The realistic analysis of sorting and searching algorithms.

Thu-Hien Nguyen-Thi

EJCIM - Perpignan

April 8, 2013

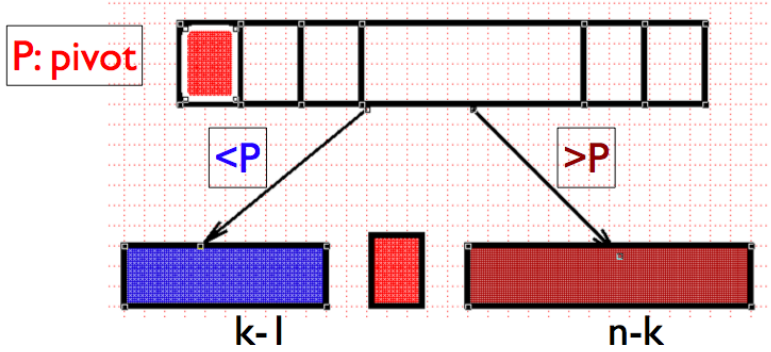
Joint work with Julien Clément and Brigitte Vallée

Plan of the talk

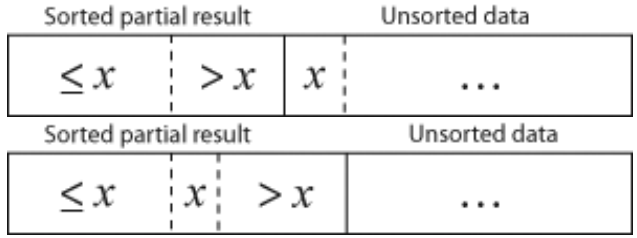
1. Motivations
   - Sorting and searching algorithms
   - Realistic analysis
2. A general framework of the realistic analysis
   - The model
   - Overview of the method
3. The results
   - Discuss about the robustness of the algorithms
   - Lower bound of sorting algorithms
4. Conclusion

Some algorithms

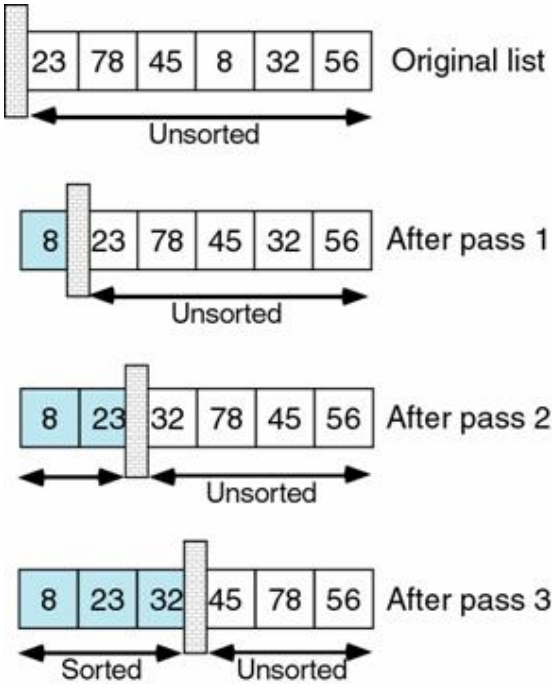
QuickSort algorithm



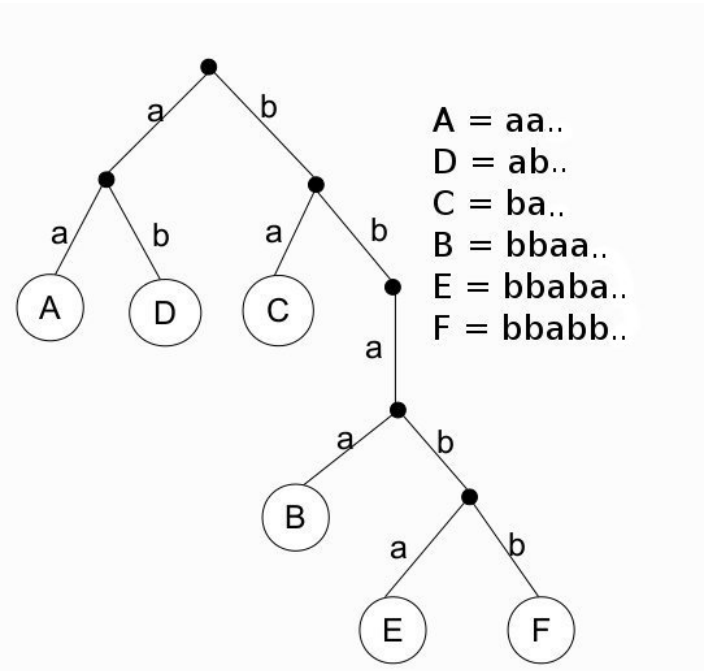
InsSort algorithm



BubSort algorithm



TrieSort algorithm



Context

- 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.
- The average-case complexity $K(n)$ of these algorithms are well known in the permutation model.

Context

- 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.
- The average-case complexity $K(n)$ of these algorithms are well known in the permutation model.

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

Context

- 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.
- The average-case complexity $K(n)$ of these algorithms are well known in the permutation model.

`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).
- However, unit measure cost of `TrieSort` is the comparison of symbols.

Context

- 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.
- The average-case complexity $K(n)$ of these algorithms are well known in the permutation model.

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).
- 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.

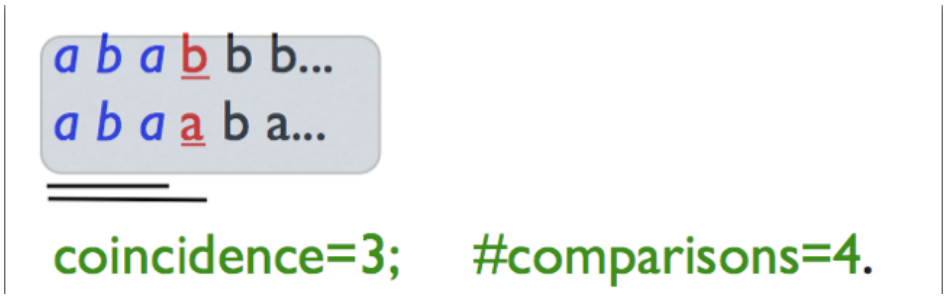
1. Motivations
   - Sorting and searching algorithms
   - Realistic analysis
2. A general framework of the realistic analysis
   - The model
   - Overview of the method
3. The results
   - Discuss about the robustness of the algorithms
   - Lower bound of sorting algorithms
4. Conclusion

Settings

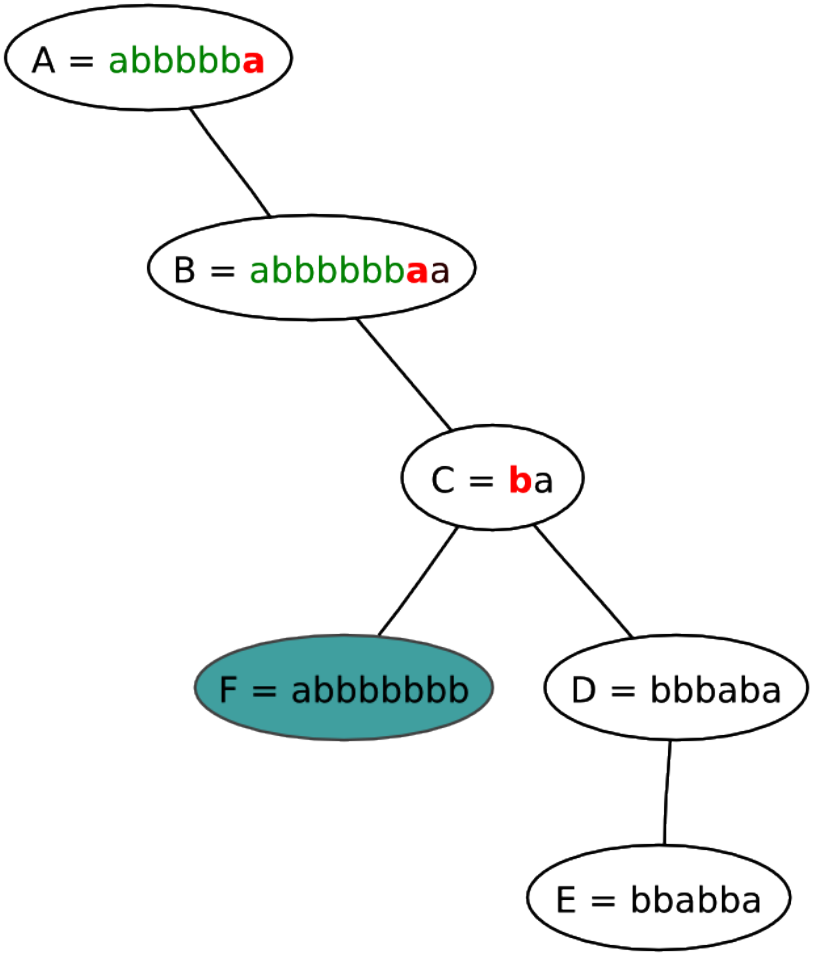
- For the realistic analysis of sorting and searching algorithms, the unit measure cost will be the comparison of symbols.

Settings

- For the realistic analysis of sorting and searching algorithms, the unit measure cost will be the comparison of symbols.
- The coincidence function measures the length of the largest common prefix between two words.

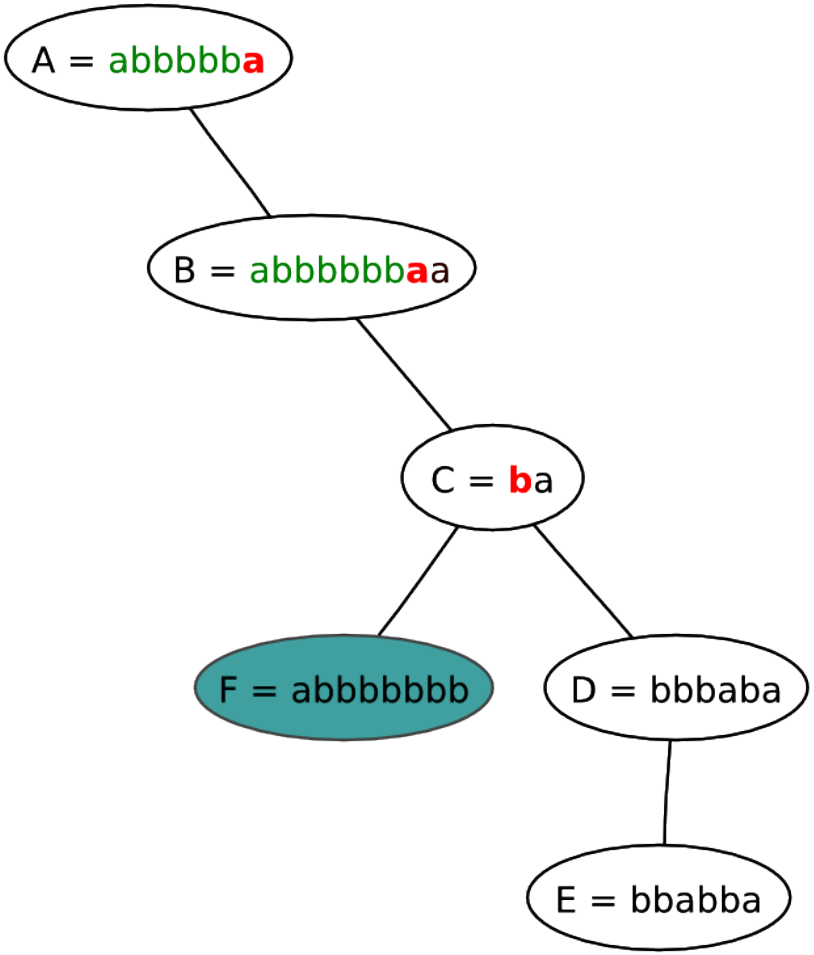


Example: Insertion of a word into a BST



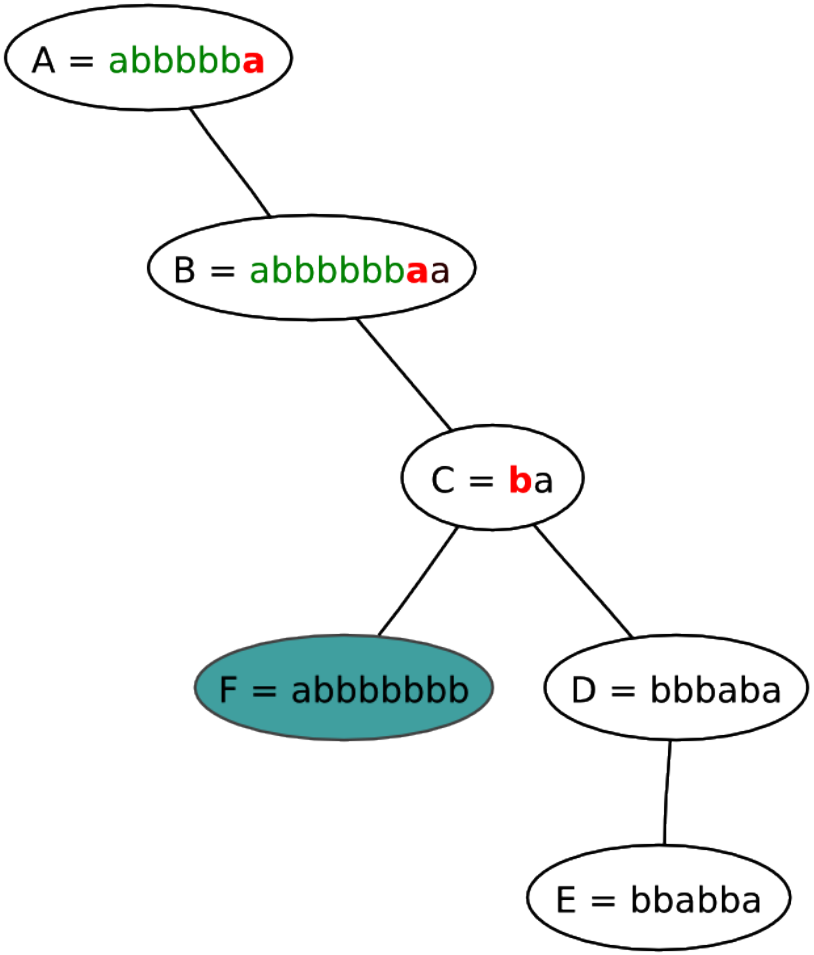
1. To insert F into BST, we must do 3 key comparisons.

Example: Insertion of a word into a BST



1. To insert F into BST, we must do 3 key comparisons.
2. How many symbol comparisons to do that?

Example: Insertion of a word into a BST



1. To insert F into BST, we must do 3 key comparisons.
2. How many symbol comparisons to do that?

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 needed is : $(c(A, F) + 1) + (c(B, F) + 1) + (c(C, F) + 1) = 7 + 8 + 1 = 16$, in compare with 3 key comparisons.

Settings

Define the input in information theory: Under the same context, the related works are the analysis of Trie, BST, DST, etc.

Settings

Define the input in information theory: Under the same context, the related works are the analysis of Trie, BST, DST, etc.

- The source: a mechanism which produces randomly and independently infinite words from an alphabet Σ.
  - The source is parameterized by p_w, the probability that an infinite word begins with the prefix w.
- The properties of the source is related to the Dirichlet series of the source: $\Lambda(s) = \sum_w {p_w}^s$.

Main results

- Define a general framework for the realistic analysis of sorting and searching algorithms.

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
$S(n)$	$\frac{1}{h(\mathcal{S})} n \log^2 n$	$\frac{c(\mathcal{S})}{4} n^2$	$\frac{1}{4h(\mathcal{S})} n^2 \log n$	$2b(\mathcal{S})n$	$a(\mathcal{S})n$

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

Main results

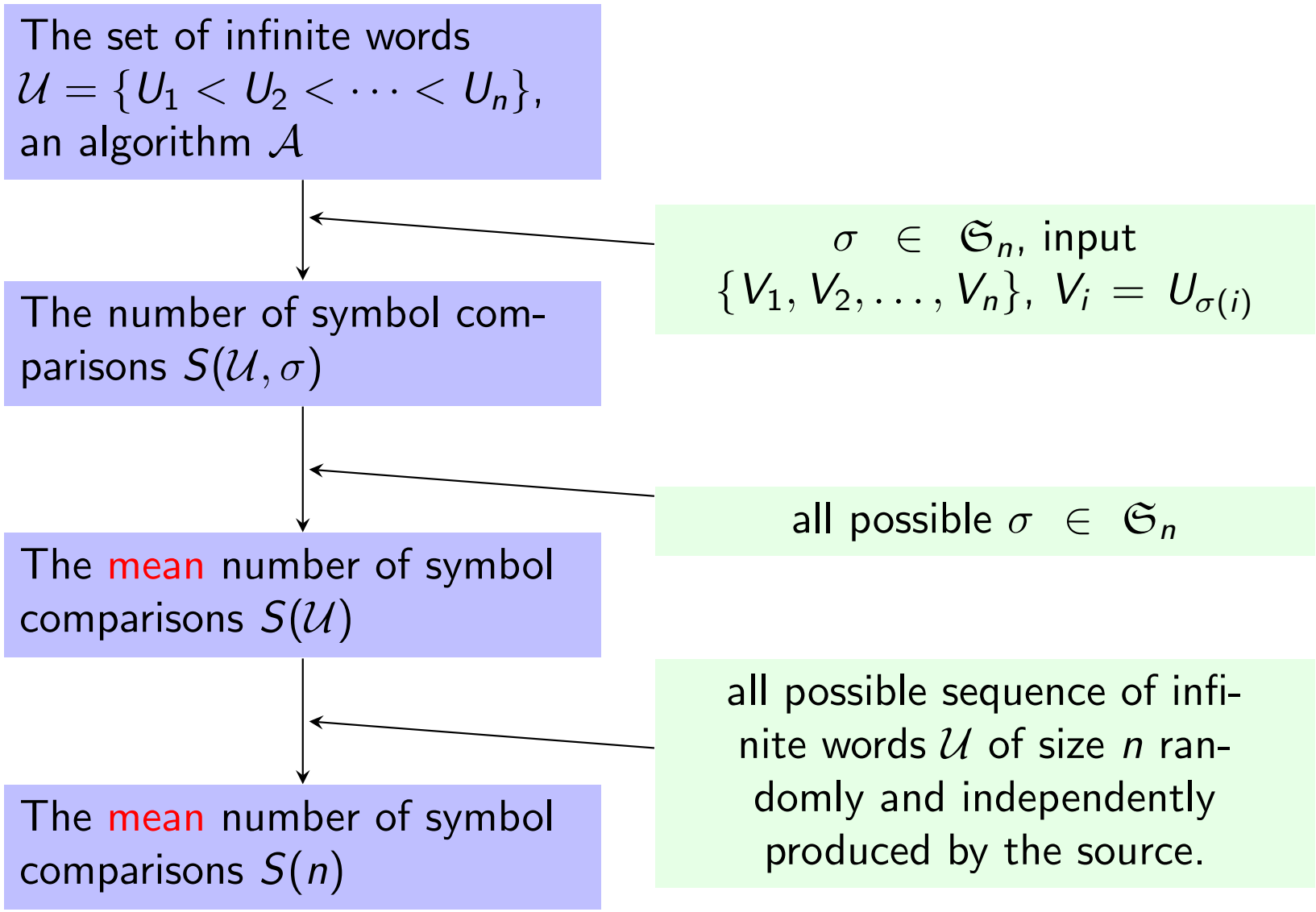
- Define a general framework for the realistic analysis of sorting and searching algorithms.
- Application for some popular algorithms: $S(n)$ is the mean number of symbol comparisons performed by the algorithm on n words.

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

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

- Discuss the role of the dominant constants: $h(\mathcal{S})$, $a(\mathcal{S}), b(\mathcal{S}), c(\mathcal{S})$ 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).

The model

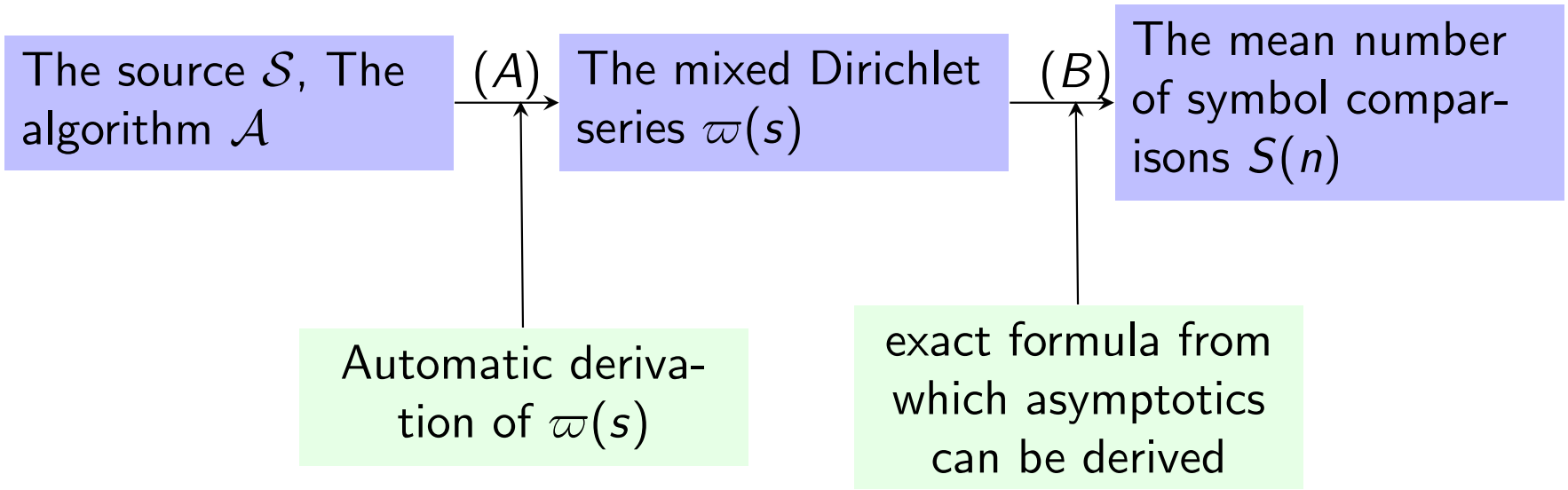

The set of infinite words $\mathcal{U} = \{U_1 < U_2 < \cdots < U_n\}$, an algorithm $\mathcal{A}$
$\sigma \in \mathfrak{S}_n$, input $\{V_1, V_2, \ldots, V_n\}$, $V_i = U_{\sigma(i)}$
The number of symbol comparisons $S(\mathcal{U}, \sigma)$
all possible $\sigma \in \mathfrak{S}_n$
The mean number of 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

- 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.

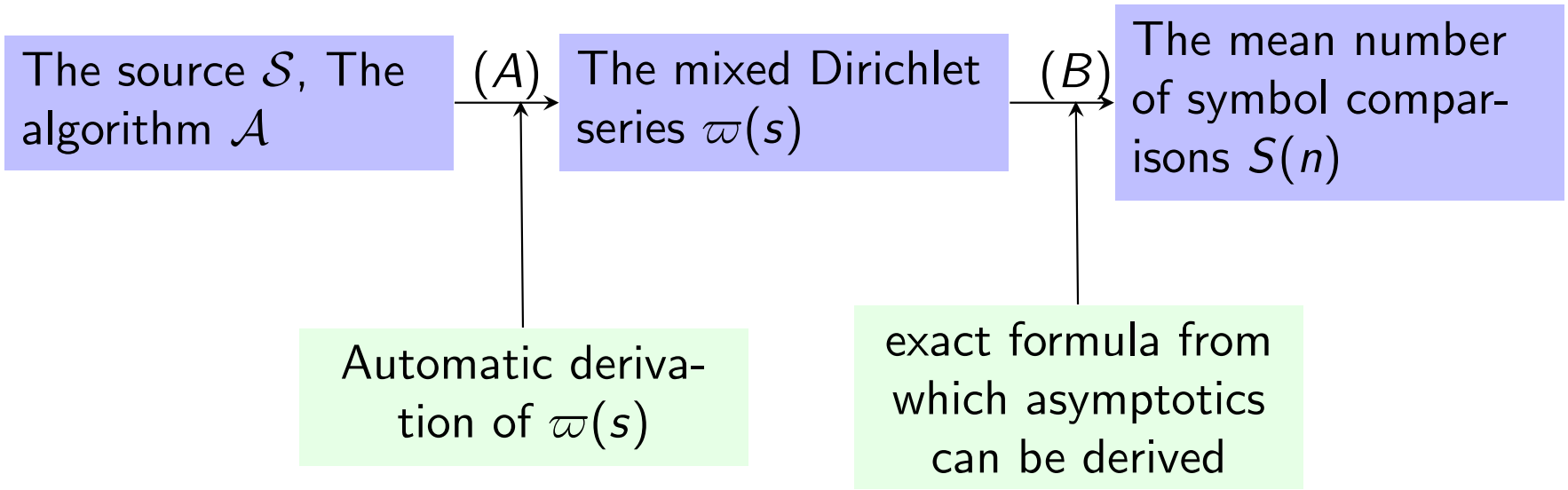
Overview of the method

- 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.



Overview of the method

- 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.

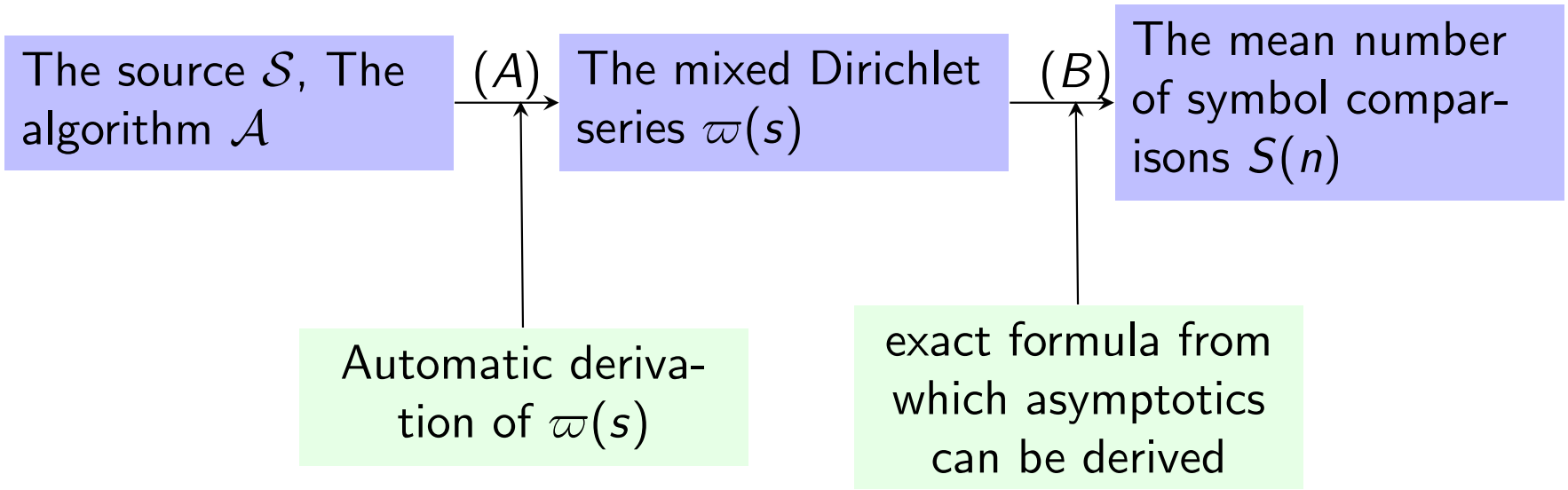


- (A) Combinatorial - algebraic step: true for all the sources. $\varpi(s)$, an important tool which links the properties of the source and the strategy of the algorithm:

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

Overview of the method

- 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.



- (A) Combinatorial - algebraic step: true for all the sources. $\varpi(s)$, an important tool which links the properties of the source and the strategy of the algorithm:

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

- (B) Analytic asymptotic step: depends on the tameness of the source or the analytic properties of $\Lambda(s) = \sum_w {p_w}^s$.

1. Motivations
   - Sorting and searching algorithms
   - Realistic analysis
2. A general framework of the realistic analysis
   - The model
   - Overview of the method
3. The results
   - Discuss about the robustness of the algorithms
   - Lower bound of sorting algorithms
4. Conclusion

Asymptotic estimates

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

(Robust algorithms (in blue): $K(n)$ and $S(n)$ are of the same order.)

Asymptotic estimates

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

(Robust algorithms (in blue): $K(n)$ and $S(n)$ are of the same order.)

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. $c(\mathcal{S}) = \sum_w {p_w}^2$
- $a(\mathcal{S})$ (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.

1. Motivations
   - Sorting and searching algorithms
   - Realistic analysis
2. A general framework of the realistic analysis
   - The model
   - Overview of the method
3. The results
   - Discuss about the robustness of the algorithms
   - Lower bound of sorting algorithms
4. Conclusion

Lower bound of sorting algorithms

For any sorting algorithms:

- It is known that the lower bound for the mean number of key comparisons is:

$$\underline{K}(n) = n \log n \qquad \text{(proof by decision tree)}$$

Lower bound of sorting algorithms

For any sorting algorithms:

- It is known that the lower bound for the mean number of key comparisons is:

$$\underline{K}(n) = n \log n \qquad \text{(proof by decision tree)}$$

- Now, we can prove that the lower bound of the mean number of symbol comparisons:

$$\underline{S}(n) = \frac{n(\log n)^2}{2h(\mathcal{S})} \qquad (\textit{Seidel}2010)$$

Lower bound of sorting algorithms

For any sorting algorithms:

- It is known that the lower bound for the mean number of key comparisons is:

$$\underline{K}(n) = n \log n \qquad \text{(proof by decision tree)}$$

- Now, we can prove that the lower bound of the mean number of symbol comparisons:

$$\underline{S}(n) = \frac{n(\log n)^2}{2h(\mathcal{S})} \qquad (\textit{Seidel}2010)$$

- The ratio $\frac{\underline{S}(n)}{\underline{K}(n)} = \frac{\log n}{2h(\mathcal{S})}$.

Lower bound of sorting algorithms

For any sorting algorithms:

- It is known that the lower bound for the mean number of key comparisons is:

$$\underline{K}(n) = n \log n \qquad \text{(proof by decision tree)}$$

- Now, we can prove that the lower bound of the mean number of symbol comparisons:

$$\underline{S}(n) = \frac{n(\log n)^2}{2h(\mathcal{S})} \qquad (\textit{Seidel}2010)$$

- The ratio $\frac{\underline{S}(n)}{\underline{K}(n)} = \frac{\log n}{2h(\mathcal{S})}$.

Observation for `QuickSort`:

- $K(n) = 2n \log n$
- $S(n) = \frac{n(\log n)^2}{h(\mathcal{S})}$
- The ratio $\frac{S(n)}{K(n)} = \frac{\log n}{2h(\mathcal{S})} = \frac{\underline{S}(n)}{\underline{K}(n)}$. An explanation?

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.

Work in progress:

- Analysis of *BinarySearch* and *HeapSort*.
- Our dream: the realistic analysis of all algorithms in the student book.