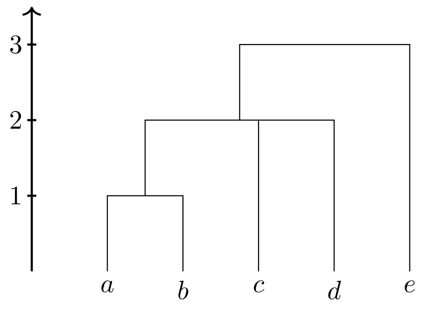


Figure 37: A dendrogram.

Exercise 5.16. Consider the metric (X, d) with $X = \{(1, 1), (0, 0), (2, 2), (3, 2), (3, 3)\} \subset \mathbb{R}^2$. Run Lloyd's algorithm (on paper) with $k = 2$ and initial cluster centers $\mu_1 = (2, 2)$, $\mu_2 = (3, 2)$. Compute the cost of the resulting clustering.

Exercise 5.17. For each of the following metric spaces (X, d),

a. $X = \{(1,1),(0,0),(2,0),(0,2),(2,2)\} \subset \mathbb{R}^2$, $d = d_{\max}$.

b. X as above, $d = d_1$.

(i) give an explicit expression for the single linkage clustering $SL(X)_r$ for each $r \in [0,\infty)$,
(ii) draw the single linkage dendrogram.

Exercise 5.18.

1. Write a Python function GaussianClusters(k,n) which takes as input a number k, and integer $n > 0$, and does the following:
   - Compute an i.i.d. sample $c_1,\dots,c_k$ from the unit square $[0,1]^2$.
   - For each $i \in \{1,\dots,k\}$, compute an i.i.d. sample X_i of n points from a Gaussian distribution with mean c_i and covariance matrix
   $$\begin{pmatrix} .1 & 0 \\ 0 & .1 \end{pmatrix}.$$
   - Output $X = \bigcup_{i=1}^k X_i$ as a list.
2. Run this function with $n = 100$ and $k = 3$, and plot the output.

Exercise 5.19.

1. Implement Lloyd's algorithm from scratch in Python, choosing the initial cluster centers randomly from the set X. (Make sure the cluster centers are distinct, i.e., chosen randomly *without* replacement.)
2. Compute a data set X as in Exercise 5.18 (ii). (Make sure that the c_i are chosen far enough apart so that a plot reveals clear three-cluster structure in X.)
3. Taking $k = 3$, run your implementation of Lloyd's algorithm on X 20 times (i.e., for 20 different random choices of initial cluster centers). For each iteration, compute the cost of the clustering. What are the lowest, highest, and mean value of the cost among the 20 runs?
4. Plot the clusterings yielding the lowest and highest values of cost, using colors to distinguish between the different clusters. Use two separate plots for the two clusterings.
5. Compute a 3-means clustering of K using scikit-learn's implementation of k-means in Python. Again, compute the cost, and plot the resulting clustering. How do the results compare?
6. Run scikit-learn's implementation of k-means for $k = 1, \dots 20$, and plot the costs of the resulting clusterings as a function of k. At which values of k do you see an "elbow?"
7. Use skikit-learn to compute the single linkage dendrogram of X. Does the dendrogram clearly reveal the presence of three clusters in X?
8. Do the same for the average linkage dendrogram of X. Does the dendrogram clearly reveal the presence of three clusters in X?

Exercise 5.20. Construct the dendrogram of the average linkage clustering over

$$X = \{(0,0),(-0.1,0.5),(0.3,1.1),(2.1,1.9),(1,2),(0.7,0)\}$$

with the Euclidean metric.

Exercise 5.21. Construct the dendrogram of the average linkage clustering over

$$X = \{(0,0), (-0.1, 0.5), (0.3, 1.1), (2.1, 1.9), (1, 2), (0.7, 0)\}$$

with the metric $d = d_1$.

Exercise 5.22. For each of the following metric spaces (X, d),

a. $X = \{(-1,-1), (2,3), (2,4), (3,3), (-1,-2), (5,5)\} \subset \mathbb{R}^2$, $d = d_{\max}$.

b. X as above, $d = d_1$.

(i) give an explicit expression for the average linkage clustering $SL(X)_r$ for each $r \in [0, \infty)$,
(ii) draw the average linkage dendrogram.

Exercise 5.23. Construct the single linkage clustering and the average linkage clustering of the three weighted graphs from Figure 13.

Exercise 5.24. Consider the Christmas tree graph in Figure 38. Give to each edge $\{u, v\}$ the weight $w(u, v) = \frac{1}{2}(deg(u) + deg(v))$. Construct the single linkage clustering and the average linkage clustering of the weighted Christmas tree graph.

Exercise 5.25. Consider the Cossack graph in Figure 38. Give to each edge $\{u, v\}$ the weight $w(u, v) = \frac{1}{2}(deg(u) + deg(v))$. Construct the single linkage clustering and the average linkage clustering of the weighted Cossack graph.

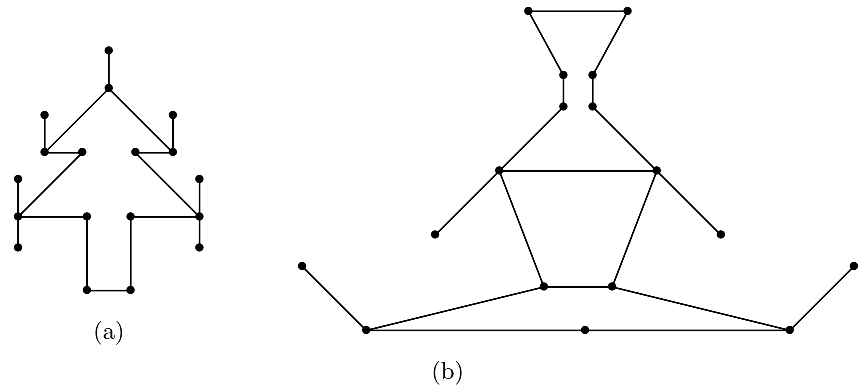


Figure 38: A Christmas tree graph (a) and a Cossack graph (b).

Exercise 5.26. Let $X = \{0, 1, 4, 5\} \subseteq \mathbb{R}$ be equipped with the standard metric. Find optimal k-means clustering(s) of X for $k = 1, 2, 3, 4$. Does any of these have several valid answers?

Exercise 5.27. Let $X = \{(0,0), (1,0), (0,2), (1,2), (2,2)\} \subseteq \mathbb{R}^2$. Pick what you believe to be the 2-means clustering of X (you do not have to show that it is the correct one) and calculate the sum $\sum_{i=1}^{k} \sum_{x \in C_i} d_2(x, \mu_i)^2$ as in Definition 5.4 for your choice of clustering.

Exercise 5.28. Let $X = \{AAD, BAAD, AA, CCCC, CDDD, C, DD\}$ be equipped with the edit distance d_e. Find the single linkage clustering of (X, d_e) and draw its dendrogram.

Exercise 5.29. Find the barcode of the dendogram in Figure 39.

Exercise 5.30. Find a dendrogram that has the barcode $\{[1, \infty), [2, 4), [3, 5), [4, 6), [3, 3.5), [3, 3.5)\}$.

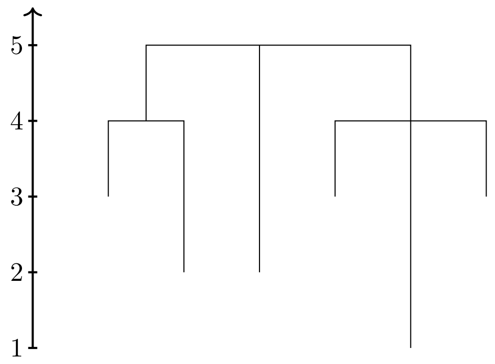


Figure 39: A dendrogram.

Exercise 5.31. Find the average linkage dendrogram of $\{0, 1, 3, 4, 7, 9\}$.

Exercise 5.32. Find the average linkage dendrogram of $\{(0,0), (3,0), (0,4), (3,4)\}$.

Exercise 5.33. Find a dendrogram that has the barcode $\{[0.5, \infty), [1, 4), [2, 2.5), [2, 2.5), [3.5, 5), [4.5, 6)\}$.

Exercise 5.34. Find the barcode of the dendrogram in Figure 40.

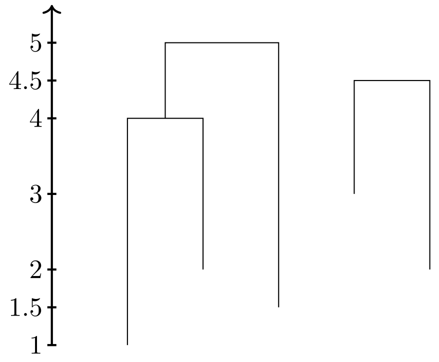


Figure 40: A dendrogram.

Exercise 5.35. Argue whether the following is a valid barcode $\{[2, \infty), [4, 5), [1, 3)\}$.

Exercise 5.36. Let D be the dendrogram in Figure 39. Label its leaves from left to right with $\{a, b, c, d, e, f\}$. Give the 0-, 1-, and 2-cut clusterings of the partition associated with D.

Exercise 5.37. Consider the dendrogram from Exercise 5.33. Find the (1.5)-cut clustering of it.

6 Algorithmic complexity

6.1 What is an algorithm?

The following is an informal definition: *An algorithm is a formal set of instructions for performing a calculation or solving a problem, which can be implemented on a computer.*

An algorithm has a well-defined set of inputs, and for each valid input, it returns a unique output. Thus, we can think of an algorithm as describing a function. But while a function is completely determined by its input and output (and codomain), an essential aspect of an algorithm is how the output is obtained.

Example 6.1. Gaussian elimination is an algorithm that takes a matrix as input and gives a matrix in row echelon form as output.

Example 6.2. We describe an algorithm NaiveSearch that takes as input a list L of numbers in increasing order and a number x, and returns "True" if x is in L, and "False" if x is not in L. Informally, our algorithm works as follows:

Starting at the beginning of L, look at each entry in order.

- If you see x, stop and return "True".
- If you reach the end of L, return "False".

The same algorithm in pseudocode:

Algorithm 5 NaiveSearch

```
Input: A list $L$ and an element $x$
1: for $i = 0$ to length($L$) $- 1$ do
2:     if $L[i] = x$ then
3:         return True
4: return False
```

The correctness of NaiveSearch is clear, but in general, the correctness of an algorithm can be quite a nontrivial matter and may require a careful proof.

Of course, there can be mutliple algorithms for solving a given problem. For searching a sorted list, another standard algorithm is *binary search*.

Definition 6.3. The *median* of a sorted list L is

- the middle entry if $|L|$ is odd,
- the left-middle entry if $|L|$ is even.

Algorithm 6 BinarySearch(L, x)

```
Input: A list $L$ and an element $x$
1: Let $y$ = median($L$)
2: if y=x then
3:     return True
4: if $|L| = 1$ then
5:     return True
6: if $y < x$ then
7:     Let $L'$ be the part of $L$ to the right of $y$
8: else
9:     Let $L'$ be the part of $L$ to the left of $y$
10: return BinarySearch$(L', x)$
```

In the pseudocode above, we have assumed tacitly that we can compute the required medians. If, e.g., L is represented by an array, we can quickly access an arbitrary element of L (in constant time). In particular, we can quickly access the median. If the list is represented differently, e.g., as a linked list

then computing median(L) might be roughly as expensive as running NaiveSearch.

$$\begin{tikzcd}
2 \arrow[r, "\text{pointer}"] & 3 \arrow[r, "\text{pointer}"] & 5 \arrow[r] & \cdots \arrow[r] & 29
\end{tikzcd}$$

Figure 41

6.2 Complexity analysis of algorithms

To analyze the complexity (or "running time") of algorithms, we use the following idea:

(1) Model the execution of an algorithm as a sequence of "elementary operations"

(2) Count (or upper bound) the number of elementary operations an algorithm requires in the worst case, as a function of the size of the input

Remark 6.4. Such "worst-case" analysis is the dominant analysis paradigm, but other paradigms are important, and are becoming increasingly prominant. For instance, it may make sense to evaluate how an algorithm performs on average instead of in the worst case.

By *elementary operation*, we mean an operation that takes at most some constant amount of time on a particular class of computers. Examples are

- adding or multiplying two numbers
- writing a number to memory,
- checking whether two numbers are equal,
- checking whether one number is less than another,
- looking up the value of a number stored in an array,
- assigning a value to a variable,
- executing an if statement

This suggests a close relationship between algorithms and data structures. This is a major theme of the subject. From now on, we will assume that all our lists are stored in arrays.

Intuitively, BinarySearch is more efficient than NaiveSearch, because it is able to rule out half the list at each step. But how do we make this precise? The engineering solution would be to code the algorithms, run them on a bunch of test problems and compare the times. But this is not sufficient:

- The answer may depend on low-level details about machine architecture, implementation and test problems.
- It is difficult to compare findings across different sets of experiments.
- Many algorithms are hard to implement; we want to know in advance if they are efficient.
- We want a conceptual solution (not just if an algorithm is fast, but why).

There are two main aspects of an algorithm:

- Time to execute (run time), and
- memory required (storage time).

Both are important and they are also related: If an algorithm uses a lot of memory, then it also requires a lot of time, because it takes time to write to memory. However an algorithm may take a lot of time to run without using a lot of memory. Consider for example Algorithm 7. It takes a long time to run if n is big (which is part of what makes online bank transactions safe), but it takes up very little memory.

Algorithm 7 Is Prime

Example 6.5. Input: A natural number n

```
1: for $i = 2$ to $\sqrt{n}$ do
2:     if $i$ divides $n$ evenly then
3:         return False
4: return True
```

For the scope of this course, we will focus on runtime complexity, but it is important that you remember that this is only part of the picture.

Above we listed some elementary operations. To better understand the concept, we now list some examples of non-elementary operations:

- Searching a list
- Sorting a list
- Adding two vectors of the same length
- Summing the values of all elements in a list

You may notice that all these operations depend heavily on the size of the input: adding two vectors in $\mathbb{R}^2$ is faster than adding two vectors in $\mathbb{R}^{253}$.

The upshot is that each of the above is in practice performing a number of elementary operations (such as adding two numbers, comparing two numbers, etc) comparable to the size of the inputs. Thus, it makes sense to use the number of elementary operations as a measure of the runtime complexity. However, if we try to count *exactly* the number of elementary operations, we run into technical issues that are not straightforward to solve, such as semantic issues (e.g. is $(2+3)4$ one operation or two?), as well as interpretation issues. For example, if I have three algorithms, one performing 9 elementary operations, and the other two respectively 581 and 575, is it really important to know the exact number? Isn't it enough to know that the first has 2 orders of magnitude fewer operations?

The asymptotic notation, which we will see next, is addressing precisely these issues.

6.3 Asymptotic notation

We begin with the *big-O notation*.

Definition 6.6. Let $S \subseteq [0, \infty)$. Given $g \colon S \to \mathbb{R}$, define

$$O(g) := \{f \colon S \to \mathbb{R} \mid \exists c, n_0 > 0 \text{ such that } 0 \le f(n) \le cg(n) \, \forall n \ge n_0\} .$$

If $f \in O(g)$ we usually write $f = O(g)$.

In other words, $f = O(g)$ means that there exists c such that $0 \le f(n) \le cg(n)$ for all n sufficiently large.

Example 6.7. Let $S = [0, \infty)$, and consider $f(x) = x + 2$, $g(x) = x^2$. We have $0 \le x + 2 \le x^2$ for all $x \ge 2$, so $f = O(g)$, or $x + 2 = O(x^2)$.

Example 6.8. Let $S = [0, \infty)$, and consider $f(x) = x + 2$, $g(x) = 3x + 1$. We have $0 \le f(x) \le g(x)$ for all $x \ge \frac{1}{2}$, so $f = O(g)$, or $x + 2 = O(3x + 1)$. In addition, $0 \le g(x) \le 3f(x)$ for all $x \in S$, so $g = O(f)$.

In general, if $f = O(g)$ and $g = O(f)$, we write $f = \Theta(g)$ or equivalently $g = \Theta(f)$. Indeed, we can define:

$$\Theta(g) := \{f \colon S \to \mathbb{R} \mid f = O(g) \text{ and } g = O(f)\} .$$

Example 6.9. Let $S = [0, \infty)$, and consider $f(x) = \sin x + 1$, $g(x) = 1$. We have $0 \leq \sin x + 1 \leq 2$ for all $x \geq 2$, so $f = O(g)$, or $\sin x + 1 = O(1)$.

The set $O(1)$ is given by all the functions bounded above by a constant for all x sufficiently large.

Proposition 6.10. *Let $f, g \colon S \to [0, \infty)$. Then $f + g = \Theta(\max(f, g))$.*

We now describe another concept, the *small-o notation*.

Definition 6.11. Let $S \subseteq [0, \infty)$ and $f, g \colon S \to \mathbb{R}$ two functions such that, for $x \in S$ sufficiently large, f, g are positive. We write $f = o(g)$ if

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$$

and

$$o(g) := \{f \colon S \to \mathbb{R} \,|\, \forall\, c > \ \exists n_c > 0 \text{ such that } f(n) < cg(n) \ \forall n \geq n_c\}\,.$$

Example 6.12. Take $S = [0, \infty)$, $f(x) = x$, and $g(x) = x^2$.

$$\lim_{x \to \infty} \frac{x}{x^2} = \lim_{x \to \infty} \frac{1}{x} = 0\,.$$

Thus, $x = o(x^2)$.

Example 6.13. Take $S = [0, \infty)$, $f(x) = x$, and $g(x) = 100x + 7$.

$$\lim_{x \to \infty} \frac{x}{100x + 7} = \lim_{x \to \infty} \frac{1}{100 + \frac{7}{x}} = \frac{1}{100} \neq 0\,.$$

Thus, $x \neq o(100x + 7)$.

Note that, in the above example, $f(x) < g(x)$ for all $x \in \mathbb{R}$. So, this example shows that $f(x) < g(x)$ does not imply $f = o(g)$. Indeed, $f = o(g)$ means that f is smaller than any constant multiple of g.

Proposition 6.14. *If $f = o(g)$ then $f = O(g)$.*

Proposition 6.15.

1. *Let $S \subseteq [0, \infty)$, and $f, g, h \colon S \to \mathbb{R}$. If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.*
2. *Let $S \subseteq [0, \infty)$, and $f, g, h \colon S \to \mathbb{R}$. If $f = o(g)$ and $g = o(h)$, then $f = o(h)$.*
3. *Let $S \subseteq [0, \infty)$, and $f \colon S \to \mathbb{R}$. Then $f = O(f)$.*

Asymptotic properties of polynomials. A polynomial is a function from $\mathbb{R}$ to $\mathbb{R}$ of the form $p(x) = \Sigma_{i=0}^{n} a_i x^i$, where $a_i \in \mathbb{R}$ for all $i = 0, \ldots, n$, and $a_n \neq 0$. The *degree of the polynomial* $p(x)$, denoted by *deg*(p), is n, and a_n is its *leading coefficient*.

Proposition 6.16. *Let p and q be two polynomials.*

1. *If deg$(p) \leq$ deg(q) then $p = O(q)$.*
2. *If deg$(p) <$ deg(q) then $p = o(q)$.*

As a consequence, if *deg*$(p) =$ *deg*(q) then $p = \Theta(q)$.

In plain words, Proposition 6.16 is telling us that, when comparing the asymptotic behavior of two polynomials, only their leading coefficients matter.

Remark 6.17. Multiplying functions by positive constants does not change the truth of the asymptotic statements.

Asymptotic properties of logarithm and exponential.

Proposition 6.18. *If $c, b > 1$, then $\log_c(x) = \Theta(\log_b(x))$.*

In plain words, in the asymptotic notation, the choice of a base for the logarithm does not matter.

Proposition 6.19. *For all $a > 0$, $\ln(x) = o(x^a)$.*

Corollary 6.20. *For all $c > 1$ and $a > 0$, $\log_c(x) = o(x^a)$.*

Proposition 6.21. *For all $a \in (1, \infty)$ and $b \in \mathbb{R}$, $x^b = o(a^x)$.*

Intuitively we think of $f = O(g)$ as "f does not grow faster than g", $f = \Theta(g)$ as "f and g grow at the same pace" and $f = o(g)$ as "f grows slower than g". This is a useful simplification, but one should be careful about what conclusions to draw from it. For instance, the following example shows that there are f and g with $f = O(g)$, but neither $f = o(g)$ nor $f = \Theta(g)$. By our simplification, this reads as "f does not grow faster, slower, nor at the same pace as g", which might sound counterintuitive. In practice, however, one usually does not encounter functions f and g with these properties.

Example 6.22. Let $g = e^x$ and $f = e^{x \sin x}$. We have $f(x) \leq g(x)$ for all x, so $f = O(g)$. At the same time, $f((2n+1)\pi) = g((2n+1)\pi)$ for all $n \in \mathbb{N}$, so $f \neq o(g)$. But $f(n\pi) = 1$ for all $n \in \mathbb{N}$, so $f \neq \Theta(g)$.

Asymptotic complexity of algorithms. In the analysis of algorithms, we want to study the asymptotic complexity of a function $f(n)$, where n is the input size, that counts the number of elementary operations required by a given algorithm to compute the output.

Example 6.23. let us compute the asymptotic complexity of the algorithm Algorithm 5. So, let us consider $f\colon \mathbb{N} \to \mathbb{R}$, where $f(n) :=$ (the O of the number of elementary operations that Algorithm 5 performs in the worst case over a list of length n). We go through all the lines in the code, and we see that in line 1 there is a loop that runs, at worst, over the full list. Thus, at worst, we need to do n times whatever is the cost of the lines inside the loop. These are lines 2 and 3. The former has one elementary operation, the latter none. So, the worst-case cost for now is $n \cdot 1 = n$. In line 4 there are no operations, and we can conclude. Thus, $f(n) = O(n)$.

6.4 Merge sort

A standard computational task is to sort a list of numbers. In many cases, having a sorted list makes it much easier to retrieve stored information, or to check if a requested element exists or not. Especially if a sorted list is combined with a good data structure, like the binary tree considered in Example 3.24, checking if an element is in the list becomes extremely efficient. To be precise, it can be done in $O(\log n)$ operations (assuming we have already constructed the binary search tree before we begin) by starting at the root and moving downwards. In this subsection, we will study a specific algorithm called *merge sort* that takes as input an unsorted list of numbers and returns a sorted list of the same set of numbers. If there are n numbers in the list, merge sort can be executed in $O(n \log n)$ operations. Since any correct sorting algorithm needs to at least look at all the n numbers in the list, it is impossible to sort in $o(n)$, so $O(n \log n)$ is very close to the best complexity we could hope for, and allows us to sort very large lists in a reasonable amount of time.

Merge sort uses a subroutine Merge, which merges two sorted list into a single sorted list. We first show how Merge acts on a specific example, then we write Merge in pseudocode.

Example 6.24. Let us merge the lists $L_1 = [1, 4, 5, 9, 10]$ and $L_2 = [3, 7]$. We start by comparing the first element in each list. $1 < 3$, so we put 1 as the first element in the merged list M. Next, we compare 4 with 3. We have $4 > 3$, so we add 3 to the M, which is now $[1, 3]$. Next, we compare 4 with 7. Since $4 < 7$, we add 4 to M and move to 5, which is also less than 7 and is therefore added to M, which is now $[1, 3, 4, 5]$. Next, we have $7 < 9$, so we add 7 to M. With this, we have reached the end of L_2, so we add the remaining elements of L_1 to M, and end up with $M = [1, 3, 4, 5, 7, 9, 10]$.

Algorithm 8 Merge

```
Input: Two sorted lists $L_1, L_2$ of natural numbers
$i = j = 0$
$M = []$
while $i <$len($L_1$) AND $j <$len($L_2$) do
    if $j =$len($L_2$) then
        Append $L_1[i]$ to $M$
        $i = i + 1$
    else if $i =$len($L_1$) then
        Append $L_2[j]$ to $M$
        $j = j + 1$
    else if $L_1[i] < L_2[j]$ then
        Append $L_1[i]$ to $M$
        $i = i + 1$
    else
        Append $L_2[j]$ to $M$
        $j = j + 1$
return $M$
```

For every time we add an element to the merged list M, we only do a constant number of operations. If k is the number of elements in M (which is the sum of the sizes of L_1 and L_2), this means that Merge runs in $O(k)$.

To run merge sort, we first split the input list L into $n = |L|$ lists with one element each. At each step of the algorithm, we have a collection of sorted lists that we put together in pairs (possibly leaving one unpaired list that we leave until the next step), and run Merge on each pair. This gives us roughly half as many lists as we had in the previous step, and we do the same procedure with the new collection of sorted lists. In pseudocode:

Algorithm 9 MergeSort

```
Input: A list $L$ of natural numbers
$n =$len($L$)
previousLists = $[[L[0]], \ldots, [L[n-1]]]$
while len(previousLists) $> 1$ do
    newLists = []
    for $i = 0$ to len(previousLists)$-1$, $i$ even do
        if $i =$len(previousLists)$-1$ then
            Append previousLists[$i$] to NewLists
        else
            Append Merge(previousLists[$i$],previousLists[$i+1$]) to newLists
    previousLists = newLists
return previousLists
```

We now analyze the running time of MergeSort, leaving out some of the detailed calculations. We start with n lists, and for each iteration of the while loop, the number of lists is roughly halved. Thus, we run thorough the while loop roughly $\log_2 n$ times. (At most $1 + \log_2 n$ times, to be precise.) For each iteration of the while loop, we need to run Merge a number of times. But we know that that Merge runs in $O(k)$, where k is the number of elements in the lists. Since there are n elements in total in all the lists, running Merge requires in total $O(n)$ time for each iteration of the while loop. Thus, we have $O(\log n)$ iterations that each takes $O(n)$ time, so the total running time of MergeSort is $O(n \log n)$.

6.5 Exercises on asymptotic complexity

Exercise 6.1. Prove Proposition 6.10.

Exercise 6.2. Rank the following functions according to their small-o behavior:

$$a)\ \frac{1}{x} \qquad b)\ \log x \qquad c)\ e^{2x} \qquad d)\ e^{x-1} \qquad e)\ x^{5/2} \qquad f)\ \log(x^5) \qquad g)\ 2x^{5/2} \qquad h)\ x^3$$

Exercise 6.3. Find functions $f_r(x)$ for all $r \in \mathbb{N}$ (even better: $r \in \mathbb{R}$) such that $f_r = O(f_s)$ for all $r < s$. That is, $f_1 = O(f_2)$, $f_2 = O(f_3)$ etc.

Exercise 6.4. Argue whether there can be a $o(n)$ time algorithm compute the dot product of two vectors in $\mathbb{R}^n$.

Algorithm 10 Shifting the evens - lists

Example 6.25. Input: A list L of natural numbers
```
for $i = 0$ to $len(L)$ do
    if 2 divides $L[i]$ evenly then
        $L[i] = L[i] + 2$
return $L$
```

Exercise 6.5. Compute the asymptotic runtime complexity of Algorithm 10. Would this complexity change if we added an **else** statement saying "$L[i] = 3L[i]$"?

Algorithm 11 Shifting the evens - matrices

Example 6.26. Input: An $(n \times m)$-matrix M of natural numbers
```
for $i = 1$ to $m$ do
    for $j = 1$ to $n$ do
        if 2 divides $M[i, j]$ evenly then
            $M[i, j] = M[i, j] + 2$
return $M$
```

Exercise 6.6. Compute the asymptotic runtime complexity of Algorithm 11. Would this complexity change if we looped first over n and them over m?

Exercise 6.7. Show that $3xe^{2x+1} = o(x \log(x) e^{2x})$.

Exercise 6.8. Show that $4 + (\cos x)^2 = \Theta(1)$.

Exercise 6.9. Show that $e^{2x} \neq O(e^x)$.

Exercise 6.10. Apply merge sort to the following lists, showing the merged lists at each step. (That is, write the lists in newLists in Algorithm 9 after each iteration of the while loop.)

(a) $[5, 9, 0, 3, 7, 6, 5]$

(b) $[2, 8, 13, 17, 7, 8, 1, 16, 12, 6, 1, 6]$

Exercise 6.11. How many iterations of the while loop does MergeSort (Algorithm 9) run given a list with

(a) 4

(b) 64

(c) 100

(d) 1000

numbers?

A Solutions to Selected Exercises

A.1 Sets and functions

Solution to Exercise 2.1.
The first and third pairs are different, but $\{1, 2, 3\} = \{3, 2, 1\}$.

Solution to Exercise 2.2.
One example is $\{\{\}, \{0\}, \{1\}\}$.

Solution to Exercise 2.3.

1. The set $\{\{1, 2\}\}$ has one element, the 2-elements set $\{1, 2\}$.
2. No.

Solution to Exercise 2.4.
Here are several possibilities:

$$\{x \in \mathbb{Z} \mid x = 2k + 1 \text{ for } k \in \mathbb{Z}\}$$

or

$$\{x \in \mathbb{Z} \mid x = 2k - 1 \text{ for } k \in \mathbb{Z}\}$$

or

$$\{x \in \mathbb{Z} \mid x \text{ is not divisible by } 2\}$$

or

$$\{\ldots, -5, -3, -1, 1, 3, 5, \ldots\}.$$

Solution to Exercise 2.5.

a. $\mathcal{P}(\{A, B\}) = \{\emptyset, \{A\}, \{B\}, \{A, B\}\}$

b. Since the cardinality of the power set of S is $2^{|S|}$, then $|\mathcal{P}(\mathcal{P}(\{A, B\})) = 2^{|\mathcal{P}(\{A,B\})|} = 2^4 = 16$.

Solution to Exercise 2.6.
If T is a finite set with n elements,

a. how many elements does $\mathcal{P}(T^2)$ have? Answer: 2^{n^2}.

b. how many elements does $(\mathcal{P}(T))^2$ have? Answer: $(2^n)^2 = 2^{2n}$.

Solution to Exercise 2.7.

a. $S \cup T = \{1, 2, 3, 5\}$, $S \cap T = \{1, 3\}$, $S \setminus T = \{2\}$, $T \setminus S = \{5\}$.

b. $S \cup T = \{a, b, c, d\} = S$, $S \cap T = \{a, d\} = T$, $S \setminus T = \{b, c\}$, $T \setminus S = \emptyset$ (note that $T \subseteq S$).

c. $S \cup T = \mathbb{Z}$, $S \cap T = \mathbb{N}$, $S \setminus T = \emptyset$, $T \setminus S = \{\text{negative integers}\}$ (note that $S \subseteq T$).

d. $S \cup T = \{2, 4, 6\} = T$, $S \cap T = \emptyset = S$, $S \setminus T = \emptyset = S$, $T \setminus S = \{2, 4, 6\} = T$ (note that $S \subseteq T$).

Solution to Exercise 2.8.

a. $S \setminus (S \cup T) = \emptyset$, we are removing from S all its elements (and the elements in T),

b. $S \setminus (S \cap T) \neq \emptyset$ in general, for example if $S \cap T = \emptyset$, $S \setminus (S \cap T) = S \setminus \emptyset = S$,

b. $S \cup \emptyset = S \neq \emptyset$ in general,

c. $S \cap \emptyset = \emptyset$,

d. $S \setminus \emptyset = S \neq \emptyset$ in general,

e. $\emptyset \setminus S = \emptyset$ (we are removing elements from a set that does not have any to start with!),

f. $(S \cap \{1\}) \cap (S \cap \{2\}) = \emptyset$, since this set is either $= \{1\} \cap \{2\}$, if $1, 2 \in S$, and thus $= \emptyset$, or the intersection of two sets, at least one of which is empty.

Solution to Exercise 2.9.

a. $\{3\}$ is an interval: $[3, 3]$;

b. $\mathbb{Z}$ is not an interval, because it is a disconnected subset of the reals;

c. $\{0, 3\}$ is not an interval, because it is a disconnected subset of the reals;

d. $\{x \in \mathbb{R} \mid x < 0\}$ is an interval: $(-\infty, 0)$;

e. $\{x \in \mathbb{R} \mid x \leq 1\}$ is an interval: $(-\infty, 1]$;

f. $\{x \in \mathbb{R} \mid -4 < x \leq 1\}$ is an interval: $(-4, 1]$;

g. $\{x \in \mathbb{R} \mid x < 3 \text{ and } x > 0\}$ is an interval: $(0, 3)$;

h. $\{x \in \mathbb{R} \mid x < 3 \text{ or } x > 0\}$ is an interval: $(-\infty, \infty)$;

i. $\{x \in \mathbb{R} \mid x < 0 \text{ or } x > 3\}$ is not an interval, because it is a disconnected subset of the reals.

Solution to Exercise 2.10.

a. $[3, \infty) = \{x \in \mathbb{R} \mid x \geq 3)\}$.

b. $(-\infty, 3) = \{x \in \mathbb{R} \mid x < 3\}$.

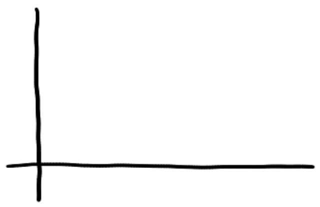
c. $[3, 5) = \{x \in \mathbb{R} \mid 3 \leq x < 5\}$.

d. $(3, 5) = \{x \in \mathbb{R} \mid 3 < x < 5\}$.

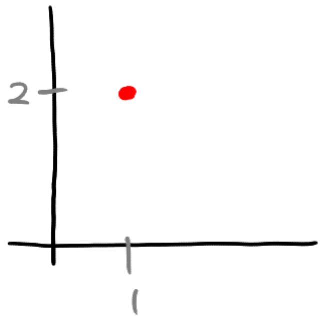
e. $[3, 5] = \{x \in \mathbb{R} \mid 3 \leq x \leq 5\}$.

Solution to Exercise 2.11.

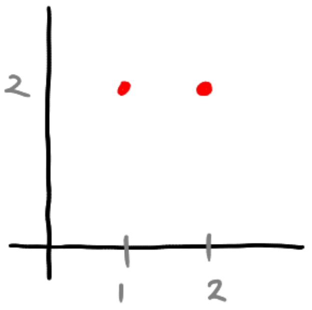
a. $S = \{1\}$, $T = \emptyset \rightarrow S \times T = \emptyset$.



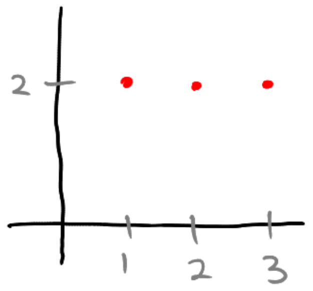
b. $S = \{1\}$, $T = \{2\} \to S \times T = \{(1,2)\}$.



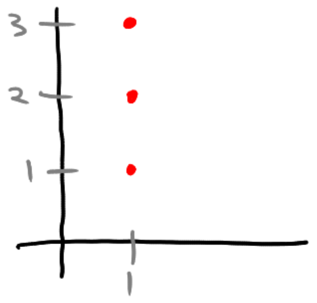
c. $S = \{1,2\}$, $T = \{2\} \to S \times T = \{(1,2),(2,2)\}$.



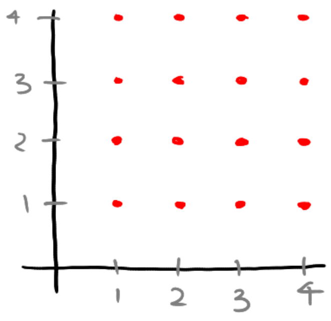
d. $S = \{1,2,3\}$, $T = \{2\} \to S \times T = \{(1,2),(2,2),(3,2)\}$.



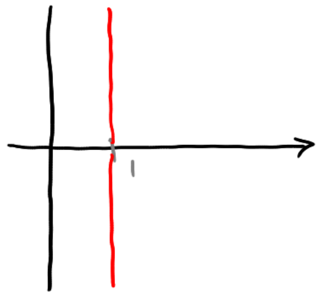
e. $S = \{1\}$, $T = \{1,2,3\} \to S \times T = \{(1,1),(1,2),(1,3)\}$.



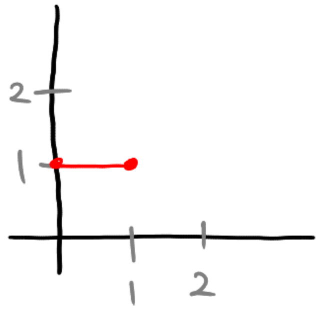
f. $S = \{1,2,3,4\}$, $T = \{1,2,3,4\} \to S \times T = \{(1,1),(1,2),(1,3),(1,4),\ (2,1),(2,2),(2,3),(2,4),$ $(3,1),(3,2),(3,3),(3,4),\ (4,1),(4,2),(4,3),(4,4)\}$.



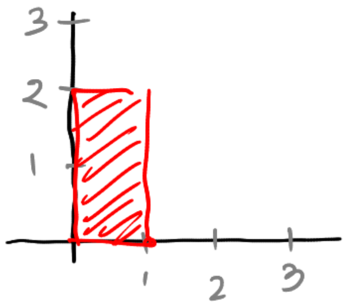
g. $S = \{1\}$, $T = \mathbb{R} \to S \times T = \{(1,x) \mid x \in \mathbb{R}\}$.



h. $S = [0,1]$, $T = \{1\} \to S \times T = \{(x,1) \mid x \in [0,1]\}$.



i. $S = [0,1]$, $T = [0,2] \to S \times T = \{(x,y) \mid x \in [0,1], y \in [0,2]\}$.



Solution to Exercise 2.14.

a. $\text{im}(f) = \mathbb{R}$;

b. $\text{im}(f) = [-1, 1)$.

Solution to Exercise 2.15.

a. f is injective but is not surjective since A is not hit by f.

b. f is surjective but is not injective since both $1, 2 \mapsto A$.

c. f is injective but is not surjective since 0 can not be hit by f, since $f(-3) = 0$ but $-3 \notin \mathbb{N}$.

d. f is injective but is not surjective since odd numbers are not hit by f.

e. f is none; it is not injective since $1, -1 \mapsto 0$ and is not surjective since nothing in $(-\infty, -1)$ is hit by f.

Solution to Exercise 2.16.
For each function, give an explicit expression for its image, and say whether the function is an injection, surjection, or bijection. If the function is a bijection, also give its inverse.

a. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^3$ **Answer**: $\text{im}(f) = \mathbb{R}$. Bijection. $f^{-1}(x) = x^{1/3}$.

b. $f : \mathbb{R} \to \mathbb{R}^2$, $f(x) = (x, x^3)$ [here, just use the bracket notation to express $\text{im}(f)$.] ans $\text{im}(f) = \{(x, x^3) \mid x \in \mathbb{R}\}$. Injective, not surjective.

c. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^4$ **Answer**: $\text{im}(f) = [0, \infty)$. Neither injective nor surjective.

d. $f : \mathbb{R} \to [0, \infty)$, $f(x) = x^4$ **Answer**: $\text{im}(f) = [0, \infty)$. Surjective but not injective.

e. $f : \mathbb{R} \to [-1, 1]$, $f(x) = \cos x$ **Answer**: $\text{im}(f) = [-1, 1]$. Surjective but not injective.

f. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = |x|$. **Answer**: $\text{im}(f) = [0, \infty)$. Neither injective nor surjective.

g. $g : S^1 \times I \to \mathbb{R}^2$, $g(x, y) = x$. (Here, it is understood that $x \in S^1$ and $y \in I$.) **Answer**: $\text{im}(f) = S^1$. Neither injective nor surjective.

h. $h : S^1 \to \mathbb{R}^2$, $h(x, y) = (|x|, |y|)$. **Answer**: $\{(x, y) \in S^1 \mid |x|, |y| \geq 0\}$. Neither injective nor surjective.

Solution to Exercise 2.17.

a. My answer is $f(x) = \frac{1}{x}$. $f^{-1} = \frac{1}{x}$. This is not the only possible answer, but I imagine it is the simplest.

b.

$$f(x) = \begin{cases} 0 & \text{if } x = \frac{1}{2}, \\ 1 & \text{if } x = \frac{1}{4}, \\ \frac{1}{2^{n-2}} & \text{if } x = \frac{1}{2^n} \text{ for } n \in \{3, 4, 5, \ldots\}, \\ x & \text{otherwise} \end{cases}$$

Solution to Exercise 2.18.*
One example is: first all even numbers with their "usual" order, and then all odd numbers with their "usual" order:

$$\ldots, -4, -2, 0, 2, 4, \ldots, -3, -1, 1, 3, \ldots \quad .$$

Solution to Exercise 2.19.

- Short answer: yes. Proof: take a subset S of $\mathbb{N}$. Let $m \in S$. If m is smaller or equal than any other element in S, we are done. So, assume it is not. Take S_m to be the subset of S given by all the elements in S that are smaller or equal than m. The key observation is that S_m is finite, since it is at most given by all natural numbers between 0 and m. Since all finite totally ordered sets have a minimum, the claim is proved.
- No, since for example $(0, 1)$ does not have a smallest element.

Solution to Exercise 2.20.
$\mathbb{R}^2/_\sim = \{S_r \mid r \geq 0\}$, where S_r is the circle of radius r centered at the origin and $S_0 = \{(0, 0)\}$.

Solution to Exercise 2.24.
Which of the following relations $\sim$ on $\mathbb{Z}$ is an equivalence relation? Explain your answer, and for each relation that is an equivalence relation, give the set $\mathbb{Z}/\sim$ of all equivalence classes.

1. $x \sim y$ iff $x - y$ is divisible by 3. (Note: 0 is considered to be divisible by 3.). HINT: Compare this problem to Example 2.51, recalling that by definition, an even integer is an integer that is divisible by 2. Yes, this is an equivalence relation. Let's check each property: Reflexivity: $x \sim x$ for all $x \in \mathbb{Z}$ because 0 is divisible by 3. Symmetry: $x \sim y$ implies $y \sim x$ because if $x - y$ is divisible by 3, then $-(x - y) = y - x$ is divisible by 3. Transitivity: If $x \sim y$ and $y \sim z$, then $x - y$ is divisible by 3 and $y - z$ is divisible by 3. Thus $(x - y) + (y - z)$ is divisible by 3. But $x - z = (x - y) + (y - z)$, so $x \sim z$.
$$\begin{aligned}\mathbb{Z}/_\sim =&\{[0], [1], [2]\}\\ =&\{\{x \in \mathbb{Z} \mid x \text{ is divisible by } 3\},\\ &\{x \in \mathbb{Z} \mid x - 1 \text{ is divisible by } 3\},\\ &\{x \in \mathbb{Z} \mid x - 2 \text{ is divisible by } 3\}\}.\end{aligned}$$
2. $x \sim y$ iff $x - y = 1$. No, this is not an equivalence relation. It suffices to see that it doesn't satisfy the reflexivity property:
$$0 - 0 = 0 \neq 1,$$
so $0 \not\sim 0$. Hence $\sim$ is not reflexive. In fact it doesn't satisfy any of the three properties.
3. $x \sim y$ iff $xy \geq 0$. This is not an equivalence relation, because transitivity fails: $1 \sim 0$ and $0 \sim -1$, but $1 \not\sim -1$.
4. $x \sim y$ iff $x = y$ or $x = -y$. Yes, this is an equivalence relation. By definition, it is reflexive. It is symmetric because $x = y$ iff and $y = x$ and $x = -y$ iff $y = -x$. Transitivity follows from a simple case analysis. Suppose that $x \sim y$ and $y \sim z$. There are four cases to consider:
1)$x = y$ and $y = z$,
2)$x = y$, $y = -z$,
3)$x = -y$, $y = z$,
4)$x = -y$, $y = -z$.
In each case, it is clear that $x \sim z$.

$\mathbb{Z}/_\sim = \{[0], [1], [2], \ldots\}$, where $[0] = \{0\}$ and $[z] = \{z, -z\}$ for $z > 0$.

Solution to Exercise 2.25.
(i) Not an equivalence relation. Transitivity fails: $b \sim a$, $a \sim d$, but $b \not\sim d$. (ii) An equivalence relation. Reflexivity holds because the diagonal contains only 1's. Symmetry holds because the matrix is symmetric. Transitivity holds because if $x \sim y, \sim z$ then either $x, y, z \in \{a, d\}$ or $x, y, z \in \{b, c\}$.

Solution to Exercise 2.26.
Suppose we are given an equivalence relation $\sim$ on a set S and elements $a, b, c \in S$ such that such that $a \not\sim b$ and $a \sim c$. It cannot be true that $b \sim c$, since transitivity and symmetry then imply that $a \sim b$, a contradiction.

Solution to Exercise 2.27.
If $\sim$ is the equivalence relation on $\mathbb{R}^2$ given by $(x_1, x_2) \sim (y_1, y_2)$ if and only if $x_1 = y_1$, then $\mathbb{R}^2/_\sim$ is the set of vertical lines in $\mathbb{R}^2$. To express this in our set-theoretic formalism,

$$\mathbb{R}^2/_\sim = \{V_a \mid a \in \mathbb{R}\},$$

where V_a is the vertical line $x = a$, i.e., $V_a = \{(a, y) \mid y \in \mathbb{R}\}$.

Solution to Exercise 2.28.
Consider an equivalence relation $\sim$ on a set S and elements $a, b, c, d \in S$ such that such that $a \not\sim b$, $b \sim c$, and $c \not\sim d$. Is *sometimes* true that $a \sim d$. For example, it is true for the equivalence relation corresponding to the partition of $\{\{a, d\}, \{b, c\}\}$ of $\{a, b, c, d\}$, but not true for the equivalence relation corresponding to the partition $\{\{a\}, \{b, c\}, \{d\}\}$ of $\{a, b, c, d\}$.

A.2 Graphs

Solution to Exercise 3.1.
$G = (V.E)$, where $V = \{a, b, c, d\}$, and $E = \{\{a, b\}, \{a, d\}, \{c, b\}, \{c, d\}\}$.

Solution to Exercise 3.2.
See Figure 42 and Figure 43.

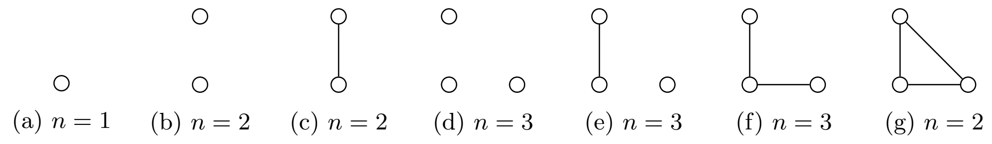


Figure 42: All graphs on 1, 2 and 3 vertices up to isomorphism

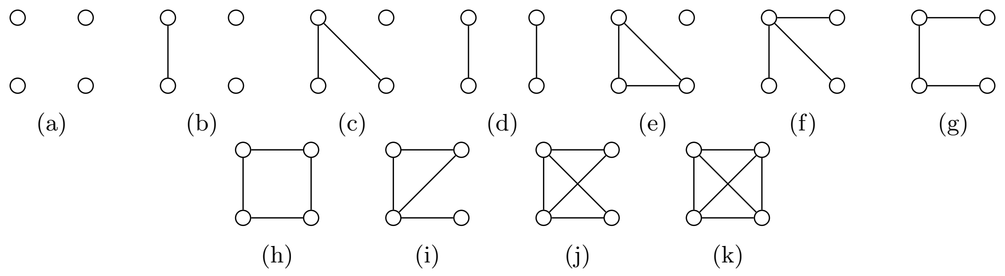


Figure 43: All graphs on 4 vertices up to isomorphism

Solution to Exercise 3.3.
See Figure 44.

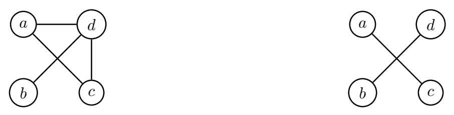
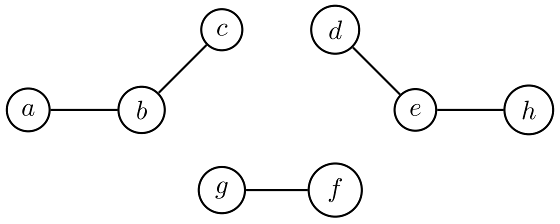


Figure 44

Solution to Exercise 3.8.
Representation as a drawing:



Set-theoretic representation:

$$\begin{aligned}
&G = (V, E),\\
&V = \{a, b, c, d, e, f, g, h\},\\
&E = \{\{a, b\}, \{b, c\}, \{d, e\}, \{f, g\}, \{e, h\}\}.
\end{aligned}$$

Adjacency matrix representation:

$$\begin{array}{c|cccccccc}
 & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} \\
\hline
\text{a} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{b} & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\text{c} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{d} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\text{e} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\text{f} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{g} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\text{h} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}$$

Solution to Exercise 3.13.
Reflexivity: There is a trivial path of length zero from any vertex to itself.

Transitivity: If there is a path from a to b and a path from b to c, then we can concatenate these paths to get a walk $a, x_1, x_1, \ldots, x_n, c$ from a to c. If there is a vertex that appears more than once in this walk, say, $x_i = x_j$ for $i < j$, then we can remove $x_{i+1}, \ldots, x_j$ and get a shorter walk from a to c. Since any walk has length at least 0, it will eventually be impossible to shorten the walk, so we get a walk from a to c without repeating vertices, so this is a path.

Symmetry: Given a path from a to b, we can reverse the order of the vertices in the path to get a path from b to a.

Solution to Exercise 3.14.

a. The maximal length of a path is six, which is realized for instance by c, d, e, f, g, b, a.

b. c, d, e, f, g, b, c.

c. Removing any of the edges in the cycle in (b) gives a spanning tree.

d. To get a spanning tree, one has to remove exactly one edge, and this edge must be in the cycle. There are six possibilities to do this, all of which give spanning trees, so there are six spanning trees.

Solution to Exercise 3.15.

a. The maximal length of a path is five, which is realized by b, c, d, e, f, g.

b. c, d, e, f, c. (All the possible solutions are "rotations" of this one.)

c. b, g and b, c, d, e, f, g.

Solution to Exercise 3.17.
Remember that the n-th power of the adjacency matrix encodes the walks of length n in the graph. Then, to obtain the number of walks of length exactly 3, we take the 3-rd power of the adjacency matrix A (see Figure 45) and sum up all its elements. The result, 38, is the double of the number of walks, since the walk between v and w is counted from v to w and from w to v.

To get the number of length at most 3, we sum all the powers of the adjacency matrix up to 3 $(A + A^2 + A^3)$ and then all the elements of the resulting matrix. The result, 64, is again the double of the wanted number.

The number of walks of length 3 from v_1 to v_3 can be read in the element $(3, 1)$ (or $(1, 3)$) of A^3, namely 1. Similarly, the number of walks of length 3 from v_3 to itself is 2.

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 \\ v_3 & 0 & 1 & 0 & 1 \\ v_4 & 0 & 1 & 1 & 0 \end{array}$$

(a) A

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 1 & 0 & 1 & 1 \\ v_2 & 0 & 3 & 1 & 1 \\ v_3 & 1 & 1 & 2 & 1 \\ v_4 & 1 & 1 & 1 & 2 \end{array}$$

(b) A^2

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 3 & 1 & 1 \\ v_2 & 3 & 2 & 4 & 4 \\ v_3 & 1 & 4 & 2 & 3 \\ v_4 & 1 & 4 & 3 & 2 \end{array}$$

(c) A^3

Figure 45

Solution to Exercise 3.18.*

Solution to Exercise 3.19.
a) See, for example, Figure 46

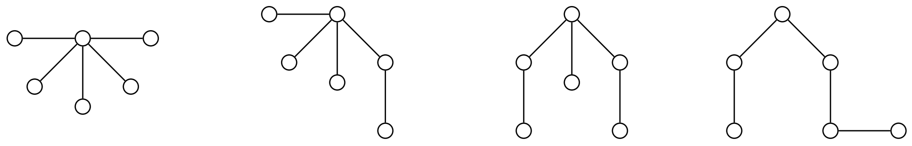


Figure 46

b)* Since in a tree on n vertices there are $n - 1$ edges, we know that the sum of all degrees of all vertices is $2n - 2$. A leaf is a vertex of degree 1. Therefore, if we had more than $n - 1$ leaves, i.e. n leaves, the sum of degrees would be n, a contradiction. If there was only one leaf, the total sum would be $2n - 1$, again a contradiction.

Solution to Exercise 3.20.
The diameter of the graph in Figure 16a is 5, realized, for example, between a and h. The closeness centrality of a, b, and c is:

$$C(a) = \frac{8-1}{1+2+3+4+5+3+2} = \frac{7}{20} = \frac{7}{20}$$
$$C(b) = \frac{8-1}{1+1++2+3+4+2+2} = \frac{7}{15} = \frac{7}{15}$$
$$C(c) = \frac{8-1}{2+1+1+2+3+3+2} = \frac{7}{14} = \frac{1}{2}$$

The diameter of the graph in Figure 14a is 4, realized, for example, by the path between 6 and 1. The closeness centrality of a, b, and c is:

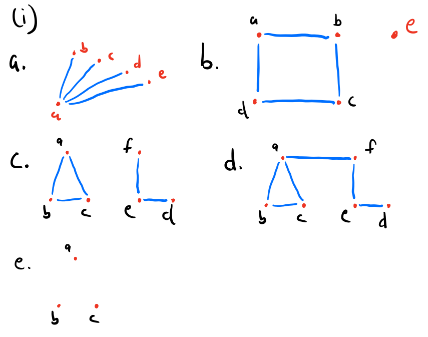
$$C(4) = \frac{6-1}{2+1+1+1+2} = \frac{6}{7}$$
$$C(6) = \frac{6-1}{4+3+3+2+1} = \frac{6}{13}$$

Solution to Exercise 3.21.
The graph (b) is not isomorphic to (a) nor to (c) since both (a) and (c) have a cycle of length 3 and one cycle of length 5, while (b) has two cycles of length 4. The graphs (a) and (c) are isomorphic, and the following map realizes the isomorphism:

$$a \mapsto 4$$
$$b \mapsto 5$$
$$c \mapsto 2$$
$$d \mapsto 3$$
$$e \mapsto 6$$
$$f \mapsto 1$$

Solution to Exercise 3.22.

(i)



(ii) For a and e , there is just one component G.
b. $G_1=(V_1,E)$, where $V_1 = V\setminus\{e\}$.
$G_2=(\{e\},\emptyset)$
c. $G_1=(V_1,E_1)$ $V_1=\{a,b,c\}$, $E_1=\{[a,b],[b,c],[a,c]\}$
$G_2=(V_2,E_2)$ $V_2=\{d,e,f\}$, $E_2=\{[d,e],[e,f]\}$
e. $G_1=(\{a\},\emptyset)$
$G_2=(\{b\},\emptyset)$
$G_3=(\{c\},\emptyset)$.

(iii) a. is a tree.
e. is a forest, not a tree
b, c, d. are not forests.

A.3 Metric Spaces

Solution to Exercise 4.1.
By the triangle inequality,

$$d(x', y) \leq d(x', x) + d(x, y)$$
$$d(x', y') \leq d(x', y) + d(y, y')$$

Plugging the first inequality into the second yields

$$d(x', y') \leq d(x, y) + d(x', x) + d(y, y') \leq d(x, y) + 2\epsilon.$$

By symmetry, the same argument shows that

$$d(x, y) \leq d(x', y') + 2\epsilon.$$

It follows that $|d(x, y) - d(x', y')| \leq 2\epsilon$, as desired.

Solution to Exercise 4.2.
Each vertex u has a trivial path to itself of length 0, so $d(u, u) = 0$. If u and v are different vertices, then any path from u to v has at least one edge. Since every edge is positively weighted, the length of the path must be positive, so $d(u, v) > 0$.

For any vertices u and v, there is a path from u to v of length $d(u, v)$. The opposite path goes from v to u and has the same length, so $d(v, u) \leq d(u, v)$. Switching u and v, we get $d(v, u) \geq d(u, v)$ and thus, $d(u, v) = d(v, u)$.

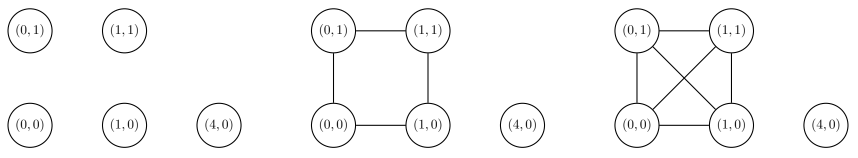
Given vertices u, v and w, there are a path from u to v of length $d(u, v)$ and a path from v to w of length $d(v, w)$. Composing these paths, we get a path from u to w of length $d(u, v) + d(v, w)$, so $d(u, w \leq d(u, v) + d(v, w)$ follows.

Solution to Exercise 4.3.
If G is disconnected, then we can pick vertices u and v in different connected components, in which case there is no path from u to v, so $d(u, v)$ is not defined. (It would make sense to let it be ∞, but a metric does not allow infinite distances.)

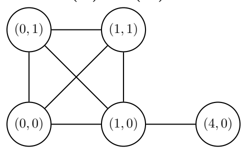
Solution to Exercise 4.4.
Properties 1 and 2 of a metric are clear. To prove the triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$, we use a case analysis:

- If $x = y = z$, then $d(x, z) = 0 \leq 0 + 0 = d(x, y) + d(y, z)$,
- If $x = y \neq z$, then $d(x, z) = 1 \leq 0 + 1 = d(x, y) + d(y, z)$,
- If $x \neq y = z$, then $d(x, z) = 1 \leq 1 + 0 = d(x, y) + d(y, z)$,
- If $x \neq y \neq z$, then $d(x, z) \leq 2 = d(x, y) + d(y, z)$.

Solution to Exercise 4.6.
See Figure 47. The graphs have 5, 2, 2 and 1 connected component, respectively. (In $N(S)_0$, each vertex induces a separate connected component. In $N(S)_1$ and $N(S)_2$, $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and $\{(4, 0)\}$ induce connected components, and in $N(S)_3$, the graph is connected.)



(a) $N(S)_0$ (b) $N(S)_1$ (c) $N(S)_2$



(d) $N(S)_3$

Figure 47: Solution to Exercise 4.6.

Solution to Exercise 4.7.
We have a metric if and only if $c > 0$. To explain, if $c < 0$, then the function d does not take values in $[0, \infty)$, as required. If $c = 0$, then property 1 of a metric is violated. If $c > 0$, then it is easily checked that the three properties of a metric are satisfied.

Solution to Exercise 4.8.

(i) For any set S, the metric of Example 4.8 has no midpoints,

(ii) The Euclidean distance on $\mathbb{R}^n$ has the property that an pair of points have exactly one midpoint,

(iii) The distances d_1 and d_∞ on $\mathbb{R}^n$ both have the property that some pair of points have exactly one midpoint. For example, consider $x = (0, 0)$ and $y = (1, 1)$. Then $(1, 0)$ and $(0, 1)$ are both midpoints of x and y for d_1. Next, let $w = (0, 0)$ and $z = (2, 1)$. Then $(1, 0)$ and $(1, 1)$ are both midpoints of w and z for d_∞.

Solution to Exercise 4.11.
See the solution to Exercise 4.1; this problem is a special case.

Solution to Exercise 4.12.
For each of the following functions $d : \mathbb{R} \times \mathbb{R} \to [0, \infty)$, say whether d is a metric. Briefly explain your reasoning.

a. $d(x, y) = |x| + |y|$. **Answer**: This is not a metric since the first property fails : if $x \neq 0$, then $d(x, x) = |x| + |x| \neq 0$.

b. $d(x, y) = \frac{|x|}{1+|y|}$. **Answer**: This is a not metric since the first property fails : if $x \neq 0$, then $d(x, x) = \frac{|x|}{1+|x|} \neq 0$.

c. $d(x, y) = |x - y|^3$. **Answer**: This is not a metric since the triangle inequality fails : $d(0, 2) = 8$, but $d(0, 1) + d(1, 2) = 1^3 + 1^3 = 2$.

Solution to Exercise 4.13.
For each of the following functions $d : \mathbb{R}^2 \times \mathbb{R}^2 \to [0, \infty)$, say whether d is a metric. Justify your answer.

a. $d(x, y) = |x_2 - y_2|$,
Answer: no, fails property 1, e.g., $d((1, 0), (2, 0)) = 0$.

b. $d(x, y) = |x_1 - y_1|^3 + |x_2 - y_2|^3$,
Answer: no, fails triangle inequality, e.g, take
$x = (0, 0)$, $y = (1, 0)$, $z = (2, 0)$. Then $d(x, z) = 8 \not\leq 2 = d(x, y) + d(y, z)$.

c. $d(x, y) = d_1(x, y)d_2(x, y)$,
Answer: no, fails triangle inequality, e.g, take
$x = (0, 0)$, $y = (1, 0)$, $z = (2, 0)$ as above. Then $d(x, z) = 4 \not\leq 2 = d(x, y) + d(y, z)$

d. $d(x, y) = 2|x_1 - y_1| + 3|x_2 - y_2|$.
Answer: Yes. $d(x, x) = 2|0| + 3|0| = 0$. Conversely, if $x \neq y$, then either $x_1 \neq y_1$ or $x_2 \neq y_2$, and so either the first or second term in the expression for $d(x, y)$ is positive. But the other terms is non-negative, so $d(x, y) > 0$. This proves Property 1. Property 2 is easily checked, as follows:

$$d(x, y) = 2|x_1 - y_1| + 3|x_2 - y_2| = 2|y_1 - x_1| + 3|y_2 - x_2| = d(y, x).$$

For property 3, we twice apply the triangle inequality for the Euclidean metric on $\mathbb{R}$:

$$\begin{aligned} d(x, z) &= 2|x_1 - z_1| + 3|x_2 - z_2| \\ &\leq 2(|x_1 - y_1| + |y_1 - z_1|) + 3(|x_2 - y_2| + |y_2 - z_2|) \\ &= (2|x_1 - y_1| + 3|x_2 - y_2|) + (2|y_1 - z_1| + 3|y_2 - z_2|) \\ &= d(x, y) + d(y, z). \end{aligned}$$

Solution to Exercise 4.15.
Compute the edit distance between the following pairs of sequences:

a. $x = AAAA$, $y = AA$ **Answer**: **2**; remove two. Since the lengths of X and Y differ by two, it is not possible to transform x into y with fewer than two elementary operations.

b. $x = AAAA$, $y = AAAT$ **Answer**: **1**; replace one.

c. $x = GTAA$, $y = TAAG$. **Answer**: **2**; remove one, add one. One can check directly that it's not possible to transform x into y with only one elementary operation.

d. $x = GGGG$, $y = TT$. **Answer**: **4**; remove two, replace two. A single elementary operation can remove at most one copy of G. Thus, since x has 4 G's and y has 0 G's, it is not possible to transform x into y with fewer than four elementary operations.

Solution to Exercise 4.16.
Let $d = d_1 = d_2 = d_\infty$. For any $c \in (0, \infty)$, cd (as defined in Exercise 4.7) is a metric on $\mathbb{R}$, and these are all different. We also have that letting $d(x, y) = c$ for all $x \neq y$ and $d(x, x)$ for all x defines a metric. These are two infinite families of different metrics, but there are also many more.

NB: As the hint says, the Manhattan, Euclidean and d_∞ distances are equal on $\mathbb{R}$, so they only count as one of the three requested metrics.

Solution to Exercise 4.17.
One example of a function $d\colon \mathbb{R} \times \mathbb{R} \to [0, \infty)$ satisfying all properties of a metric except the first is the zero function, i.e., $d(x, y) = 0$ for all $x, y \in \mathbb{R}$.

Solution to Exercise 4.18.
$W(f, g) = 1$.

Solution to Exercise 4.19.
$W(f, g) = 3$.

Solution to Exercise 4.20.*

We have to verify that $\sum_{j\in[m]} \Gamma_3(i,j) = f(i)$ and $\sum_{i\in[m]} \Gamma_3(i,j) = h(j)$.

$$\begin{aligned}
\sum_{j\in[m]} \Gamma_3(i,j) &= \sum_{j\in[m]} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} \\
&= \sum_{\substack{k\in[m]\\ g(k)\neq 0}} \frac{\Gamma_1(i,k)}{g(k)} \sum_{j\in[m]} \Gamma_2(k,j) \\
&= \sum_{\substack{k\in[m]\\ g(k)\neq 0}} \frac{\Gamma_1(i,k)}{g(k)} g(k) \\
&= \sum_{\substack{k\in[m]\\ g(k)\neq 0}} \Gamma_1(i,k) \\
&= f(i),
\end{aligned}$$

where we have used the fact that both f and g are transport plans. A similar calculation shows that $\sum_{i\in[m]} \Gamma_3(i,j) = h(j)$.

Solution to Exercise 4.21.*

Any transport plan Γ with cost 0 has $\Gamma(i,j) = 0$ for all $i \neq j$, which gives a valid transport plan from f to g if and only if $f = g$. (In this case, $\Gamma(i,i) = f(i)$ for all i.)

If Γ is a transport plan from f to g, the Γ' is a transport plan from g to f with the same cost, where we define $\Gamma'(i,j) = \Gamma(j,i)$ for all i and j. This gives symmetry.

The triangle inequality is more complicated. Given Γ_1 and Γ_2 as in Exercise 4.19, the cost of Γ_3 is bounded above by the sum of the costs of Γ_1 and Γ_2:

$$\begin{aligned}
\mathrm{cost}(\Gamma_3) &= \sum_{i=1}^{m}\sum_{j=1}^{m} \Gamma(i,j)|i-j| \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |i-j| \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} \\
&\leq \sum_{i=1}^{m}\sum_{j=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |i-k| \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} + \sum_{i=1}^{m}\sum_{j=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |k-j| \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} \\
&= \sum_{i=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |i-k| \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} + \sum_{j=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |k-j| \frac{\Gamma_1(i,k)\Gamma_2(k,j)}{g(k)} \\
&= \sum_{i=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |i-k|\Gamma_1(i,k) + \sum_{j=1}^{m} \sum_{\substack{k\in[m]\\ g(k)\neq 0}} |k-j|\Gamma_2(k,j) \\
&= \mathrm{cost}(\Gamma_1) + \mathrm{cost}(\Gamma_2)
\end{aligned}$$

The triangle inequality follows from this.

A.4 Clustering

Solution to Exercise 5.1.

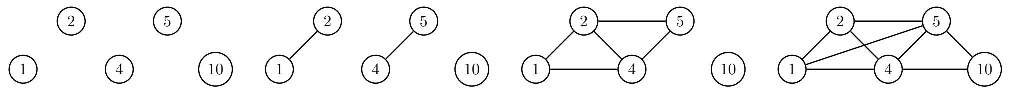
- For $k = 1$, we need a partition of X with one set. There is only one option: $\{X\}$.
- $k = 2$: there are a number of partitions of X with two sets, but there are three main candidates that have a small cost: $C_1 = \{\{1\}, \{2, 3, 4\}\}$, $C_2 = \{\{1, 2\}, \{3, 4\}\}$ and $C_3 = \{\{1, 2, 3\}, \{4\}\}$. (For instance, $\{\{1, 3\}, \{2, 4\}\}$ clearly has a higher cost than C_2.) The means of the clusters of C_1 are 1 and 3, which gives a cost of $(1-1)^2 + (2-3)^2 + (3-3)^2 + (4-3)^2 = 2$. By symmetry, the cost of C_3 is the same. The means of the clusters of C_2 are 1.5 and 3.5. This gives a cost of $4 \cdot (\frac{1}{2})^2 = 1$. Since C_2 has the smallest cost of all partitions of X, C_2 is the 2-clustering of X.
- $k = 3$: To choose a partition of X with three sets, we only need to pick which two elements are in the same set. Some easy calculations show that the cheapest options are $\{\{1\}, \{2\}, \{3, 4\}\}$, $\{\{1\}, \{4\}, \{2, 3\}\}$ and $\{\{3\}, \{4\}, \{1, 2\}\}$. These are all valid 3-clusterings of X.
- $k = 4$: There is only one partition of X with 4 sets: $\{\{1\}, \{2\}, \{3\}, \{4\}\}$.

Solution to Exercise 5.2.
The 2-clustering is $\{\{(0, 0), (1, 0)\}, \{(0, 2), (1, 2)\}\}$, with means $\mu_1 = (\frac{1}{2}, 0)$ and $\mu_2 = (\frac{1}{2}, 2)$ The cost is

$$\sum_{i=1}^{k} \sum_{x \in C_i} d_2(x, \mu_i)^2 = d_2((0,0),(\frac{1}{2},0))^2 + d_2((1,0),(\frac{1}{2},0))^2 + d_2((0,2),(\frac{1}{2},2))^2 + d_2((1,2),(\frac{1}{2},2))^2 = 4(\frac{1}{2})^2 = 1.$$

Solution to Exercise 5.3.
The two clusters in Proposition 5.7 cannot be separated by a line. By Figure 21, this means that they cannot be clusters in a k-means clustering.

Solution to Exercise 5.4.
We construct $N(X)_\epsilon$ for $\epsilon = 0, 1, 3, 7$. By taking connected components of these graphs, illustrated in Figure 48, we get $C_0 = \{\{1\}, \{2\}, \{4\}, \{5\}, \{10\}\}$, $C_1 = \{\{1, 2\}, \{4, 5\}, \{10\}\}$, $C_3 = \{\{1, 2, 4, 5\}, \{10\}\}$ and $C_4 = \{X\}$.



(a) $k = 0$ (b) $k = 1$ (c) $k = 3$ (d) $k = 7$

Figure 48

Solution to Exercise 5.5.
For $r = 1$, we have the clustering $\{\{(0, 0), (1, 0)\}, \{(0, 2), (1, 2)\}, \{(4, 6)\}\}$. For $r = 2$, we get $\{\{(0, 0), (1, 0), (0, 2), (1, 2)\}, \{(4, 6)\}\}$. To get one cluster, we need $(4, 6)$ to connect to the big cluster, which happens for $r \geq d_2((1, 2), (4, 6)) = 5$.

Solution to Exercise 5.5.
For $r = 1$, we have the clustering $\{\{(0, 0), (1, 0)\}, \{(0, 2), (1, 2)\}, \{(4, 6)\}\}$. For $r = 2$, we get $\{\{(0, 0), (1, 0), (0, 2), (1, 2)\}, \{(4, 6)\}\}$. To get one cluster, we need $(4, 6)$ to connect to the big cluster, which happens for $r \geq d_2((1, 2), (4, 6)) = 5$.

Solution to Exercise 5.6.
We can choose an r that is smaller than the shortest distance between a red and a blue point in Figure 21, but bigger than any gap in the red and blue clusters. (Formally, we choose r such that if we divide the blue (resp. red) cluster in two, then the smallest distance between the two parts is smaller than r.) Then $N(X)_r$ will have two connected components: one containing the blue points, and one containing the red points.

Solution to Exercise 5.7.
There are two words with distance 1: $AAAA$ and $BAAA$. Apart from this, every pair of words have distance at least 2. This gives the clustering $\{\{AAAA, BAAA\}, \{AA\}, \{CCCC\}, \{CDDD\}\}$ for $r = 1$.

$BAAA$ and AA have distance 2 (delete B and A from $BAAA$ to get AA), so their respective clusters merge. Both $CCCC$ and $CDDD$ have distance at least 3 from any other word in X, so we get the clustering $\{\{AAAA, BAAA, AA\}, \{CCCC\}, \{CDDD\}\}$ for $r = 2$.

$CCCC$ and $CDDD$ have distance 3 (turn the Ds of $CDDD$ into Cs in 3 steps), so they merge for $r = 3$. But they have distance 4 to every other word in X, so we get $\{\{AAAA, BAAA, AA\}, \{CCCC, CDDD\}\}$ for $r = 3$.

Solution to Exercise 5.8.
The dendrogram is depicted in Figure 49.

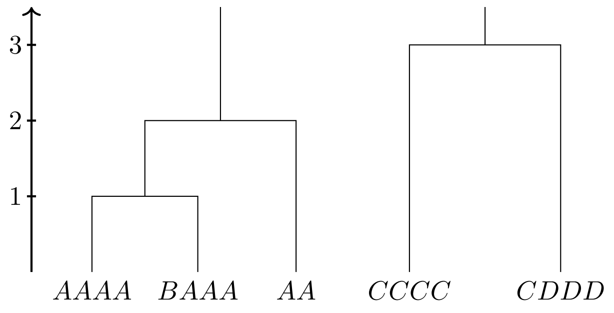


Figure 49

Solution to Exercise 5.12.
The dendrogram is depicted in Figure 50.

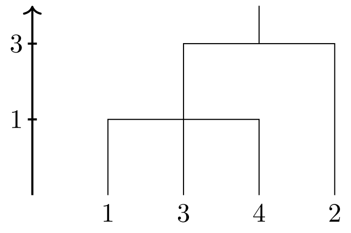
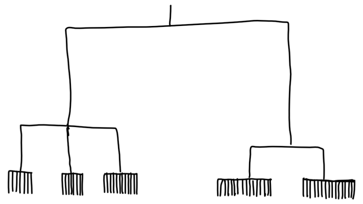


Figure 50

Solution to Exercise 5.13.
The dendrogram is depicted in Figure 51.

Solution to Exercise 5.14.
A solution is depicted below:



Solution to Exercise 5.15. (a) A possible solution is $\{-1, 0, 2, 4, 7\} \subseteq \mathbb{R}$.

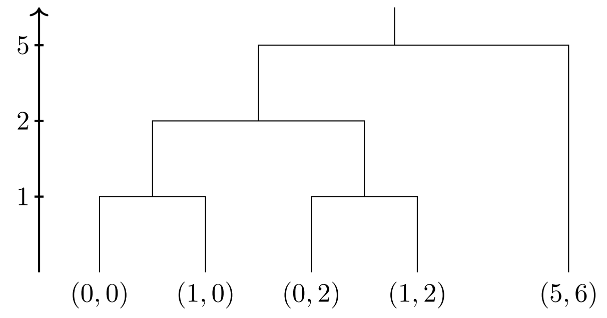


Figure 51

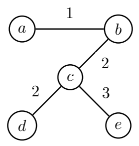


Figure 52

(b) A possible solution is depicted in Figure 52

As we can easily see by cutting the dendrogram at the appropriate high, $N(X)_1$ has 4 connected components and $N(X)_6$ has 1.

Solution to Exercise 5.20.
Computing all the mutual distances, we have that the minimum is attained by the pair of points $(0, 0)$ and $(-.1, .5)$, which are thus the first two merge at height .51. We now compute the average distances between the set $\{(0, 0), (-.1, .5)\}$ and all the other singletons. The minimum of these distances is .82, realized with the set $\{(0.7, 0)\}$. This is smaller than any other mutual distance. So, we have

$$P_{.82} = \{\{(0, 0), (-.1, .5), (.7, 0)\}, \{(0.3, 1.1)\}, \{(2.1, 1.9)\}, \{(1, 2)\}\}.$$

We now compute the average distances between the set $\{(0, 0), (-.1, .5), (.7, 0)\}$ and all the other singletons. The minimum of these distances is 1.01, realized with the set $\{(.3, 1.1)\}$. This is smaller than any other mutual distance. So, we have

$$P_{1.01} = \{\{(0, 0), (-.1, .5), (.7, 0), (0.3, 1.1)\}, \{(2.1, 1.9)\}, \{(1, 2)\}\}.$$

We now compute the average distances between the set $\{(0, 0), (-.1, .5), (.7, 0), (0.3, 1.1)\}$ and all the other singletons. The minimum of these distances is 1.82, which is bigger than the distance between $(2.9, 1.9)$ and $(1, 2)$, so these latter are now merged at 1.1 and we have

$$P_{1.1} = \{\{(0, 0), (-.1, .5), (.7, 0), (0.3, 1.1)\}, \{(2.1, 1.9), (1, 2)\}\}.$$

There are only two clusters left, so the only thing we need to do is compute their average distance to find out when they merge. This is 2.84. The dendrogram is depicted in Figure 53.

Solution to Exercise 5.22.
First, we compute all the mutual distances, for $d_{max} = d_\infty$ and d_1, between the points. In both cases, we have that $(-1, -1)$ and $(-1, -2)$, $(2, 3)$ and $(2, 4)$, and $(3, 3)$ and $(2, 3)$ have distance 1. However, $d_{max}((2, 3), (3, 3)) = 1$ and $d_1((2, 3), (3, 3)) = 2$. We did not cover this case in the lecture, but we can adopt the convention that the partition at 1 in both cases is

$$P_1 = \{\{(-1, -1), (-1, -2)\}, \{(2, 3), (3, 3), (2, 4)\}, \{(5, 5)\}\},$$

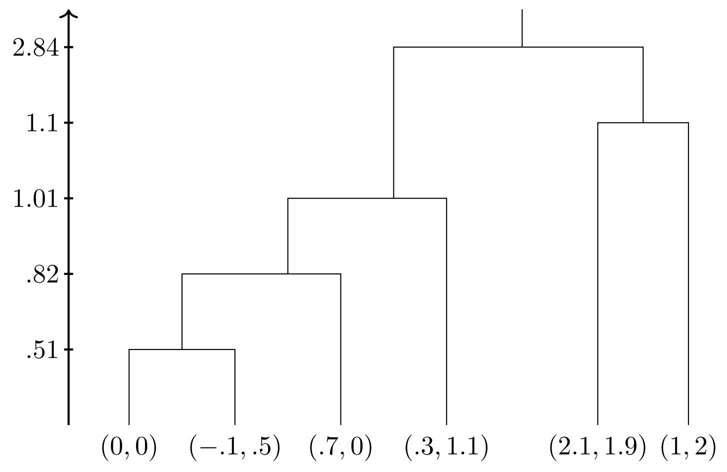


Figure 53

i.e., we merge the three points even if two of them have mutual distance bigger than 1. Next, we need to compute the average distances between the three sets, using d_{max} and d_1. We have

$$d_{max}(\{(2,3),(3,3),(2,4)\},\{(-1,-1),(-1,-2)\}) = 4.8 \qquad d_1(\{(2,3),(3,3),(2,4)\},\{(-1,-1),(-1,-2)\}\}) = 8.2$$
$$d_{max}(\{(2,3),(3,3),(2,4)\},\{(5,5)\}) = 2.7 \qquad d_1(\{(2,3),(3,3),(2,4)\},\{(5,5)\}) = 4.3$$
$$d_{max}(\{(-1,-1),(-1,-2)\},\{(5,5)\}) = 6.5 \qquad d_1(\{(-1,-1),(-1,-2)\},\{(5,5)\}) = 12.5$$

Thus, in both cases, we next merge $\{(2,3),(3,3),(2,4)\}$ and $\{(5,5)\}$, but at different values in the two distances. Next we have only one possibility, so we only need to compute when the merge happens.

$$d_{max}(\{(2,3),(3,3),(2,4),(5,5)\},\{(-1,-1),(-1,-2)\}) = 5.25$$
$$d_1(\{(2,3),(3,3),(2,4),(5,5)\},\{(-1,-1),(-1,-2)\}\}) = 9.25$$

The dendrograms are depicted in Figure 54

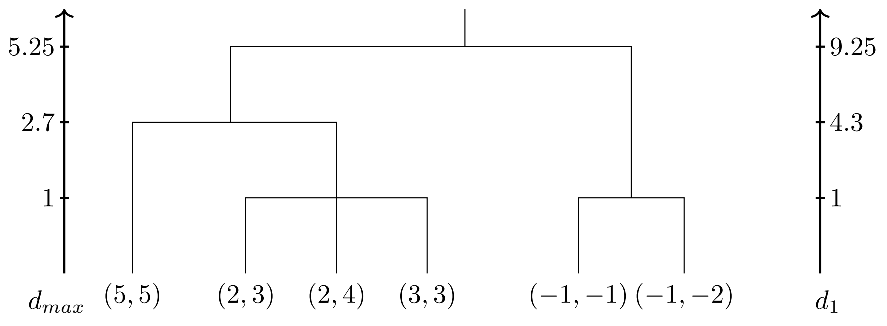


Figure 54

Solution to Exercise 5.23.
First, we put the inverse of the degree as density over each vertex. So, for the first graph $a, b \leftarrow \frac{1}{2}$, $d, e \leftarrow 1$, and $c \leftarrow \frac{1}{4}$, for the second graph $a, d, e \leftarrow 1$, $b \leftarrow \frac{1}{2}$, and $c \leftarrow \frac{1}{3}$, and the third graph

$b, d, e \leftarrow 1$, $a \leftarrow \frac{1}{2}$, and $c \leftarrow \frac{1}{3}$. The three totally ordered sets indexing the filtrations are:

$$T_1 = \{\frac{1}{4}, \frac{1}{2}, 1, 3, 4\}$$
$$T_2 = \{\frac{1}{3}, \frac{1}{2}, 1, 3, 4\}$$
$$T_3 = \{\frac{1}{3}, \frac{1}{2}, 1, 3, 4\}$$

The hierarchical partition of the first graph is:

$$\begin{aligned} P_{\frac{1}{4}} &= \{\{c\}\} \\ P_{\frac{1}{2}} &= \{\{c\}, \{a\}, \{b\}\} \\ P_1 &= \{\{c, d, e\}, \{a\}, \{b\}\} \\ P_3 &= \{\{c, d, e\}, \{a, b\}\} \\ P_4 &= \{\{c, d, e, a, b\}\} \end{aligned}$$

The hierarchical partition of the second graph is:

$$\begin{aligned} P_{\frac{1}{3}} &= \{\{c\}\} \\ P_{\frac{1}{2}} &= \{\{c\}, \{b\}\} \\ P_1 &= \{\{c, d, e\}, \{a\}, \{b\}\} \\ P_3 &= \{\{c, d, e\}, \{a, b\}\} \\ P_4 &= \{\{c, d, e, a, b\}\} \end{aligned}$$

The hierarchical partition of the third graph is:

$$\begin{aligned} P_{\frac{1}{3}} &= \{\{c\}\} \\ P_{\frac{1}{2}} &= \{\{c\}, \{a\}\} \\ P_1 &= \{\{c, d, e\}, \{a\}, \{b\}\} \\ P_3 &= \{\{c, d, e\}, \{a, b\}\} \\ P_4 &= \{\{c, d, e, a, b\}\} \end{aligned}$$

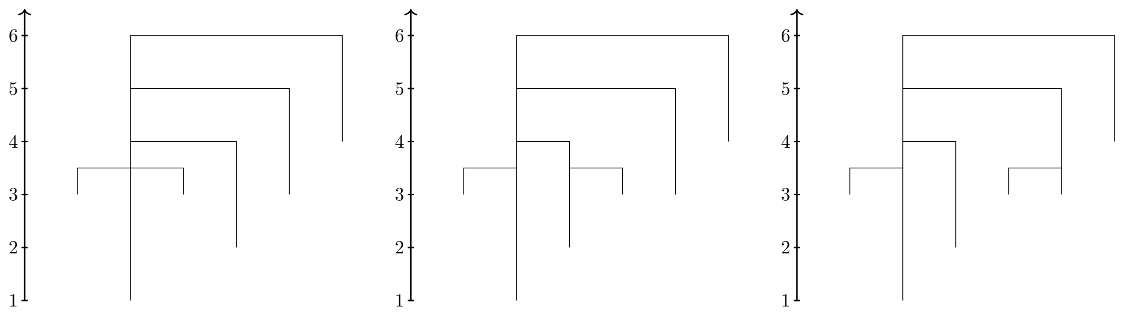
Solution to Exercise 5.29.
The barcode is $\{[1, \infty), [2, 5), [2, 5), [3, 4), [3, 4), [3, 4)\}$. Things to remember:

- The intervals are half-open with a closed left end and open right end.
- There is an infinite length interval that contains all the other intervals.
- The barcode is a multiset, which means that we may have to write several copies of the same interval.

Solution to Exercise 5.30.
A few possible solutions are illustrated in Figure 55.

Solution to Exercise 5.31.
At height 1, $\{0\}$ and $\{1\}$ merge, and so do $\{3\}$ and $\{4\}$, giving a partition of $\{\{0, 1\}, \{3, 4\}, \{7\}, \{9\}\}$. We have

$$\begin{aligned} \delta(\{0, 1\}, \{3, 4\}) &= \frac{1}{2 \cdot 2}(3 + 2 + 4 + 3) = 3 \\ \delta(\{3, 4\}, \{7\}) &= \frac{1}{2 \cdot 1}(3 + 2) = 3.5 \\ \delta(\{7\}, \{9\}) &= \frac{1}{1 \cdot 1}(2) = 2. \end{aligned}$$



(a) (b) (c)

Figure 55: A few solutions to Exercise 5.30.

Thus, we have to merge $\{7\}$ with $\{9\}$ at height 2. This gives a partition of $\{\{0,1\},\{3,4\},\{7,9\}\}$. We have $\delta(\{0,1\},\{3,4\}) < \delta(\{3,4\},\{7,9\})$, so we merge $\{0,1\}$ and $\{3,4\}$ at $\delta(\{0,1\},\{3,4\}) = 3$. We now have the partition $\{\{0,1,3,4\},\{7,9\}\}$. We have

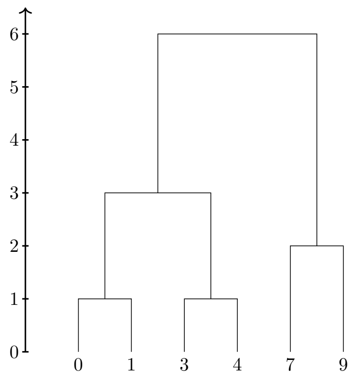
$$\delta(\{0,1,3,4\},\{7,9\}) = \frac{1}{4\cdot 2}(7+6+4+3+9+8+6+5) = 6,$$

which gives the last merging height. See Figure 56a for the dendrogram.

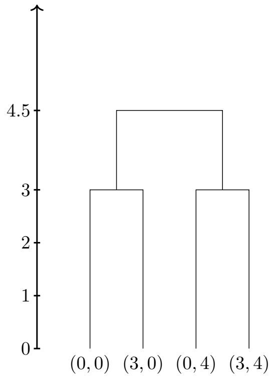
Solution to Exercise 5.32.
At height 3, $\{(0,0)\}$ and $\{(3,0)\}$ merge, and so do $\{(0,4)\}$ and $\{(3,4)\}$. Therefore, we have the partition $P_3 = \{\{(0,0),(3,0)\},\{(0,4),(3,4)\}\}$. We have

$$\delta(\{(0,0),(3,0)\},\{(0,4),(3,4)\}) = \frac{1}{2\cdot 2}(4+5+5+4) = 4.5,$$

which is the height at which the last two clusters merge. See Figure 56b for the dendrogram.



(a) Solution to Exercise 5.31.



(b) Solution to Exercise 5.32.

Figure 56

Solution to Exercise 5.33.
A possible solution is depicted in Figure 57.

Solution to Exercise 5.34.
Since there are two connected components, the solution is $\{[1,\infty),[1.5,5),[2,5),[2,\infty),[3,4.5)\}$.

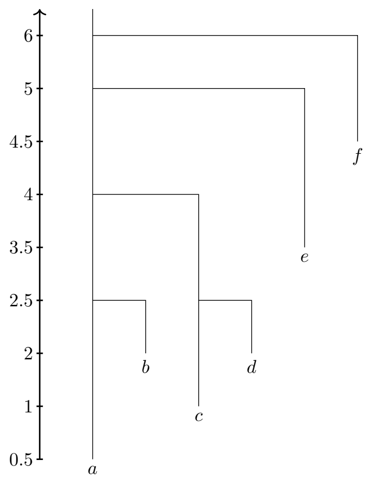


Figure 57: A possible solution to Exercise 5.33.

Solution to Exercise 5.35.
No, the bar $[1, 3)$ begins before the infinite bar $[2, \infty)$.

Solution to Exercise 5.36.
For $\tau = 0$, we do not remove any bar/cut, which means that the clustering is

$$\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}\}\,.$$

For $\tau = 1$, we remove all the bars/cut shorter than 1, and we merge their vertices to the bars they are attached to. Therefore, we obtain

$$\{\{a, b\}, \{c\}, \{d, e, f\}\}\,.$$

For $\tau = 2$, we obtain the same as in $\tau = 1$ since there are no bars of length smaller or equal 2 and greater than 1.

Solution to Exercise 5.37.
Using the dendrogram drawn in the solution of Exercise 5.33 and its labelling, we have

$$\{\{a, b, e, f\}, \{c, d\}\}\,.$$

A.5 Asymptotic complexity

Solution to Exercise 6.2.
$\frac{1}{x}$ is a small-o of both $\log x$ and $\log(x^5)$, but $\log x \neq o(\log(x^5))$ and $\log(x^5) \neq o(\log x)$. (In fact, this follows from $\log x = \Theta(\log(x^5))$, which holds because $\log(x^5) = 5 \log x$.) Then $\log x, \log(x^5) = o(x^{5/2}), o(2x^{5/2})$, and $2x^{5/2}, x^{5/2} = o(x^3)$, but $x^{5/2} \neq o(2x^{5/2})$ and $2x^{5/2} \neq o(x^{5/2})$. Finally, we have $x^3 = o(e^{x-1})$ and $e^{x-1} = o(e^{2x})$. Thus, from fastest-growing to slowest-growing, we have

- e^{2x}
- e^{x-1}
- x^3
- $x^{\frac{5}{2}}, 2x^{\frac{5}{2}}$
- $\log(x^5), \log x$

- $\frac{1}{x}$

Solution to Exercise 6.3.
A simple solution is to take $f_r(x) = x^r$, if $r \in \mathbb{N}$. Another (trivial) solution, since every function is a big-O of itself, is to take $f_r(x) = x$ for all $r \in \mathbb{R}$.

Solution to Exercise 6.4.
No, because to compute the dot product we need to access to all the n elements of the vectors, which means that we need at least n iteration in general (i.e. without any hypothesis on the sparsity of the vectors). Since $n \neq o(n)$, there cannot be an algorithm with complexity $o(n)$ (but there is an algorithm with complexity $O(n)$).

Solution to Exercise 6.5.
The asymptotic complexity is $O(n)$, where n is the length of the input list L. This complexity is achieved in the for loop in the first line, since inside the loop there are only elementary operations but the loop iterates over the input list.

The complexity would not change with the additional else statement as it consists only of an elementary operation.

Solution to Exercise 6.6.
The asymptotic complexity is $O(nm)$. This complexity is achieved in the two nested for loops in the first and second line, since inside the inner loop there are only elementary operations but the loops iterate over n and m.

The complexity would not change switching the loops since values commutes in $\mathbb{R}$.

Solution to Exercise 6.7.
We have to show that

$$\lim_{x\to\infty} \frac{3xe^{2x+1}}{x\log(x)e^{2x}} = 0.$$

The fraction is equal to

$$\frac{3e \cdot e^{2x}}{\log(x)e^{2x}} = \frac{3e}{\log(x)},$$

which goes to zero as x goes to infinity, since $3e$ is a constant and $\log(x)$ goes to infinity as $x \to \infty$.

Solution to Exercise 6.8.
Since $-1 \leq \cos x \leq 1$ for all x, we have $0 \leq (\cos x)^2 \leq 1$ and thus $4 \cdot 1 \leq 4 + (\cos x)^2 \leq 5 \cdot 1$ for all x, so $4 + (\cos x)^2 = \Theta(1)$.

Solution to Exercise 6.9.
Suppose $e^{2x} = O(e^x)$. Then there is a constant c such that $e^{2x} \leq ce^x$ for all x. But we can pick a (large) x such that $c < e^x$. Then $ce^x < e^x \cdot e^x = e^{2x}$, which contradicts our assumption, so we cannot have $e^{2x} = O(e^x)$.

Solution to Exercise 6.10.
(a)

$$\begin{aligned}
&[5],[9],[0],[3],[7],[6],[5] \\
\to&[5,9],[0,3],[6,7],[5] \\
\to&[0,3,5,9],[5,6,7] \\
\to&[0,3,5,5,6,7,9]
\end{aligned}$$

(b)

$$\begin{aligned}
&[2],[8],[13],[17],[7],[8],[1],[16],[12],[6],[1],[6]\\
\to&[2,8],[13,17],[7,8],[1,16],[6,12],[1,6]\\
\to&[2,8,13,17],[1,7,8,16],[1,6,6,12]\\
\to&[1,2,7,8,8,13,16,17],[1,6,6,12]\\
\to&[1,1,2,6,6,7,8,8,12,13,16,17]
\end{aligned}$$

Solution to Exercise 6.11.
We start with n lists, where n is the number of elements in the list, and for each iteration we go from having k to $\lceil \frac{k}{2} \rceil$ numbers (where $\lceil \cdot \rceil$ means that we are rounding up to the nearest integer). This gives a total of $\lceil \log_2 n \rceil$ iterations.

(a) $\lceil \log_2 4 \rceil = 2$. (The number of lists is 4, then 2, then 1, so two steps/iterations.)

(b) $\lceil \log_2 64 \rceil = 6$. (The number of lists is $64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$.)

(c) $\lceil \log_2 100 \rceil = 7$. ($100 \to 50 \to 25 \to 13 \to 7 \to 4 \to 2 \to 1$.)

(d) $\lceil \log_2 100 \rceil = 10$. ($1000 \to 500 \to 250 \to 125 \to 63 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$.)