NUMBER OF SYMBOL COMPARISONS

IN QUICKSORT

Brigitte Vallée
(CNRS and Université de Caen, France)

Joint work with Julien Clément, Jim Fill and Philippe Flajolet

Plan of the talk.

– Presentation of the study

– Statement of the results

– The general model of source

– The main steps of the method

– Sketch of the proof.

– What is a tamed source ?

Plan of the talk.

- Presentation of the study
- Statement of the results
- The general model of source
- The main steps of the method
- Sketch of the proof
- What is a tamed source ?

The classical framework for sorting.

The main sorting algorithms or searching algorithms

e.g., `QuickSort`, `BST-Search`, `InsertionSort`,...

deal with n (distinct) keys $U_1, U_2, \ldots, U_n$ of the same ordered set Ω.

They perform comparisons and exchanges between keys.

The unit cost is the key–comparison.

The classical framework for sorting.

The main sorting algorithms or searching algorithms

e.g., `QuickSort`, `BST-Search`, `InsertionSort`,...

deal with n (distinct) keys $U_1, U_2, \ldots, U_n$ of the same ordered set Ω.

They perform comparisons and exchanges between keys.

The unit cost is the key–comparison.

The behaviour of the algorithm (wrt to key–comparisons)

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.

The classical framework for sorting.

The main sorting algorithms or searching algorithms

e.g., `QuickSort`, `BST-Search`, `InsertionSort`,...

deal with n (distinct) keys $U_1, U_2, \ldots, U_n$ of the same ordered set Ω.

They perform comparisons and exchanges between keys.

The unit cost is the key–comparison.

The behaviour of the algorithm (wrt to key–comparisons)

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.

Here, realistic analysis of the `QuickSort` algorithm

```
QuickSort (n, A): sorts the array A
        Choose a pivot;
        (k, A−, A+) := Partition(A);
        QuickSort (k − 1, A−);
        QuickSort (n − k, A+).
```

Here, realistic analysis of the `QuickSort` algorithm

```
QuickSort (n, A): sorts the array A
        Choose a pivot;
        (k, A_-, A_+) := Partition(A);
        QuickSort (k - 1, A_-);
        QuickSort (n - k, A_+).
```

Mean number K_n of key–comparisons

$$K_n \sim 2n \log n$$

A more realistic framework for sorting.

Keys are viewed as words. The domain Ω of keys is a subset of Σ^∞.

$\Sigma^\infty = \{$the infinite words on some ordered alphabet $\Sigma\}$.

The words are compared [wrt the lexicographic order].

The realistic unit cost is now the symbol–comparison.

A more realistic framework for sorting.

Keys are viewed as words. The domain Ω of keys is a subset of Σ^∞.

$\Sigma^\infty = \{$the infinite words on some ordered alphabet $\Sigma\}$.

The words are compared [wrt the lexicographic order].

The realistic unit cost is now the symbol–comparison.

The realistic cost of the comparison between two words A and B,

$$A = a_1\, a_2\, a_3 \ldots a_i \ldots \qquad \text{and} \qquad B = b_1\, b_2\, b_3 \ldots b_i \ldots$$

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

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

We are interested in this new cost for each algorithm:

the number of symbol–comparisons...and its mean value S_n(for n words)

We are interested in this new cost for each algorithm:

the number of symbol–comparisons...and its mean value S_n(for n words)

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.

We are interested in this new cost for each algorithm:
the number of symbol–comparisons...and its mean value S_n(for n words)

How is S_n compared to K_n? That is the question....

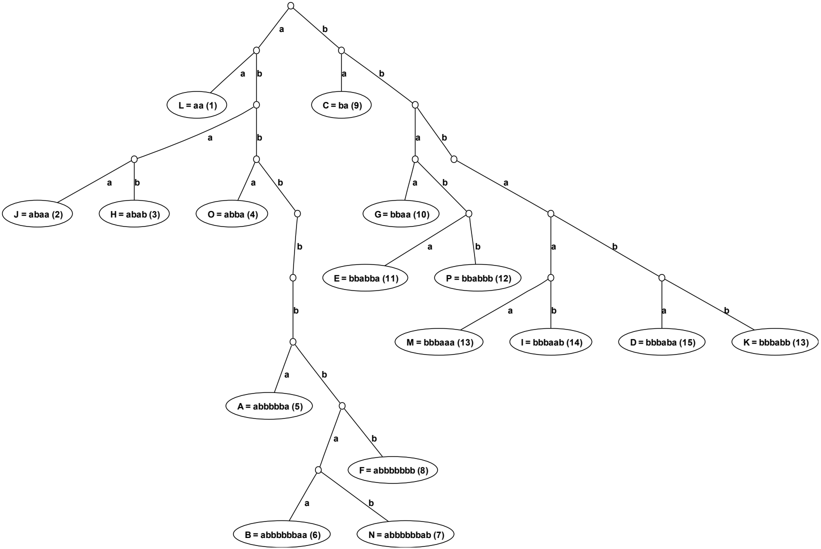
An initial question asked by Sedgewick in 2000...
... In order to also compare with text algorithms based on tries.

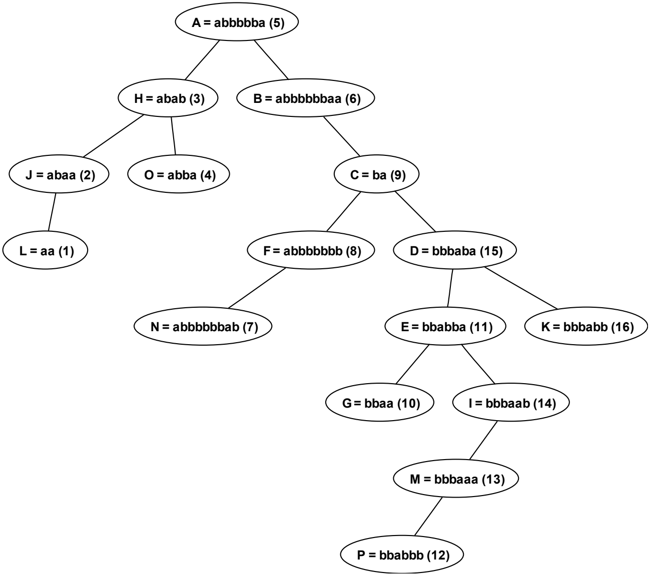
An example.

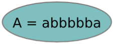
Sixteen words drawn from the memoryless source $p(a) = 1/3, p(b) = 2/3$.
We keep the prefixes of length 12.

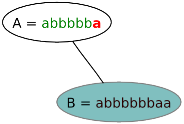
A = abbbbbaaabab	B = abbbbbbaabaa	C = baabbbabbbba
D = bbbababbbaab	E = bbabbaababbb	F = abbbbbbbbabb
G = bbaabbabbaba	H = ababbbabbbab	I = bbbaabbbbbbb
J = abaabbbbaabb	K = bbbabbbbbbaa	L = aaaabbabaaba
M = bbbaaabbbbbb	N = abbbbbbabbaa	O = abbababababbbb
	P = bbabbbaaaabb	

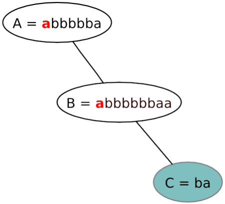
A = abbbbbaaabab B = abbbbbbaabaa C = baabbbabbbba D = bbbababbbaab E = bbabbaababbb
F = abbbbbbbbabb G = bbaabbabbaba H = ababbbabbbab I = bbbaabbbbbbb J = abaabbbbaabb
K = bbbabbbbbbaa L = aaaabbabaaba M = bbbaaabbbbbb N = abbbbbbabbaa O = abbabababbbb P = bbabbbaaaabb

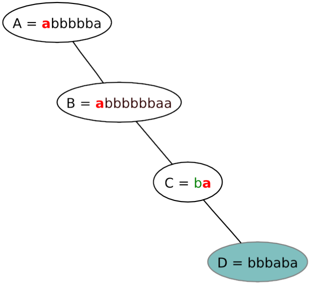


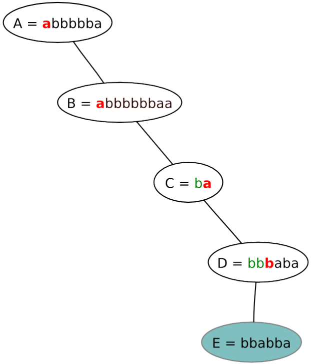

A = abbbbba (5)
H = abab (3)
B = abbbbbbaa (6)
J = abaa (2)
O = abba (4)
C = ba (9)
L = aa (1)
F = abbbbbbb (8)
D = bbbaba (15)
N = abbbbbbab (7)
E = bbabba (11)
K = bbbabb (16)
G = bbaa (10)
I = bbbaab (14)
M = bbbaaa (13)
P = bbabbb (12)

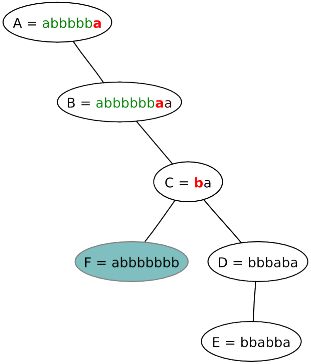

A = abbbbba

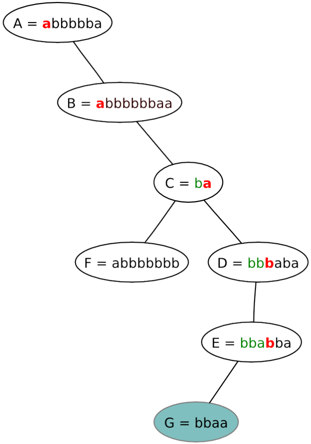

A = abbbbba
B = abbbbbbaa

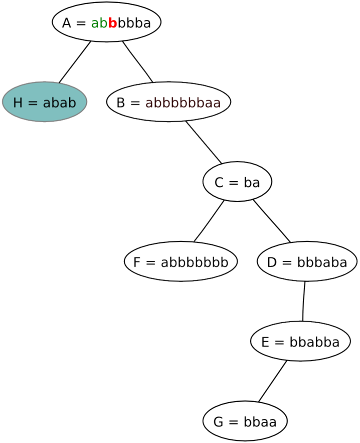

A = abbbbba
B = abbbbbbaa
C = ba

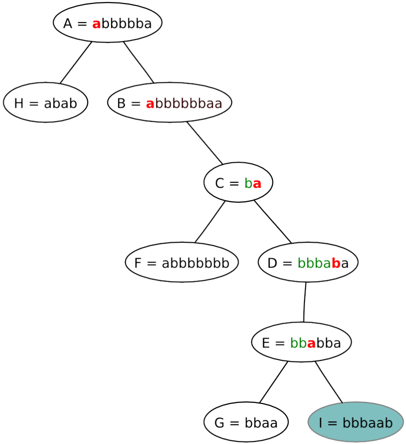

A = abbbbba
B = abbbbbbaa
C = ba
D = bbbaba


A = abbbbba
B = abbbbbbaa
C = ba
D = bbbaba
E = bbabba


A = abbbbba
B = abbbbbbaa
C = ba
F = abbbbbbb
D = bbbaba
E = bbabba


A = abbbbba
B = abbbbbbaa
C = ba
F = abbbbbbb
D = bbbaba
E = bbabba
G = bbaa


A = abbbbba
H = abab
B = abbbbbbaa
C = ba
F = abbbbbbb
D = bbbaba
E = bbabba
G = bbaa



A = abbbbba
H = abab
B = abbbbbbaa
C = ba
F = abbbbbbb
D = bbbaba
E = bbabba
G = bbaa
I = bbbaab

Plan of the talk.

– Presentation of the study

– Statement of the results

– The general model of source

– The main steps of the method

– Sketch of the proof

– What is a tamed source ?

Case of `QuickSort`(n)[CFFV 08]

Theorem. *For any tamed 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

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

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

Case of QuickSort(n)[CFFV 08]

Theorem. *For any tamed 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

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

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

Compared to $K_n \sim 2n \log n$, there is an extra factor equal to $1/(2h_{\mathcal{S}}) \log n$

Compared to $T_n \sim (1/h_{\mathcal{S}})\, n\, \log n$, there is an extra factor of $\log n$.

The (general) model of source

A general source $\mathcal{S}$ produces infinite words on an ordered alphabet Σ.

For $w \in \Sigma^\star$, $p_w :=$ probability that a word begins with the prefix w.

Define $$a_w := \sum_{\substack{w',\, |w'| = |w| \\ w' < w,\, p_{w'} \neq 0}} p_{w'} \qquad b_w := \sum_{\substack{w',\, |w'| = |w| \\ w' \leq w,\, p_{w'} \neq 0}} p_{w'}$$

Then: $\forall u \in [0,1], \quad \forall k \geq 1, \quad \exists w = M_k(u) \in \Sigma^k$ such that $u \in [a_w, b_w[$.

The (general) model of source

A general source $\mathcal{S}$ produces infinite words on an ordered alphabet Σ.
For $w \in \Sigma^\star$, $p_w :=$ probability that a word begins with the prefix w.

Define $$a_w := \sum_{\substack{w',\, |w'| = |w| \\ w' < w,\, p_{w'} \neq 0}} p_{w'} \qquad b_w := \sum_{\substack{w',\, |w'| = |w| \\ w' \leq w,\, p_{w'} \neq 0}} p_{w'}$$

Then: $\forall u \in [0,1], \quad \forall k \geq 1, \quad \exists w = M_k(u) \in \Sigma^k$ such that $u \in [a_w, b_w[$.

Example $\qquad p_0 = 1/3,\ p_1 = 2/3 \Rightarrow M_1(1/2) = 1, M_1(1/4) = 0$

The (general) model of source

A general source $\mathcal{S}$ produces infinite words on an ordered alphabet Σ.

For $w \in \Sigma^\star$, $p_w :=$ probability that a word begins with the prefix w.

Define $$a_w := \sum_{\substack{w',\, |w'| = |w| \\ w' < w,\, p_{w'} \neq 0}} p_{w'} \qquad b_w := \sum_{\substack{w',\, |w'| = |w| \\ w' \leq w,\, p_{w'} \neq 0}} p_{w'}$$

Then: $\forall u \in [0,1], \quad \forall k \geq 1, \quad \exists w = M_k(u) \in \Sigma^k$ such that $u \in [a_w, b_w[$.

Example $\quad p_0 = 1/3,\ p_1 = 2/3 \Rightarrow M_1(1/2) = 1, M_1(1/4) = 0$

$p_{00} = 1/12,\ p_{01} = 3/12,\ p_{10} = 1/2,\ p_{11} = 1/6 \Rightarrow M_2(1/2) = 10, M_2(1/4) = 01$

The (general) model of source

A general source $\mathcal{S}$ produces infinite words on an ordered alphabet Σ.

For $w \in \Sigma^\star$, $p_w :=$ probability that a word begins with the prefix w.

Define $$a_w := \sum_{\substack{w',\, |w'| = |w| \\ w' < w,\, p_{w'} \neq 0}} p_{w'} \qquad b_w := \sum_{\substack{w',\, |w'| = |w| \\ w' \leq w,\, p_{w'} \neq 0}} p_{w'}$$

Then: $\forall u \in [0,1], \quad \forall k \geq 1, \quad \exists w = M_k(u) \in \Sigma^k$ such that $u \in [a_w, b_w[$.

Example $p_0 = 1/3,\, p_1 = 2/3 \Rightarrow M_1(1/2) = 1, M_1(1/4) = 0$

$p_{00} = 1/12,\, p_{01} = 3/12,\, p_{10} = 1/2,\, p_{11} = 1/6 \Rightarrow M_2(1/2) = 10, M_2(1/4) = 01$

If $p_w \to 0$ for $|w| \to \infty$, the sequences (a_w) and (b_w) are adjacent

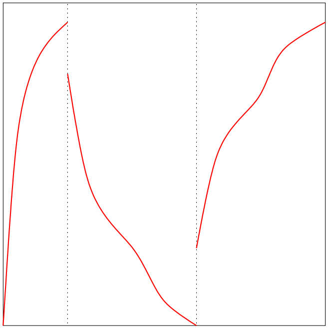
They define an infinite word $M(u) := \lim_{k\to\infty} M_k(u)$.

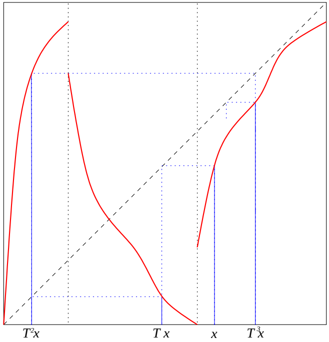
Then, the source is alternatively defined by a mapping $M : [0,1] \to \Sigma^\infty$.

Fundamental interval $[a_w, b_w] := \{u, M(u)$ begins with prefix $w\}$

Natural instances of sources: Dynamical sources

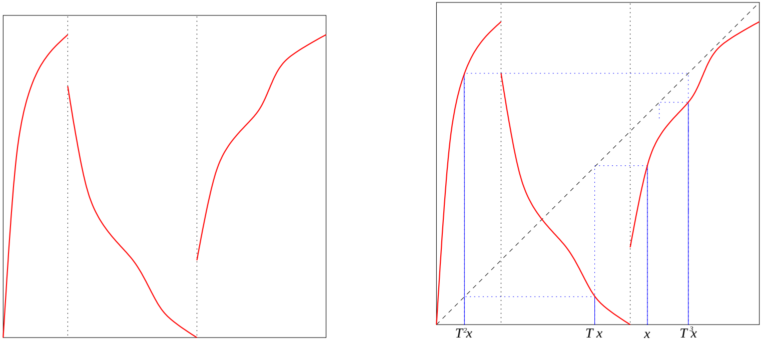
With a shift map $T : \mathcal{I} \to \mathcal{I}$ and an encoding map $\tau : \mathcal{I} \to \Sigma$,
the emitted word is $M(x) = (\tau x, \tau T x, \tau T^2 x, \ldots \tau T^k x, \ldots)$





Natural instances of sources: Dynamical sources

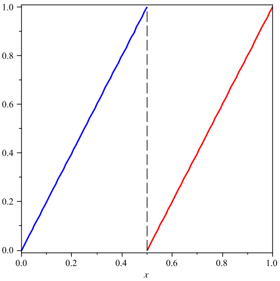
With a shift map $T : \mathcal{I} \to \mathcal{I}$ and an encoding map $\tau : \mathcal{I} \to \Sigma$,
the emitted word is $M(x) = (\tau x, \tau T x, \tau T^2 x, \dots \tau T^k x, \dots)$

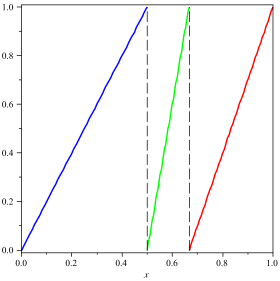


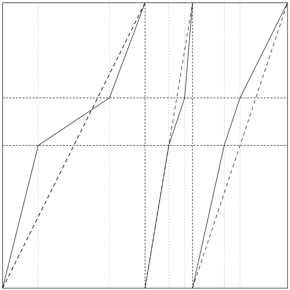
A dynamical system, with $\Sigma = \{a, b, c\}$ and a word $M(x) = (c, b, a, c \dots)$.

Memoryless sources or Markov chains.

= Dynamical sources with affine branches....







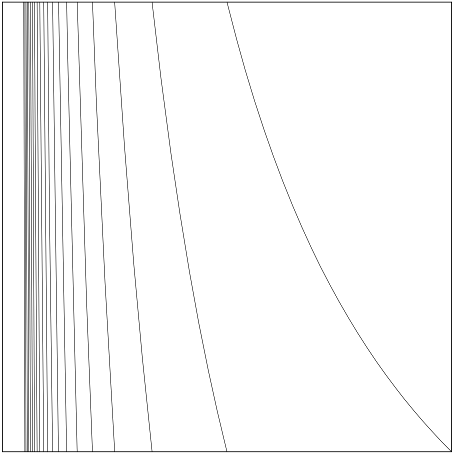
The dynamical framework leads to more general sources.

The curvature of branches entails correlation between symbols

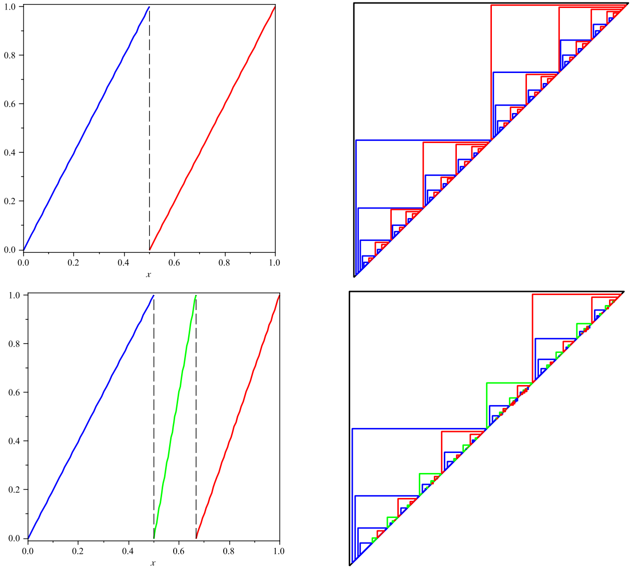
The dynamical framework leads to more general sources.

The curvature of branches entails correlation between symbols

Example : the Continued Fraction source



Fundamental intervals and fundamental triangles.



Plan of the talk.

- Presentation of the study
- Statement of the results
- The general model of source
- The main steps of the method
- Sketch of the proof
- What is a tamed source ?

Three main steps for the analysis
of the mean number S_n of symbol comparisons

Three main steps for the analysis
of the mean number S_n of symbol comparisons

(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.

We first obtain nice expressions for $\widetilde{S}_Z$ ....

Three main steps for the analysis
of the mean number S_n of symbol comparisons

(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.

We first obtain nice expressions for $\widetilde{S}_Z$ ....

(B) It is now possible to retur m to the model where the number of keys is fixed. We obtain a nice exact formula for S_n ....

from which it is not easy to obtain the asymptotics...

Three main steps for the analysis
of the mean number S_n of symbol comparisons

(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.

We first obtain nice expressions for $\widetilde{S}_Z$ ....

(B) It is now possible to retumn to the model where the number of keys is fixed. We obtain a nice exact formula for S_n ....

from which it is not easy to obtain the asymptotics...

(C) Then, the Rice formula provides the asymptotics of S_n ($n \to \infty$), as soon as the source is "tamed".

Plan of the talk.

– Presentation of the study

– Statement of the results

– The general model of source

– The main steps of the method

– Sketch of the proof

– What is a tamed source ?

(A) **Dealing with the Poisson Model.**

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

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

the mean number $\widetilde{S}(Z)$ of symbol comparisons for `QuickSort` is

$$\widetilde{S}(Z) = \int_{\mathcal{T}} [\gamma(u,t) + 1]\, \pi(u,t)\, du\, dt$$

where $\mathcal{T} := \{(u,t), \quad 0 \le u \le t \le 1\}$ is the unit triangle

$\gamma(u,t)$:= coincidence between $M(u)$ and $M(t)$

$\pi(u,t)\, du\, dt$:= Mean number of key-comparisons between $M(u')$ and $M(t')$ with $u' \in [u, u+du]$ and $t' \in [t-dt, t]$.

First Step in the Poisson model : The coincidence $\gamma(u,t)$

An (easy) alternative expression for

$$\widetilde{S}(Z) = \int_{\mathcal{T}} [\gamma(u,t)+1]\, \pi(u,t)\, du\, dt$$

$$= \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} \pi(u,t)\, du\, dt$$

It involves the fundamental triangles

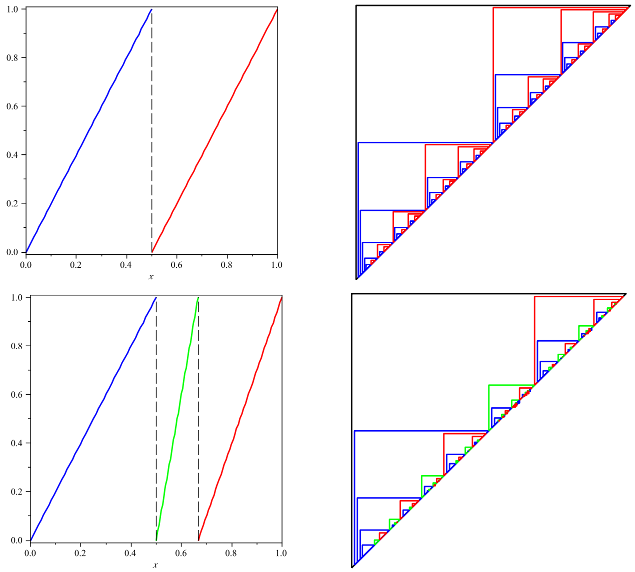
and separates the rôles of the source and the algorithm.

It is then sufficient to

– study the key–probability $\pi(u,t)$ of the algorithm (the second step).

– take its integral on each fundamental triangle (the third step)

Fundamental intervals and fundamental triangles.



Study of the key probability $\pi(u,t)$ of the algorithm (I)

$\pi(u,t)\,du\,dt :=$ Mean number of key-comparisons between $M(u')$ and $M(t')$ with $u' \in [u, u+du]$ and $t' \in [t-dt, t]$.

Study of the key probability $\pi(u,t)$ of the algorithm (I)

$\pi(u,t)\,du\,dt :=$ Mean number of key-comparisons between $M(u')$ and $M(t')$ with $u' \in [u, u+du]$ and $t' \in [t-dt, t]$.

Case of QuickSort. $M(u)$ and $M(t)$ are compared iff the first pivot chosen in $\{M(v), v \in [u,t]\}$ is $M(u)$ or $M(t)$

Study of the key probability $\pi(u,t)$ of the algorithm (I)

$\pi(u,t)\,du\,dt :=$ Mean number of key-comparisons between $M(u')$
and $M(t')$ with $u' \in [u, u+du]$ and $t' \in [t-dt, t]$.

Case of QuickSort. $M(u)$ and $M(t)$ are compared
iff the first pivot chosen in $\{M(v), v \in [u,t]\}$ is $M(u)$ or $M(t)$

```
QuickSort (n, A): sorts the array A
        Choose a pivot;
        (k, A_−, A_+) := Partition(A);
        QuickSort (k − 1, A_−);
        QuickSort (n − k, A_+).
```

Study of the key probability $\pi(u, t)$ of the algorithm (I)

$\pi(u, t)\, du\, dt :=$ Mean number of key-comparisons between $M(u')$ and $M(t')$ with $u' \in [u, u + du]$ and $t' \in [t - dt, t]$.

Case of QuickSort. $M(u)$ and $M(t)$ are compared iff the first pivot chosen in $\{M(v), v \in [u, t]\}$ is $M(u)$ or $M(t)$

Study of the key probability $\pi(u,t)$ of the algorithm (I)

$\pi(u,t)\,du\,dt :=$ Mean number of key-comparisons between $M(u')$ and $M(t')$ with $u' \in [u, u+du]$ and $t' \in [t-dt, t]$.

Case of QuickSort. $M(u)$ and $M(t)$ are compared iff the first pivot chosen in $\{M(v), v \in [u,t]\}$ is $M(u)$ or $M(t)$

$$\pi(u,t)\,du\,dt = Z du \cdot Z dt \cdot \mathbb{E}\left[\frac{2}{2+N_{[u,t]}}\right]$$

Here, $N_{[u,t]}$ is the number of words $M(v)$ with $v \in [u,t]$,
It follows a Poisson law of parameter $Z(t-u)$.

Then: $\pi(u,t) = 2\,Z^2\,f_1(Z(t-u))$ with $f_1(\theta) := \theta^{-2}\,[e^{-\theta} - 1 + \theta]$

Finally:

$$\widetilde{S}(Z) = 2\,Z^2 \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} f_1(Z(t-u))\,du\,dt$$

(B) Return to the model where n is fixed.

With the expansion of f_1, the mean value

$$\widetilde{S}(Z) = \sum_{k=2}^{\infty} (-1)^k \varpi(-k) \frac{Z^k}{k!},$$

is expressed with a series $\varpi(s)$ of Dirichlet type,

which depends both on the algorithm and the source.

$$\varpi(s) = 2 \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} (t-u)^{-(s+2)} du dt$$

$$\int_{\mathcal{T}_w} (t-u)^{-(s+2)} du dt = \frac{p_w^{-s}}{s(s+1)} \quad \Longrightarrow \quad \varpi(s) = 2 \frac{\Lambda(s)}{s(s+1)}$$

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

(B) Return to the model where n is fixed.

With the expansion of f_1, the mean value

$$\widetilde{S}(Z) = \sum_{k=2}^{\infty} (-1)^k \varpi(-k) \frac{Z^k}{k!},$$

is expressed with a series $\varpi(s)$ of Dirichlet type,

which depends both on the algorithm and the source.

$$\varpi(s) = 2 \sum_{w \in \Sigma^\star} \int_{\mathcal{T}_w} (t-u)^{-(s+2)} du dt$$

$$\int_{\mathcal{T}_w} (t-u)^{-(s+2)} du dt = \frac{p_w^{-s}}{s(s+1)} \quad \Longrightarrow \quad \varpi(s) = 2 \frac{\Lambda(s)}{s(s+1)}$$

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

Since $\frac{S_n}{n!} = [Z^n] \left(e^Z \cdot \widetilde{S}(Z) \right)$, there is an exact formula for S_n

$$S_n = 2 \sum_{k=2}^{n} (-1)^k \binom{n}{k} \varpi(-k) = 2 \sum_{k=2}^{n} (-1)^k \binom{n}{k} \frac{\Lambda(-k)}{k(k-1)}.$$

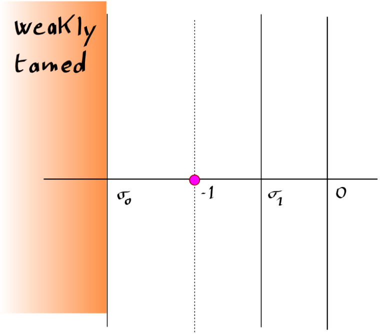
(C) Using Rice formula

As soon as $\varpi(s)$ is "weakly tamed" in $\Re(s) < \sigma_0$ with $\sigma_0 > -2$,

the residue formula transforms the sum into an integral:

$$S_n = \sum_{k=2}^{n} (-1)^k \binom{n}{k} \varpi(-k) = \frac{1}{2i\pi} \int_{d-i\infty}^{d+i\infty} \varpi(s) \frac{n!}{s(s+1)\dots(s+n)} ds,$$

with $-2 < d < \min(-1, \sigma_0)$.



(C) **Using Rice formula**

As soon as $\varpi(s)$ is "weakly tamed" in $\Re(s) < \sigma_0$ with $\sigma_0 > -2$,
the residue formula transforms the sum into an integral:

$$S_n = \sum_{k=2}^{n} (-1)^k \binom{n}{k} \varpi(-k) = \frac{1}{2i\pi} \int_{d-i\infty}^{d+i\infty} \varpi(s) \frac{n!}{s(s+1)\dots(s+n)} ds,$$

with $-2 < d < \min(-1, \sigma_0)$.

Where are the leftmost singularities for $\varpi(s)$?

Recall: $\varpi(s) = 2\dfrac{\Lambda(s)}{s(s+1)}$

where $\Lambda(s) := \sum_{w \in \Sigma^\star} p_w^{-s}$ has always a singularity at $s = -1$.

What type of singularity? Is it the dominant singularity?

Plan of the talk.

- Presentation of the study
- Statement of the results
- The general model of source
- The main steps of the method
- Sketch of the proof
- What is a tamed source?

What can be expected about $\Lambda(s)$?

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

What can be expected about $\Lambda(s)$?

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

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

What can be expected about $\Lambda(s)$?

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

— For a tamed 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 $\quad \frac{\varpi(s)}{s+1} = 2\frac{\Lambda(s)}{s(s+1)^2}$

and $$\frac{\varpi(s)}{s+1} \sim \frac{2}{h_{\mathcal{S}}}\frac{1}{(s+1)^3} \qquad s \to -1$$

For shifting the integral to the right, past... $d = -1$,

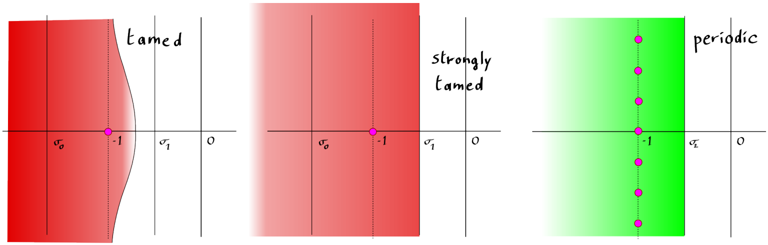
other properties of $\Lambda(s)$ are needed on $\Re s \geq -1$, –more subtle–

Different behaviours of $\Lambda(s)$ for $\Re s \geq -1$ where one can past $d = -1$...

For shifting the integral to the right, past... $d = -1$,

other properties of $\Lambda(s)$ are needed on $\Re s \geq -1$, –more subtle–

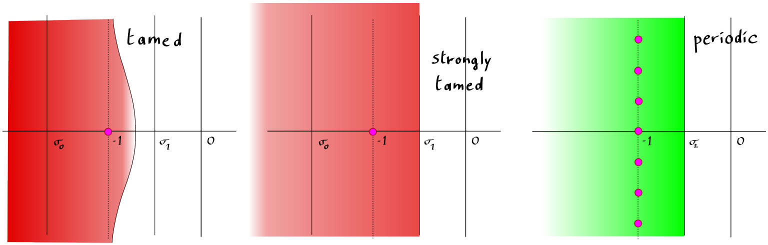
Different behaviours of $\Lambda(s)$ for $\Re s \geq -1$ where one can past $d = -1$...



In colored domains, $\Lambda(s)$ is meromorphic and of polynomial growth for $|s| \to \infty$.

For shifting the integral to the right, past... $d = -1$,
other properties of $\Lambda(s)$ are needed on $\Re s \geq -1$, –more subtle–

Different behaviours of $\Lambda(s)$ for $\Re s \geq -1$ where one can past $d = -1$...



In colored domains, $\Lambda(s)$ is meromorphic and of polynomial growth for $|s| \to \infty$.

For dynamical sources, we provide sufficient conditions
(of geometric or arithmetic type), under which these behaviours hold.
For a memoryless source, they depend on the approximability of ratios $\log p_i / \log p_j$

Conclusions.

If the source $\mathcal{S}$ *is tamed, then* $S_n \sim \dfrac{1}{h_{\mathcal{S}}}\, n \log^2 n.$ *We are done !*

Conclusions.

If the source $\mathcal{S}$ is tamed, then $S_n \sim \dfrac{1}{h_{\mathcal{S}}}\, n \log^2 n.$ *We are done !*

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

$$\textit{For a tamed source,} \quad T_n \sim \frac{1}{h_{\mathcal{S}}}\, n \log n.$$

Conclusions.

If the source $\mathcal{S}$ is tamed, then $S_n \sim \frac{1}{h_{\mathcal{S}}} n \log^2 n.$ *We are done !*

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

$$\textit{For a tamed source,} \quad T_n \sim \frac{1}{h_{\mathcal{S}}} n \log n.$$

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

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

Conclusions.

If the source $\mathcal{S}$ is tamed, then $S_n \sim \frac{1}{h_{\mathcal{S}}} n \log^2 n.$ *We are done !*

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

$$\textit{For a tamed source,} \quad T_n \sim \frac{1}{h_{\mathcal{S}}} n \log n.$$

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

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

— It is easy to adapt our results to the intermittent sources, which emits "long" sequences of the same symbols. In this case,

$$S_n = \Theta(n \log^3 n), \qquad T_n = \Theta(n \log^2 n).$$

Open problems.

— What about the distribution of the average search cost in a BST?

Is it asymptotically normal?

We know that is the case if one counts the number of key–comparisons.

We already know that, for a tamed source,

the average depth of a trie is asymptotically normal (Cesaratto-V, '07).

Open problems.

— What about the distribution of the average search cost in a BST?

Is it asymptotically normal?

We know that is the case if one counts the number of key–comparisons.

We already know that, for a tamed source,

the average depth of a trie is asymptotically normal (Cesaratto-V, '07).

Long term research projects...

— Revisit the complexity results of the main classical algorithms, and take into account the number of symbol-comparisons...

instead of the number of key-comparisons.

Open problems.

— What about the distribution of the average search cost in a BST?

Is it asymptotically normal?

We know that is the case if one counts the number of key–comparisons.

We already know that, for a tamed source,

the average depth of a trie is asymptotically normal (Cesaratto-V, '07).

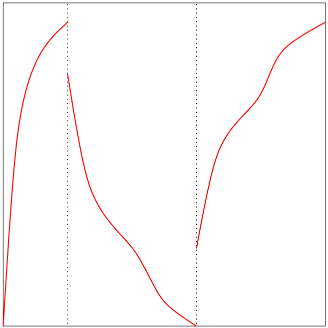
Long term research projects...

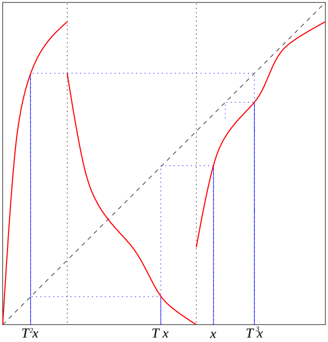
— Revisit the complexity results of the main classical algorithms,
and take into account the number of symbol-comparisons...
instead of the number of key-comparisons.

— Provide a sharp "analytic" classification of sources:
Transfer geometric properties of sources into analytical properties of $\Lambda(s)$

Natural instances of sources: Dynamical sources

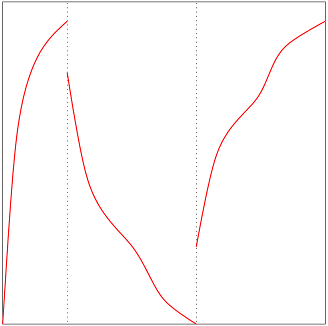
With a shift map $T : \mathcal{I} \to \mathcal{I}$ and an encoding map $\tau : \mathcal{I} \to \Sigma$,
the emitted word is $M(x) = (\tau x, \tau T x, \tau T^2 x, \ldots \tau T^k x, \ldots)$

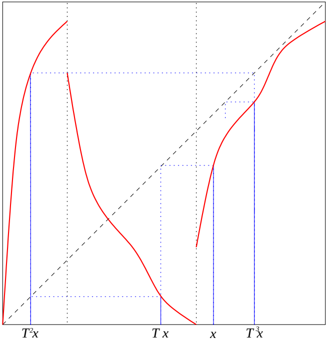




Natural instances of sources: Dynamical sources

With a shift map $T : \mathcal{I} \to \mathcal{I}$ and an encoding map $\tau : \mathcal{I} \to \Sigma$,
the emitted word is $M(x) = (\tau x, \tau T x, \tau T^2 x, \dots \tau T^k x, \dots)$

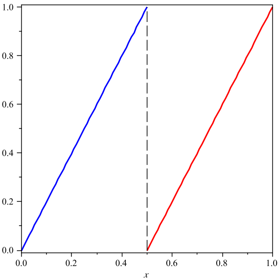


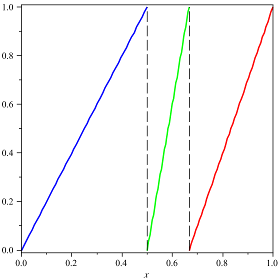


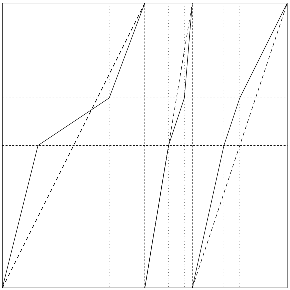
A dynamical system, with $\Sigma = \{a, b, c\}$ and a word $M(x) = (c, b, a, c \dots)$.

Memoryless sources or Markov chains.

= Dynamical sources with affine branches....







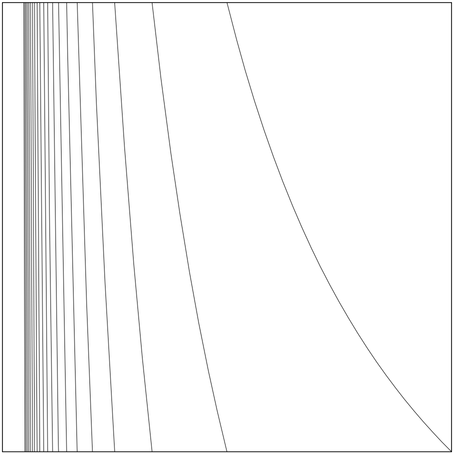
The dynamical framework leads to more general sources.

The curvature of branches entails correlation between symbols

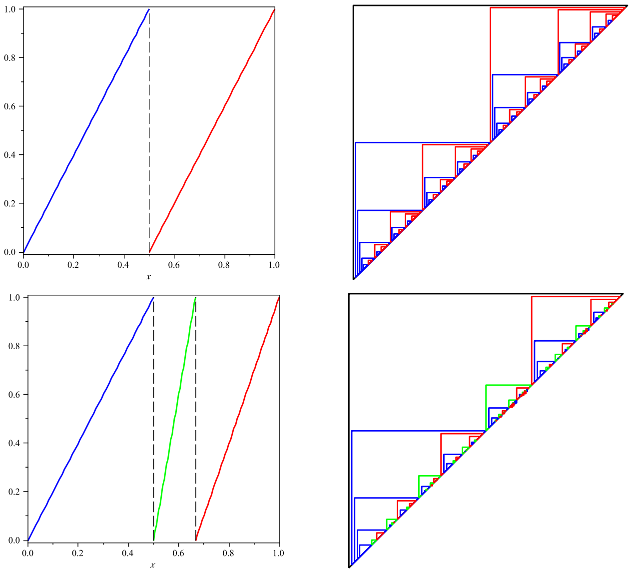
The dynamical framework leads to more general sources.

The curvature of branches entails correlation between symbols

Example : the Continued Fraction source



Fundamental intervals and fundamental triangles.



Case of `QuickMin`(n), `QuickMax`(n), [CFFV 08]

Theorem 2. *For any weakly tamed source, the mean numbers of symbol comparisons used by* `QuickMin`(n) *and* `QuickMax`(n)

$$T_n^{(-)} \sim c_{\mathcal{S}}^{(-)}\, n \quad \text{and} \quad T_n^{(+)} \sim c_{\mathcal{S}}^{(+)}\, n,$$

involve the constants $c_{\mathcal{S}}^{(\epsilon)}$ *which depend on probabilities* p_w *and* $p_w^{(\epsilon)}$, $(\epsilon = \pm)$

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

Here $p_w^{(-)}, p_w^{(+)}, p_w$ *are the probabilities that a word begins with a prefix* w', *with* $|w'| = |w|$ *and* $w' < w$ *or* $w' > w$ *or* $w' = w$.

Case of `QuickRand`(n) [CFFV 08]

Theorem 3. *For any weakly tamed source, the mean number of symbol comparisons used by* `QuickRand`(n) *(randomized wrt rank), satisfies* $T_n \sim c_{\mathcal{S}}\, n,$ *with*

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

Here $p_w^{(-)}, p_w^{(-)}, p_w$ *are the probabilities that a word begins with a prefix* w', *with* $|w'| = |w|$ *and* $w' < w$ *or* $w' > w$ *or* $w' = w$.

Case of $\texttt{QuickQuant}_\alpha(n)$ [CFFV 09] Work yet in progress

Theorem 4. *For any weakly tamed source, the mean number of symbol comparisons used by* $\texttt{QuickQuant}_\alpha(n)$ *satisfies* $q_n^{(\alpha)} \sim \rho_{\mathcal{S}}(\alpha)\, n$ *with*

$$\begin{aligned}\rho_{\mathcal{S}}(\alpha) \quad &= \quad 2 \sum_{w \in \mathcal{S}(\alpha)} p_w + p_w \log p_w - p_w^{(\alpha,+)} \log p_w^{(\alpha,+)} - p_w^{(\alpha,-)} \log p_w^{(\alpha,-)} \\ &\quad + 2 \sum_{w \in \mathcal{R}(\alpha)} p_w \left[1 + \left(\frac{p_w^{(\alpha,-)}}{p_w} - 1 \right) \log \left(1 - \frac{p_w}{p_w^{(\alpha,-)}} \right) \right] \\ &\quad + 2 \sum_{w \in \mathcal{L}(\alpha)} p_w \left[1 + \left(\frac{p_w^{(\alpha,+)}}{p_w} - 1 \right) \log \left(1 - \frac{p_w}{p_w^{(\alpha,+)}} \right) \right].\end{aligned}$$

Three parts depending on the position of probabilities $p_w^{(\epsilon)}$ wrt α:

$$w \in \mathcal{L}(\alpha) \quad \text{iff} \quad p_w^{(+)} \geq 1 - \alpha, \qquad w \in \mathcal{R}(\alpha) \quad \text{iff} \quad p_w^{(-)} \geq \alpha$$

$$w \in \mathcal{S}(\alpha) \quad \text{iff} \quad p_w^{(-)} < \alpha < 1 - p_w^{(+)},$$

$$p_w^{(\alpha,-)} = 1 - \alpha - p_w^{(+)}, \qquad p_w^{(\alpha,-)} = \alpha - p_w^{(-)}$$