Chapter 1

Fundamentals

1.1 Models of Computation

When designing algorithms, one has to agree on a model of computation according to which the algorithms are executed. There are various models to choose from, but when it comes to geometry, a Turing machine type model, for instance, would be rather inconvenient to represent and manipulate the frequent encounters of real numbers. Remember that even elementary geometric operations—such as taking the center of a circle defined by three points or computing the length of a circular arc—could quickly leave the realms of rational and even algebraic numbers.

Therefore, other models of computation are more prominent in the area of geometric algorithms and data structures. In this course we will be mostly concerned with two models: the *Real RAM* and the *algebraic computation/decision tree* model. The former is rather convenient when designing algorithms, as it abstracts away the aforementioned representation issues by simply *assuming* that it can be done. The latter typically appears in the context of lower bounds, that is, in proofs that solving a given problem requires at least certain amount of resource (as a function of the input size and possibly other parameters).

Let us look into these models in more detail.

Real RAM Model. RAM stands for random access machine, that is a machine whose memory cells are indexed by integers, and any specified cell can be accessed in constant time. "Real" means that each cell can store a real number. Any single arithmetic operation (addition, subtraction, multiplication, division, and k-th root, for small constant k) or comparison can be computed in constant time. [1]

This is a quite powerful model of computation, as a single real number in principle can encode an arbitrary amount of information. On the positive side, it allows to abstract

[1] In addition, sometimes also logarithms, other analytic functions, indirect addressing (integral), or floor and ceiling are used. As adding some of these operations makes the model more powerful, it is usually specified and emphasized explicitly when an algorithm uses them.

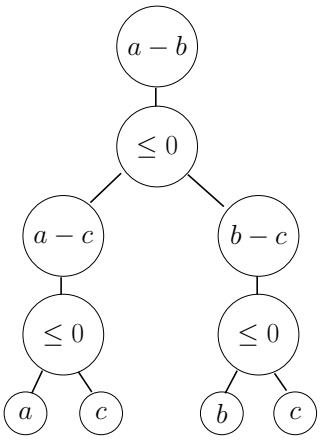
from the lowlands of numeric and algebraic computation and to concentrate on the algorithmic core from a combinatorial point of view.

But there are also downsides. First, the model is somewhat unrealistic, and it poses a challenge to efficiently implement an algorithm designed for it on an actual computer. With bounded memory there is no way to represent general real numbers explicitly, and operations using a symbolic representation can hardly be considered constant time. Therefore we have to ensure that we do not abuse the power of this model. For instance, we may want to restrict the numbers that are manipulated by any single arithmetic operation to be some fixed polynomial in the numbers that appear in the input.

Second, it is difficult if not impossible to derive reasonable lower bounds in the real RAM model. So when interested in lower bounds, it is convenient to use a different, less powerful model of computation. One such model is the computation tree model, which encompasses and explicitly represents all possible execution paths of an algorithm.

Algebraic Computation Trees (Ben-Or [1]). A model is as a rooted binary tree, where each node has at most two children. The computation starts at the root and proceeds down to leaves.

- Every node v with one child has an associated operation in $+, -, *, /, \sqrt{}, \ldots$. The operands of this operation are constant input values, or among v's ancestors in the tree.
- Every node v with two children is associated with a branching of the form > 0, $\geqslant 0$, or $= 0$. The branch is with respect to the result of v's parent node. If the expression yields `true`, the computation continues with the left child of v; otherwise, it continues with the right child of v.
- Every leaf contains the result of the computation.



The term *decision tree* is used if all of the final results (leaves) are either `true` or `false`. If every branch is based on a linear function in the input values, we face a *linear decision tree*. Analogously one can define, say, quadratic decision trees.

The complexity of a computation or decision tree is the maximum number of nodes among all root-to-leaf paths. It is well known that $\Omega(n \log n)$ comparisons are required to sort n numbers. But also for some problems that appear easier than sorting at first glance, the same lower bound holds. Consider, for instance, the following problem.

Element Uniqueness

Input: $\{x_1, \ldots, x_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$.

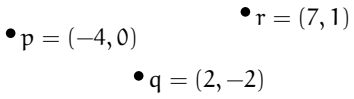
Output: Is $x_i = x_j$, for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$?

Ben-Or [1] has shown that any algebraic decision tree to solve Element Uniqueness for n elements has complexity $\Omega(n \log n)$.

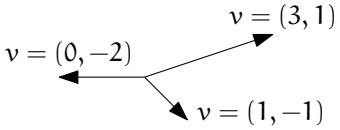
1.2 Basic Geometric Objects

We will mostly be concerned with the d-dimensional Euclidean space $\mathbb{R}^d$ for small $d \in \mathbb{N}$; typically $d = 2$ or $d = 3$. The basic objects of interest in $\mathbb{R}^d$ are the following.

Points. A point $p \in \mathbb{R}^d$ is typically described by its d Cartesian coordinates $p = (x_1, \ldots, x_d)$.

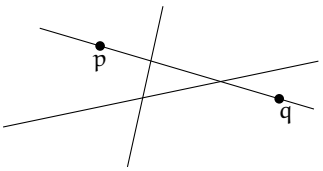


Vectors. A vector $v \in \mathbb{R}^d$ is typically described by its d Cartesian coordinates $v = (x_1, \ldots, x_d)$. Its length (in Euclidean metric) is denoted as $\|v\| := \sqrt{\sum_{i=1}^{d} {x_i}^2}$. If v has unit length, we also call it a direction.



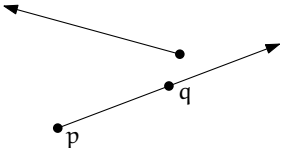
What is the difference between a point and a vector? Mathematically they are both elements of Euclidean space $\mathbb{R}^d$, hence they are the same. The different terms are used to indicate how we think of such an element in a given context. We think of a point as a location in space (a "dot"), while we think of a vector as a translation in space (an "arrow" starting from the origin). The point view is dominant in geometry. But since every point can be understood as a vector (the arrow from the origin to the dot), we can seamlessly apply vector space operations (addition, scalar multiplication) to points; and the resulting vector can be cast as a point again (the dot at the arrowhead). There are a number of sources that insist differentiating points and vectors, and they are right when it comes to how we interpret them. But when it comes to what they actually are, there is no need to make a difference.

Lines. A line is a one-dimensional affine subspace in $\mathbb{R}^d$. It can be described by two distinct points p and q as the point set $\{p + \lambda(q - p) : \lambda \in \mathbb{R}\}$.



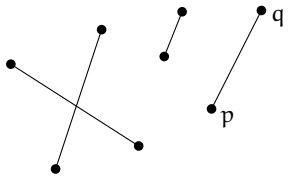
While any pair of distinct points defines a unique line, a line in $\mathbb{R}^2$ contains infinitely many points and so it may happen that a collection of three or more points lie on a line. Such a collection of points is termed *collinear*[2].

Rays. If we split a line at a point and only look into one direction from the point, then we obtain a ray. It can be described by two distinct points p and q as the point set $\{p + \lambda(q - p) : \lambda \geqslant 0\}$. The *orientation* of a ray is the vector $q - p$.



[2]Not *colinear*, which refers to a notion in the theory of coalgebras.

Line segments. A line segment is, as its name suggests, the segment between two points p, q on a line. It can be described as the point set $\{p + \lambda(q - p) : 0 \leqslant \lambda \leqslant 1\}$. We will also denote this line segment by $\overline{pq}$. Depending on the context we may allow or disallow *degenerate* line segments consisting of a single point only ($p = q$ in the above equation).



Hyperplanes and halfspaces. A hyperplane h is a $(d-1)$-dimensional affine subspace in $\mathbb{R}^d$. It can be described algebraically by $d + 1$ coefficients $h_1, \ldots, h_{d+1} \in \mathbb{R}$ as the point set $\{(x_1, \ldots, x_d) \in \mathbb{R}^d : \sum_{i=1}^{d} h_i x_i = h_{d+1}\}$. Usually, we require at least one of $h_1, \ldots, h_d$ to be nonzero. Otherwise, the equation is satisfied by either all points (if $h_{d+1} = 0$) or no point (if $h_{d+1} \neq 0$), which we call a *degenerate* hyperplane. Degeneracy is useful in some contexts, and we will explicitly say so where we allow them. If we change "$=$" in the definition to "$\geqslant$", the obtained object is called a halfspace (or halfplane in $\mathbb{R}^2$).

Spheres and balls. A sphere is the set of all points that are equidistant to a fixed point. It can be described by its center $c \in \mathbb{R}^d$ and radius $r \in \mathbb{R}$ as the point set $\{x \in \mathbb{R}^d : \|x-c\| = r\}$. Likewise, the ball of radius r around c is the point set $\{x \in \mathbb{R}^d : \|x-c\| \leqslant r\}$. In $\mathbb{R}^2$, spheres and balls are called circles and disks, respectively.

1.3 Topology

In this section we review some basic concepts and notation from set-theoretic topology, on a level of what you also encounter in courses on real analysis typically. These concepts arise here and there, for instance, when we formalize intuitive objects such as "curves" and "polygons". They are also indispensible when we study certain abstract objects such as convex sets.

A set $P \subseteq \mathbb{R}^d$ is *bounded*, if it is contained in some ball $B_r := \{x \in \mathbb{R}^d : \|x\| \leqslant r\}$ of radius $r > 0$ around the origin.

A point $p \in \mathbb{R}^d$ is *interior* to $P \subseteq \mathbb{R}^d$, if for some $\varepsilon > 0$, there exists a ball $B_\varepsilon(p) = \{x \in \mathbb{R}^d : \|x - p\| \leqslant \varepsilon\}$ around it that is completely contained in P. A set is *open* if all of its points are interior; and it is *closed* if its complement is open. Beware that a set can be both open and closed (e.g. $\mathbb{R}^d$), or neither open nor closed (e.g. the interval $(0, 1]$ in one-dimension).

Finally, a set is *compact* if it is both bounded and closed. An important fact from analysis states that every continuous function from a compact set $P \subseteq \mathbb{R}^d$ to $\mathbb{R}$ attains its minimum/maximum at some point $p \in P$.

Exercise 1.1. *Determine for each of the following sets whether they are open or closed in* $\mathbb{R}^2$. *a)* $B_1(0)$ *b)* $\{(1, 0)\}$ *c)* $\mathbb{R}^2$ *d)* $\mathbb{R}^2 \backslash \mathbb{Z}^2$ *e)* $\mathbb{R}^2 \backslash \mathbb{Q}^2$ *f)* $\{(x, y) : x \in \mathbb{R}, y \geqslant 0\}$

Exercise 1.2. *Show that the union of countably many open sets in $\mathbb{R}^d$ is open. Show that the union of a finite number of closed sets in $\mathbb{R}^d$ is closed. (For the curious reader: These are two of the axioms of an abstract topology. So here we show that the Euclidean space is a topology.) What follows for intersections of open and closed sets? Finally, show that the union of countably many closed sets in $\mathbb{R}^d$ is not necessarily closed.*

The *boundary* ∂P of $P \subseteq \mathbb{R}^d$ consists of all points in $\mathbb{R}^d$ (not necessarily in P) that are neither interior to P nor to $\mathbb{R}^d \setminus P$. In other words, $p \in \partial P$ if every ball $B_\varepsilon(p)$ intersects both P and $\mathbb{R}^d \setminus P$.

Sometimes one wants to approximate a set $P \subseteq \mathbb{R}^d$ by an open/closed set. In the former scenario we can resort to its *interior* P° formed by all points interior to P. It is not hard to see that $P^\circ = P \setminus \partial P$. Similarly, in the latter scenario one can use its *closure* $\overline{P} = P \cup \partial P$.

Exercise 1.3. *Show that for any $P \subseteq \mathbb{R}^d$ the interior P° is open, and the closure $\overline{P}$ is closed. (Why is there something to show to begin with?)*

What is the interior of a lower-dimensional object living in a higher dimensional space, such as a line segment in $\mathbb{R}^2$ or a triangle in $\mathbb{R}^3$? The answer is $\emptyset$, because the balls considered in the definition are higher-dimensional creatures which will always contain points from the "outer space". To overcome this undesirable artefact, we can use the notion of a *relative interior*, denoted by $\operatorname{relint}(S)$. It refers to the interior of S where all the "balls" in the definition are restricted to live in the smallest affine subspace that contains S.

For instance, the smallest affine subspace that contains the line segment $\overline{pq}$ in $\mathbb{R}^2$ is the line through p, q. So all the "balls" considered by the relative interior will be intervals of that line, thus $\operatorname{relint}(\overline{pq}) = \overline{pq} \setminus \{p, q\}$. Similarly, the smallest affine subspace that contains a triangle in $\mathbb{R}^3$ is a plane. Hence the relative interior is just the interior of the triangle, considered as a two-dimensional object.

1.4 Graphs

Next we review some basic definitions and properties of graphs. For more details and proofs, refer to any standard textbook on graph theory [2, 3, 5].

A (simple undirected) graph $G = (V, E)$ is defined on a set V of *vertices* whose pairwise relations are captured by the set $E \subseteq \binom{V}{2}$ of edges. Unless stated otherwise, V is always finite. Two vertices u, v are *adjacent* if $\{u, v\} \in E$, in which case both vertices are *incident* to the edge $\{u, v\}$. To avoid clutter we often omit brackets and write uv for edge $\{u, v\}$.

For a vertex $v \in V$, its *neighborhood* in G, denoted $N_G(v)$, consists of all vertices from G that are adjacent to v. Similarly, for a set $W \subset V$ of vertices its neighborhood $N_G(W)$ is defined as $\bigcup_{w \in W} N_G(w)$. The *degree* $\deg_G(v)$ of a vertex $v \in V$ is the size of its

neighborhood, that is, the number of edges from E incident to v. The subscript is often omitted if the graph under consideration is clear from the context.

Lemma 1.4 (Handshaking Lemma). *In any graph* $G = (V, E)$ *we have*

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Two graphs $G = (V, E)$ and $H = (U, F)$ are *isomorphic*, denoted $G \simeq H$, if there is a bijection $\phi : V \to U$ such that $\{u, v\} \in E \iff \{\phi(u), \phi(v)\} \in F$. Such a bijection ϕ is called an *isomorphism* between G and H. The structure of isomorphic graphs is identical and often we do not distinguish between them when looking at them as graphs.

For a graph G denote by $V(G)$ the set of vertices and by $E(G)$ the set of edges. A graph $H = (U, F)$ is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$. In case that $U = V$ the graph H is a *spanning* subgraph of G. For a set $U \subseteq V$ of vertices denote by $G[U]$ the *induced subgraph* of G on U, that is, the graph $(U, E \cap \binom{U}{2})$. For $F \subseteq E$ denote $G \setminus F := (V, E \setminus F)$. Similarly, for $U \subseteq V$ denote $G \setminus U := G[V \setminus U]$. In particular, for a vertex or edge $x \in V \cup E$ we write $G \setminus x$ for $G \setminus \{x\}$. The *union* of two graphs $G = (V, E)$ and $H = (U, F)$ is the graph $G \cup H := (V \cup U, E \cup F)$.

For an edge $e = uv \in E$ the graph G/e is obtained from $G \setminus \{u, v\}$ by adding a new vertex w with $N_{G/e}(w) := (N_G(u) \cup N_G(v)) \setminus \{u, v\}$. This process is called *contraction* of e in G. Similarly, for a set $F \subseteq E$ of edges the graph G/F is obtained from G by contracting all edges from F (the order in which the edges from F are contracted does not matter).

Graph traversals. A *walk* in G is a sequence $W = (v_1, \ldots, v_k)$, $k \in \mathbb{N}$, of vertices such that v_i and v_{i+1} are adjacent in G, for all $1 \leqslant i < k$. The vertices v_1 and v_k are referred to as the walk's *endpoints*, and the other vertices its *interior*. A walk with endpoints v_1 and v_k is sometimes called a walk *between* v_1 and v_k. If the endpoints coincide (namely $v_1 = v_k$), then the walk is *closed*; otherwise it is *open*. For a walk W denote by $V(W)$ its set of vertices and by $E(W)$ its set of edges (that is, pairs of consecutive vertices along W). We say that W *visits* its vertices and edges.

A walk that uses each edge of G at most once is called a *trail*. A closed walk that visits each edge (hence also each vertex) at least once is called a *tour* of G. An *Euler tour* is both a trail and a tour of G, that is, it visits each edge of G exactly once. A graph that contains an Euler tour is termed *Eulerian*.

If the vertices $v_1, \ldots, v_k$ of a closed walk W are pairwise distinct except for $v_1 = v_k$, then W is a *cycle* of size $k - 1$. If the vertices $v_1, \ldots, v_k$ of a walk W are pairwise distinct, then W is a *path* of size k. A *Hamilton cycle (path)* is a cycle (path) that visits every vertex of G. A graph that contains a Hamilton cycle is *Hamiltonian*.

Two trails are *edge-disjoint* if they do not share any edge. Two paths are called *internally vertex-disjoint* if they do not share any vertices (except for potential common endpoints). For two vertices $s, t \in V$ any path with endpoints s and t is called an (s, t)-*path* or a path *between* s and t.

Connectivity. Define an equivalence relation "$\sim$" on V by setting $a \sim b$ if and only if there is a path between a and b in G. The equivalence classes with respect to "$\sim$" are called *components* of G. A graph G is *connected* if it has only one component, and *disconnected* otherwise.

A set $C \subset V$ of vertices in a connected graph $G = (V, E)$ is a *cut-set* of G if $G \setminus C$ is disconnected. A graph is *k-connected*, for a positive integer k, if $|V| \geqslant k + 1$ and every cut-set has at least k vertices. Similarly a graph $G = (V, E)$ is *k-edge-connected*, if $G \setminus F$ is connected, for any set $F \subseteq E$ of at most $k - 1$ edges. Connectivity and cut-sets are related via the following well-known theorem.

Theorem 1.5 (Menger [4]). *For any two nonadjacent vertices u, v of a graph $G = (V, E)$, the minimum size of a cut-set that disconnects u and v is the same as the maximum number of pairwise internally vertex-disjoint paths between u and v.*

Specific families of graphs. A graph with all potential edges present, that is $(V, \binom{V}{2})$, is called a *clique*. Up to isomorphism there is only one clique on n vertices; it is referred to as the *complete graph* K_n, for $n \in \mathbb{N}$. At the other extreme, the *empty graph* $\overline{K_n}$ consists of n isolated vertices, so no edge is present. A set U of vertices in a graph G is *independent* if $G[U]$ is an empty graph. A graph whose vertex set can be partitioned into two independent sets is *bipartite*. An equivalent characterization states that a graph is bipartite if and only if it does not contain any odd cycle. The bipartite graphs with a maximum number of edges (unique up to isomorphism) are the *complete bipartite graphs* $K_{m,n}$, for $m, n \in \mathbb{N}$. They consist of two disjoint independent sets of size m and n, respectively, and all mn edges in between.

A *forest* is a graph that is *acyclic*, that is, it does not contain any cycle. A connected forest is called *tree* and its *leaves* are the vertices of degree one. Every connected graph contains a spanning subgraph which is a tree—a so-called *spanning tree*. Beyond the definition given above, there are several equivalent characterizations of trees.

Theorem 1.6. *The following statements for a graph G are equivalent.*

(1) G is a tree (that is, it is connected and acyclic).

(2) G is a connected graph with n vertices and $n - 1$ edges.

(3) G is an acyclic graph with n vertices and $n - 1$ edges.

(4) Any two vertices in G are connected by a unique path.

(5) G is minimally connected, that is, G is connected but removal of any single edge yields a disconnected graph.

(6) G is maximally acyclic, that is, G is acyclic but adding any single edge creates a cycle.

Directed graphs. In a directed graph or, short, *digraph* $D = (V, E)$ the set E consists of ordered pairs of vertices, that is, $E \subseteq V^2$. The elements of E are referred to as *arcs*. To avoid clutter we often omit brackets and write uv for an arc (u, v). An arc $uv \in E$ is said to be directed from its *source* u to its *target* v. For $uv \in E$ we also say "there is an arc from u to v in D". Usually, we consider *loop-free* graphs, that is, arcs of the type vv, for some $v \in V$, are not allowed.

The *in-degree* $\deg_D^-(v) := |\{(u, v) | uv \in E\}|$ of a vertex $v \in V$ is the number of *incoming* arcs at v. Similarly, the *out-degree* $\deg_D^+(v) := |\{(v, u) \mid vu \in E\}|$ of a vertex $v \in V$ is the number of *outgoing* arcs at v. Again the subscript is often omitted when the graph under consideration is clear from the context.

From any undirected graph G one can obtain a digraph on the same vertex set by specifying a direction for each edge of G. Each of these $2^{|E(G)|}$ different digraphs is called an *orientation* of G. Similarly every digraph $D = (V, E)$ has an *underlying* undirected graph $G = (V, \{\{u, v\} \mid (u, v) \in E \text{ or } (v, u) \in E\})$. Hence most of the terminology for undirected graphs carries over to digraphs.

A *directed walk* in a digraph D is a sequence $W = (v_1, \ldots, v_k)$, for some $k \in \mathbb{N}$, of vertices such that there is an arc from v_i to v_{i+1} in D, for all $1 \leqslant i < k$. In the same way we define *directed trails*, *directed tours*, *directed paths*, and *directed cycles*.

Multigraphs. Sometimes we also consider *multigraphs*, where each edge may have multiple copies. Unless forbidden explictly, a multigraph may contain loops. Just as simple graphs/digraphs, multigraphs may be undirected or directed, and also most of the other basic notions for graphs discussed above naturally generalize to multigraphs.

References

[1] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[2] John Adrian Bondy and U. S. R. Murty, *Graph Theory*, vol. 244 of *Graduate texts in Mathematics*, Springer, London, 2008.

[3] Reinhard Diestel, *Graph Theory*, vol. 173 of *Graduate texts in Mathematics*, Springer, Heidelberg, 5th edn., 2016.

[4] Karl Menger, Zur allgemeinen Kurventheorie. *Fund. Math.*, 10/1, (1927), 96–115.

[5] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, 2nd edn., 2001.