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

Thu-Hien Nguyen-Thi

GREYC - Caen

March 21, 2013

Joint work with Julien Clément and Brigitte Vallée

Plan of the talk

1. Background
   - Sorting and searching algorithms
   - Realistic analysis
2. The source
   - Parameterization of the source
   - Geometry of the source
3. The general framework of the realistic analysis
   - The model
   - Combinatorial - algebraic step
   - Analytic step
4. Discuss about the robustness of the algorithms

Sorting and searching algorithms: Problems

The algorithms perform comparisons and exchange between keys. The execution of the algorithms depend only on the key relative order, i.e, on the permutation.

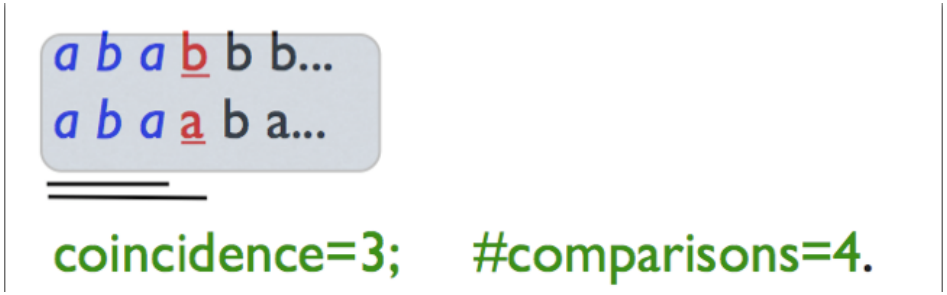
QuickSort	InsSort	BubSort	QuickMin	MinSel	TrieSort
$2n \log n$	$\frac{n^2}{4}$	$\frac{n^2}{2}$	$2n$	n	$O(n \log n)$

- The unit measure cost of the algorithms (`QuickSort, InsSort, BubSort, SelMin, QuickMin`) is the comparison of data items (key).
  The average-case complexity $K(n)$ of these algorithms are well known in the permutation model.
- However, unit measure cost of `TrieSort` is the comparison of symbols.
- Problem?
  - The unit measure cost is not the same. How can we compare the efficiency of the algorithms?
  - It is not realistic to consider comparison of data item (key) as unit comparison because items might have complex structure.

Settings

Settings:

- For the realistic analysis of sorting and searching algorithms, the unit measure cost will be the comparison of symbols.
- Define the input in information theory: Source and coincidence function.
  - The source: a mechanism which produces randomly and independently infinite words of $\Sigma^{\mathbb{N}}$, with an alphabet Σ.
  - The coincidence function measures the length of the largest common prefix between two words. It allows to count the number of symbol comparisons between two words.

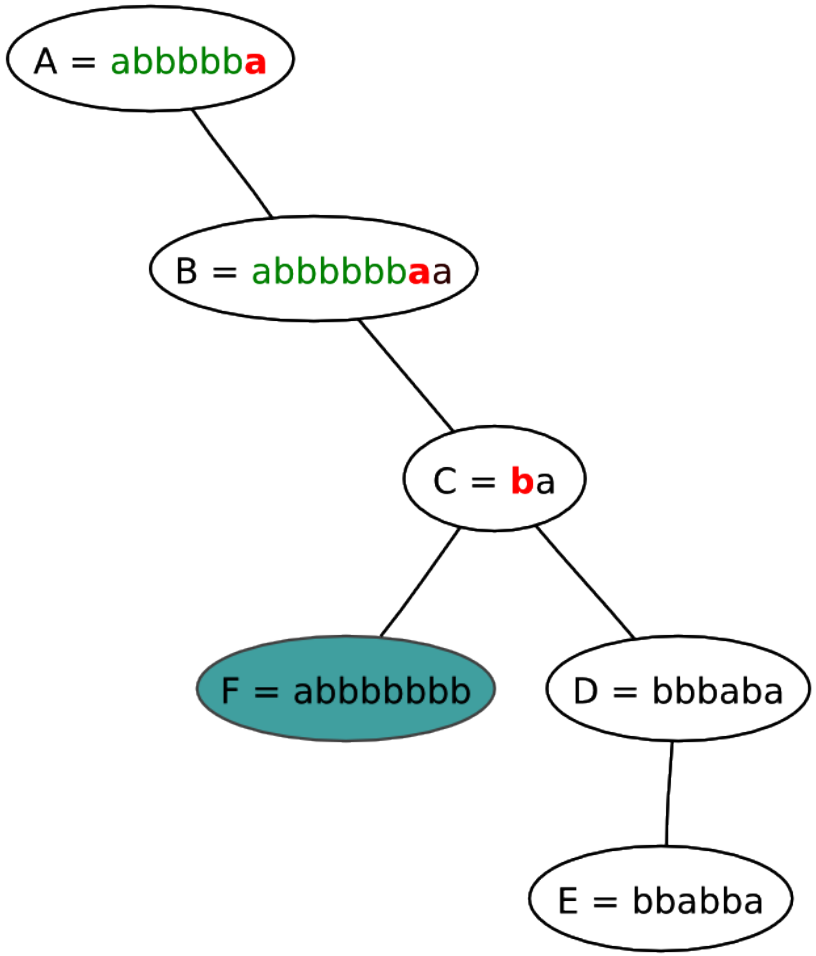


Example: Insertion of a word into a BST

We must do 3 key comparisons to insert F into the BST.
But how many symbol comparisons needed?

The coincidence $c(X, Y)$ of two words X and Y is the length of their largest common prefix.
The realistic comparison cost of X and Y is $c(X, Y) + 1$.

The number of symbol comparisons for inserting F into the BST is :
$(c(A, F) + 1) + (c(B, F) + 1) + (c(C, F) + 1) = 7 + 8 + 1 = 16$,



Main results

- Define a general framework for the realistic analysis of sorting and searching algorithms.
- Application for some popular algorithms: $S(n)$ is the mean number of symbol comparisons performed by the algorithm on n words.

	`QuickSort`	InsSort	BubSort	`QuickMin`	SelMin
$K(n)$	$2n \log n$	$\frac{n^2}{4}$	$\frac{n^2}{2}$	$2n$	n
$S(n)$	$\frac{1}{h(\mathcal{S})} n \log^2 n$	$\frac{c(\mathcal{S})}{4} n^2$	$\frac{1}{4h(S)} n^2 \log n$	$2b(\mathcal{S})n$	$a(\mathcal{S})n$

The first results for `QuickSort` and `QuickMin` are due to Clément-Fill-Flajolet-Vallée (2009).

- Discuss the role of the dominant constants: $h(\mathcal{S})$ is the entropy of the source and $a(\mathcal{S}), b(\mathcal{S}), c(\mathcal{S})$ are the constants relative to the source.
- Discuss the robustness of the algorithms (i.e., $\frac{S(n)}{K(n)}$, the change in the complexity behaviors, from the number of key comparisons to the number of symbol comparisons).

Parameterization of the source

Denote $L(S)$ the set of infinite words produced by the source S.

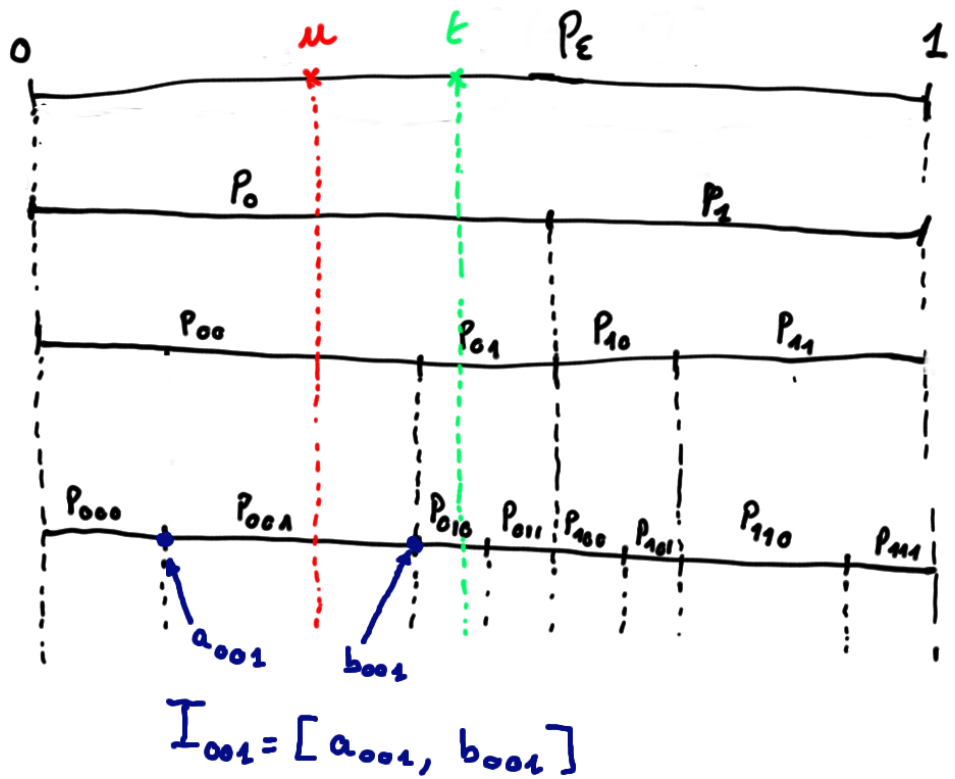
Fundamental probability p_w: probability that an infinite word begins with a finite prefix w. $\sum_{i \in \Sigma} p_{w.i} = p_w, \qquad \sum_{w \in \Sigma^k} p_w = 1$

There exists a mapping

$$\mathcal{M} : \mathcal{I} = [0,1] \to L(\mathcal{S})$$
$$u \mapsto M(u)$$

Each word is associated with a real $u \in [0,1]$.

The fundamental $\mathcal{I}_w = [a_w, b_w]$ is the set of reals u such that $M(u)$ begins with the prefix w.



Geometry of the source

Given the whole triangle $\mathcal{T} = \{(u,t) : 0 \leq x \leq y \leq 1\}$, the fundamental triangle associated with the prefix w is $\mathcal{T}_w = \{(u,t), \quad a_w \leq u < t \leq b_w\}$.
In fact, $\mathcal{T}_w = \{(u,t), \quad M(u)$ and $M(t)$ have the common prefix $w\}$.

The coincidence function $\gamma(u,t)$ is the length of the largest common prefix of two infinite words $M(u)$ and $M(t)$.

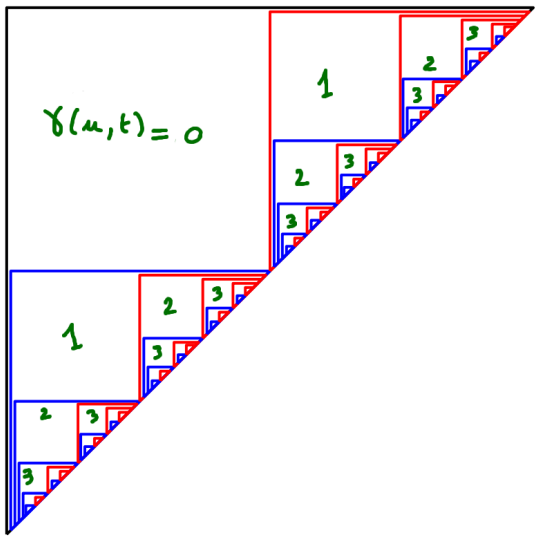


Figure : The case of $\Sigma := \{a, b\}$ with $p_a = p_b = 1/2$.

Correspondence between the coincidence function and the fundamental triangle:

$$\{(u,t), \gamma(u,t) \geq k\} = \bigcup_{|w|=k} \mathcal{T}_w$$

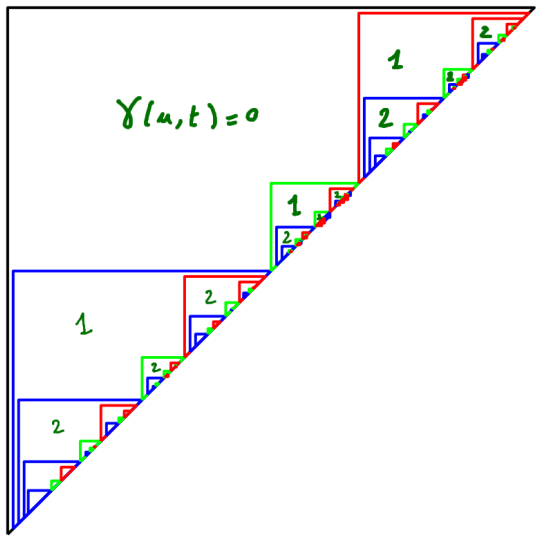
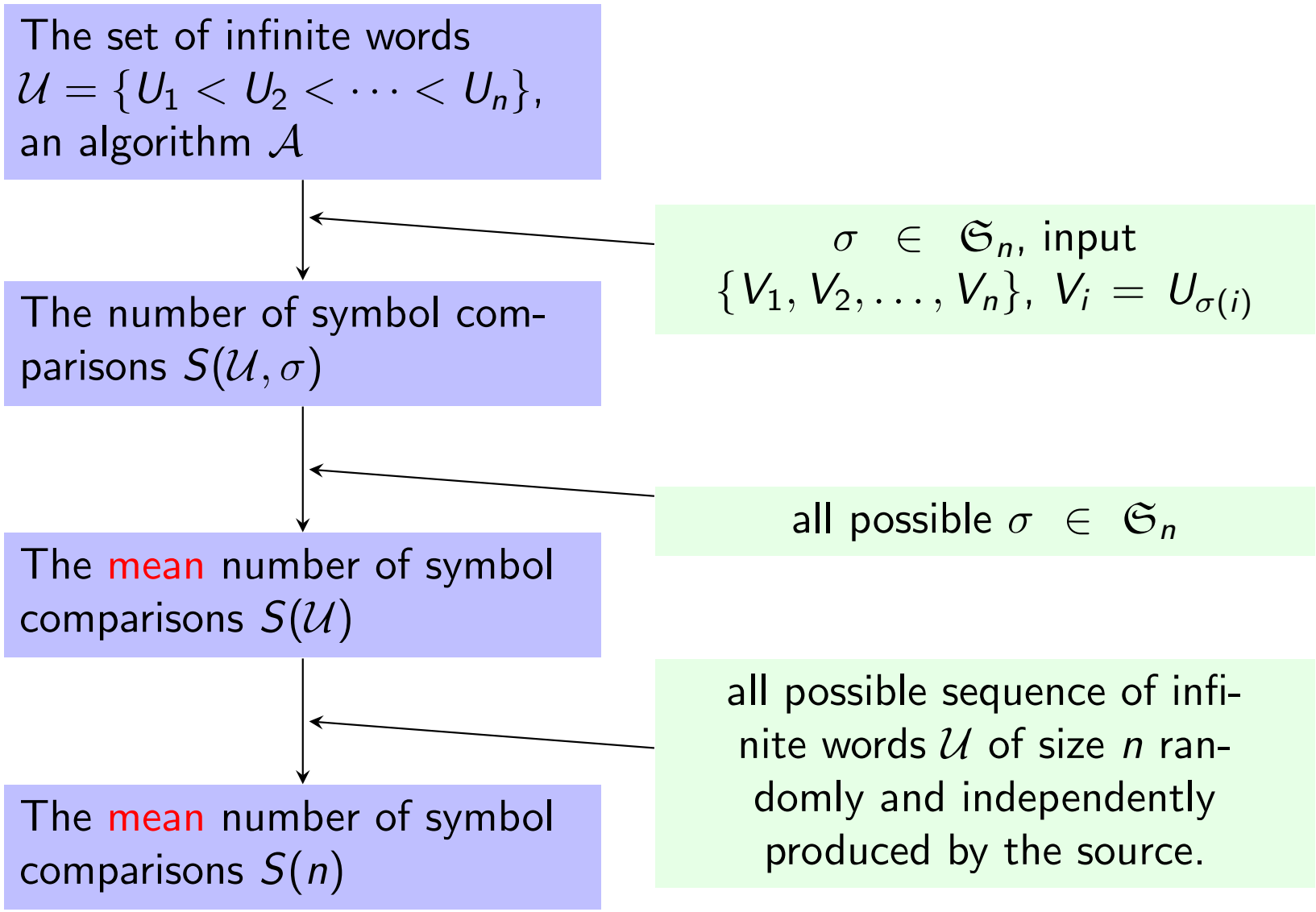


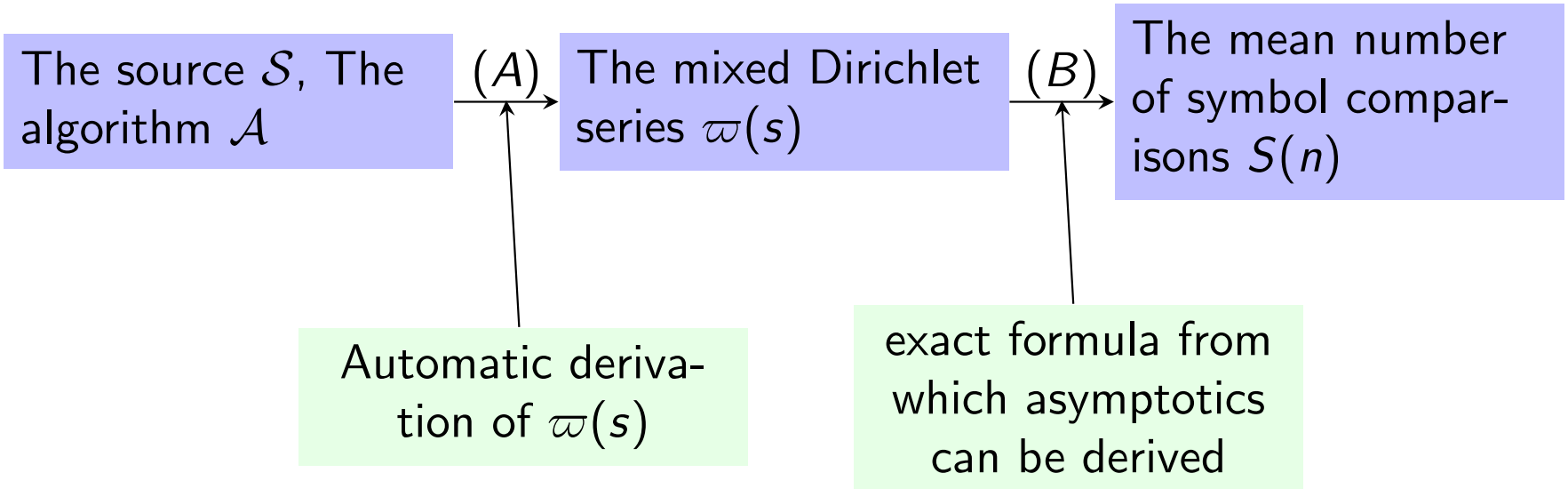
Figure : The case of $\Sigma := \{a, b, c\}$ with $p_a = 1/2, p_b = 1/6, p_c = 1/3$

The model


The set of infinite words $\mathcal{U} = \{U_1 < U_2 < \cdots < U_n\}$, an algorithm $\mathcal{A}$
$\sigma \in \mathfrak{S}_n$, input $\{V_1, V_2, \ldots, V_n\}$, $V_i = U_{\sigma(i)}$
The number of symbol comparisons $S(\mathcal{U}, \sigma)$
all possible $\sigma \in \mathfrak{S}_n$
The mean number of symbol comparisons $S(\mathcal{U})$
all possible sequence of infinite words $\mathcal{U}$ of size n randomly and independently produced by the source.
The mean number of symbol comparisons $S(n)$

Overview of the method

A sequence of n words are randomly and independently produced by the source $\mathcal{S}$. Initial input of the algorithm: $\{\mathcal{U}, \sigma \in \mathfrak{S}_n\}$. The execution of the algorithm actually only depends on $\mathcal{U}$ and σ.



In our analysis, we use two dictionaries: combinatorial - algebraic (A) (true for all the sources), and analytic (B) (depends on the source).
$S(n)$ might be expressed in function of $\varpi(s)$, an important tool which links the properties of the source and the strategy of the algorithm:

$$S(n) = \sum_k (-1)^k \binom{n}{k} \varpi(k)$$

Local strategy of the algorithm

If $\mathcal{U}$ is fixed and σ varies, the "local" mean number of key comparisons between two keys ranked i and j is $\pi(i,j)$. When keys are infinite words, the variable change is : $\hat{\pi}(u,t) = \pi(N_{[0,u[} + 1, N_{[0,t[} + 2)$.
Vary both $\mathcal{U}$ and σ: $\varpi(s,u,t)$ is automatically computed from $\hat{\pi}(u,t)$. Then we have:

	$\pi(i,j)$	$\varpi(s,u,t)$
`QuickSort`	$\frac{2}{j-i+1}$	$2(t-u)^{s-2}$
`InsSort`	$\frac{1}{2} + \frac{1}{(j-i+1)(j-i)}$	$(s-1)(t-u)^{s-2}$
`BubSort`	$\frac{1}{2} + \frac{1}{(j-i+1)(j-i)} + \frac{2(i-1)}{(j-i+2)(j-i+1)(j-i)}$	$(s-1)(t-u)^{s-3}[t-(s-1)u]$
`QuickMin`	$\frac{2}{j}$	$2t^{s-2}$
`SelMin`	$\frac{1}{i(i+1)} + \frac{1}{j(j-1)}$	$(s-1)[u^{s-2} + t^{s-2}]$

Then we have:

$$\varpi(s) = \int_{\mathcal{T}} (\gamma(u,t)+1)\varpi(s,u,t)dudt:$$

The mean number of comparisons

The density $\phi(u,t)dudt$ of the algorithm is the "local mean" number of key comparisons between $M(u')$ and $M(t')$, with $u' \in [u-du, u]$ and $t' \in [t, t+dt]$. The total "mean" number of key and symbol comparisons are respectively

$$\mathbb{E}[K] = \int_T \phi(u,t)dudt$$

$$\mathbb{E}[S] = \int_T (\gamma(u,t)+1)\phi(u,t)dudt = \sum_{w\in\Sigma^*} \int_{T_w} \phi(u,t)dudt$$

When the number of words is exactly n, $S(n)$ is computed using $\varpi(s)$.

	σ_0	$\varpi(s)$
`QuickSort`	1	$\frac{2}{s(s-1)}(\sum_w {p_w}^s)$
`InsSort`	2	$\frac{1}{s}\sum_w {p_w}^s$
`BubSort`	2	$-\sum_{w\in\Sigma^\star} a_w p_w^{s-1}$
`QuickMin`	1	$2\sum_{w\in\Sigma^\star} \int_{a_w}^{b_w} (t-a_w)t^{s-2}dt$
`SelMin`	1	$(s-1)\sum_{w\in\Sigma^\star}(b_w-a_w)\int_{a_w}^{b_w} u^{s-2}du$

σ_0 is the dominant singularity of $\varpi(s)$.

Asymptotic estimates

The results are the following:

	QuickSort	InsSort	BubSort	QuickMin	SelMin
$K(n)$	$2n \log n$	$\frac{n^2}{4}$	$\frac{n^2}{2}$	$2n$	n
Dom. term of $S(n)$	$\frac{1}{h(S)}\, n \log^2 n$	$\frac{c(\mathcal{S})}{4}\, n^2$	$\frac{1}{4h(S)}\, n^2 \log n$	$2b(\mathcal{S})\, n$	$a(\mathcal{S})\, n$

The constants of interest :

- $h(\mathcal{S})$ the entropy of the source: $h(\mathcal{S}) = \lim_{k \to +\infty} -\frac{1}{k} \sum_{|w|=k} p_w \log p_w$.
- $c(\mathcal{S})$ (uniform coincidence): the mean number of comparisons between two words randomly produced by the source. $c(\mathcal{S}) = 2 \sum_w \int_{\mathcal{T}_w} dudt = \sum_w {p_w}^2$
- $a(\mathcal{S})$ (min coincidence): the mean number of comparisons between an uniform random word and the minimum word produced by the source.
- $b(\mathcal{S})$ (logarithmic coincidence): "logarithmic" because the density $\frac{1}{t}$ intervenes in the distribution of two words.

Discuss about the robustness of the algorithms

Two kinds of algorithms:

- Robust algorithms (`InsertionSort`, `QuickMin`, `SelectionMin`): $K(n)$ and $S(n)$ are of the same asymptotic order.
  The ratio $\frac{S(n)}{K(n)}$ involve coincidence of various types $a(\mathcal{S})$, $b(\mathcal{S})$ and $c(\mathcal{S})$.
- The other algorithms (`QuickSort`, `BubSort`): $K(n)$ and $S(n)$ are not of the same asymptotic order.
  The ratio $\frac{S(n)}{K(n)}$ is asymptotic to $\frac{1}{2h(\mathcal{S})} \log n$.

	`QuickSort`	`InsSort`	`BubSort`	`QuickMin`	`SelMin`
$K(n)$	$2n \log n$	$\frac{n^2}{4}$	$\frac{n^2}{2}$	$2n$	n
Dom. term of $S(n)$	$\frac{1}{h(S)}\, n \log^2 n$	$\frac{c(\mathcal{S})}{4}\, n^2$	$\frac{1}{4h(S)}\, n^2 \log n$	$2b(\mathcal{S})\, n$	$a(\mathcal{S})\, n$

Seidel notion about the faithfulness of the algorithms

[Seidel]

- An algorithm is strongly faithful if and only if the mean number of key comparisons $\pi(i,j)$ between two keys U_i and U_j depends only on the difference of their ranks $(j-i)$.
- For a strongly faithful algorithm which sorts words independently drawn from the same source.
  - (a) If the mean number $K(n)$ of key comparisons is asymptotic to An^2, then $S(n)/K(n)$ is asymptotic to $c(\mathcal{S})$.
  - (b) If $K(n)$ is asymptotic to $An\log n$, then $S(n)/K(n)$ is asymptotic to $\frac{\log n}{2h(\mathcal{S})}$.

  (This allows to recover the results of `QuickSort` and `InsSort`, two strongly faithful algorithms.)

Idea of the proof: Build the trie $\mathcal{T}(\mathcal{U})$. $P(\mathcal{U})$ is the set of all prefixes of strings in $\mathcal{U}$.
If the algorithm is strongly faithful, $S(\mathcal{U})$ depends only on N_w (the number of leaves of the subtrie of root $w \Leftrightarrow$ the number of infinite words having the prefix w:

$$S(\mathcal{U}) = \sum_{w \in P(\mathcal{U})} K(N_w)$$

Conclusion

So we have seen:

- A new point of view about the mean number of symbol comparisons
- A general framework for the realistic analysis.
- Applications to some popular algorithms.
- Notion of faithful algorithms introduced by Seidel

The related works is the analysis of trie, ABR, etc.

- Work in progress: Analysis of *BinarySearch* and *HeapSort* (?).
- Our dream: the realistic analysis of all algorithms in the student book.