1

Tools

This chapter presents the algorithmic and combinatorial framework in which are developed the following chapters. It first specifies the concepts and notation used to work on strings, languages, and automata. The rest is mainly devoted to the introduction of chosen data structures for implementing automata, to the presentation of combinatorial results, and to the design of elementary pattern matching techniques. This organization is based on the observation that efficient algorithms for text processing rely on one or the other of these aspects.

Section 1.2 provides some combinatorial properties of strings that occur in numerous correctness proofs of algorithms or in their performance evaluation. They are mainly periodicity results.

The formalism for the description of algorithms is presented in Section 1.3, which is especially centered on the type of algorithm presented in the book, and introduces some standard objects related to queues and automata processing.

Section 1.4 details several methods to implement automata in memory, these techniques contribute, in particular, to results of Chapters 2, 5, and 6.

The first algorithms for locating strings in texts are presented in Section 1.5. The sliding window mechanism, the notions of search automaton and of bit vectors that are described in this section are also used and improved in Chapters 2, 3, and 8, in particular.

Section 1.6 is the algorithmic jewel of the chapter. It presents two fundamental algorithmic methods used for text processing. They are used to compute the border table and the prefix table of a string that constitute two essential tables for string processing. They synthesize a part of the combinatorial properties of a string. Their utilization and adaptation is considered in Chapters 2 and 3, and also punctually come back in other chapters.

Finally, we can note that intuition for combinatorial properties or algorithms sometimes relies on figures whose style is introduced in this chapter and kept thereafter.

1.1 Strings and automata

In this section, we introduce notation on strings, languages, and automata.

Alphabet and strings

An ***alphabet*** is a finite nonempty set whose elements are called ***letters***. A ***string*** on an alphabet A is a finite sequence of elements of A. The zero letter sequence is called the ***empty string*** and is denoted by ε. For the sake of simplification, delimiters, and separators usually employed in sequence notation are removed and a string is written as the simple juxtaposition of the letters that compose it. Thus, ε, a, b, and baba are strings on any alphabet that contains the two letters a and b. The set of all the strings on the alphabet A is denoted by A^*, and the set of all the strings on the alphabet A except the empty string ε is denoted by A^+.

The ***length*** of a string x is defined as the length of the sequence associated with the string x and is denoted by $|x|$. We denote by $x[i]$, for $i = 0, 1, \ldots, |x| - 1$, the letter at index i of x with the convention that indices begin with 0. When $x \neq \varepsilon$, we say more specifically that each index $i = 0, 1, \ldots, |x| - 1$ is a ***position*** **on** x. It follows that the ith letter of x is the letter at position $i - 1$ on x and that:

$$x = x[0]x[1] \ldots x[|x| - 1].$$

Thus an elementary definition of the identity between any two strings x and y is:

$$x = y$$

if and only if

$$|x| = |y| \text{ and } x[i] = y[i] \text{ for } i = 0, 1, \ldots, |x| - 1.$$

The set of letters that occur in the string x is denoted by alph(x). For instance, if $x =$ abaaab, we have $|x| = 6$ and alph(x) = {a, b}.

The ***product*** – we also say the ***concatenation*** – of two strings x and y is the string composed of the letters of x followed by the letters of y. It is denoted by xy or also $x \cdot y$ to show the decomposition of the resulting string. The neutral element for the product is ε. For every string x and every natural number n, we define the nth ***power*** of the string x, denoted by x^n, by $x^0 = \varepsilon$ and $x^k = x^{k-1}x$ for $k = 1, 2, \ldots, n$. We denote respectively by zy^{-1} and $x^{-1}z$ the strings x and y when $z = xy$. The ***reverse*** – or ***mirror image*** – of the string x is the string $x^\sim$ defined by:

$$x^\sim = x[|x| - 1]x[|x| - 2] \ldots x[0].$$

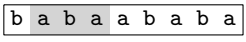

Figure 1.1. An occurrence of string aba in string babaababa at (left) position 1.

A string x is a ***factor*** of a string y if there exist two strings u and v such that $y = uxv$. When $u = \varepsilon$, x is a ***prefix*** of y; and when $v = \varepsilon$, x is a ***suffix*** of y. The string x is a ***subsequence***[1] of y if there exist $|x| + 1$ strings $w_0, w_1, \ldots, w_{|x|}$ such that $y = w_0x[0]w_1x[1]\ldots x[|x| - 1]w_{|x|}$; in a less formal way, x is a string obtained from y by deleting $|y| - |x|$ letters. A factor or a subsequence x of a string y is ***proper*** if $x \neq y$. We denote respectively by $x \preceq_{\text{fact}} y$, $x \prec_{\text{fact}} y$, $x \preceq_{\text{pref}} y$, $x \prec_{\text{pref}} y$, $x \preceq_{\text{suff}} y$, $x \prec_{\text{suff}} y$, $x \preceq_{\text{sseq}} y$, and $x \prec_{\text{sseq}} y$ when x is a factor, a proper factor, a prefix, a proper prefix, a suffix, a proper suffix, a subsequence, and a proper subsequence of y. One can verify that $\preceq_{\text{fact}}$, $\preceq_{\text{pref}}$, $\preceq_{\text{suff}}$, and $\preceq_{\text{sseq}}$ are orderings on A^*.

The ***lexicographic ordering***, denoted by $\leq$, is an ordering on strings induced by an ordering on the letters and denoted by the same symbol. It is defined as follows. For $x, y \in A^*$, $x \leq y$ if and only if, either $x \preceq_{\text{pref}} y$, or x and y can be decomposed as $x = uav$ and $y = ubw$ with $u, v, w \in A^*$, $a, b \in A$, and $a < b$. Thus, ababb $<$ abba $<$ abbaab assuming a $<$ b.

Let x be a nonempty string and y be a string, we say that there is an ***occurrence*** of x in y, or, more simply, that x ***occurs*** **in** y, when x is a factor of y. Every occurrence, or every appearance, of x can be characterized by a position on y. Thus we say that an occurrence of x ***starts*** at the ***left position*** i on y when $y[i\,..\,i + |x| - 1] = x$ (see Figure 1.1). It is sometimes more suitable to consider the ***right position*** $i + |x| - 1$ at which this occurrence ***ends***. For instance, the left and right positions where the string $x =$ aba occurs in the string $y =$ babaababa are:

i	0	1	2	3	4	5	6	7	8
$y[i]$	b	a	b	a	a	b	a	b	a
left positions		1			4		6		
right positions				3			6		8

The ***position of the first occurrence*** $pos(x)$ of x in y is the minimal (left) position at which starts the occurrence of x in yA^*. With the notation on the languages recalled thereafter, we have:

$$pos(x) = \min\{|u| : uxA^* \cap yA^* \neq \emptyset\}.$$

[1] We avoid the common use of "subword" because it has two definitions in literature: one of them is factor and the other one is subsequence.

The square bracket notation for the letters of strings is extended to factors. We define the factor $x[i\,..\,j]$ of the string x by:

$$x[i\,..\,j] = x[i]x[i+1]\ldots x[j]$$

for all integers i and j satisfying $0 \le i \le |x|, -1 \le j \le |x|-1$, and $i \le j+1$. When $i = j+1$, the string $x[i\,..\,j]$ is the empty string.

Languages

Any subset of A^* is a ***language*** on the alphabet A. The product defined on strings is extended to languages as follows:

$$XY = X \cdot Y = \{xy : (x, y) \in X \times Y\}$$

for every languages X and Y. We extend as well the notion of power as follows $X^0 = \{\varepsilon\}$ and $X^k = X^{k-1}X$ for $k \ge 1$. The ***star*** of X is the language:

$$X^* = \bigcup_{n \ge 0} X^n.$$

We denote by X^+ the language defined by

$$X^+ = \bigcup_{n \ge 1} X^n.$$

Note that these two latter notation are compatible with the notation A^* and A^+. In order not to overload the notation, a language that is reduced to a single string can be named by the string itself if it does not lead to any confusion. For instance, the expression $A^*\mathtt{abaaab}$ denotes the language of the strings in A^* having the string `abaaab` as suffix, assuming $\{\mathtt{a}, \mathtt{b}\} \subseteq A$.

The notion of length is extended to languages as follows:

$$|X| = \sum_{x \in X} |x|.$$

In the same way, we define $\mathrm{alph}(X)$ by

$$\mathrm{alph}(X) = \bigcup_{x \in X} \mathrm{alph}(x)$$

and $X^{\sim}$ by

$$X^{\sim} = \{x^{\sim} : x \in X\}.$$

The sets of factors, prefixes, suffixes, and subsequences of the strings of a language X are particular languages that are often considered in the rest of the book; they are respectively denoted by $\mathrm{Fact}(X)$, $\mathrm{Pref}(X)$, $\mathrm{Suff}(X)$, and $\mathrm{Subs}(X)$.

The ***right context*** of a string y relatively to a language X is the language:

$$y^{-1}X = \{y^{-1}x : x \in X\}.$$

The equivalence relation defined by the identity of right contexts is denoted by $\equiv_X$, or simply[2] $\equiv$. Thus

$$y \equiv z \text{ if and only if } y^{-1}X = z^{-1}X$$

for $y, z \in A^*$. For instance, when $A = \{\mathtt{a}, \mathtt{b}\}$ and $X = A^*\{\mathtt{aba}\}$, the relation $\equiv$ admits four equivalence classes: $\{\varepsilon, \mathtt{b}\} \cup A^*\{\mathtt{bb}\}$, $\{\mathtt{a}\} \cup A^*\{\mathtt{aa}, \mathtt{bba}\}$, $A^*\{\mathtt{ab}\}$, and $A^*\{\mathtt{aba}\}$. For every language X, the relation $\equiv$ is an equivalence relation that is compatible with the concatenation. It is called the ***right syntactic congruence*** associated with X.

Regular expressions and languages

The ***regular expressions*** on an alphabet A and the languages they describe, the ***regular languages***, are recursively defined as follows:

- `0` and `1` are regular expressions that respectively describe $\emptyset$ (the empty set) and $\{\varepsilon\}$,
- for every letter $a \in A$, $\boldsymbol{a}$ is a regular expression that describes the singleton $\{a\}$,
- if $\boldsymbol{x}$ and $\boldsymbol{y}$ are regular expressions respectively describing the regular languages X and Y, then `(`$\boldsymbol{x}$`)+(`$\boldsymbol{y}$`)`, `(`$\boldsymbol{x}$`)·(`$\boldsymbol{y}$`)`, and `(`$\boldsymbol{x}$`)*` are regular expressions that respectively describe the regular languages $X \cup Y$, $X \cdot Y$, and X^*.

The priority order of operations on the regular expressions is `*`, $\cdot$, then `+`. Possible writing simplifications allow one to omit the symbol $\cdot$ and some parentheses pairs. The language described by a regular expression $\boldsymbol{x}$ is denoted by $\mathrm{Lang}(\boldsymbol{x})$.

Automata

An ***automaton*** M on the alphabet A is composed of a finite set Q of ***states***, of an ***initial*** state[3] q_0, of a set $T \subseteq Q$ of ***terminal*** states, and of a set $F \subseteq Q \times A \times Q$

[2] As in all the rest of the book, the notation is indexed by the object to which they refer only when it could be ambiguous.

[3] The standard definition of automata considers a set of initial states rather than a single initial state as we do in the entire book. We leave the reader to convince himself that it is possible to build a correspondence between any automaton defined in the standard way and an automaton with a single initial state that recognizes the same language.

of ***arcs*** – or ***transitions***. We denote the automaton M by the quadruplet:

$$(Q, q_0, T, F).$$

We say of an arc (p, a, q) that it leaves the state p and that it enters the state q; state p is the ***source*** of the arc, letter a its ***label***, and state q its ***target***. The number of arcs outgoing a given state is called the ***outgoing degree*** of the state. The ***incoming degree*** of a state is defined in a dual way. By analogy with graphs, the state q is a ***successor*** by the letter a of the state p when $(p, a, q) \in F$; in the same case, we say that the pair (a, q) is a ***labeled successor*** of the state p.

A ***path*** of length n in the automaton $M = (Q, q_0, T, F)$ is a sequence of n consecutive arcs

$$\langle (p_0, a_0, p'_0), (p_1, a_1, p'_1), \ldots, (p_{n-1}, a_{n-1}, p'_{n-1}) \rangle,$$

that satisfies

$$p'_k = p_{k+1}$$

for $k = 0, 1, \ldots, n-2$. The ***label*** of the path is the string $a_0 a_1 \ldots a_{n-1}$, its ***origin*** the state p_0, its ***end*** the state p'_{n-1}. By convention, there exists for each state p a path of null length of origin and of end p; the label of such a path is ε, the empty string. A path in the automaton M is ***successful*** if its origin is the initial state q_0 and if its end is in T. A string is ***recognized*** – or ***accepted*** – by the automaton if it is the label of a successful path. The language composed of the strings recognized by the automaton M is denoted by $\mathrm{Lang}(M)$.

Often, more than its formal notation, a diagram illustrates how an automaton works. We represent the states by circles and the arcs by directed arrows from source to target, labeled by the corresponding letter. When several arcs have the same source and the same target, we merge the arcs and the label of the resulting arc becomes an enumeration of the letters. The initial state is distinguished by a short incoming arrow and the terminal states are double circled. An example is shown in Figure 1.2.

A state p of an automaton $M = (Q, q_0, T, F)$ is ***accessible*** if there exists a path in M starting at q_0 and ending in p. A state p is ***co-accessible*** if there exists a path in M starting at p and ending in T.

An automaton $M = (Q, q_0, T, F)$ is ***deterministic*** if for every pair $(p, a) \in Q \times A$ there exists at most one state $q \in Q$ such that $(p, a, q) \in F$. In such a case, it is natural to consider the ***transition function***

$$\delta : Q \times A \to Q$$

of the automaton defined for every arc $(p, a, q) \in F$ by

$$\delta(p, a) = q$$

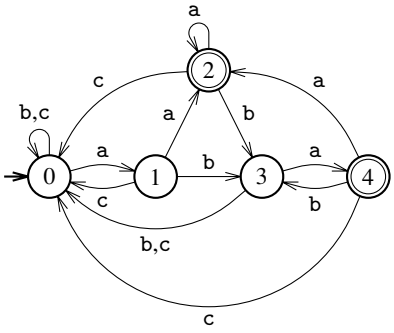

Figure 1.2. Representation of an automaton on the alphabet $A = \{\texttt{a}, \texttt{b}, \texttt{c}\}$. The states of the automaton are numbered from 0 to 4, its initial state is 0, and its terminal states are 2 and 4. The automaton possesses $3 \times 5 = 15$ arcs. The language that it recognizes is described by the regular expression `(a+b+c)*(aa+aba)`, that is, the set of strings on the three letter alphabet `a`, `b`, and `c` ending by `aa` or `aba`.

and not defined elsewhere. The function δ is easily extended to strings. It is enough to consider its extension $\bar{\delta}\colon Q \times A^* \to Q$ recursively defined by $\bar{\delta}(p, \varepsilon) = p$ and $\bar{\delta}(p, wa) = \delta(\bar{\delta}(p, w), a)$ for $p \in Q$, $w \in A^*$, and $a \in A$. It follows that the string w is recognized by the automaton M if and only if $\bar{\delta}(q_0, w) \in T$. Generally, the function δ and its extension $\bar{\delta}$ are denoted in the same way.

The automaton $M = (Q, q_0, T, F)$ is ***complete*** when for every pair $(p, a) \in Q \times A$ there exists at least one state $q \in Q$ such that $(p, a, q) \in F$.

Proposition 1.1
For every automaton, there exists a deterministic and complete automaton that recognizes the same language. ∎

To complete an automaton is not difficult: it is enough to add to the automaton a ***sink*** state, then to make it the target of all undefined transitions. It is a bit more difficult to ***determinize*** an automaton, that is, to transform an automaton $M = (Q, q_0, T, F)$ into a deterministic automaton recognizing the same language. One can use the so-called method of ***construction by subsets***: let M' be the automaton whose states are the subsets of Q, the initial state is the singleton $\{q_0\}$, the terminal states are the subsets of Q that intersect T, and the arcs are the triplets (U, a, V) where V is the set of successors by the letter a of the states p belonging to U; then M' is a deterministic automaton that recognizes the same language as M. In practical applications, we do not construct the automaton M' entirely, but only its accessible part from the initial state $\{q_0\}$.

A language X is ***recognizable*** if there exists an automaton M such that $X = \text{Lang}(M)$. The statement of a fundamental theorem of automata theory that establishes the link between recognizable languages and regular languages on a given alphabet follows.

Theorem 1.2 (Kleene's Theorem)
A language is recognizable if and only if it is regular. ∎

If X is a recognizable language, the ***minimal automaton*** of X, denoted by $\mathcal{M}(X)$, is determined by the right syntactic congruence associated with X. It is the automaton whose set of states is $\{w^{-1}X : w \in A^*\}$, the initial state is X, the set of terminal states is $\{w^{-1}X : w \in X\}$, and the set of arcs is $\{(w^{-1}X, a, (wa)^{-1}X) : (w, a) \in A^* \times A\}$.

Proposition 1.3
The minimal automaton $\mathcal{M}(X)$ of a language X is the automaton having the smallest number of states among the deterministic and complete automata that recognize the language X. The automaton $\mathcal{M}(X)$ is the homomorphic image of every automaton recognizing X. ∎

We often say of an automaton that it is minimal though it is not complete. Actually, this automaton is indeed minimal if one takes care to add a sink state.

Each state of an automaton, or even sometimes each arc, can be associated with an ***output***. It is a value or a set of values associated with the state or the arc.

1.2 Some combinatorics

We consider the notion of periodicity on strings for which we give the basic properties. We begin with presenting two families of strings that have interesting combinatorial properties with regard to questions of periodicities and repeats examined in several chapters.

Some specific strings

Fibonacci numbers are defined by the recurrence:

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1, \\ F_n &= F_{n-1} + F_{n-2} \quad \text{for } n \geq 2. \end{aligned}$$

These famous numbers satisfy properties all more remarkable than the others. Among those, we just give two:

- for every natural number $n \geq 2$, $\gcd(F_n, F_{n-1}) = 1$,
- for every natural number n, F_n is the nearest integer of $\Phi^n/\sqrt{5}$, where $\Phi = \frac{1}{2}(1 + \sqrt{5}) = 1{,}61803\ldots$ is the ***golden ratio***.

Fibonacci strings are defined on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ by the following recurrence:

$$\begin{aligned} f_0 &= \varepsilon, \\ f_1 &= \mathtt{b}, \\ f_2 &= \mathtt{a}, \\ f_n &= f_{n-1} f_{n-2} \quad \text{for } n \geq 3. \end{aligned}$$

Note that the sequence of lengths of the strings is exactly the sequence of Fibonacci numbers, that is, $F_n = |f_n|$. Here are the first ten Fibonacci numbers and strings:

n	F_n	f_n
0	0	ε
1	1	b
2	1	a
3	2	ab
4	3	aba
5	5	abaab
6	8	abaababa
7	13	abaababaabaab
8	21	abaababaabaababaababa
9	34	abaababaabaababaababaabaababaabaab

The interest in Fibonacci strings is that they satisfy many combinatorial properties and they contain a large number of repeats.

The de Bruijn strings considered here are defined on the alphabet $A = \{\mathtt{a}, \mathtt{b}\}$ and are parameterized by a non-null natural number. A nonempty string $x \in A^+$ is a ***de Bruijn string*** of order k if each string on A of length k occurs once and only once in x. A first example: ab and ba are the only two de Bruijn strings of order 1. A second example: the string aaababbbaa is a de Bruijn string of order 3 since its factors of length 3 are the eight strings of A^3, that is, aaa, aab, aba, abb, baa, bab, bba, and bbb, and each of them occurs exactly once in it.

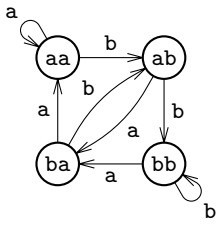

Figure 1.3. The order 3 de Bruijn automaton on the alphabet {a, b}. The initial state of the automaton is not specified.

The existence of a de Bruijn string of order $k \geq 2$ can be verified with the help of the ***automaton*** defined by

- states are the strings of the language A^{k-1},
- arcs are of the form (av, b, vb) with $a, b \in A$ and $v \in A^{k-2}$,

the initial state and the terminal states are not given (an illustration is shown in Figure 1.3). We note that exactly two arcs exit each of the states, one labeled by a, the other by b; and that exactly two arcs enter each of the states, both labeled by the same letter. The graph associated with the automaton thus satisfies the Euler condition: the outgoing degree and the incoming degree of each state are identical. It follows that there exists an Eulerian circuit in the graph. Now, let

$$\langle (u_0, a_0, u_1), (u_1, a_1, u_2), \ldots, (u_{n-1}, a_{n-1}, u_0) \rangle$$

be the corresponding path. The string $u_0 a_0 a_1 \ldots a_{n-1}$ is a de Bruijn string of order k, since each arc of the path is identified with a factor of length k. It follows in the same way that a de Bruijn string of order k has length $2^k + k - 1$ (thus $n = 2^k$ with the previous notation). It can also be verified that the number of de Bruijn strings of order k is exponential in k.

The de Bruijn strings are often used as examples of limit cases in the sense that they contain all the factors of a given length.

Periodicity and borders

Let x be a nonempty string. An integer p such that $0 < p \leq |x|$ is called a ***period*** of x if:

$$x[i] = x[i + p]$$

for $i = 0, 1, \ldots, |x| - p - 1$. Note that the length of a nonempty string is a period of this string, such that every nonempty string has at least one period. We define thus without any ambiguity ***the period*** of a nonempty string x as the