Chapter 1

Fundamentals

1.1 Models of Computation

When designing algorithms, one has to agree on a model of computation according to which these algorithms can be executed. There are various such models, but when it comes to geometry some are more convenient to work with than others. Even using very elementary geometric operations—such as taking the center of a circle defined by three points or computing the length of a given circular arc—the realms of rational and even algebraic numbers are quickly left behind. Representing the resulting real numbers/coordinates would be a rather painful task in, for instance, a Turing machine type model of computation.

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, because it sort of abstracts from the aforementioned representation issues by simply *assuming* that it can be done. The latter model typically appears in the context of lower bounds, that is, proofs that certain problems cannot be solved more efficiently than some function depending on the problem size (and possibly some other parameters).

So let us see what these models are in more detail.

Real RAM Model. A memory cell stores a real number (that is what the "Real" stands for)[1]. Any single arithmetic operation (addition, subtraction, multiplication, division, and k-th root, for small constant k) or comparison can be computed in constant time.[2] This is a quite powerful (and somewhat unrealistic) model of computation, as a single real number in principle can encode an arbitrary amount of information. Therefore we

[1] RAM stands for random access machine, meaning that every memory cell can be accessed in constant time. Not like, say, a list where one always has to start from the first element.

[2] 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.

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 bounded by some fixed polynomial in the numbers that appear in the input.

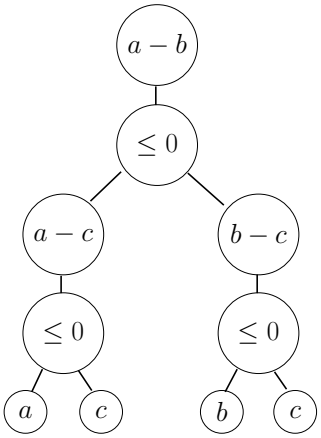
On the positive side, the real RAM model allows to abstract from the lowlands of numeric and algebraic computation and to concentrate on the algorithmic core from a combinatorial point of view.

But there are also downsides to using such a powerful model. In particular, it may be a challenge to efficiently implement a geometric algorithm designed for the real RAM 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.

When interested in lower bounds, it is convenient to use a model of computation that encompasses and represents explicitly all possible execution paths of an algorithm. This is what the following model is about.

Algebraic Computation Trees (Ben-Or [1]). A computation is regarded as a binary tree.

- The leaves contain the (possible) results of the computation.
- Every node v with one child has an operation of the form $+, -, *, /, \sqrt{}, \ldots$ associated to it. The operands of this operation are constant input values, or among the ancestors of v in the tree.
- Every node v with two children has associated to it 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.



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 vertices along any root-to-leaf path. 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: $\{x_1, \ldots, x_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$.

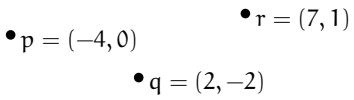
Output: Is $x_i = x_j$, for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$?

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

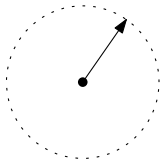
1.2 Basic Geometric Objects

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.

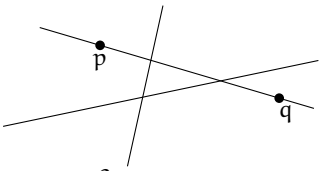
Points. A point p, typically described by its d Cartesian coordinates $p = (x_1, \ldots, x_d)$.



Directions. A vector $v \in \mathbb{S}^{d-1}$ (the $(d-1)$-dimensional unit sphere), typically described by its d Cartesian coordinates $v = (x_1, \ldots, x_d)$, with $\|v\| = \sqrt{\sum_{i=1}^{d} {x_i}^2} = 1$.

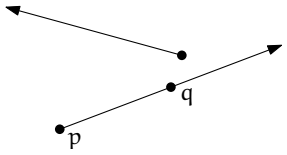


Lines. A line is a one-dimensional affine subspace. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in \mathbb{R}$.

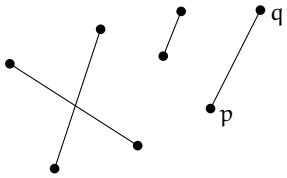


While any pair of distinct points defines a unique line, a line in $\mathbb{R}^2$ contains infinitely many points and so it may happen that a collection of three or more points lie on a line. Such a collection of points is termed *collinear*[3].

Rays. If we remove a single point from a line and take the closure of one of the connected components, then we obtain a ray. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \geqslant 0$. The *orientation* of a ray is the direction $(q - p)/\|q - p\|$.



Line segment. A line segment is a compact connected subset of a line. It can be described by two points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in [0, 1]$. We will denote the line segment through p and q 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. A hyperplane $\mathcal{H}$ is a $(d-1)$-dimensional affine subspace. It can be described algebraically by $d + 1$ coefficients $\lambda_1, \ldots, \lambda_{d+1} \in \mathbb{R}$, where $\|(\lambda_1, \ldots, \lambda_{d+1})\| = 1$, as the set of all points $(x_1, \ldots, x_d)$ that satisfy the linear equation $\mathcal{H} : \sum_{i=1}^{d} \lambda_i x_i = \lambda_{d+1}$.

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

If the above equation is converted into an inequality, we obtain the algebraic description of a *halfspace* (in $\mathbb{R}^2$: halfplane).

Spheres and balls. A sphere is the set of all points that are equidistant to a fixed point. It can be described by a point c (center) and a number $\rho \in \mathbb{R}$ (radius) as the set of all points p that satisfy $\|p - c\| = \rho$. The *ball* of radius ρ around p consists of all points p that satisfy $\|p - c\| \leqslant \rho$.

1.3 Graphs

In this section 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].

An (undirected) graph $G = (V, E)$ is defined on a set V of *vertices*. Unless explicitly stated otherwise, V is always finite. Vertices are associated to each other through *edges* which are collected in the set $E \subseteq \binom{V}{2}$. The two vertices defining an edge are *adjacent* to each other and *incident* to the edge.

For a vertex $v \in V$, denote by $N_G(v)$ the *neighborhood* of v in G, that is, the set of vertices from G that are adjacent to v. Similarly, for a set $W \subset V$ of vertices define $N_G(W) := \bigcup_{w \in W} N_G(w)$. 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 when it is clear which graph it refers to.

Lemma 1.1 (Handshaking Lemma). *In any graph* $G = (V, E)$ *we have* $\sum_{v \in V} \deg(v) = 2|E|$.

Two graphs $G = (V, E)$ and $H = (U, W)$ are *isomorphic* if there is a bijection $\phi : V \to U$ such that $\{u, v\} \in E \iff \{\phi(u), \phi(v)\} \in W$. 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 $W \subseteq V$ of vertices denote by $G[W]$ the *induced subgraph* of W in G, that is, the graph $(W, E \cap \binom{W}{2})$. For $F \subseteq E$ let $G \setminus F := (V, E \setminus F)$. Similarly, for $W \subseteq V$ let $G \setminus W := G[V \setminus W]$. 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 $H = (W, F)$ is the graph $G \cup H := (V \cup W, E \cup F)$.

For an edge $e = \{u, v\} \in E$ 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*, the other vertices are called *interior*. A walk with endpoints v_1 and v_k is sometimes referred to as a *walk between* v_1 and v_k. For a walk W denote by $V(W)$ its set of vertices and by $E(W)$ its set of edges (pairs of vertices adjacent along W). We say that W *visits* the vertices and edges in $V(W) \cup E(W)$. A walk for which both endpoints coincide, that is, $v_1 = v_k$, is called *closed*. Otherwise the walk is *open*.

If a walk uses each edge of G at most once, it is a *trail*. A closed walk that visits each edge and 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 possibly 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 and their number is denoted by $\omega(G)$. A graph G is *connected* if $\omega(G) = 1$ 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 there is no cut-set of size less than k. Similarly a graph $G = (V, E)$ is *k-edge-connected*, if $G \setminus F$ is connected, for any set $F \subseteq E$ of less than k edges. Connectivity and cut-sets are related via the following well-known theorem.

Theorem 1.2 (Menger [4]). *For any two nonadjacent vertices u, v of a graph $G = (V, E)$, the size of a minimum cut 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 a maximum number of edges, 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, $n \in \mathbb{N}$. At the other extreme, the *empty graph* $\overline{K_n}$ consists of n isolated vertices that are not connected by any edge. 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 at most 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 1.3. *The following statements for a graph* G *are equivalent.*

(1) G *is a tree (i.e., 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 (edge-)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*. An arc $(u, v) \in E$ is said to be directed from its *source* u to its *target* v. For $(u, v) \in E$ 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 (v, v), for some $v \in V$, are not allowed.

The *in-degree* $\deg_D^-(v) := |\{(u, v) \mid (u, v) \in E\}|$ of a vertex $v \in V$ is the number of *incoming* arcs at v. Similarly, the *out-degree* $\deg_D^+(v) := |\{(v, u) \mid (v, u) \in E\}|$ 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 paths*, *directed cycles*, and *directed tours*.

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-Verlag, London, 2008.

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

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

[5] Douglas B. West, *An introduction to graph theory*, Prentice Hall, Upper Saddle River, NJ, 2nd edn., 2001.