# A general framework for the realistic analysis of sorting and searching algorithms. Application to some popular algorithms

Julien Clément, Thu Hien Nguyen Thi, Brigitte Vallée

# A GENERAL FRAMEWORK FOR THE REALISTIC ANALYSIS OF SORTING AND SEARCHING ALGORITHMS. APPLICATION TO SOME POPULAR ALGORITHMS

JULIEN CLÉMENT [1] AND THU HIEN NGUYEN THI [1] AND BRIGITTE VALLÉE [1]

[1] Université de Caen / ENSICAEN / CNRS - GREYC - Caen, France
*E-mail address*: {julien.clement,thu-hien.nguyen-thi,brigitte.vallee}@unicaen.fr

Abstract. We describe a general framework for realistic analysis of sorting and searching algorithms, and we apply it to the average-case analysis of five basic algorithms: three sorting algorithms (*QuickSort*, *InsertionSort*, *BubbleSort*) and two selection algorithms (*QuickMin* and *SelectionMin*). Usually, the analysis deals with the mean number of key comparisons, but, here, we view keys as words produced by the same source, which are compared via their symbols in the lexicographic order. The "realistic" cost of the algorithm is now the total number of symbol comparisons performed by the algorithm, and, in this context, the average–case analysis aims to provide estimates for the mean number of symbol comparisons used by the algorithm. For sorting algorithms, and with respect to key comparisons, the average-case complexity of *QuickSort* is asymptotic to $2n \log n$, *InsertionSort* to $n^2/4$ and *BubbleSort* to $n^2/2$. With respect to symbol comparisons, we prove that their average-case complexity becomes $\Theta(n \log^2 n), \Theta(n^2), \Theta(n^2 \log n)$. For selection algorithms, and with respect to key comparisons, the average-case complexity of *QuickMin* is asymptotic to $2n$, of *SelectionMin* is $n - 1$. With respect to symbol comparisons, we prove that their average-case complexity remains $\Theta(n)$. In these five cases, we describe the dominant constants which exhibit the probabilistic behaviour of the source (namely, entropy, and various notions of coincidence) with respect to the algorithm.

## Introduction

There are two main classes for sorting and searching algorithms: the first class gathers the algorithms which deal with keys, while the algorithms of the second class deal with words (or strings). Of course, any data is represented inside a computer as a sequence of bits (that is a binary string). However, the point of view is different: the key is viewed as a "whole", and its precise representation is not taken into account, whereas the structure of a word,

as a sequence of symbols, is essential in text algorithms. Hence, for basic algorithms of the first class (sorting, searching), the unit operation is the comparison between keys, whereas for text algorithms of the second class, comparisons between symbols are considered.

There exist two important drawbacks to this usual point of view. First, it is difficult to compare algorithms belonging to these two different classes, since they are analyzed with respect to different costs. Second, when the keys are complex items, not reduced to single machine words, it is not realistic to consider the total cost of their comparison as unitary. This is why Sedgewick proposed in 1998 to analyze basic algorithms (sorting and searching) when dealing with words rather than with "atomic" keys; in this case, the realistic cost for comparing two words is the number of symbols comparisons needed to distinguish them in the lexicographic order and is closely related to the length of their longest common prefix, called here the *coincidence*. There are two factors which influence the efficiency of such an algorithm: the strategy of the algorithm itself (*which words are compared?*) and the mechanism which produces words, called the source (*what makes two words distinguishable?*).

The first results in the area are due to Fill and Janson [5], Fill and Nakama [6], who dealt with data composed of random uniform bits. Then, in the paper [19], a general framework towards a realistic analysis based on the number of symbol comparisons is provided, when the source which emits symbols is (almost completely) general. Furthermore, these principles are applied to two algorithms, QuickSort and QuickSelect. Later on, a study of the distribution of the complexity was performed in the same framework [7, 8].

**Main results.** The present paper follows the lines of the article [19], and works within the same general framework, with four specific aims:

($a$) The general method has been already described in [19]: it was shown that a Dirichlet series denoted by $\varpi(s)$ characterizes the behavior of an algorithm with respect to the source. We wish here to highlight the main principles, in order to make easier its application to various algorithms. As it is often the case in analytical combinatorics, there are two main phases in the method, a first phase where the series $\varpi(s)$ is built, and a second phase where it is analyzed. We explain here how the first phase may be performed in an "automatic" way. For such an example, see Proposition of Appendix B.1.

($b$) We apply the method to three other popular algorithms: InsertionSort, BubbleSort and SelectionMinimum, respectively denoted in the sequel by the short names `InsSort`, `BubSort`, `SelMin` are succinctly described in Appendix A (see for instance the book [16] for a thorough description of these algorithms). With this approach we also easily recover the results about algorithms `QuickSort` and `QuickMin` already obtained in [19]. Thus we provide an unified framework for the analysis of these five algorithms in Section 2.2.

($c$) We exhibit in each case the probabilistic features of the source which play a role in the analysis: each algorithm of interest is related to a particular constant of the source, which describes the interplay between the algorithm and the source, and explains how the efficiency of the algorithm depends on the source, via various notions of coincidence between words (See Proposition 5). This type of coincidence provides a good characterization of the algorithm, and our study is a tool for a better understanding of the algorithmic strategy.

($d$) We discuss the robustness of the algorithms, i.e., the possible changes in the complexity behaviors, due to the change in the complexity measure, from the number of key comparisons to the number of symbol comparisons (see Discussion p. 11).

**Plan of the paper.** Section 1 first presents the general method, with its main steps. Then, Section 2 states the main results. Finally, Appendices A and B are devoted to the proofs.

## 1. Main steps for the "realistic" analysis of a sorting algorithm

Here, we describe our general framework, already provided in [19]. We insist on the main steps, and the notions developed here are somewhat different from the previous paper. We first characterize in Section 1.1 the strategy of the algorithm (which keys are compared? with which probability?), then we describe the source, and the central notion of coincidence (Sections 1.2 and 1.3). We obtain an exact formula for the mean number of symbol comparisons, which involves the mixed Dirichlet series $\varpi(s)$ (depending on the source *and* the algorithm) introduced in Section 1.4 and 1.5. In order to obtain asymptotic estimates, we deal with tameness properties of the source, which entail tameness for the series $\varpi(s)$, and finally the asymptotic estimates (Sections 1.6 and 1.7).

### 1.1. The classical probabilistic model: permutations and arrival times

Consider a totally ordered set of keys $\mathcal{U} = \{U_1 < U_2 < \cdots < U_n\}$ and any algorithm $\mathcal{A}$ which only performs comparisons and exchanges between keys. The initial input is the sequence $(V_1, V_2, \ldots, V_n)$ defined from $\mathcal{U}$ by the permutation $\sigma \in \mathfrak{S}_n$ via the equalities $V_i = U_{\sigma(i)}$. The execution of the algorithm does not actually depend on the input sequence, but only on the permutation $\sigma$ which defines the input sequence from the final (ordered) sequence. Then, the permutation $\sigma$ is the actual input of the algorithm and the set of all possible inputs is the set $\mathfrak{S}_n$ (usually endowed with the uniform distribution).

There is another point of view, given by the arrival times. The arrival time of $U_i$, denoted by $\tau(U_i)$ is the position of $U_i$ in the input array. Of course, there is a simple relation between the two points of view since $\tau(U_i) = j$ if and only if $V_j = U_i$ (meaning also $\sigma(j) = i$ since there is a bijection between arrival times and permutations).

The strategy of the algorithm $\mathcal{A}$ defines, for each pair $(i, j)$, with $1 \leq i < j \leq n$, the subset of $\mathfrak{S}_n$ which gathers the permutations $\sigma$ (or the arrival times) for which $U_i$ and $U_j$ are compared by the algorithm $\mathcal{A}$, when the input sequence is $(U_{\sigma(1)}, U_{\sigma(2)}, \ldots, U_{\sigma(n)})$. For efficient algorithms, the two keys $U_i$ and $U_j$ are compared only once, but there exist other algorithms (the `BubSort` algorithm for instance) where $U_i$ and $U_j$ may be compared several times. In all cases, $\pi(i, j)$ denotes the mean number of comparisons between $U_i$ and $U_j$. The computation of $\pi(i, j)$ is the first step, described in Section 2.1, and proven in Appendix A.

There are two types of comparisons between two keys $U_i$ and $U_j$: the *positive* comparisons which occur when $U_i$ and $U_j$ arrive in the good order in the initial array ($\tau(U_i) < \tau(U_j)$), and the *negative* comparisons which occur when $U_i$ and $U_j$ arrive in the wrong order ($\tau(U_i) > \tau(U_j)$). The mean number of positive and negative comparisons between two keys $U_i$ and $U_j$ is denoted respectively by $\pi^+(i, j)$ and $\pi^-(i, j)$. These mean numbers $\pi^{\pm}(i, j)$ are often computed in a separate way, with direct probabilistic arguments dealing with the arrival times. A remarkable feature is that the expectations $\pi^{\pm}(i, j)$ are always expressed as sums of rational functions depending on $i, j$ or $j - i$. The mean number of key comparisons is $\pi(i, j) = \pi^+(i, j) + \pi^-(i, j)$.

### 1.2. General sources

Here, we consider that the keys are words produced by a general source.

**Definition 1.1.** Let $\Sigma$ be a totally ordered alphabet of cardinality $r$. A *general source* produces infinite words of $\Sigma^{\mathbb{N}}$, and is specified by the set $\{p_w, w \in \Sigma^\star\}$ of *fundamental probabilities* $p_w$, where $p_w$ is the probability that an infinite word begins with the finite prefix $w$. It is (only) assumed that $\sup\{p_w : w \in \Sigma^k\}$ tends to 0, as $k \to \infty$.

For any prefix $w \in \Sigma^\star$, we denote by $|w|$ the length of $w$ (i.e., the number of the symbols that it contains) and $a_w$, $b_w$, $p_w$ the probabilities that a word produced by the source begins with a prefix $\alpha$ of the same length as $w$, which satisfies $\alpha < w$, $\alpha \leq w$, or $\alpha = w$, meaning

$$a_w := \sum_{\substack{\alpha, |\alpha|=|w|,\\ \alpha<w}} p_\alpha, \qquad b_w := \sum_{\substack{\alpha, |\alpha|=|w|,\\ \alpha\leq w}} p_\alpha, \qquad p_w = b_w - a_w. \tag{1.1}$$

Denote by $\mathcal{L}(\mathcal{S})$ the set of (infinite) words produced by the source $\mathcal{S}$, ordered via the lexicographic order. Given an infinite word $X \in \mathcal{L}(\mathcal{S})$, denote by $w_k$ its prefix of length $k$. The sequence $(a_{w_k})$ is increasing, the sequence $(b_{w_k})$ is decreasing, and $b_{w_k} - a_{w_k} = p_{w_k}$ tends to 0. Thus a unique real $P(X) \in [0,1]$ is defined as the common limit of $(a_{w_k})$ and $(b_{w_k})$, and $P(X)$ can be viewed as the probability that an infinite word $Y$ be smaller than $X$. The mapping $P : \mathcal{L}(\mathcal{S}) \to [0,1]$ is strictly increasing outside the exceptional set formed with words of $\mathcal{L}(\mathcal{S})$ which end with an infinite sequence of the smallest symbol or with an infinite sequence of the largest symbol.

Conversely, almost everywhere, except on the set $\{a_w, w \in \Sigma^\star\}$, there is a mapping $M$ which associates, to a number $u$ of the interval $\mathcal{I} := [0,1]$, a word $M(u) \in \mathcal{L}(\mathcal{S})$. Hence the probability that a word $Y$ be smaller than $M(u)$ equals $u$. The lexicographic order on words is then compatible with the natural order on the interval $\mathcal{I}$. The interval $\mathcal{I}_w := [a_w, b_w]$, of length $p_w$, gathers (up to a denumerable set) all the reals $u$ for which $M(u)$ begins with the finite prefix $w$. This is the fundamental interval of the prefix $w$.

### 1.3. Coincidence

Here, we are interested by a more realistic cost related to the number of symbol comparisons performed by these algorithms, when the keys are words independently produced by the same source. The words are ordered with respect to the lexicographic order, and the cost for comparing two words (measured as the number of symbol comparisons needed) is closely related to the coincidence, defined as follows.

**Definition 1.2.** The *coincidence function* $\gamma(u,t)$ is the length of the largest common prefix of $M(u)$ and $M(t)$.

More precisely, the realistic cost of the comparison between $M(u)$ and $M(t)$ equals $\gamma(u,t)+1$. The coincidence $\gamma(u,t)$ is at least $\ell$ if and only if $M(u)$ and $M(t)$ have the same common prefix $w$ of length $\ell$, so that the parameters $u$ and $t$ belong to the same fundamental interval $\mathcal{I}_w$ relative to a prefix $w$ of length $\ell$. We thus introduce the triangles

$$\mathcal{T} := \{(u,t) : 0 \leq u \leq t \leq 1\}, \quad \mathcal{T}_w = (\mathcal{I}_w \times \mathcal{I}_w) \cap \mathcal{T} = \{(u,t) : a_w \leq u \leq t \leq b_w\}. \tag{1.2}$$

Using the two relations

$$\mathcal{T} \cap [\gamma \geq \ell] = \bigcup_{w \in \Sigma^\ell} \mathcal{T}_w, \qquad \sum_{\ell \geq 0} \mathbf{1}_{[\gamma \geq \ell]} = \sum_{\ell \geq 0} (\ell+1) \mathbf{1}_{[\gamma = \ell]},$$

the following equality holds, for any integrable function $g$ on the unit triangle $\mathcal{T}$, and will be extensively used in the sequel,

$$\int_{\mathcal{T}} [\gamma(u,t)+1] g(u,t)\, du\, dt = \sum_{w\in\Sigma^\star} \int_{\mathcal{T}_w} g(u,t)\, du\, dt. \tag{1.3}$$

### 1.4. Average-case analysis – various models

The purpose of average–case analysis of structures (or algorithms) is to characterize the mean value of their parameters under a well-defined probabilistic model that describes the initial distribution of its inputs.

Here, we adopt the following general model for the set of inputs: we consider a finite sequence $\mathcal{V} = (V_1, \ldots, V_n)$ of infinite words independently produced by the same source $\mathcal{S}$. Such a sequence $\mathcal{V}$ is obtained by $n$ independent drawings $v_1, v_2, \ldots, v_n$ in the interval $\mathcal{I}$ via the mapping $M$, and we set $V_i := M(v_i)$. We assume moreover that $\mathcal{V}$ contains two given words $M(u)$ and $M(t)$, with $u < t$. The variables $N_{[0,u[}$, $N_{[0,t[}$ respectively denote the number of words of $\mathcal{V}$ strictly less than $M(u)$, strictly less than $M(t)$. These variables define the ranks of $M(u)$ and $M(t)$ inside the set $\mathcal{V}$, via the relations, valid for $u < t$,

$$\operatorname{Rank} M(u) = N_{[0,u[} + 1, \operatorname{Rank} M(t) = N_{[0,t[} + 2,$$

where the respective translations of 1 and 2 express that $M(u)$ and $M(t)$ belong to $\mathcal{V}$.

We first consider the number of key comparisons between $M(u)$ and $M(t)$, and deal with the mean number $\widehat{\pi}(u,t)$ of key comparisons performed by the algorithm between $M(u)$ and $M(t)$, where the mean is taken with respect to all the permutations of $\mathcal{V}$. The mean number $\widehat{\pi}(u,t)$ is related to the mean number $\pi(i,j)$ via the equality

$$\widehat{\pi}(u,t) = \pi(N_{[0,u[} + 1, N_{[0,t[} + 2). \tag{1.4}$$

In our framework, expressions obtained for $\pi(i,j)$ ensure that $\widehat{\pi}(u,t)$ is always a sum of rational functions in variables $N_{[0,u[}$, $N_{[0,t[}$ and $N_{[u,t[}$, (with the relation $N_{[0,t[} = N_{[0,u[} + N_{]u,t[} + 1$).

When the cardinality $n$ of $\mathcal{V}$ is fixed, and words $V_i \in \mathcal{V}$ are independently emitted by the source $\mathcal{S}$, this is the Bernoulli model denoted by $(\mathcal{B}_n, \mathcal{S})$. However, it proves technically convenient to consider that the sequence $\mathcal{V}$ has a variable number $N$ of elements that obeys a Poisson law of rate $Z$,

$$\Pr\{N = k\} = e^{-Z} \frac{Z^k}{k!}. \tag{1.5}$$

In this model, called the Poisson model of rate $Z$, the rate $Z$ plays a role much similar to the cardinality of $\mathcal{V}$. When it is relative to probabilistic source $\mathcal{S}$, the model, denoted by $(\mathcal{P}_Z, \mathcal{S})$, is composed with two main steps:

($a$) The number $N$ of words is drawn according to the Poisson law of rate $Z$;

($b$) Then, the $N$ words are independently drawn from the source $\mathcal{S}$.

Note that, in the Poisson model, the variables $N_{[0,u[}, N_{]u,t[}$ are themselves independent Poisson variables of parameters $Zu$ and $Z(t-u)$ (respectively). The expectation $\widehat{\pi}(u,t)$ is itself a random variable which involves these variables.

### 1.5. Exact formula for the mean number of symbol comparisons

The density of the algorithm in the Poisson model, denoted by $\phi_Z(u,t)$ and defined as

$$\phi_Z(u,t)\,du\,dt = Z^2 \cdot \mathbb{E}_Z[\widehat{\pi}(u,t)]\,du\,dt = (Z\,du)\cdot(Z\,dt)\cdot\mathbb{E}_Z[\widehat{\pi}(u,t)],$$

is the mean number of key comparisons between two words $M(u')$ and $M(t')$ for $u' \in [u-du,u]$ and $t' \in [t,t+dt]$. In the model $(\mathcal{P}_Z,\mathcal{S})$, this is a main tool for computing, not only the mean number of key comparisons $K_Z$ performed by the algorithm, but also the mean number of symbol comparisons $S_Z$ via the formulae

$$K_Z = \int_{\mathcal{T}} \phi_Z(u,t)\,du\,dt, \qquad S_Z = \int_{\mathcal{T}} [\gamma(u,t)+1]\phi_Z(u,t)\,du\,dt.$$

To return to the Bernoulli model $(\mathcal{B}_n,\mathcal{S})$, the coefficients $\varphi(n,u,t)$ in the series expansion of $\phi_Z(u,t)$ defined as

$$\varphi(n,u,t) := (-1)^n\, n![Z^n]\phi_Z(u,t), \tag{1.6}$$

are computed in an "automatic way" from the probabilities $\widehat{\pi}(u,t)$, themselves closely related to $\pi(i,j)$. This is the second step precisely described in Appendix B.1 leading to results in Table 1 p. 12. Using Eq. (1.3), the sequence $\varphi(n)$ is now defined for any $n \geq 2$,

$$\varphi(n) := \int_{\mathcal{T}} (\gamma(u,t)+1)\,\varphi(n,u,t)\,du\,dt = \sum_{w\in\Sigma^\star} \int_{\mathcal{T}_w} \varphi(n,u,t)\,du\,dt, \tag{1.7}$$

and is easy to obtain via computations of the integral of $\varphi(n,u,t)$ on the triangles $\mathcal{T}_w$. Now, the mean number $S(n)$ of symbol comparisons used by the algorithm when it deals with $n$ words independently drawn from the same source is related to $\varphi(n)$ by the equality

$$S(n) = \sum_{k=2}^{n} (-1)^k \binom{n}{k} \varphi(k), \tag{1.8}$$

which provides an exact formula for $S(n)$, described in Section 2.2. The expression of $S(n)$ is obtained in an "automatic" way, from the expectations $\pi(i,j)$.

### 1.6. Asymptotic estimates for the mean number of symbol comparisons

However, the previous formula does not give an easy or straightforward access to the asymptotic behaviour of $S(n)$ (when $n \to \infty$). In order to get asymptotic estimates, we first need an analytic lifting $\varpi(s,u,t)$ of the coefficients $\varphi(k,u,t)$, that is an analytic function $\varpi(s,u,t)$ which coincides with $\varphi(k,u,t)$ at integer values $s=k$ in the summation of Eq. (1.8). This analytic lifting gives rise to the mixed Dirichlet series itself,

$$\varpi(s) := \int_{\mathcal{T}} [\gamma(u,t)+1]\varpi(s,u,t)\,du\,dt = \sum_{w\in\Sigma^\star} \int_{\mathcal{T}_w} \varpi(s,u,t)\,du\,dt,$$

which depends both on the algorithm (via $\varpi(s,u,t)$) and the source (via the fundamental triangles $\mathcal{T}_w$). For each algorithm, the existence of this analytic lifting is granted in a domain $\Re s > \sigma_0$. However, the value of $\sigma_0$ depends on the algorithm. One has $\sigma_0 = 1$, except for the algorithms `InsSort` and `BubSort` where $\sigma_0$ equals 2. This is due to constant term $1/2$ appearing in the expectation $\pi(i,j)$, as seen in Table 1 p. 12 (see also Section 2.2).

The Rice Formula [13, 14] transforms a binomial sum into an integral in the complex plane. For any real $\sigma_1 \in ]\sigma_0, \sigma_0+1[$, one has

$$T(n) := \sum_{k=1+\sigma_0}^{n} (-1)^k \binom{n}{k} \varpi(k) = \frac{(-1)^{n+1}}{2i\pi} \int_{\Re s=\sigma_1} G(s)\, ds, \text{ with } G(s) := \frac{n!\, \varpi(s)}{s(s-1)\dots(s-n)}. \tag{1.9}$$

Then, along general principles in analytic combinatorics [10, 11], the integration line can be pushed to the left, as soon as $G(s)$ (closely related to $\varpi(s)$) has good analytic properties: we need a region $\mathcal{R}$ on the left of $\Re s = \sigma_0$, where $\varpi(s)$ is of polynomial growth (for $\Im s \to \infty$) and meromorphic. With a good knowledge of its poles, we finally obtain a residue formula

$$T(n) = (-1)^{n+1} \left[ \sum_s \operatorname{Res}\left[G(s)\right] + \frac{1}{2i\pi} \int_{\mathcal{C}_2} G(s)\, ds \right],$$

where $\mathcal{C}_2$ is a curve of class $\mathcal{C}^1$ enclosed in $\mathcal{R}$ and the sum is extended to all poles $s$ of $G(s)$ inside the domain delimited by the vertical line $\Re s = \sigma_1$ and the curve $\mathcal{C}_2$.

The dominant singularities of $G(s)$ provide the asymptotic behaviour of $T(n)$, and the remainder integral is estimated using the polynomial growth of $G(s)$ when $|\Im(s)| \to \infty$. According to Eq. (1.8) and(1.9), and in the cases where $\sigma_0 = 2$, we have to add to $T(n)$ the term corresponding to the index $k = 2$, where the analytical lifting $\varpi$ does not coincides with $\varphi$. For algorithms `BubSort` and `InsSort`, the additional term is of the form $\varphi(2)\binom{n}{2}$.

### 1.7. Tameness of sources

We first describe three cases of possible regions $\mathcal{R}$ where good properties of $\varpi(s)$ will make possible such a shifting to the left in the Rice formula.

**Definition 1.3.** A function $\varpi(s)$ is tame at $\sigma_0$ if one of the three following properties holds:

($a$) [$S$–shape] (shorthand for Strip shape) there exists a vertical strip $\Re(s) > \sigma_0 - \delta$ for some $\delta > 0$ where $\varpi(s)$ is meromorphic, has only a pole (of order $k_0 \geq 0$) at $s = \sigma_0$ and is of polynomial growth as $|\Im s| \to +\infty$.

($b$) [$H$–shape] (shorthand for Hyperbolic shape) there exists an hyperbolic region $\mathcal{R}$, defined as, for some $A, B, \rho > 0$

$$\mathcal{R} := \{s = \sigma + it;\ \ |t| \geq B,\ \ \sigma > \sigma_0 - \frac{A}{t^\rho}\} \bigcup \{s = \sigma + it;\ \ \sigma > \sigma_0 - \frac{A}{B^\rho}, |t| \leq B\},$$

where $\varpi(s)$ is meromorphic, with an only pole (of order $k_0 \geq 0$) at $s = \sigma_0$ and is of polynomial growth in $\mathcal{R}$ as $|\Im s| \to +\infty$.

($c$) [$P$–shape] (shorthand for Periodic shape) there exists a vertical strip $\Re(s) > \sigma_0 - \delta$ for some $\delta > 0$ where $\varpi(s)$ is meromorphic, has only a pole (of order $k_0 \geq 0$) at $s = \sigma_0$ and a family $(s_k)$ (for $k \in \mathbb{Z}, k \neq 0$) of simple poles at points $s_k = \sigma_0 + 2ki\pi t$ with $t \neq 0$, and is of polynomial growth as $|\Im s| \to +\infty$[1].

There are three parameters relative to the tameness: the integer $k_0$ is the *order*, and, when they exist, the real $\delta$ is the *abscissa*, and the real $\rho$ is the *exponent*.

[1]More precisely, this means that $\varpi(s)$ is of polynomial growth on a family of horizontal lines $t = t_k$ with $t_k \to \infty$, and on vertical lines $\Re(s) = \sigma_0 - \delta'$ with some $\delta' < \delta$.

Here, the main Dirichlet series $\varpi(s)$ of interest are closely related to the Dirichlet series of the source, which involve the fundamental probabilities $p_w$, and the ends $a_w, b_w$ of the fundamental intervals (see Section 1.1), via a function $F : [0,1]^2 \to \mathbb{R}^+$ of class $\mathcal{C}^1$,

$$\Lambda[F](s) := \sum_{w\in\Sigma^\star} F(a_w, b_w)\, p_w^s, \qquad \Lambda_k[F](s) := \sum_{w\in\Sigma^k} F(a_w, b_w)\, p_w^s. \tag{1.10}$$

For $F \equiv 1$, we omit the reference to $F$, and we let $\Lambda := \Lambda[1]$. These series satisfy, for $\Re s > 1$, the relation[2] $|\Lambda(F,s)| \leq \|F\|\Lambda(\sigma)$. Since the equality $\Lambda_k(1) = 1$ holds for all $k$, the series $\Lambda(s)$ is divergent at $s = 1$, and many probabilistic properties of the source can be expressed in terms of the behavior of $\Lambda(s)$, when $\Re s$ is close to 1. For instance, the entropy $h(\mathcal{S})$ of the source $\mathcal{S}$ is defined as the limit (if it exists),

$$h(\mathcal{S}) := \lim_{k\to\infty} \frac{-1}{k} \sum_{w\in\Sigma^k} p_w \log p_w = \lim_{k\to\infty} \frac{-1}{k} \frac{d}{ds} \Lambda_k(s)_{|s=1}. \tag{1.11}$$

Two types of properties of the source may entail tameness for the mixed series $\varpi(s)$.

**Definition 1.4.** [Tameness of Sources.]

(*a*) A source is *weakly tame* if the function $s \mapsto \Lambda(s)$ is analytic on $\Re s > 1$, and of polynomial growth when $\Im s \to \infty$ on any $\Re s \geq \sigma_1 > 1$

(*b*) Denote by $\mathcal{F}$ the set of functions $F : [0,1]^2 \to \mathbb{R}^+$ of class $\mathcal{C}^1$. A source is $\Lambda$–*tame* if $\Lambda(s)$ admits at $s = 1$ a simple pole, with a residue equal to $1/h(\mathcal{S})$, (where $h(\mathcal{S})$ is the entropy of the source)[3] and if one of the following conditions is fulfilled:

(1) [$S$–shape] for any $F \in \mathcal{F}$, the series $\Lambda[F](s)$ is tame at $s = 1$ with a $S$–shape;
(2) [$H$–shape] for any $F \in \mathcal{F}$, the series $\Lambda[F](s)$ is tame at $s = 1$ with a $H$–shape;
(3) [$P$–shape] for any $F \in \mathcal{F}$, the series $\Lambda[F](s)$ is tame at $s = 1$, with a $P$–shape for $F \equiv 1$. For $F \not\equiv 1$, $\Lambda[F](s)$ has either a $S$–shape, or a $P$–shape.

This definition is in fact very natural, since it describes various possible behaviors of classical sources. "Most of the time", the simple sources (memoryless sources or aperiodic Markov chains) are $\Lambda$–tame. They never have a $S$–shape, but they may have a $H$–shape or a $P$–shape, according to arithmetic properties of their probabilities [9]. Dynamical sources, introduced by Vallée and defined in [18], may have a $P$–shape only if they are "similar" to simple sources. Adapting deep results of Dolgopyat [3, 4], it is possible to prove that dynamical sources are "most of the time" $\Lambda$–tame with a $S$–shape [1], but they may also have a $H$–shape [15]. See the cited papers for more details, where all these facts, here described in a informal way, are stated in a formal way and proven.

This definition is also well-adapted to our framework since it describes situations where the mixed series $\varpi(s)$ may be proven tame. Then, the contour of the Rice integral may be shifted to the left, providing an asymptotic expansion for the mean number $S(n)$.

The weak tameness of the source is sufficient to entail the tameness at $s = 1$ (with a $S$–shape, and an exponent $k_0 = 0$) of series $\varpi(s)$ related to selection algorithms (namely `QuickMin` and `SelMin`). The $\Lambda$–tameness of the source is central in the analysis of sorting

[2] The norm $\|\cdot\|$ is the sup-norm on $[0,1] \times [0,1]$.

[3] Then, (as we prove it in Appendix B.2), any series $\Lambda[F](s)$ for any $F \in \mathcal{F}, F > 0$, admits at $s = 1$ a simple pole, with a residue equal to

$$\frac{1}{h(\mathcal{S})} \int_0^1 F(x,x)dx.$$

algorithms, as it ensures the tameness of $\varpi(s)$ related to algorithms `QuickSort`, `InsSort` and `BubSort`); moreover, the tameness shape $\varpi(s)$ is inherited from the one of the source.

## 2. Summary of our results.

We recall the main steps of the method.
**Step 1.** Computation of probabilities $\pi(i, j)$.
**Step 2.** Automatic derivation of $\varpi(s, u, t)$; determination of the abscissa $\sigma_0$.
**Step 3.** Expression for the mixed Dirichlet series $\varpi(s)$, and description of the main term of the singular expression of $\varpi(s)/(s - \sigma_0)$. Interpretation of the "dominant" constants.
**Step 4.** Relation between tameness of the source and tameness of the mixed series $\varpi(s)$. Application of the Rice Formula. Statement of the final results.

This Section presents the results with three tables (found at the end), five propositions and a theorem. Section 2.1 summarizes Steps 1 and 2 with Propositions 2.1 and 2.2, and Table 1. Section 2.2 summarizes Step 3 with Propositions 2.3, 2.4, 2.5, and Table 2. Finally, Section 2.3 states the final result (Theorem 2.6) with Table 3. The proofs are given in Appendix A for Step 1, and in Appendix B for the other steps.

### 2.1. Summary of the results for Step 1 and 2

We present in the leftmost part of Table 1 the expressions for the mean number $\pi(i, j)$ of key comparisons between $U_i$ and $U_j$, for each algorithm of interest. The proof of these estimates is found in Appendix A. With these expressions, it is easy to recover the estimates for the mean number $K(n)$ of key comparisons (recalled in the third column).

**Proposition 2.1.** *Consider the permutation model described in Section 1.1, and denote by $\pi(i, j)$ the mean number of comparisons between the keys of rank $i$ and $j$, with $i \leq j$. Then, for any of the five algorithms, the mean numbers $\pi(i, j)$ admit the expressions described in the second column of Table 1 p. 12.*

We then obtain the expressions for the analytic lifting $\varpi(s, u, t)$, via an "automatic" derivation further described in Appendix B.1.

**Proposition 2.2.** *Denote by $\varpi(s, u, t)$ the function which provides an analytical lifting of the sequence $\varphi(n, u, t)$ defined in Eq. (1.6), and by $\sigma_0$ the integer which defines the domain $\Re s > \sigma_0$ of validity of this lifting. Then, for any of the five algorithms, the functions $\varpi(s, u, t)$ admit the expressions described in the fifth column of Table 1 p. 12.*

### 2.2. Summary of the results for Step 3 – the mixed Dirichlet series

**Proposition 2.3.** *Consider any general source, assumed to be weakly tame, together with the fundamental intervals $[a_w, b_w]$ defined in (1.1) and its Dirichlet series defined in Eq. (1.10). Then, for any of the five algorithms, the mixed Dirichlet series $\varpi(s)$ (defined in Section 1.6) admit in the domain $\Re s > \sigma_0$, the expressions displayed in the second column of Table 2, together with the values of $\sigma_0$ in the third column. Depending on the value of $\sigma_0$ the mean number $S(n)$ of symbol comparisons is*

$$S(n) = \sum_{k=2}^{n} (-1)^k \binom{n}{k} \varpi(k) \text{ (if } \sigma_0 = 1), \; S(n) = \binom{n}{2} \frac{\Lambda(2)}{2} + \sum_{k=3}^{n} (-1)^k \binom{n}{k} \varpi(k) \text{ (if } \sigma_0 = 2).$$

We now study the relation between tameness of the source and tameness of the mixed Dirichlet series.

**Proposition 2.4.** *Assume the source $\mathcal{S}$ to be weakly tame. Then, the mixed Dirichlet series $\varpi(s)$ relative to selection algorithms satisfy the following:*

(*a*) `[QuickMin]` *$\varpi(s)$ is $\Lambda$–tame at $\sigma_0 = 1$ with order $k_0 = 0$ with an abscissa $\delta \geq 1/3$.*
(*b*) `[SelMin]` *$\varpi(s)$ is $\Lambda$–tame at $\sigma_0 = 1$ with order $k_0 = 0$ with an abscissa $\delta$ which depends on the exponent $a$ defined in (B.10).*

*Assume the source $\mathcal{S}$ to be $\Lambda$–tame. Then, the mixed Dirichlet series $\varpi(s)$ relative to sorting algorithms satisfy the following:*

(*a*) `[QuickSort]` *$\varpi(s)$ is $\Lambda$–tame at $\sigma_0 = 1$ with order $k_0 = 2$.*
(*b*) `[InsSort]` *$\varpi(s)$ is $\Lambda$–tame at $\sigma_0 = 2$ (order $k_0 = 1$) and at $\sigma_0 = 1$ (order $k_0 = 1$).*
(*c*) `[BubSort]` *$\varpi(s)$ is $\Lambda$–tame at $\sigma_0 = 2$ with order $k_0 = 1$.*

*Moreover, the source $\mathcal{S}$ gives its shape of tameness to the series $\varpi(s)$.*

We finally describe the main term of the singular expression of $\varpi(s)/(s-\sigma_0)$ at $s = \sigma_0$.

**Proposition 2.5.** *The constants of interest which intervene in the main terms displayed in the last column of Table 2 p. 12 are:*

(*i*) *The entropy $h(\mathcal{S})$ of the source.*
(*ii*) *The coincidence $c(\mathcal{S})$, namely the mean number of symbols needed to compare two random words produced by the source.*
(*iii*) *The min–coincidence $a(\mathcal{S})$: this is the mean number of symbols needed to compare a uniform random word and the smallest word of the source.*
(*iv*) *The logarithmic coincidence $b(\mathcal{S})$: this is the mean number of symbols needed to compare two words $X$ and $Y$ randomly chosen as follows: the word $X$ is uniformly drawn from the source, and $Y$ is drawn with $Y \geq X$, according to density $1/t$.*

The entropy is defined in (1.11). The constants $a(\mathcal{S}), b(\mathcal{S}), c(\mathcal{S})$ satisfy the inequalities $a(\mathcal{S}) < b(\mathcal{S}), c(\mathcal{S}) < 2b(\mathcal{S})$ and are defined as follows

$$a(\mathcal{S}) = \sum_{\ell \geq 0} q_\ell, \quad b(\mathcal{S}) = \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} \frac{1}{t}\, du\, dt \quad c(\mathcal{S}) = 2 \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} du\, dt = \sum_{w \in \Sigma^\star} p_w^2 = \Lambda(2).$$

Here $q_\ell$ is the probability of the prefix of length $\ell$ of the smallest word of the source, $\mathcal{T}_w$ is the fundamental triangle defined in (1.2) and $\Lambda(s)$ is defined in (1.10).

The constants $a(\mathcal{S}), c(\mathcal{S})$ and $h(\mathcal{S})$ are easy to compute for any memoryless source. For the unbiased source $\mathcal{M}_r$, or for the source $\mathcal{B}_p$ on the alphabet $\{0,1\}$, with $p := p_0$, one has:

$$a(\mathcal{M}_r) = c(\mathcal{M}_r) = \frac{r}{r-1}, \qquad h(\mathcal{M}_r) = \log r,$$

$$a(\mathcal{B}_p) = \frac{1}{1-p}, \quad c(\mathcal{B}_p) = \frac{1}{2p(1-p)}, \qquad h(\mathcal{B}_p) = -p \log p - (1-p)\log(1-p).$$

The constant $b(\mathcal{S})$ is more difficult to compute even in the memoryless case. But, for the source $\mathcal{M}_r$, one has (see [12] for details)

$$b(\mathcal{M}_r) = \sum_{\ell \geq 0} \left( 1 + \frac{1}{r^\ell} \sum_{k=1}^{r^\ell - 1} \log \frac{k}{r^\ell} \right), \qquad b(\mathcal{M}_2) \doteq 2.639689120.$$

### 2.3. Final step

**Theorem 2.6.** *Consider a general source $\mathcal{S}$. For selection algorithms* `QuickMin`, `SelMin`, *we assume the source to be weakly–tame, and, for sorting algorithms* `QuickSort`, `InsSort`, `BubSort`, *we assume the source to be $\Lambda$–tame. Then, the mean number $S(n)$ of symbol comparisons performed by each algorithm on a sequence of $n$ words independently drawn from the same source $\mathcal{S}$ admits the asymptotic behaviour described in Table 3. Here, the constants $\kappa_i$ in the subdominant terms*[4] *involve the Euler constant $\gamma$ together with the subdominant constant of the source*[5] *$d(\mathcal{S})$:*

$$\kappa_0 = \frac{2}{h(\mathcal{S})}(\gamma - 2) + 2d(\mathcal{S}), \qquad \kappa_1 = \frac{1}{8h(\mathcal{S})}(2\gamma - 3) + \frac{d(\mathcal{S})}{4}.$$

*The errors terms $E(n), F(n)$ are not of the same type for sorting algorithms and selection algorithms.*

*For selection algorithms, still assuming the source is weakly tame. The error term $F(n)$ is of order $O(n^{1-\delta})$, with $\delta = 1/3$ for* `QuickMin`. *For* `SelMin`, *the constant $\delta$ depends on the exponent $a$ (if it exists) defined in (B.10).*

*For sorting algorithms, assuming a $\Lambda$–tame source with a given shape, we have*

- *if the source has a $S$–shape with abscissa $\delta$, then $E(n) = O(n^{1-\delta})$;*
- *if the source has a $H$–shape with exponent $\rho$, then $E(n) = n \cdot O\left(\exp[-(\log n)^{\rho}]\right)$;*
- *if the source has a $P$–shape with abscissa $\delta$, then $E(n) = n \cdot \Phi(n) + O(n^{1-\delta})$ where $n \cdot \Phi(n)$ is the expansion given by the family of imaginary poles $(s_k)$.*

**Discussion.** We now compare the asymptotic estimates for the two mean numbers, the mean number $K(n)$ of key–comparisons (column 2 of Table 3) and the mean number $S(n)$ (column 3 of Table 3). There are two types of algorithms

($a$) The "robust" algorithms for which $K(n)$ and $S(n)$ are of the same order. This is the case for three algorithms: `InsSort`, `QuickMin` and `SelMin`. Of course, the constants are different for $K(n)$ and $S(n)$, and the ratios $S(n)/K(n)$ involve coincidences of various types always between *two* words, respectively uniform coincidence $c(\mathcal{S})$, logarithmic-coincidence $b(\mathcal{S})$, or min-coincidence $a(\mathcal{S})$.

($b$) The algorithms for which $S(n)$ and $K(n)$ are not of the same order, here `QuickSort` and `BubSort`. In both cases, the ratio $S(n)/K(n)$ satisfies

$$\frac{S(n)}{K(n)} \sim \frac{1}{2}D(n), \qquad \text{with} \quad D(n) = \frac{1}{h(\mathcal{S})}\log n.$$

As it is proven[6] in [2] for a $\Lambda$–tame source, the factor $nD(n)$ is asymptotic to the mean path length of a trie built on $n$ words independently drawn from the source. This coincides with the mean number of symbol comparisons needed to completely distinguish these $n$ words. Then $D(n)$ is the expected depth of the complete trie built on these $n$ words. The surprise comes from the factor $1/2$; this suggests that the complete trie is not necessary to distinguish pairs of words which will be compared in `QuickSort` and `BubSort`: define the "lazy" trie as the trie which is built in a "lazy" way, only when the algorithm asks the comparison between two words. Then the "lazy" trie has an expected depth equal to the half of the expected depth of the "complete" trie.

[4]The constant $\kappa_2$ is not computed here. Note that the computation of the subdominant term for `InsSort` needs the singular expansion of $\varpi(s)/(s-1)$ at $s = 1$.

[5]This constant, defined as the constant term in the singular expansion of $\Lambda(s)$ at $s = 1$, is easy to compute for any source $\mathcal{B}_p$: $d(\mathcal{B}_p) = (1/h(\mathcal{B}_p))^2(p\log^2 p + (1-p)\log^2(1-p))$.

[6]The paper [2] extends the result already known for simple sources [17] to dynamical sources, but the proof is easily adapted for a general $\Lambda$–tame source

| Algorithms | $\pi(i,j)$ | $K(n)$ | $\sigma_0$ | $\varpi(s,u,t),\ \Re s > \sigma_0$ |
|---|---|---|---|---|
| `QuickSort` | $\dfrac{2}{j-i+1}$ | $2n\log n$ | 1 | $2(t-u)^{s-2}$ |
| `InsSort` | $\dfrac{1}{2}+\dfrac{1}{(j-i+1)(j-i)}$ | $\dfrac{n^2}{4}$ | 2 | $(s-1)(t-u)^{s-2}$ |
| `BubSort` | $\dfrac{1}{2}+\dfrac{1}{(j-i+1)(j-i)}+$ $+\dfrac{2(i-1)}{(j-i+2)(j-i+1)(j-i)}$ | $\dfrac{n^2}{2}$ | 2 | $(s-1)(t-u)^{s-3}[t-(s-1)u]$ |
| `QuickMin` | $\dfrac{2}{j}$ | $2n$ | 1 | $2t^{s-2}$ |
| `SelMin` | $\dfrac{1}{i(i+1)}+\dfrac{1}{j(j-1)}$ | $n$ | 1 | $(s-1)[u^{s-2}+t^{s-2}]$ |

(A) Table 1: results for Steps 1 and 2 (Section 2.1)

| Algorithms | $\varpi(s)$ | $\sigma_0$ | Main term of $\varpi(s)/(s-\sigma_0)$ |
|---|---|---|---|
| `QuickSort` | $\dfrac{2\Lambda(s)}{s(s-1)}$ | 1 | $\dfrac{2}{h(\mathcal{S})}\dfrac{1}{(s-1)^3}$ |
| `InsSort` | $\dfrac{\Lambda(s)}{s}$ | 2 | $\dfrac{c(\mathcal{S})}{2}\dfrac{1}{(s-2)}$ |
| `BubSort` | $-\Lambda[F_0](s-1) = -\sum_{w\in\Sigma^\star} a_w p_w^{s-1}$ | 2 | $-\dfrac{1}{2h(\mathcal{S})}\dfrac{1}{(s-2)^2}$ |
| `QuickMin` | $2\sum_{w\in\Sigma^\star}\int_{a_w}^{b_w}(t-a_w)t^{s-2}dt$ | 1 | $2b(\mathcal{S})\dfrac{1}{s-1}$ |
| `SelMin` | $(s-1)\sum_{w\in\Sigma^\star}(b_w-a_w)\int_{a_w}^{b_w}u^{s-2}du$ | 1 | $a(\mathcal{S})\dfrac{1}{s-1}$ |

(B) Table 2: results for Step 3 (Section 2.2)

| Algorithms | $K(n)$ | Dominant term for $S(n)$ | Subdominant terms | Remainder term |
|---|---|---|---|---|
| `QuickSort` | $2n\log n$ | $\dfrac{1}{h(S)}n\log^2 n$ | $\kappa_0 n\log n\ +\ \kappa_2 n$ | $E(n)$ |
| `InsSort` | $\dfrac{n^2}{4}$ | $\dfrac{c(\mathcal{S})}{4}n^2$ | $\dfrac{1}{h(S)}n\log n\ +\ \left(\kappa_0-\dfrac{c(\mathcal{S})}{4}\right)n$ | $E(n)$ |
| `BubSort` | $\dfrac{n^2}{2}$ | $\dfrac{1}{4h(S)}n^2\log n$ | $\left(\kappa_1+\dfrac{c(\mathcal{S})}{4}\right)n^2$ | $nE(n)$ |
| `QuickMin` | $2n$ | $2b(\mathcal{S})\,n$ | | $F(n)$ |
| `SelMin` | $n$ | $a(\mathcal{S})\,n$ | | $F(n)$ |

(C) Table 3: results for Theorem 1 (Section 2.3)

FIGURE 1. Tables summarizing results.

**Conclusion.** We show here the applicability of the method which has been described in the paper [19]. We describe a new point of view on the basic algorithms, and their analysis, which can be (partially) automatized. Our dream is to revisit all standard algorithms from a student book, with this point of view, and perform their realistic analysis.

**Acknowledgements.** This paper greatly benefited from many discussions we had with Philippe Flajolet, on the topics of the Rice formula and the tameness of sources. For these, we are truly grateful.

## References

[1] Cesaratto, E. and Vallée, B. Gaussian distribution of trie depth for dynamical sources, *submitted*.

[2] Clément, J., Flajolet, P., and Vallée, B. Dynamical sources in information theory: A general analysis of trie structures. *Algorithmica 29*, 1/2 (2001), 307–369.

[3] Dolgopyat, D. On decay of correlations in Anosov flows, *Ann. of Math.* 147 (1998) 357-390.

[4] Dolgopyat, D. Prevalence of rapid mixing (I) *Ergodic Theory and Dynamical Systems* 18 (1998) 1097-1114.

[5] Fill, J. A., and Janson, S. The number of bit comparisons used by Quicksort: An average-case analysis. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA04)* (2001), pp. 293–300, long version *Electronic Journal of Probability* 17, Article 43, 1-22 (2012).

[6] Fill, J. A., and Nakama, T. Analysis of the expected number of bit comparisons required by Quickselect. *Algorithmica* 58:730-769 (2010).

[7] Fill, J. A. Distributional convergence for the number of symbol comparisons used by QuickSort, *Annals of Applied Probability* (2012), to appear, available from `http://www.ams.jhu.edu/~fill`.

[8] Fill, J. A., and Nakama, T. Distributional Convergence for the Number of Symbol Comparisons Used by QuickSelect. *Submitted* 2012.

[9] Flajolet, P., Roux, M. and Vallée, B. Digital trees and memoryless sources: from arithmetics to analysis Proceedings of AofA'10, *DMTCS*, proc AM, pp 231–258 (2010)

[10] Flajolet, P., and Sedgewick, R. Mellin transforms and asymptotics: finite differences and Rice's integrals. *Theoretical Computer Science 144*, 1–2 (June 1995), 101–124.

[11] Flajolet, P., and Sedgewick, R. *Analytic Combinatorics*. Cambridge University Press, 2008.

[12] Grabner, P., and Prodinger, H. On a constant arising in the analysis of bit comparisons in Quickselect. *Quaest. Math.* 31, (2008), 303–306.

[13] Nörlund, N. E. Leçons sur les équations linéaires aux différences finies. In *Collection de monographies sur la théorie des fonctions*. Gauthier-Villars, Paris, 1929.

[14] Nörlund, N. E. *Vorlesungen über Differenzenrechnung*. Chelsea Publishing Company, New York, 1954.

[15] Roux, M. and Vallée, B. Information theory: Sources, Dirichlet series, and realistic analysis of data structures, Proceedings of Words, 11, *Electronic Proceedings of Theoretical Computer Science*, Volume 63, pp 199-214 (2011)

[16] Sedgewick, R. *Algorithms in C, Parts 1–4*, third ed. Addison–Wesley, Reading, Mass., 1998.

[17] Szpankowski, W. *Average-Case Analysis of Algorithms on Sequences*. John Wiley, 2001.

[18] Vallée, B. Dynamical sources in information theory: Fundamental intervals and word prefixes. *Algorithmica 29*, 1/2 (2001), 262–306.

[19] Vallée, B., Clément, J., Fill, J. A., and Flajolet, P. The number of symbol comparisons in QuickSort and QuickSelect. In *ICALP 2009, Part I* (2009), S. A. *et al.*, Ed., vol. 5555 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 750–763. Proceedings of the 36th International Colloquium on Automata, Languages and Programming.

## Appendix A. Probabilistic study of key–comparisons. Proofs for Step 1.

The present appendix aims at describing proofs for Proposition 2.1 that provides expressions for expectations $\pi(i,j)$. We will see that the event "$U_i$ and $U_j$ are compared" is generally "similar" to an event of the type "The arrival times of the keys $U_i$ and $U_j$ into a given subset $\mathcal{U}$ of keys are the first two (resp. the last two)". For a subset $\mathcal{U}$ of cardinality $\ell$, the probability of such an event is $1/\ell(\ell-1)$. Moreover, the subset $\mathcal{U}$ is often a subset $U_{[x,y]}$ which gathers all the keys whose rank belongs to the interval $[x,y]$, with three main cases, according to the algorithms: $[x,y]=[1,i]$, $[x,y]=[1,j]$, or $[x,y]=[i,j]$, which entails that $\ell$ belongs to $\{i, j, j-i+1\}$.

Then, the mean numbers $\pi^{\pm}(i,j)$, and their analogs $\widehat{\pi}(u,t)$ admit a general form which will makes easy the sequel of the computations, namely the automatic transfer obtained in Appendix B.1.

We first briefly recall in Figure 2 five algorithms (for precisions, see [16]).

`QuickSort` algorithm

```
Input: An array V[1..n]
Result: The sorted array V[1..n]
pivot ⟵ V[1];
(V⁻, V⁺) ⟵ Partition(pivot, V);
QuickSort(V⁻);
QuickSort(V⁺);
```

`QuickMin` algorithm

```
Input: An array V[1..n]
Output: The minimum key of
        V[1..n]
pivot ⟵ V[1];
(V⁻, V⁺) ⟵ Partition(pivot, V);
if V⁻ = ∅ then
    return pivot
else return QuickMin(V⁻);
```

`InsSort` algorithm

```
Input: An array V[1..n]
Result: The sorted array V[1..n]
for i from 2 to n do
    for j from i downto 2 do
        if V[j − 1] ≥ V[j] then
            swap(V[j], V[j − 1])
```

`BubSort` algorithm

```
Input: An array V[1..n]
Result: The sorted array V[1..n]
for i from 1 to n − 1 do
    for j from n downto i + 1 do
        if V[j − 1] > V[j] then
            swap(V[j − 1], V[j])
```

`SelMin` algorithm

```
Input: An array V[1..n]
Output: The minimum key of
        V[1..n]
Min ⟵ V[1];
for i from 2 to n do
    if V[i] < Min then Min = V[i]
return Min
```

FIGURE 2. Five basic algorithms: `QuickSort`, `QuickMin`, `InsSort`, `BubSort`, `SelMin`.

### A.1. Algorithms `QuickSort` and `QuickMin`

These algorithms are based on the "Divide and Conquer" principle. All the keys are compared to the first key of the array that is used as a pivot. During the `Partition` stage, the keys that are smaller than the pivot are placed on its left (in the sub–array $V^-$), whereas the keys that are greater are placed on its right (in the subarray $V^+$). After this partitioning, the pivot is at the right place.

Then, the `QuickSort` algorithm recursively sorts the two sub-arrays, $V^-$ and $V^+$. While the pivot does not belong to the subset $U_{[i,j]}$, this set is not separated by the pivot. When the pivot belongs to the subset $U_{[i,j]}$, the keys $U_i$ and $U_j$ may be compared only if $U_i$ or $U_j$ is a pivot. This event coincides with the event "$U_i$ or $U_j$ is the first key-in inside the subset $U_{[i,j]}$". After such a comparison, the keys are separated and no longer compared. Then, the probability $\pi(i,j)$ equals $2/(j-i+1)$.

The algorithm `QuickMin` is a particular case (for $m=1$) of the `QuickSelect`$(m)$ algorithm which returns the key of rank $m$. The `QuickMin` algorithm also uses the first key of the array as a pivot and performs the partition operation. If $V^-$ is not empty, the minimum belongs to $V^-$. Otherwise the pivot is the minimum. In the `QuickMin` algorithm, as in the `QuickSort` algorithm, the keys $U_i$ and $U_j$ are compared only if $U_i$ or $U_j$ is a pivot. This event coincides with the event "$U_i$ or $U_j$ is the first key-in inside the subset $U_{[1,j]}$". Then, the probability $\pi(i,j)$ equals $2/j$.

### A.2. Algorithm `InsSort`

There are $n-1$ phases in the algorithm. During the $i$-th phase, the key $V_i$ of the array is inserted into the left sub-array which contains an already sorted sequence built on the set $\{V_1, V_2, \ldots, V_{i-1}\}$.

*First case. $U_i$ and $U_j$ arrive in the wrong order in the initial array* ($\tau(U_i) > \tau(U_j)$). In the phase when $U_i$ is inserted into the left sub-array, this sub–array already contains $U_j$ with $U_j > U_i$, and the key $U_i$ is always compared and exchanged with $U_j$. This event is defined as "Inside the two keys set $\{U_i, U_j\}$, $U_j$ is the first–in key, and $U_i$ is the second–in key" and the probability of such an event is $\pi^-(i,j) = 1/2$.

*Second case. $U_i$ and $U_j$ arrive in the good order in the initial array* ($\tau(U_i) < \tau(U_j)$). The comparison does not always occur. In the phase when $U_j$ is inserted into the left sub–array, this left sub–array already contains the key $U_i$. If this left sub-array contains one of the keys of the subset $U_{[i,j[}$, then $U_j$ "meets" (i.e., is compared to) this key before meeting $U_i$ and remains on its right. Finally, the comparison between $U_i$ and $U_j$ occurs only if the subset $U_{]i,j[}$ arrives after $U_j$. This defines the event "$U_i$ is the first–in key and $U_j$ is the second–in key inside the set $U_{[i,j]}$". The probability of such an event is

$$\pi^+(i,j) = \frac{1}{(j-i+1)(j-i)}.$$

### A.3. Algorithm `BubSort`

As its name says, the algorithm pushes the smallest keys to the left of the array as the air bubbles on to the surface of a liquid. The algorithm performs $n-1$ phases. During each phase, the algorithm steps through the array, compares each pair of adjacent keys and

swaps them if they are in the wrong order. The $i$-th phase aims at finding the key of rank $i$ and place it in the position $i$ of the array. After the $i$-th phase, the keys of $U_{[1..i]}$ are at their right places. The `BubSort` algorithm may perform several comparisons between two keys $U_i$ and $U_j$. We are now interested in the first comparison between $U_i$ and $U_j$ and we distinguish two cases:

*First case. $U_i$ and $U_j$ arrive in the right order in the initial array* ($\tau(U_i) < \tau(U_j)$). If there is one key of $U_{]i,j[}$ which arrives after $U_i$ and before $U_j$, it will stay between $U_i$ and $U_j$ in the array thereafter, and will prevent $U_i$ and $U_j$ from meeting each other. If it arrives after $U_j$, it will eventually come between $U_i$ and $U_j$ in the array before these two keys meet each other. Hence, there is a comparison between $U_i$ and $U_j$ only if all the keys of the subset $U_{]i,j[}$ arrive before both $U_i$ and $U_j$. This coincides with the event "the key $U_j$ is the last–in and the key $U_i$ arrived just before inside the subset $U_{[i,j]}$". The probability that the first comparison between $U_i$ and $U_j$ occurs is

$$\frac{1}{(j-i+1)(j-i)}.$$

*Second case. $U_i$ and $U_j$ arrive in the wrong order in the initial array* ($\tau(U_j) < \tau(U_i)$). The first comparison between $U_i$ and $U_j$ occurs just before they are swapped. The probability of the event "$U_j$ is the first–in key and $U_j$ is the second–in key in $\{U_i, U_j\}$" is $1/2$.

*Subsequent comparisons.* There might be subsequent comparisons between two keys. Note that, in both previous cases, immediately after the first comparison (either positive or negative) $U_i$ and $U_j$ are in the right order and in consecutive positions. A necessary condition for having at least one subsequent comparison between $U_i$ and $U_j$ is that all the keys of $U_{]i,j[}$ are still on the left of $U_i$ after this point (for the same reasons exposed previously in Case 1). Now we also remark that any key $U_\ell$ with $\ell \in [1, i[$ which arrived after $U_{]i,j[}$ and before $U_i$ in the first case, and after $U_{]i,j[}$ and before $U_j$ in the second case, will be the cause of a stop of key $U_i$ during some latter phases (such a key $U_\ell$ will never be swapped with $U_i$ because of its smaller value). Also each time a key $U_i$ is stopped during a phase by a key from $U_{[1,i[}$, the set of keys from $U_{[1,i[}$ between $U_{]i,j[}$ and $U_i$ decreases by one during the same phase. After such a phase, as all keys to the right of $U_i$ are in $U_{[j,n]}$, the key $U_j$ during the next phase will be swapped until reaching $U_i$ (and results in a comparison). In conclusion the number of subsequent comparisons is exactly the number of keys from $U_{[1,i[}$ which arrived after $U_{]i,j[}$ and before $U_i$ in the first case and before $U_j$ in the second case. For any $\ell \in [1..i[$, the probabilities that $U_\ell$ arrives after $U_{]i,j[}$ and before $U_i$ (and $U_j$ arrives after $U_i$ – Case 1) or after $U_{]i,j[}$ and before $U_j$ (and $U_i$ arrives after $U_j$ – Case 2) have the same expression

$$\frac{1}{(j-i+2)(j-i+1)(j-i)}.$$

Using independence of events for $\ell \in [1, i[$, this yields that the mean number of subsequent (positive) comparisons (summing up for both Cases 1 an 2) is

$$\frac{2(i-1)}{(j-i+2)(j-i+1)(j-i)}.$$

To conclude, one has

$$\pi^+(i,j) = \frac{1}{(j-i+1)(j-i)} + \frac{2(i-1)}{(j-i+2)(j-i+1)(j-i)}, \quad \pi^-(i,j) = \frac{1}{2}.$$

### A.4. Algorithm `SelMin`.

The algorithm `SelMin` is the first phase of `SelectionSort`. This is the most natural strategy for finding the minimum key of an array. The variable called `Min` is initiated with the first key $V_1$. While stepping through the array, each key is compared with `Min` and replaces it if it is smaller. Then, the variable `Min` memorises all the possible *déjà vus* minima, namely the successive left to right minima of the array. We recall that a left to right minimum of an array is the smallest key amongst all the keys which are on its left. If two keys $U_i$ and $U_j$ are compared, the first–in key of the set $\{U_i, U_j\}$ is a left to right minimum.

*First case. $U_i$ and $U_j$ arrive in the right order* ($\tau(U_i) < \tau(U_j)$). Then $U_i$ is a left-to-right minimum, and $U_j$ must arrive before the following left-to-right minimum, namely before all the keys of $U_{[1,i[}$. Finally, inside the set $U_{[1,i]} \cup \{U_j\}$, of cardinality $i+1$, the key $U_i$ is the first–in, and $U_j$ is the second–in. The probability of this event is $\pi^+(i,j) = 1/(i(i+1))$.

*Second case. $U_i$ and $U_j$ arrive in the wrong order.* ($\tau(U_i) > \tau(U_j)$). Then $U_j$ is a left-to-right minimum and $U_i$ is the following left-to-right minimum. This means that all the keys of the set $U_{[1,i[} \cup U_{]i,j[}$ arrived after $U_i$. Inside the set $U_{[1,j]}$ of cardinality $j$, $U_j$ is the first–in key, $U_i$ is the second–in key. The probability of such an event is $\pi^-(i,j) = 1/(j(j-1))$.

## Appendix B. Proofs for Steps 2 and 3.

This appendix provides elements of proofs for Propositions 2.2, 2.3, 2.4, 2.5. Appendix B.1 focusses on the "automatic" transfer between expectations $\pi(i,j)$ and coefficients $\varphi(n,u,t)$ which will entail, together with Proposition 2.1, Proposition 2.2 of Section 2.1. Then, the sequel of this appendix is devoted to the proofs of Propositions 2.3, 2.4, 2.5. Appendix B.2 focusses on the case of sorting algorithms, whereas Appendix B.3 describes the proofs for selection algorithms.

### B.1. Automatic transfer from Step 1 to Step 2

We first explain how we transfer the mean number $\pi(i,j)$ of key–comparisons into the Dirichlet terms $\varpi(s,u,t)$.

**Proposition B.1.** *The following holds:*

(*a*) *Consider a variable $X$ which follows a Poisson law of parameter $Z$, and, for $m \geq 1$, the variable $\pi_m(X) := 1/(X+1)(X+2)\dots(X+m)$. Denote by $F_m(Z)$ the expectation of the variable $\pi_m(X)$. Then, the two sequences*

$$\beta_m(n,\lambda) = (-1)^n n![Z^n]\left(Z^2\, F_m(\lambda Z)\right), \qquad \gamma_m(n,\lambda) := (-1)^n n![Z^n]\left(Z^3\, F_m(\lambda Z)\right).$$

*admit the following expressions, resp. for $n > 1$ and $n > 2$,*

$$\beta_m(n,\lambda) = \frac{1}{(m-1)!}\frac{n(n-1)}{n+m-2}\lambda^{n-2}, \qquad \gamma_m(n,\lambda) = \frac{-1}{(m-1)!}\frac{n(n-1)(n-2)}{n+m-3}\lambda^{n-3}. \tag{B.1}$$

(*b*) *For any of the five algorithms, the random variable $\widetilde{\pi}(u,t)$, equal to $\widehat{\pi}(u,t)$ up to the possible constant term $1/2$, can be expressed in the "basis" $\pi_m$, as displayed in the second column of the following table.*

(*c*) *For any of the five algorithms, there exists an integer $\sigma_0$, for which the coefficients $\varphi(n,u,t)$ of the density $\Phi_Z(u,t)$ can be expressed for $n > \sigma_0$ as a linear combination of $\beta_m(n,\lambda)$ and $\gamma_m(n,\lambda)$ for $\lambda \in \{u, t, t-u\}$, as displayed in the third column of the table. The integer $\sigma_0$ is displayed in the fourth (and last) column of the table.*

| Algorithms | $\widetilde{\pi}(u,t)$ (in the "basis" $\pi_i$) | $\varphi(n,u,t)$ (in the "basis" $\beta_i, \gamma_j$) | $\sigma_0$ |
|---|---|---|---|
| `QuickSort` | $2[\pi_1(N_{[u,t[}) - \pi_2(N_{[u,t[})]$ | $2[\beta_1(n,t-u) - \beta_2(n,t-u)]$ | 1 |
| `InsSort` | $\pi_2(N_{[u,t[})$ | $\beta_2(n,t-u)$ | 2 |
| `BubSort` | $\pi_2(N_{[u,t[}) + 2N_{[0,u[} \cdot \pi_3(N_{[u,t[})$ | $\beta_2(n,t-u) + 2u\gamma_3(n,t-u)$ | 2 |
| `QuickMin` | $2[\pi_1(N_{[0,t[}) - \pi_2(N_{[0,t[})]$ | $2[\beta_1(n,t) - \beta_2(s,t)]$ | 1 |
| `SelMin` | $\pi_2(N_{[0,u[}) + \pi_2(N_{[0,t[})$ | $\beta_2(n,u) + \beta_2(n,t)$ | 1 |

*Proof.* We first compute the coefficients $\alpha_m(n,\lambda)$ of $F_m(Z)$

$$\alpha_m(n) := (-1)^n n![Z^n]F_m(Z) = \frac{1}{(m-1)!}\frac{1}{n+m}. \tag{B.2}$$

Then, the coefficients $\beta_m(n,\lambda)$ and $\gamma_m(n,\lambda)$ are related to $\alpha_m(n)$, resp. for $n > 1$ and $n > 2$

$$\beta_m(n,\lambda) = n(n-1)\,\lambda^{n-2}\alpha_m(n-2), \qquad \gamma_m(n,\lambda) = -n(n-1)(n-2)\,\lambda^{n-3}\alpha_m(n-3),$$

which proves, with (B.2), the expressions (B.1) of Assertion (*a*).

*Assertion* (*b*). We begin with the expressions of $\pi(i,j)$ displayed in the Table 1 p. 12, and we obtain with (1.4) expressions for the random variable $\widetilde{\pi}(u,t)$ displayed in the second column of the table above.

*Assertion* (*c*). Now, the third column is obtained from the second one by "taking the expectations" in the Poisson model, multiply by $Z^2$ and extracting the coefficient of order $n$. All the expressions of the second column are linear combinations, except the term $N_{[0,u[}\cdot\pi_3(N_{]u,t[})$, which involves the product of two independent variables, whose expectation is thus the product of expectations, namely

$$Z^2 \cdot \mathbb{E}_Z(N_{[0,u[}) \cdot \mathbb{E}_Z\left(\pi_3(N_{]u,t[})\right) = u \cdot Z^3 F_3(Z(t-u)).$$

■

The explicit expressions of $\varphi(n,u,t)$ deduced from the decompositions described in the third column together with the expressions (B.1) yields expressions for $\varpi(s,u,t)$ reported in the Table 1 p. 12 (for $s > \sigma_0$). We recall that the link between $\varphi(n,u,t)$ and $\varpi(s,u,t)$ (essentially – but not only – a change of variable $n \to s$) is given in Section 1.6.

### B.2. Analytic Study of the mixed Dirichlet series for sorting algorithms.

In the three cases, the mixed Dirichlet series is closely related to the Dirichlet series $\Lambda(s)$ of the source, and the transfer of tameness between $\Lambda(s)$ and $\varpi(s)$ is easy.

In the sequel, $\sigma$ denotes the real part of $s$, i.e., $\sigma := \Re s$.

**Case of `QuickSort` and `InsSort`.** The integral of $\varpi(s,u,t) = (s-1)(t-u)^{s-2}$ on the fundamental triangle $\mathcal{T}_w$ equals $(1/s)p_w^s$. This entails the nice formulae for both $\varpi(s)$,

$$\varpi_Q(s) = \frac{\Lambda(s)}{s(s-1)}, \qquad \varpi_I(s) = \frac{\Lambda(s)}{s}.$$

Then, the functions $s \mapsto \varpi(s)$ are tame at $s=1$. Moreover, the shape of tameness of $\varpi(s)$ at $s=1$ coincides with the shape of $\Lambda$–tameness of the source. For `InsSort`, the function $\varpi_I(s)$ has a simple dominant pole at $s=1$ with a residue equal to $1/h(S)$, whereas, for `QuickSort`, the function $\varpi_I(s)$ has a dominant pole at $s=1$ of order 2. Moreover, the singular expressions of the functions $\varpi(s)/(s-1)$ can be easily computed from the singular expression of $\Lambda(s)$.

**Case of `BubSort`.** The integral of $\varpi(s,u,t) = (s-1)(t-u)^{s-3}[t-(s-1)u]$ on the fundamental triangle equals $-a_w {p_w}^{s-1}$. Then, the Dirichlet series $\varpi(s)$ admits the expression

$$\varpi(s) = -\sum_{w\in\Sigma^\star} a_w {p_w}^{s-1} = -\Lambda[F_0](s-1),$$

where $F_0(x,y) = x$. By hypothesis, the series $s \mapsto \Lambda[F_0](s)$ is tame at $s=1$. Then, the series $\varpi(s)$ is tame at $s=2$, with the same shape of tameness as the series $s \mapsto \Lambda[F_0](s)$. We now study its precise behaviour at $s=2$. We remark, with the relation $\Lambda_\ell(1) = \Lambda_\ell(1)^2 = 1$, the equality

$$2\Lambda_\ell[F_0](1) = 2\sum_{w\in\Sigma^\ell} a_w p_w = 2\sum_{w\in\Sigma^\ell}\left[\sum_{w'<w} p_{w'}\right] p_w = \Lambda_\ell(1)^2 - \sum_{w\in\Sigma^\ell} p_w^2 = \Lambda_\ell(1) - \Lambda_\ell(2). \quad \text{(B.3)}$$

The series

$$L(s) := \sum_{\ell\geq 0} L_\ell(s) \qquad \text{with} \quad L_\ell(s) := 2\Lambda_\ell[F_0](s) - \Lambda_\ell(s),$$

is convergent at $s=1$ and satisfies

$$L(1) = \sum_{\ell\geq 0} L_\ell(1) = -\sum_{\ell\geq 0} \Lambda_\ell(2) = -\Lambda(2) = -c(\mathcal{S}),$$

where $c(\mathcal{S})$ is the coincidence of the source defined in Proposition 2.5. Since $\Lambda(s)$ admits a simple pole at $s=1$ with a residue equal to $1/h(\mathcal{S})$, then $\Lambda[F_0](s)$ admits a simple pole at $s=1$ with a residue equal to $1/2h(\mathcal{S})$. More precisely, as the singular expansion of $\Lambda(s)$ at $s=1$ is

$$\Lambda(s) = \frac{1}{h(\mathcal{S})}\frac{1}{s-1} + d(\mathcal{S}) + O(s-1),$$

the singular expansion of $\varpi(s)$ at $s=2$ is

$$\varpi(s) = -\frac{1}{2h(\mathcal{S})}\frac{1}{s-2} + \frac{1}{2}(c(\mathcal{S}) - d(\mathcal{S})) + O(s-2).$$

Remark that the equation (B.3) can be generalized to any function $F$ of class $\mathcal{C}^1$. The sum of interest can be viewed as a Riemann sum on the fundamental intervals of depth $\ell$, so that

$$\sum_{w\in\Sigma^\ell} F(a_w, b_w) p_w = I[F] + \rho_\ell[F], \quad \text{with} \quad I[F] := \int_0^1 F(t,t)dt \quad \text{and} \quad |\rho_\ell[F]| \leq \|F\|_1 \cdot \Lambda_\ell(2)\,.$$

This entails that, for any function $F$ whose integral $I[F]$ is not zero, the Dirichlet series $\Lambda[F](s)$ has a residue at $s=1$ equal to $I[F]/h(\mathcal{S})$.

### B.3. Analytic study of the mixed Dirichlet series for `QuickMin` and `SelMin`

In this case, the analytic study of $\varpi(s)$ is less easy, because $\varpi(s)$ involves not only the probabilities $p_w$, but also the integral of a function over each fundamental interval $[a_w, b_w]$. We first consider the mixed series $\varpi_\ell(s)$ of depth $\ell$, relative to fundamental intervals of depth $\ell$, and we prove that the series of general term $\varpi_\ell(s)$ is normally convergent.

**Case of `QuickMin`.** The Dirichlet series of depth $\ell$ is

$$\varpi_\ell(s) := 2 \sum_{w\in\Sigma^\ell} \int_{a_w}^{b_w} (t-a_w) t^{s-2} dt.$$

For $\sigma \geq 2$, each term of the series is at most equal to $p_w^2$ in modulus and the series $\varpi_\ell(s)$ satisfies $|\varpi_\ell(s)| \leq \Lambda_\ell(2)$.

For $\sigma \in ]0,2]$, we consider some real $A \in [0,1]$ (to be fixed later as a function of $s$ and $\ell$) and split the sum into three sums, each of them relative to a subset of prefixes: the prefixes $w$ for which $b_w < A$, the prefixes $w$ for which $a_w > A$ and finally the unique prefix $\alpha$ for which $A \in [a_\alpha, b_\alpha]$. These sums are respectively denoted by $\varpi_\ell^{(-)}(s), \varpi_\ell^{(+)}(s), \varpi_\ell^{(=)}(s)$.

For $b_w \leq A$, we use the inequality

$$\int_{a_w}^{b_w} (t-a_w) t^{\sigma-2} dt \leq \int_{a_w}^{b_w} t^{\sigma-1} dt, \qquad \text{so that} \quad |\varpi_\ell^{(-)}(s)| \leq 2 \int_0^A t^{\sigma-1} dt = \frac{2}{\sigma} A^\sigma.$$

For $a_w \geq A$, we observe that

$$\int_{a_w}^{b_w} (t-a_w) t^{\sigma-2} dt \leq \frac{1}{2} A^{\sigma-2} p_w^2, \qquad \text{so that} \quad |\varpi_\ell^{(+)}(s)| \leq A^{\sigma-2} \Lambda_\ell(2).$$

We now choose $A$ such that the two previous bounds are equal, namely

$$A = \Big(\frac{\sigma}{2}\, \Lambda_\ell(2)\Big)^{1/2}, \qquad \text{so that} \quad |\varpi_\ell^{(-)}(s)| + |\varpi_\ell^{(+)}(s)| \leq C_1(\sigma)\, \Lambda_\ell(2)^{\sigma/2},$$

where $C_1(\sigma)$ is bounded for $\sigma \geq \sigma_2$ (for any $\sigma_2 > 0$). The middle part $\varpi_\ell^{(=)}(s)$ corresponds to the fundamental interval $[a_\alpha, b_\alpha]$ of length $p_\alpha \leq \Lambda_\ell(2)^{1/2}$, and

$$|\varpi_\ell^{(=)}(s)| \leq 2 \int_{a_\alpha}^{b_\alpha} t^{\sigma-1} dt \leq \frac{2}{\sigma} (A+p_\alpha)^\sigma \leq C_2(\sigma)\, \Lambda_\ell(2)^{\sigma/2},$$

where $C_2(\sigma)$ is bounded for $\sigma \geq \sigma_2$ (for any $\sigma_2 > 0$). The well–known log-convexity of the function $s \mapsto \Lambda_\ell(\sigma)$ and the equality $\Lambda_\ell(1) = 1$, imply the inequalities

$$\Lambda_\ell(2)^{\sigma/2} = \Lambda_\ell(2)^{\sigma/2} \times \Lambda_\ell(1)^{\sigma/2} \leq \Lambda_\ell\left(2 \times \frac{\sigma}{2} + 1 \times \frac{\sigma}{2}\right) = \Lambda_\ell\left(\frac{3\sigma}{2}\right), \tag{B.4}$$

which leads to the final bounds, with $C(\sigma) := \max\left(C_1(\sigma), C_2(\sigma)\right)$

$$|\varpi_\ell(s)| \leq C(\sigma)\, \Lambda_\ell\left(\frac{3\sigma}{2}\right), \qquad |\varpi(s)| \leq C(\sigma)\, \Lambda\left(\frac{3\sigma}{2}\right).$$

When the source is weakly tame, then, the function $\varpi(s)$ is tame at $\sigma_0 = 1$, of order 0, with a $S$–shape, related to the vertical strip $\Re s \geq \sigma_1$ for any $\sigma_1 > 2/3$.

The value $\varpi(1)$, denoted by $2b(\mathcal{S})$ in Proposition 2.5, is easily computed,

$$\varpi(s) := \sum_{w\in\Sigma^\star} \int_{\mathcal{T}_w} \frac{1}{t}\, du\, dt\,.$$

The set $\mathcal{P}$ which gathers all the prefixes of the smallest word emitted by the source is particular. If $q_\ell$ denotes the probability of the prefix of $\mathcal{P}$ of depth $\ell$, the contribution of the set $\mathcal{P}$ to $\varpi(s)$ is

$$a(\mathcal{S}) = \sum_{w\in\mathcal{P}} p_w = \sum_{\ell\geq 0} q_\ell,$$

whereas the contribution of the set $\Sigma^\star \setminus \mathcal{P}$ is equal to

$$\sum_{w\in\Sigma^\star\setminus\mathcal{P}} p_w \left(1 - \frac{a_w}{p_w}\log\left(1+\frac{p_w}{a_w}\right)\right).$$

Finally,

$$b(\mathcal{S}) = a(\mathcal{S}) + \sum_{w\in\Sigma^\star\setminus\mathcal{P}} p_w \left(1 - \frac{a_w}{p_w}\log\left(1+\frac{p_w}{a_w}\right)\right), \qquad a(\mathcal{S}) = \sum_{w\in\mathcal{P}} p_w = \sum_{\ell\geq 0} q_\ell.$$

We will see later that the constant $a(\mathcal{S})$ is the constant which occurs in `SelMin`.

**Case of `SelMin`.** The two Dirichlet series of interest are

$$\varpi^+(s) = (s-1)\sum_{w\in\Sigma^*}\int_{a_w}^{b_w} u^{s-2}(u-a_w)du, \qquad \varpi^-(s) = (s-1)\sum_{w\in\Sigma^*}\int_{a_w}^{b_w} u^{s-2}(b_w-u)du.$$

Then, the total Dirichlet series is

$$\varpi(s) = \varpi^+(s) + \varpi^-(s) = (s-1)\sum_{w\in\Sigma^*}(b_w-a_w)\int_{a_w}^{b_w} u^{s-2}du.$$

Under this form, the upper bound, valid for $\sigma \geq 2$,

$$|\varpi(s)| \leq |s-1|\sum_{w\in\Sigma^*}(b_w-a_w)\int_{a_w}^{b_w} u^{\sigma-2}du \leq |s-1|\Lambda(2),$$

proves that $\varpi(s)$ is tame for $\Re s \geq 2$. The sequel of the proof is devoted to the case $\sigma \leq 2$.

We first consider the Dirichlet series of depth $\ell$, namely

$$\varpi_\ell(s) = (s-1)\sum_{w\in\Sigma^\ell}(b_w-a_w)\int_{a_w}^{b_w} u^{s-2}du,$$

where $\Sigma^\ell$ is the set of the prefixes of length $\ell$. As in the case of `QuickMin`, the first term of the sum which defines $\varpi_\ell$ is particular. It is relative to the prefix $\alpha_\ell$ of depth $\ell$ of the smallest word of the source (whose probability is denoted by $q_\ell$) and equals

$$(s-1)q_\ell\int_0^{q_\ell} u^{s-2}du = q_\ell^s.$$

We then consider the remainder of the sum

$$R_\ell(s) = \sum_{\substack{w\in\Sigma^\ell,\\ w>\alpha_\ell}} (b_w-a_w)(b_w^{s-1}-a_w^{s-1}),$$

and prove that the series of general term $R_\ell(s)$ is normally convergent for $\Re s > 1 - \sigma_3$ for some $\sigma_3 > 0$. Then, the sum $R(s)$ of this series defines a tame function at $\sigma_0$ of abscissa $\sigma_2$. And the equality $R(1) = 0$ holds since on the real axis $R(s)$ is positive for $s > 1$ and negative for $s < 1$. Finally,

$$\varpi(1) = a(\mathcal{S}) = \sum_{\ell \geq 0} q_\ell, \qquad \text{with} \quad q_\ell := \text{probability of the smallest prefix of length } \ell. \tag{B.5}$$

We then study $R_\ell(s)$ for $\sigma \leq 2$. As previously, we consider some real $A \in [0, 1]$ (to be fixed later as a function of $s$ and $\ell$) and split the sum into three sums, each of them relative to a subset of prefixes: the prefixes $w$ for which $b_w < A$, the prefixes $w$ for which $a_w > A$ and finally the unique prefix $\beta$ for which $A \in [a_\beta, b_\beta]$. These sums are respectively denoted by $R_\ell^{(-)}(s), R_\ell^{(+)}(s), R_\ell^{(=)}(s)$.

When $a_w > A$, we use the mean-value theorem. There exists $c_w \in [a_w, b_w]$ for which

$$|b_w^{s-1} - a_w^{s-1}| \leq |s-1| c_w^{\sigma-2}(b_w - a_w) \leq |s-1| A^{\sigma-2}(b_w - a_w) \quad \text{and} \quad |R_\ell^{(+)}(s)| \leq |s-1| A^{\sigma-2} \Lambda_\ell(2). \tag{B.6}$$

There are two cases for the other two sums, according to the position of $\sigma$ with respect to 1.

*First case: Case when* $1 \leq \sigma \leq 2$. In this case, when $b_w < A$, we remark

$$|b_w^{s-1} - a_w^{s-1}| \leq |b_w^{s-1}| + |a_w^{s-1}| = b_w^{\sigma-1} + a_w^{\sigma-1} \leq 2A^{\sigma-1}, \quad |R_\ell^{(-)}(s)| \leq 2A^{\sigma-1} \cdot A = 2A^\sigma. \tag{B.7}$$

For the sum $R_\ell^{(=)}(s)$, we use the upper bound

$$|b_w^{s-1} - a_w^{s-1}| \leq 2, \qquad R_\ell^{(=)}(s) \leq 2\, p_\beta \leq 2\pi_\ell \leq 2\Lambda_\ell(2)^{1/2}, \tag{B.8}$$

which deals with the length $p_\beta$ of the interval $[a_\beta, b_\beta]$. Finally, with (B.6), (B.7), (B.8), one obtains in the case $1 \leq \sigma < 2$,

$$R_\ell(s) \leq A^{\sigma-2}\Lambda_\ell(2) + 2A^\sigma + 2\Lambda_\ell(2)^{1/2}.$$

We choose the value of $A$ so that the first two terms are equal, namely

$$2A^\sigma = |s-1| A^{\sigma-2} \Lambda_\ell(2), \qquad \text{i.e.,} \quad A = \left(\frac{|s-1|}{2}\right)^{1/2} \Lambda_\ell(2)^{1/2}.$$

Then, with the log-convexity bound $\Lambda_\ell(2)^{\sigma/2} \leq \Lambda_\ell(3\sigma/2)$, one has

$$|R_\ell^{(-)}(s) + R_\ell^{(+)}(s)| \leq 4 \left(\frac{|s-1|}{2}\right)^{\sigma/2} \Lambda_\ell(2)^{\sigma/2} \leq 4 \left(\frac{|s-1|}{2}\right)^{\sigma/2} \Lambda_\ell\left(\frac{3\sigma}{2}\right).$$

Finally, if the source is weakly tame, $R(s)$ is analytic and of polynomial growth for $\Re(s) \geq 1$.

*Second case: Case when* $\sigma \leq 1$. The upper bounds for $R_\ell^{(=)}(s)$ and $R_\ell^{(-)}(s)$ are

$$|R_\ell^{(-)}(s)| \leq 2q_\ell^{\sigma-1} A, \qquad |R_\ell^{(=)}(s)| \leq 2q_\ell^{\sigma-1}\pi_\ell, \tag{B.9}$$

and involve the probability $\pi_\ell := \max\{p_w, w \in \Sigma^\ell\}$. The upper bound for $|R_\ell^{(+)}(s)|$ is the same as previously (see (B.6)). We choose $A$ such that the bounds, for $|R_\ell^{(-)}(s)|$ in (B.9),

and $|R_\ell^{(+)}(s)|$ in (B.6), are equal

$$2q_\ell^{\sigma-1}A = |s-1|\Lambda_\ell(2)A^{\sigma-2}, \qquad \text{i.e.,} \quad A = \left(\frac{|s-1|}{2}\right)^{1/(3-\sigma)} \left(\Lambda_\ell(2){q_\ell}^{1-\sigma}\right)^{1/(3-\sigma)}.$$

Then, one has

$$|R_\ell^{(-)}(s) + R_\ell^{(+)}(s)| \le 4\left(\frac{|s-1|}{2}\right)^{1/(3-\sigma)} \left(\Lambda_\ell(2){q_\ell}^{1-\sigma}\right)^{1/(3-\sigma)} \cdot q_\ell^{\sigma-1}$$

To conclude in this case, we need to compare the two sequences $q_\ell$ and $\Lambda_\ell(2)$ which already satisfy $q_\ell \le \Lambda_\ell(2)^{1/2}$ and we assume that $q_\ell$ is not too small with respect to $\Lambda_\ell(2)$, namely

$$a := \limsup \left|\frac{\log q_\ell}{\log \Lambda_\ell(2)}\right| < +\infty. \tag{B.10}$$

Then, for $\ell$ large enough, one has $q_\ell \ge \Lambda_\ell(2)^a$, and

$$|R_\ell^{(-)}(s) + R_\ell^{(+)}(s)| \le |C(s)|\,\Lambda_\ell(2)^b, \qquad \text{with} \quad b = \frac{1}{(3-\sigma)}\left(1 + a(1-\sigma)(\sigma-2)\right),$$

where the function $C(s)$ is of polynomial growth. For the remainder term $R_\ell^{(=)}(s)$, one has, with (B.10) and (B.9),

$$\pi_\ell \le \Lambda_\ell(2)^{1/2} \qquad \text{so that} \quad |R_\ell^{(=)}(s)| \ll \Lambda_\ell(2)^c \qquad \text{with} \quad c = \frac{1}{2} + a(\sigma-1).$$

The two exponents $b$ and $c$ depend continuously of $\sigma$ and equal $1/2$ for $\sigma = 1$. Then, for any $b_0 > 1/3$, there exist $\sigma_3 > 0$ such that $b, c$ satisfy $b, c > b_0$ for $\sigma > 1 - \sigma_3$. Then, the convexity argument already used in (B.4) proves that

$$|R_\ell(s)| \le |C(s)|\,\Lambda_\ell(3b_0),$$

which defines a convergent series. Then, if the source is weakly tame, the related series $R(s)$ is analytic and of polynomial growth in the vertical strip $\sigma \in ]1-\sigma_3, 1]$.

Finally, we have proven that the series $\varpi(s)$ is tame, with a $S$–shape, related to the half plane $\Re s > 1 - \sigma_3$ for some $\sigma_3 > 0$. Moreover, at $s = 1$, the series satisfies (B.5).