

**NUMBER OF SYMBOL COMPARISONS
IN QUICKSORT AND QUICKSELECT**

Brigitte VALLÉE (CNRS and Université de Caen, France)

Joint work with Julien CLÉMENT and Philippe FLAJOLET

Plan of the talk.

- General framework of the study
- Statement of our results
- Description of the methods
- Sketch of the proof in the case of QuickSort.

The classical framework.

The main sorting algorithms or searching algorithms e.g., InsertionSort, SelectionSort, QuickSort, QuickSelect,... deal with n (distinct) keys U_1, U_2, \dots, U_n of the same ordered set Ω and only perform comparisons and exchanges.

Since the behaviour of the algorithm (wrt to the comparisons performed between keys) only depends on the relative order between the keys, it is sufficient to restrict to the case when $\Omega = [1..n]$. The input set is then \mathfrak{S}_n , with uniform probability.

Then, the analysis of all these algorithms is very well known, with respect to the number of key-comparisons performed in the worst-case, or in the average case.

In real life (?!)

However, **in real life**, the domain Ω of the keys is often **complex**.

It is more realistic to consider

$$\Omega \subset \Sigma^\infty = \{\text{the infinite words on some (finite) alphabet } \Sigma\}.$$

Then, the algorithm **compares words** [wrt the **lexicographic order**]
and the unit cost is now the **comparison** between **symbols**.

The **cost** of the comparison between two words (wrt this unit cost)

$$A = a_1 a_2 a_3 \dots a_i \dots$$

$$B = b_1 b_2 b_3 \dots b_i \dots$$

equals $k + 1$, where k is the **length** of their **largest common prefix**

$$k := \max\{i; \quad \forall j \leq i, \quad a_j = b_j\},$$

k is also called the **coincidence**.

The (general) model of source.

Source. A general **source** \mathcal{S} produces words on an **alphabet** Σ .
To $u \in \mathcal{I} := [0, 1]$ it associates a word $M(u) \in \Sigma^\infty$.
The lexicographic order on Σ^∞ is **compatible** with the order on \mathcal{I} .
When \mathcal{I} is endowed with a **density** f , this is a **probabilistic source**.

Fundamental intervals and fundamental probabilities.

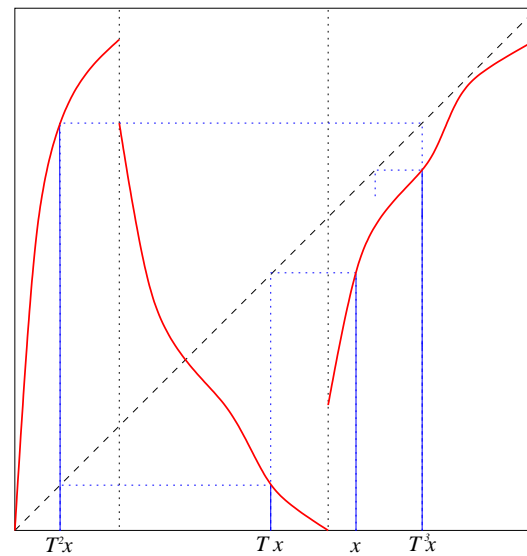
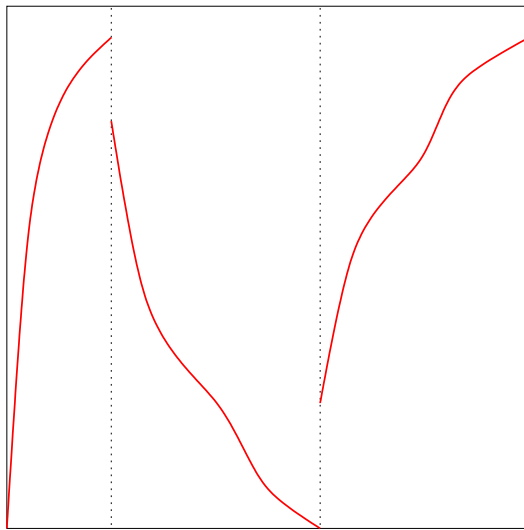
For a prefix $w \in \Sigma^*$, the interval \mathcal{I} is divided into **three** intervals.
Up to a finite number of points, each of them gathers the reals u for which $M(u)$ and w are **comparable**,

$$\begin{aligned}\mathcal{I}_w^{(-)} &:= [0, a_w] \sim \{u, \quad M(u) < w\}, \\ \mathcal{I}_w^{(+)} &:= [b_w, 1] \sim \{u, \quad M(u) > w\}, \\ \mathcal{I}_w &:= [a_w, b_w] \sim \{u, \quad M(u) \text{ begins with } w\}.\end{aligned}$$

The interval \mathcal{I}_w is the **fundamental interval** relative to the **prefix** w .

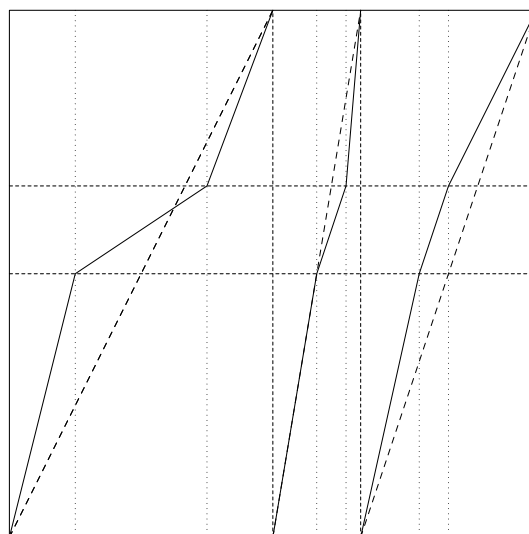
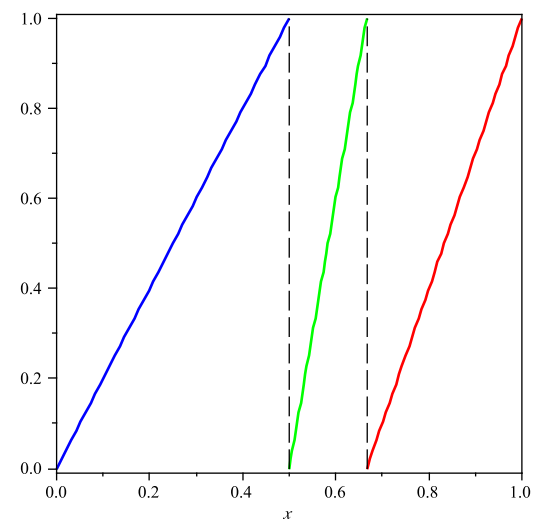
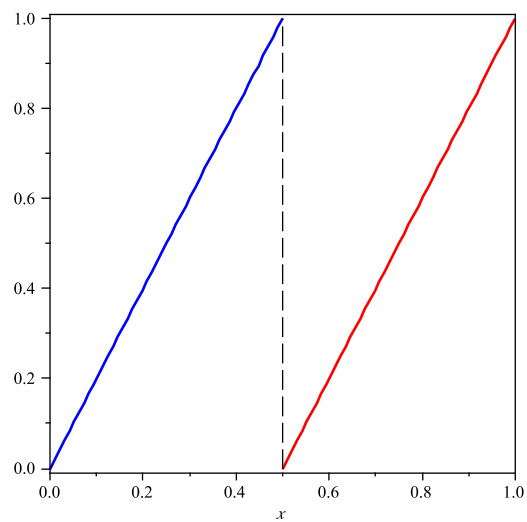
The **measure** of each interval denoted by p_w or $p_w^{(\pm)}$ is the **probability** that $M(u)$ **begins** with a prefix $w' = w, w' < w$ or $w' > w$.

Natural instances of sources: Dynamical sources

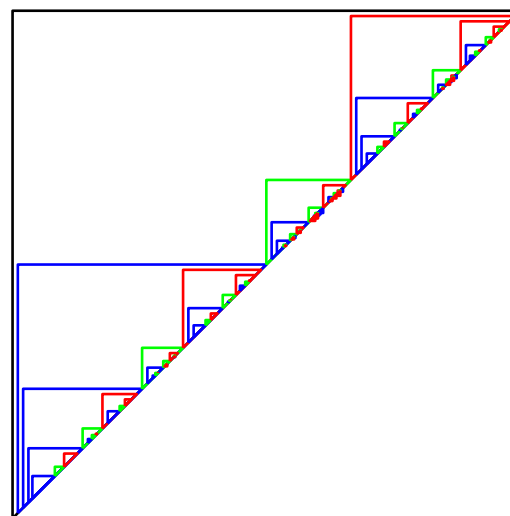
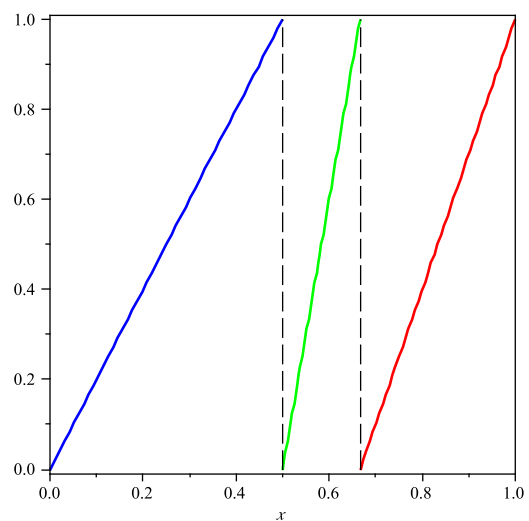
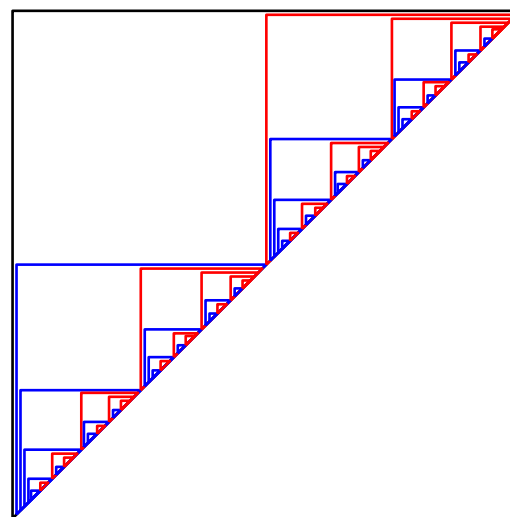
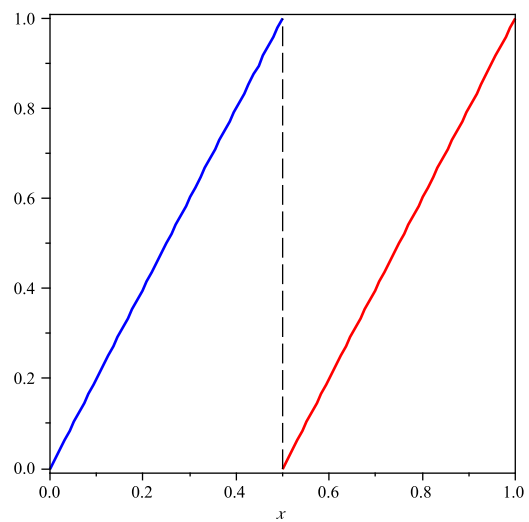


A dynamical system, with $\Sigma = \{a, b, c\}$ and a word $M(x) = (c, b, a, c \dots)$.

Instances of memoryless sources or Markov chains.



Fundamental intervals and fundamental triangles.



The mean number of symbol comparisons

Consider an algorithm $\mathcal{A}(n)$ which deals with n keys U_1, U_2, \dots, U_n independently drawn from the same **probabilistic source** \mathcal{S} .

The **mean** number $S(n)$ of **symbol-comparisons** of $\mathcal{A}(n)$ is

$$S(n) = \int_{\mathcal{T}} [\gamma(u, t) + 1] \phi_n(u, t) du dt \quad \text{and involves three main objects}$$

- (i) the **unit triangle** $\mathcal{T} := \{(u, t) : 0 \leq u < t \leq 1\}$,
- (ii) the **coincidence** (of the source) $\gamma(u, t)$ between $M(u)$ and $M(t)$
- (iii) the **density** ϕ_n of the algorithm, defined as

$\phi_n(u, t) du dt :=$ Mean number of key-comparisons performed
by the algorithm $\mathcal{A}(n)$ between a key $M(u')$ and a key $M(t')$
with $u' \in [u, u + du], t' \in [t, t + dt]$

The **mean** number $K(n)$ of **key-comparisons** of $\mathcal{A}(n)$ is $K(n) = \int_{\mathcal{T}} \phi_n(u, t) du dt$

An alternative expression for $S(n) = \int_{\mathcal{T}} [\gamma(u, t) + 1] \phi_n(u, t) du dt$

Two properties for the coincidence:

$$\sum_{\ell} (\ell + 1) \mathbf{1}_{[\gamma(u, t) = \ell]} = \sum_{\ell} \mathbf{1}_{[\gamma(u, t) \geq \ell]}$$

$$[\gamma(u, t) \geq \ell] = \bigcup_{w \in \Sigma^{\ell}} (\mathcal{I}_w \times \mathcal{I}_w) \quad \text{so that} \quad \mathcal{T} \cap [\gamma(u, t) \geq \ell] = \bigcup_{w \in \Sigma^{\ell}} \mathcal{T}_w$$

where \mathcal{T}_w is the **fundamental triangle** built on the interval \mathcal{I}_w .

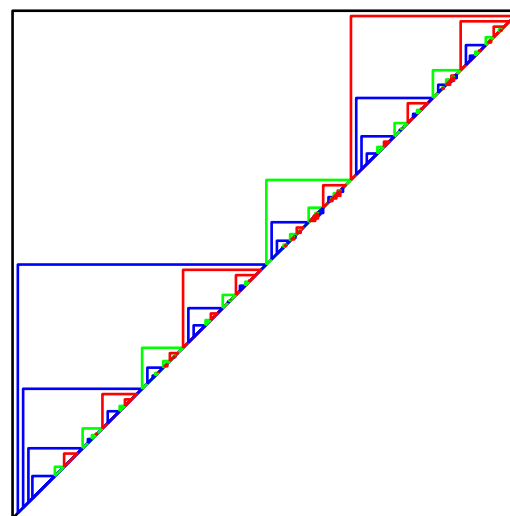
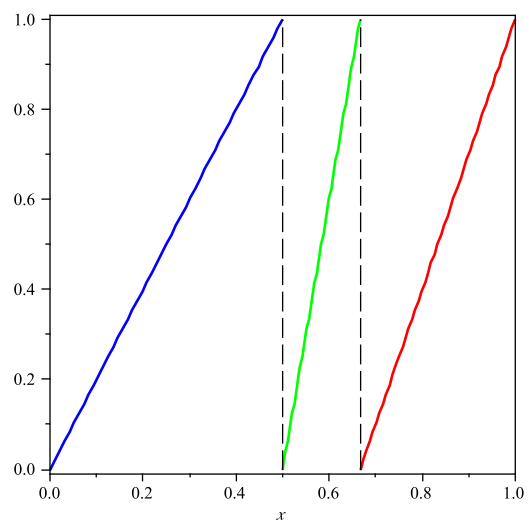
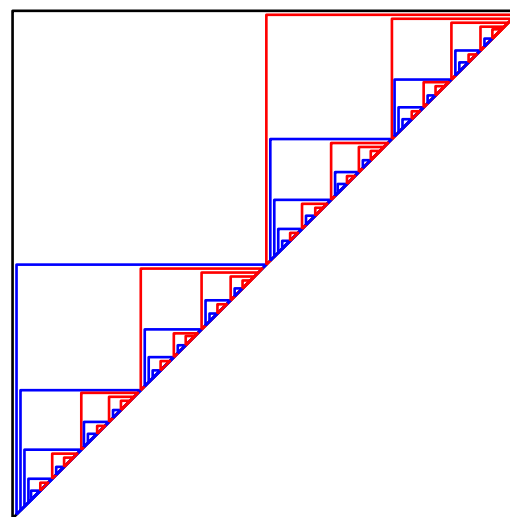
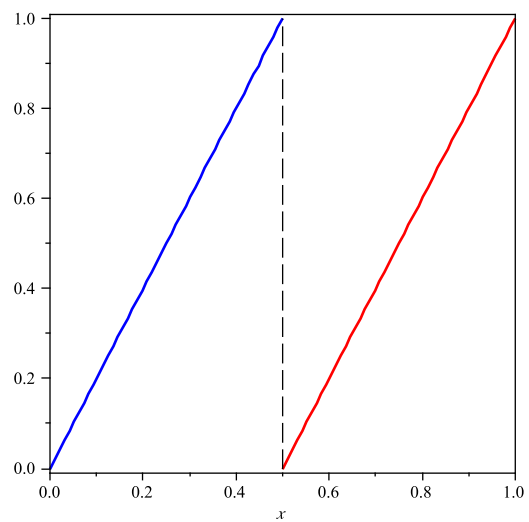
Lemma 1. *The mean number of key-comparisons of $\mathcal{A}(n)$ on \mathcal{S} satisfies:*

$$S(n) = \sum_{w \in \Sigma^*} \int_{\mathcal{T}_w} \phi_n(u, t) du dt$$

where \mathcal{T}_w is the **fundamental triangle** built on the interval \mathcal{I}_w .

The density ϕ_n and the fundamental triangles \mathcal{T}_w are then the **two main objects** of the analysis.

Fundamental intervals and fundamental triangles.



How to compute the density ϕ_n ? (I)

It involves two objects defined with the key $U^{(i)}$ of rank i .

(i) The joint distribution $\delta_{i,j,n}$ of $U^{(i)}$ and $U^{(j)}$, characteristic of the density f and defined for $i < j$ and $(u, t) \in \mathcal{T}$ as

$$\delta_{i,j,n}(u, t) du dt := \Pr[U^{(i)} = M(u') \text{ and } U^{(j)} = M(t') \\ \text{with } u' \in [u, u + du] \text{ and } t' \in [t, t + dt]]$$

(ii) The probability $\pi_{i,j,n}$, characteristic of the algorithm $\mathcal{A}(n)$,

$$\pi_{i,j,n} := \Pr[\mathcal{A}(n) \text{ compares } U^{(i)} \text{ and } U^{(j)}]$$

Lemma 2. *The density ϕ_n of $\mathcal{A}(n)$ defined by*

$$\phi_n(u, t) du dt := \text{Mean number of key-comparisons performed by} \\ \mathcal{A}(n) \text{ between a key } M(u') \text{ and a key } M(t') \text{ with} \\ u' \in [u, u + du], t' \in [t, t + dt]$$

$$\text{satisfies} \quad \phi_n(u, t) = \sum_{1 \leq i < j \leq n} \pi_{i,j,n} \delta_{i,j,n}(u, t)$$

How to compute the density ϕ_n ? (II)

Recall : $\delta_{i,j,n}(u, t) du dt := \Pr[U^{(i)} = M(u') \text{ and } U^{(j)} = M(t')$
with $u' \in [u, u + du]$ and $t' \in [t, t + dt]$]

$\delta_{i,j,n}$ only depends on the density f on \mathcal{I} . We focus on a uniform density $f = 1$ (Not an actual restriction). Then:

$$\delta_{i,j,n}(u, t) du dt = \binom{n}{i-1, 1, j-i-1, 1, n-j} \\ \times u^{i-1} \cdot du \cdot (t-u)^{j-i-1} \cdot dt \cdot (1-t)^{n-j}$$

Conclusion. As soon as $\pi_{i,j,n}$ is known, it is possible to compute ϕ_n , (even if this expression is often involved)....

It is then possible to compare $S(n)$ and $K(n)$,

$$K(n) = \int_{\mathcal{I}} \phi_n(u, t) du dt, \quad \text{and} \quad S(n) = \int_{\mathcal{I}} [\gamma(u, t) + 1] \phi_n(u, t) du dt,$$

Two main classes of algorithms.

— **Isotropic** algorithms where $\pi_{i,j,n}$ is **independent** of the pair (i, j) . The probability of comparing two keys is independent of their ranks. A main instance is **InsertionSort**(n) where $\pi_{i,j,n} = 1/2$.

$$\text{Then } \phi_n(u, t) = \frac{1}{2}n(n-1), \quad K(n) = \frac{1}{4}n(n-1),$$

$$S(n) = \frac{1}{2}n(n-1) \left(\frac{1}{2} \sum_{w \in \Sigma^*} p_w^2 \right) = K(n) \cdot \Lambda(2) \quad \text{with} \quad \Lambda(s) = \sum_{w \in \Sigma^*} p_w^s.$$

— **Anisotropic** algorithms where $\pi_{i,j,n}$ **depends** on the pair (i, j) . A main instance is **QuickSort**(n) where $\pi_{i,j,n} = 2/(j-i-1)$. The algorithm performs comparisons between keys whose rank is often close, and whose coincidence is probably high.

How is $S(n)$ compared to $K(n)$? **That is the question....**

An initial question asked by Sedgewick in 2000...

... In order to also compare with text algorithms based on **tries**.

For **QuickSort**(n), a first answer given by **Janson and Fill** ('06), only for the classical **binary** source.

For **QuickSelect**(n), this was studied by **Fill and Nakama** ('07), only in the case of the **binary** source.

- A quite **involved** form for the constants of the analysis.
- More than **twenty pages** of computation....

Here, we answer these **two** questions, in the case of a **general** source.

- There are precise **restrictive hypotheses** on the source, and we provide sufficient conditions under which these hypotheses hold.
- We provide a **closed form** for the constants of the analysis, for any source of the previous type.
- We use **different** methods, with **(almost) no** computation...

Case of QuickSort(n)

Theorem 1. [CFV 08] For any “good” source, the mean number $S(n)$ of symbol comparisons used by QuickSort(n) satisfies

$$S(n) \sim \frac{1}{h_{\mathcal{S}}} n \log^2 n.$$

and involves the *entropy* $h_{\mathcal{S}}$ of the source \mathcal{S} , defined as

$$h_{\mathcal{S}} := \lim_{k \rightarrow \infty} \left[\frac{-1}{k} \sum_{w \in \Sigma^k} p_w \log p_w \right],$$

where p_w is the probability that a word *begins* with prefix w .

Compared to $K(n) = 2n \log n$, there is an extra factor equal to $\frac{\log n}{2h_{\mathcal{S}}}$

Case of QuickSelect(n)

Theorem 2. [CFV 08] For any “good” source, the following holds:

(i) The mean numbers of symbol comparisons used by

QuickSelectMinimum(n) and **QuickSelectMaximum**(n)

$$T^{(-)}(n) \sim c_S^{(-)} n \quad \text{and} \quad T^{(+)}(n) \sim c_S^{(+)} n,$$

involve the constants $c_S^{(\epsilon)}$ which depend on probabilities p_w and $p_w^{(\epsilon)}$,

$$c_S^{(\epsilon)} := \sum_{w \in \Sigma^*} p_w \left[1 - \frac{p_w^{(\epsilon)}}{p_w} \log \left(1 + \frac{p_w}{p_w^{(\epsilon)}} \right) \right].$$

(ii) The mean number of symbol comparisons used by **QuickSelect**(n) (randomized wrt rank), satisfies $T(n) \sim c_S n$, with

$$c_S = \sum_{w \in \Sigma^*} p_w^2 \left[2 + \frac{1}{p_w} + \sum_{\epsilon = \pm 1} \left[\log \left(1 + \frac{p_w^{(\epsilon)}}{p_w} \right) - \left(\frac{p_w^{(\epsilon)}}{p_w} \right)^2 \log \left(1 + \frac{p_w}{p_w^{(\epsilon)}} \right) \right] \right],$$

The constants of the analysis for the binary source.

$$h_{\mathcal{B}} = \log 2$$

$$c_{\mathcal{B}}^{(+)} = c_{\mathcal{B}}^{(-)} = c_{\mathcal{B}}^{(\pm)}$$

$$c_{\mathcal{B}}^{(\pm)} = 4 + 2 \sum_{\ell \geq 0} \frac{1}{2^\ell} + 2 \sum_{\ell \geq 0} \frac{1}{2^\ell} \sum_{k=1}^{2^\ell - 1} \left[1 - k \log \left(1 + \frac{1}{k} \right) \right]$$

$$c_{\mathcal{B}} = 2 + 2 \sum_{\ell=0}^{\infty} \frac{1}{2^{2\ell}} + 2 \sum_{\ell=0}^{\infty} \frac{1}{2^{2\ell}} \sum_{k=1}^{2^\ell - 1} \left[k + 1 + \log(k + 1) - k^2 \log \left(1 + \frac{1}{k} \right) \right]$$

Numerically,

$$c_{\mathcal{B}}^{(\pm)} = 5.279378241080958373865627037785815380836411493490316288078288.$$

The methods used in the analysis

We **begin** with the framework, due to Janson and Fill, previously described :

- We deal with the density ϕ_n of the algorithm
- We express it with the joint distribution $\delta_{i,j,n}$ and the probability $\pi_{i,j,n}$ of the algorithm [Lemma 2 of this talk]

Then, our method **differs** from the method of Janson and Fill:

- We use the decomposition into **fundamental triangles**, [Lemma 1]
- We deal with two main tools: the **Poisson model** and the **Mellin transform**, together with their inverse processes, the **Inverse Mellin transform** and the **Depoissonisation** Process.

With **these four steps**,

the computations become simpler and more transparent....

Four main steps for the asymptotics of $S(n)$

(A) The **Poisson model** \mathcal{P}_Z does not deal with a fixed number n of keys. The number N of keys is now a **random variable** which follows a Poisson law of parameter Z .

A **nice** expression for $\tilde{\phi}_Z$ replaces an **involved** expression for ϕ_n .
But, the expression of $\tilde{S}(Z)$ involves a sum over all the \mathcal{T}_w 's....

(B) The **Mellin transform** \mathcal{M} separates the rôle of the **variable** Z (which tends to ∞) from the other **variables** (u, t) which live in \mathcal{T}_w .
The **singularities** of $\hat{S} := \mathcal{M}[\tilde{S}]$ are easy to find (**position, nature**).

(C) With the **inverse** Mellin transform, this provides the **asymptotics** of $\tilde{S}(Z)$ when $Z \rightarrow \infty$.

(D) With the **Depoissonisation** process, one returns to the initial Bernoulli model \mathcal{B}_n where the number n of keys is fixed, and obtains the **asymptotics of $S(n)$** , our initial goal

(A) Dealing with the Poisson Model.

In the \mathcal{P}_Z model, where the number N of keys follows the law

$$\Pr[N = n] = e^{-Z} \frac{Z^n}{n!},$$

the density of the algorithm involves $f_0(\theta) := 2[e^{-\theta} - 1 + \theta,]$ and

$$\tilde{\phi}_Z(u, t) := e^{-Z} \sum_{n \geq 0} \frac{Z^n}{n!} \phi_n(u, t) = \frac{1}{(t-u)^2} f_0(Z(t-u)).$$

The mean number of symbol comparisons is

$$\begin{aligned} \tilde{S}(Z) &= \int_{\mathcal{T}} [\gamma(u, t) + 1] \tilde{\phi}_Z(u, t) \, dudt \\ &= \sum_{w \in \Sigma^*} \int_{\mathcal{T}_w} \frac{1}{(t-u)^2} f_0(Z(t-u)) \, dudt \end{aligned}$$

(B) Dealing with the Mellin transform.

The Mellin transform is defined by $\mathcal{M}[g](s) := \int_0^\infty g(Z)Z^{s-1} dZ$.

For $g_\lambda : Z \mapsto g(\lambda Z)$, one has : $\mathcal{M}[g_\lambda](s) = \lambda^{-s} \mathcal{M}[g](s)$. Then,

$$\mathcal{M}[\tilde{S}](s) = \mathcal{M}[f_0](s) \cdot \left(\sum_{w \in \Sigma^*} \int_{\mathcal{I}_w} (t-u)^{-(2+s)} dudt \right)$$

Nice factorisation, since $\mathcal{M}[f_0] = 2\Gamma$ (the Γ function), and

$$\int_{\mathcal{I}_w} (t-u)^{-(2+s)} dudt = \frac{p_w^{-s}}{s(s+1)},$$

involves the measure p_w of the fundamental interval \mathcal{I}_w . Finally,

$$\mathcal{M}[\tilde{S}](s) = 2\Gamma(s) \cdot \frac{1}{s(s+1)} \cdot \Lambda(-s)$$

where $\Lambda(s) := \sum_{w \in \Sigma^*} p_w^s$ is the **Dirichlet series** of fundamental probabilities.

(B) Singularities of $\mathcal{M}[\tilde{S}](s) = 2\Gamma(s) \cdot \frac{1}{s(s+1)} \cdot \Lambda(-s)$

The **properties** of the source arise here for the first time.

They can be **translated** into **analytical** properties for $\Lambda(s)$.

— For any source, $\Lambda(s)$ has a **singularity** at $s = 1$.

— For a **good** source \mathcal{S} , the **dominant** singularity of $\Lambda(s)$ is located at $s = 1$, this is a **simple pôle**, whose residue equals $(-1)/h_{\mathcal{S}}$.

— In this case, there is a **triple** pôle at $s = -1$ for $\mathcal{M}[\tilde{S}]$

$$\text{and } \mathcal{M}[\tilde{S}](s) \sim \frac{2}{h_{\mathcal{S}}} \frac{1}{(s+1)^3} \quad \text{near } s = -1$$

What about the last two steps (C) and (D)?

Other properties of $\Lambda(s)$ are needed on $\Re s \leq 1$, –more subtle–,

... even for a memoryless source (p_i) , where $\Lambda(s) = \frac{1}{1 - \sum_i p_i^s}$

Conclusions.

*If the source \mathcal{S} satisfies all these conditions, then $S(n) \sim \frac{1}{h_{\mathcal{S}}} n \log^2 n$,
and we are done...*

— QuickSort must be compared to the algorithm which uses **tries** for sorting n words. We have studied its mean cost in 2001.

For a general “good” source, it is asymptotics to $\frac{1}{h_{\mathcal{S}}} n \log n$.

— Our methods apply to **QuickSelect**(n), and the hypotheses needed for the source are different (less strong).

They are related to properties of $\Lambda(s)$ for $\Re s > 1$.

— However, we have not yet a result on **QuickMedian**(n).

A work in progress