I.1 Connected Components

A theme that goes through this entire book is the transfer back and forth between discrete and continuous models of reality. In this first section, we compare the notion of connectedness in discrete graphs and continuous spaces.

Simple graphs. The abstract concept of a *graph* is a pair $G = (V, E)$ consisting of set of *vertices*, V, and a set of *edges*, E, each a pair of vertices. We draw the vertices as points or little circles and edges as line segments or curves connecting the points. Crossings between curves are allowed as they only represent the edges. The graph is *simple* if the edge set is a subset of the set of unordered pairs, $E \subseteq \binom{V}{2}$. For $n = \operatorname{card} V$ vertices, the number of edges is $m = \operatorname{card} E \leq \binom{n}{2}$. Every graph with n vertices is a subgraph of the *complete graph*, K_n, that has n vertices and all possible $m = \binom{n}{2}$ edges; see Figure I.1.

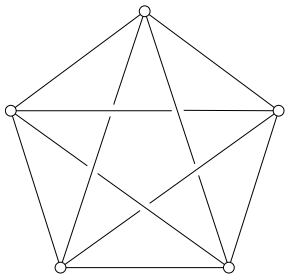

Figure I.1: The complete graph with five vertices, K_5. It has ten edges which form five crossings if drawn as sides and diagonals of a convex pentagon.

Connectivity. In a simple graph, a *path* from $u \in V$ to $v \in V$ can be described by a sequence of vertices, $u = u_0, u_1, u_2, \ldots, u_k = v$, where we have an edge from u_i to u_{i+1} for each $0 \leq i \leq k-1$. Vertices can repeat allowing the path to cross itself or fold onto itself.

DEFINITION A. A simple graph G is *connected* if there is a path between every pair of vertices.

The smallest connected graphs are the *trees*, which are characterized by having a unique (non-self-intersecting) path between every pair of vertices; see Figure I.2. Removing any one edge disconnects the tree. Alternatively, we can

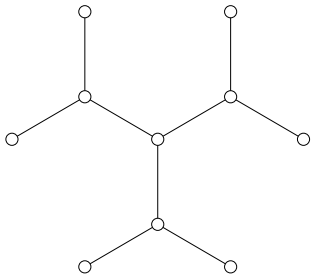

Figure I.2: A tree with ten vertices.

characterize a connected graph by the impossibility to cut it in two.

DEFINITION B. A *separation* is a non-trivial partition of the vertices, $V = U \cup W$, such that no edge connects a vertex in U with a vertex in W. A simple graph G is *connected* if it has no separation.

Topological spaces. A topology of a point set is a collection of subsets that implicitly defines which points are near each other without specifying the distance between them. Think of an abstraction of Euclidean space in which neighborhoods are open balls around points.

DEFINITION. A *basis* of a topology on a point set $\mathbb{X}$ is a collection $\mathcal{B} \subseteq 2^{\mathbb{X}}$ of *basis elements* such that

(i) each $x \in \mathbb{X}$ is contained in at least one $B \in \mathcal{B}$;

(ii) $x \in B_1 \cap B_2$ implies there is a third basis element with $x \in B_3 \subseteq B_1 \cap B_2$.

The *topology* $\mathcal{U}$ *generated* by $\mathcal{B}$ consists of all sets $U \subseteq \mathbb{X}$ for which $x \in U$ implies there is a basis element $x \in B \subseteq U$. The pair $(\mathbb{X},\mathcal{U})$ is a *topological space*. The sets in $\mathcal{U}$ are its *open sets*.

Note that $\emptyset, \mathbb{X} \in \mathcal{U}$, the union of open sets is an open set, and the finite intersection of open sets is an open set. These are the requirements for $(\mathbb{X},\mathcal{U})$ to be a topological space. We often call $\mathbb{X}$ a topological space, tacitly assuming that $\mathcal{U}$ is understood.

As an example consider $\mathbb{X} = \mathbb{R}$ and let $\mathcal{B}$ be the collection of open intervals. This gives the usual topology of the real line. Note that the intersection of the

intervals $(-\frac{1}{n}, +\frac{1}{n})$, for the infinitely many integers $n \geq 1$, is the point 0. This is not an open set which illustrates the need for restricting the intersections to finite collections.

We often encounter sets inside other sets, $\mathbb{Y} \subseteq \mathbb{X}$, and in these cases we can borrow the topology of the latter for the former. Specifically, if $\mathcal{U}$ is a topology of $\mathbb{X}$ then the collection of sets $\mathbb{Y} \cap U$, for $U \in \mathcal{U}$, is the *induced topology* of $\mathbb{Y}$. As an example consider the closed interval $[0, 1] \subseteq \mathbb{R}$. We have seen that the open intervals form the basis for a topology of the real line. The intersections of open intervals with $[0, 1]$ form the basis of the induced topology of the closed interval.

Paths and path-connectedness. To formulate Definition A for topological spaces, we need the notion of a path, which is a special continuous function. A function $f : \mathbb{Y} \to \mathbb{X}$ between topological spaces is *continuous* if the preimage of every open set is open. A *path* is a continuous function $\gamma : [0, 1] \to \mathbb{X}$. It *connects* the point $\gamma(0)$ to the point $\gamma(1)$ in $\mathbb{X}$. We allow self-intersections, calling γ a simple path if it is injective. Figure I.3 illustrates the definitions.

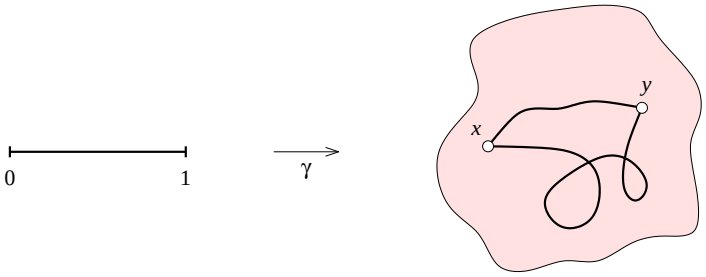

Figure I.3: A simple and a non-simple path from $x = \gamma(0)$ to $y = \gamma(1)$.

DEFINITION. A topological space $\mathbb{X}$ is *path-connected* if every pair of points is connected by a path.

Connectedness. There is also a counterpart of Definition B for topological spaces.

DEFINITION. A *separation* of a topological space $\mathbb{X}$ is a partition $\mathbb{X} = U \dot{\cup} W$ into two non-empty, open subsets. $\mathbb{X}$ is *connected* if it has no separation.

It turns out connectedness is strictly weaker than path-connectedness, although for most spaces we encounter they are the same. An example of a space that is connected but not path-connected is the comb with a single tooth deleted; see Figure I.4. It is constructed by gluing vertical closed intervals to a horizontal

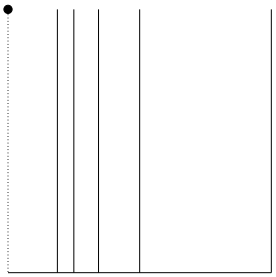

Figure I.4: The comb is $[0,1] \times 0$ union $\frac{1}{n} \times [0,1]$, for all positive integers n, union $0 \times [0,1]$. From this we delete $0 \times (0,1)$.

closed interval and finally deleting the interior of the leftmost vertical interval. To construct a topology, we take the collection of open disks as the basis of a topology on $\mathbb{R}^2$ and we use the induced topology for the comb. This space is connected because it is the union of a path-connected set and a limit point. It is not path-connected.

Disjoint set systems. We return to graphs and look at the algorithmic question of deciding connectedness. We can do this by adding edges one at a time and maintaining the components as sets in a disjoint set system. Formulated as an abstract data type, we have three operations manipulating a system $\mathcal{S}$:

ADD(i):	$\mathcal{S} = \mathcal{S} \dot\cup \{i\}$;
FIND(i):	`return` $A \in \mathcal{S}$ with $i \in A$;
UNION(A, B):	$\mathcal{S} = \mathcal{S} - \{A, B\} \dot\cup \{A \dot\cup B\}$.

We need the FIND operation to prevent forming the union of a set with itself. Each union operation reduces the number of sets in the system by one. Starting with n elements, it therefore takes $n-1$ union operations to get to a single set. Equivalently, every tree with n vertices has $m = n - 1$ edges.

A standard data structure implementing a disjoint set system stores each set as a tree, with each element pointing to its parent, as shown in Figure I.5. It is convenient to assume each root points to itself. We implement FIND with

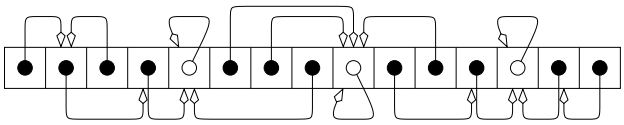

Figure I.5: Three trees representing the same number of disjoint sets stored in a linear array. From left to right, the sets have size six, four, and five.

path-compression, which means each time we traverse a path up to the root, we connect each node along the path directly to the root.

```
int FIND(i)
  if i.parent ≠ i then return i.parent = FIND(i.parent) endif;
  return i.
```

We execute a UNION operation whenever we find elements in different sets. We merge two trees in a weighted manner, connecting the root of the smaller tree to the root of the larger tree.

```
A = FIND(i);  B = FIND(j);
if A ≠ B then UNION(A, B) endif.

void UNION(A, B)
  if A.size > B.size then A ↔ B endif;
  A.parent = B.
```

Using path-compression together with weighted merging implies that the time needed to execute m UNION, FIND, and ADD operations is at most $O(m\alpha(n))$, where $\alpha(n)$ is the notoriously slowly growing inverse of the Ackermann function.

Size functions. Suppose we have a process in time that translates into a sequence of ADD and UNION operations, each with a time-stamp. Given two moments in time $t_1 < t_2$, the two-parameter *size function* is the number of components at time t_1 that are still disjoint at time t_2, which we denote as $s(t_1, t_2)$. Draw the process as a tree in which each point represents a component. Decompose the tree into paths, ending the shorter and continuing the longer path at every junction, like in Figure I.6 on the left. Then $s(t_1, t_2)$ is equal to the number of paths that cross the entire slab between time t_1 and time t_2.

We can get the same information from a simpler diagram. Draw each path as a point in the plane with ordinate the time of birth and abscissa the time of

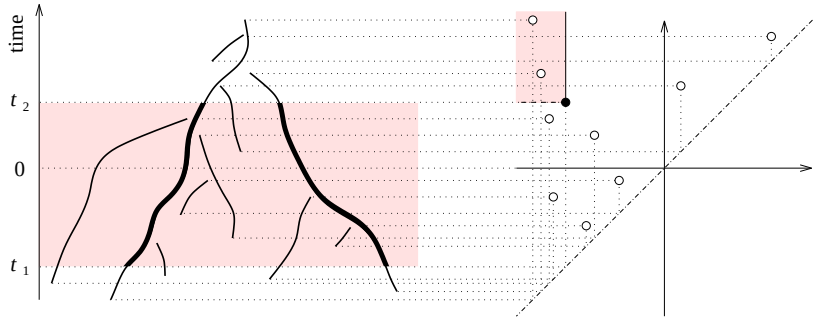

Figure I.6: Left: history of components drawn as a collection of paths forming a tree. Of the five components at time t_1, two are still disjoint at time t_2, hence $s(t_1, t_2) = 2$. Right: Diagram in which each point represents a path of the tree. A point lies in the upper left quadrant iff the corresponding path crosses the entire slab.

death, as in Figure I.6 on the right. Then $s(t_1, t_2)$ is the number of points in the upper left quadrant of the point (t_1, t_2).

Bibliographic notes. Graphs are ubiquitous objects and appear in most disciplines. Within mathematics, the theory of graphs is considered part of combinatorics. There are many good books on the subject, including the one by Tutte [4]. The basic topological notions of connectedness are treated in books on point-set or general topology, including the text by Munkres [2]. The computational problem of maintaining a system of disjoint sets is a classic topic in the field of algorithms and discussed in detail in the text by Tarjan [3]. Size functions have been introduced relatively recently by Frosini and Landi [1]. We mention them in this first section because they are elementary and will later lead naturally to the more involved idea of topological persistence.

[1] P. Frosini and C. Landi. Size functions and morphological transformations. *Acta Applicandae Mathematicae* **49** (1997), 85–104

[2] J. R. Munkres. *Topology. A First Course.* Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[3] R. E. Tarjan. *Data Structures and Network Algorithms.* SIAM, Philadelphia, Pennsylvania, 1983.

[4] W. T. Tutte. *Graph Theory.* Addison-Wesley, Reading, Massachusetts, 1984.