

# Distributed Collaborative Editing

## LSEQ: an Adaptive Distributed Sequence Data Structure

### On the Fly Order Preserving Object Renaming

Achour Mostefaoui

joint work with Emmanuel Desmontils, Pascal Molli and Brice Nédelec



# Distributed Collaborative Editors

Untitled document ☆ 📁



Comments

Share

File Edit View Insert Format Tools Table Help All changes saved in Drive

🖨️ ↶ ↷ 📄 Normal text Arial 8 B I U A 🔗 🗨️ ☰ ☱ ☲ ☳ More ⬆

1 2 3 4 5 6 7 8  
urcum dignissim, turpis risus moribus urna, et gravida quam ante sed erat. Aenean eleifend sapien mori, quis vehicula libero  
semper ut. Fusce bibendum risus at lorem rutrum volutpat. Praesent convallis consectetur nisi, vitae fringilla orci rutrum eu.  
Donec ultrices, lectus eget solli **Brice Nédelec**, mi leo commodo augue, quis sagittis mi augue at quam. Vestibulum pharetra  
vestibulum dolor eget dignissim.

## porttitor felis

Aliquam molestie elit volutpat malesuada molestie. Etiam libero dolor, ultrices consectetur turpis malesuada, porta porttitor felis.  
Nam consectetur iaculis vehicula. Proin porta egestas lectus, vitae porta lectus eleifend in. Nam vitae porttitor orci. Quisque  
vulputate accumsan metus at dignissim. Nullam consequat tempor est, ac tincidunt mi consequat quis. Integer rhoncus facilisis  
vel.

## lacus vulputate

Suspendisse tellus orci, luctus vel fringilla at, mattis condimentum velit. Nam nec magna massa. Duis imperdiet vitae nulla non  
laoreet. Interdum et malesuada fames ac ante ipsum primis in faucibus. In eget est sit amet elit feugiat iaculis. Sed sed felis non  
lacus vulputate malesuada. Aliquam laoreet scelerisque turpis et vehicula. Duis hendrerit turpis tortor, in accumsan lacus viverra  
non. Aliquam erat volutpat. Donec porta tincidunt eges

# Distributed Collaborative Editors

- 1 Across space, time, organizations.

Distributed Collaborative editors

Optimistic replication

CRDT      OT

```
graph TD; A[Distributed Collaborative editors] --- B[Optimistic replication]; B --- C[CRDT]; B --- D[OT];
```

# Distributed Collaborative Editors

- 1 Across space, time, organizations.
- 2 Two phases :
  - a locally prepare operations to send
  - b execute remote operations

Distributed Collaborative editors

Optimistic replication

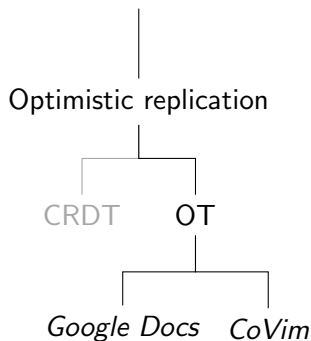
CRDT      OT

```
graph TD; A[Distributed Collaborative editors] --- B[Optimistic replication]; B --- C[CRDT]; B --- D[OT];
```

# Distributed Collaborative Editors

- 1 Across space, time, organizations.
- 2 Two phases :
  - a locally prepare operations to send
  - b execute remote operations
- 3 Operational transform
  - + local operations cheap
  - remote operations complex

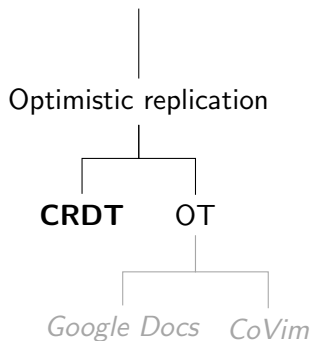
## Distributed Collaborative editors



# Distributed Collaborative Editors

- 1 Across space, time, organizations.
- 2 Two phases :
  - a locally prepare operations to send
  - b execute remote operations
- 3 Operational transform
  - + local operations cheap
  - remote operations complex
- 4 Conflict-free Replicated Data Type
  - 2 phases share computational cost

## Distributed Collaborative editors



# Distributed Collaborative Editors

- 1 Across space, time, organizations.
  - 2 Two phases :
    - a locally prepare operations to send
    - b execute remote operations
  - 3 Operational transform
    - + local operations cheap
    - remote operations complex
  - 4 Conflict-free Replicated Data Type
    - 2 phases share computational cost
- ↗ **collaborators** ⇒ **quadratic** ↗ **remote operations**

## Distributed Collaborative editors

### Optimistic replication

**CRDT**

**OT**

*Google Docs*

*CoVim*

# Distributed Collaborative Editors

A document can be seen as a sequence of basic elements (characters, words, lines, etc.). The problem is non trivial because it is necessary that the edition (updating of the document) ensures the following three properties (CCI) :

- 1 Convergence : the different copies need to converge to a same copy



# Distributed Collaborative Editors

A document can be seen as a sequence of basic elements (characters, words, lines, etc.). The problem is non trivial because it is necessary that the edition (updating of the document) ensures the following three properties (CCI) :

- 1 Convergence : the different copies need to converge to a same copy
- 2 Causality : any operation needs to reflect the operations that occurred causally before it

# Distributed Collaborative Editors

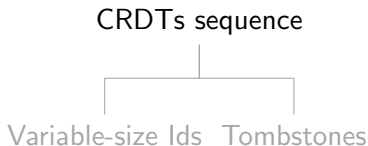
A document can be seen as a sequence of basic elements (characters, words, lines, etc.). The problem is non trivial because it is necessary that the edition (updating of the document) ensures the following three properties (CCI) :

- 1 Convergence : the different copies need to converge to a same copy
- 2 Causality : any operation needs to reflect the operations that occurred causally before it
- 3 Intention : the effect of an operation needs to meet the intention of the user that ordered it

# CRDTs for sequences

1 Two commutative operations :

- Insert / delete
- **Identify the basic elements**
- The set of ids is totally ordered
- The ids make the sequence



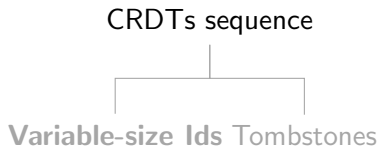
# CRDTs for sequences

## 1 Two commutative operations :

- Insert / delete
- **Identify the basic elements**
- The set of ids is totally ordered
- The ids make the sequence

## 2 The operations :

- $insert(p, elem, q)$   
⇒ **basic function**  $alloc(p, q)$
- $delete(id_{elem})$
- $id_{elem}$  : immutable



# CRDTs for sequences

## 1 Two commutative operations :

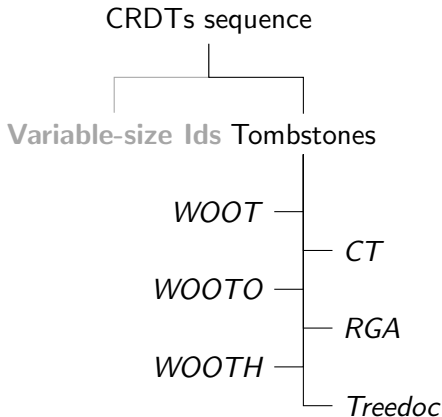
- Insert / delete
- **Identify the basic elements**
- The set of ids is totally ordered
- The ids make the sequence

## 2 The operations :

- $insert(p, elem, q)$   
⇒ **basic function**  $alloc(p, q)$
- $delete(id_{elem})$
- $id_{elem}$  : immutable

## 3 Deleted elements are only marked

⇒ eventually needs purge



# CRDTs for sequences

## 1 Two commutative operations :

- Insert / delete
- **Identify the basic elements**
- The set of ids is totally ordered
- The ids make the sequence

## 2 The operations :

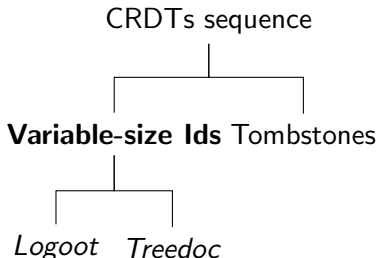
- $insert(p, elem, q)$   
⇒ **basic function**  $alloc(p, q)$
- $delete(id_{elem})$
- $id_{elem}$  : immutable

## 3 Deleted elements are only marked

⇒ eventually needs purge

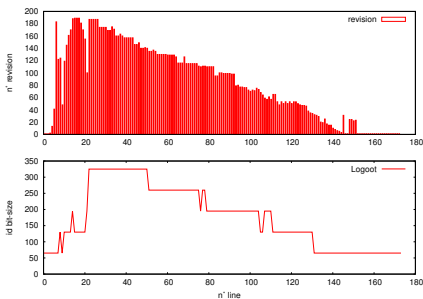
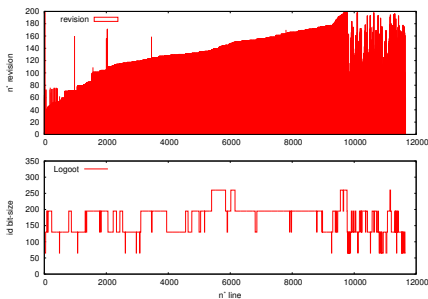
## 4 The size of identifiers may grow

- linearly wrt # operations



# Motivations

Spectrum of two Wikipedia documents.

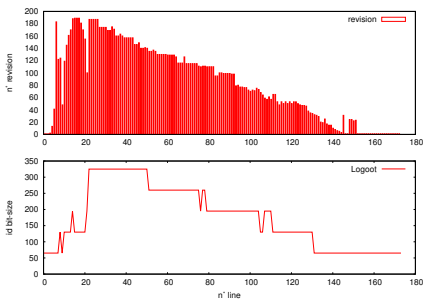
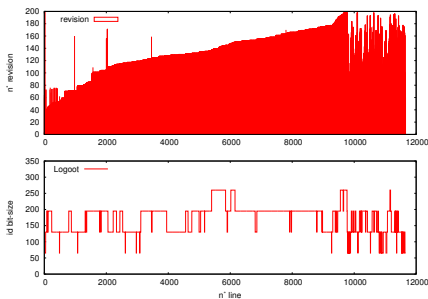


(a) Page edited in the end.  $\Rightarrow$  169.7 bits/id.

(b) Page edited in front.  $\Rightarrow$  172.25 bits/id.

# Motivations

Spectrum of two Wikipedia documents.



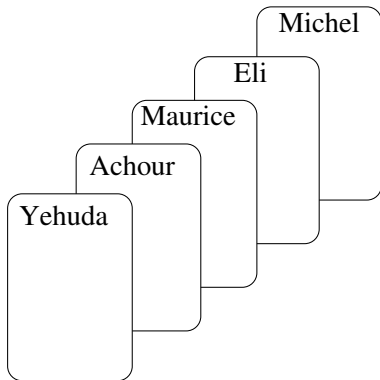
(c) Page edited in the end.  $\Rightarrow$  169.7 bits/id.

(d) Page edited in front.  $\Rightarrow$  172.25 bits/id.

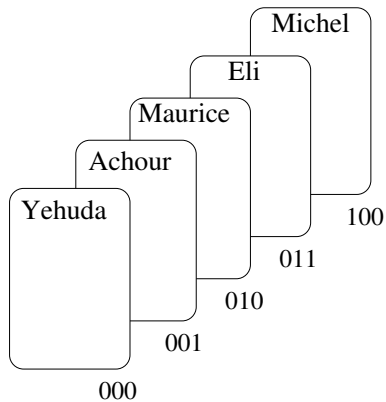
$\Rightarrow$  Allocation strategies are **CRUCIAL**



# Abstract Problem (1)

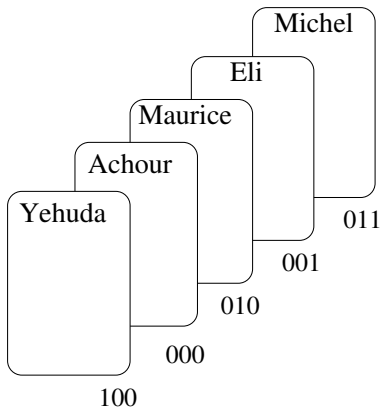


## Abstract Problem (1)



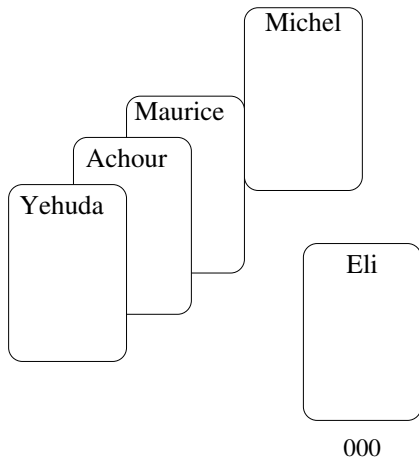
$n$  cards can be named using ids of size  $O(\log n)$

## Abstract Problem (1)



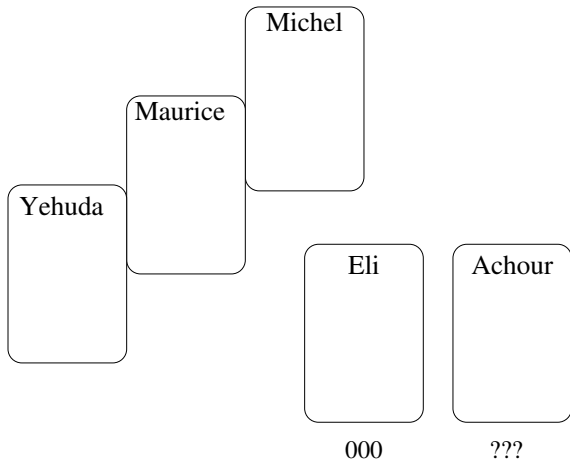
Even if one wants to preserve the order defined by the original names,  $n$  cards can be renamed with ids of size  $O(\log n)$

## Abstract Problem (2)



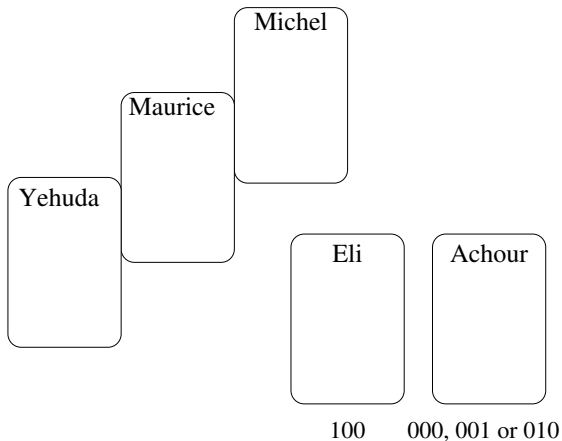
How about if the original names are not a priori known ?

## Abstract Problem (2)



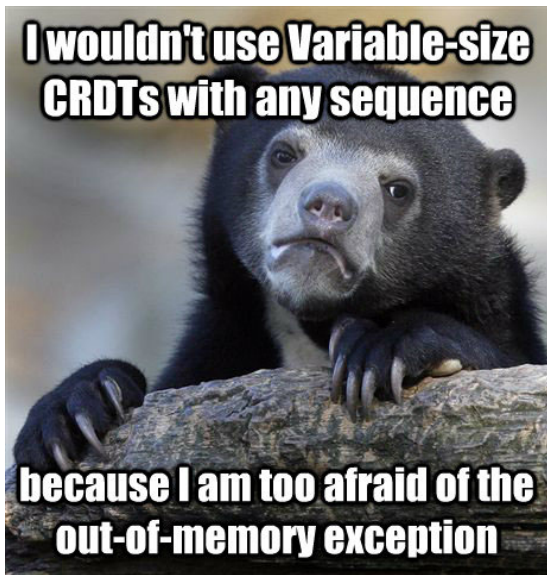
One needs to have spare space (dense set of ids)

## Abstract Problem (2)



Is it possible to avoid all this loss of space?

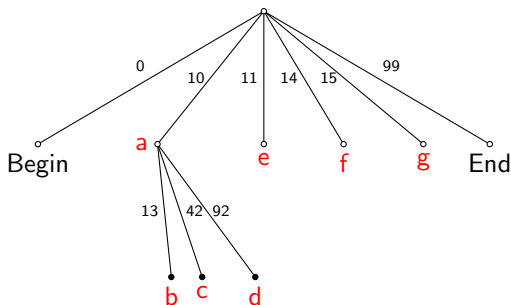
Bear confesses...



# Problem

## Variable-size identifier

A variable-size identifier  $id$  is a sequence of numbers  $id = [p_1.p_2 \dots p_n]$  which can designate a path in a tree.



## Problem statement

Let  $\mathcal{D}$  a document on which  $n$  insert operations have been performed. Let  $\mathcal{I}(\mathcal{D}) = \{id | (-, id) \in \mathcal{D}\}$ . The function  $alloc(id_p, id_q)$  should provide identifiers such as :

$$\sum_{id \in \mathcal{I}} \frac{|id|_2}{n} < O(n)$$

$|id|_2$  means  $\log_2(id)$  aka. bit-length



# Proposal : LSEQ

Three components :

- base doubling,
- multiple allocation strategies,
- random strategy choice.

## Intuition

As it is complex to predict the editing behaviour, some depths of the tree on a given path can be lost if the reward **compensates** the loss.

In other terms, even if LSEQ chooses the wrong strategy at a given time, it will eventually choose the good one, and that choice will **amortize** the cost of all previous lost depths.

# Base doubling

Exponential trees :

- Under uniform distribution :
  - Spatial complexity :  $O(n \log \log n)$ . Where  $n$  the number of Ids.

$[p_1 \cdot p_2 \dots p_n] \Rightarrow |p_n|_2 = |p_{n-1}|_2 + \mathbf{1}$ . Where  $|p_1| = \text{base}$

+ **1** bit  $\Rightarrow$  **x2** identifiers

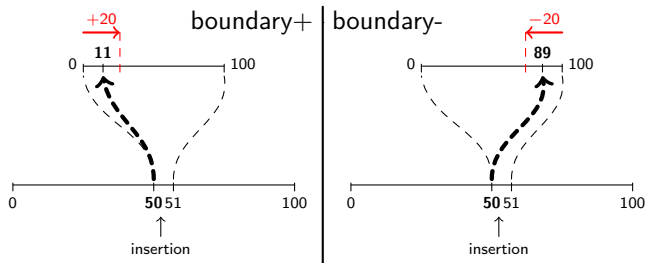
## Intuition

If the number of insert operations is **low**, the id bit-length can stay **small**.  
On the other hand, when the number of insertions **increases**, it is profitable to allocate **larger** identifiers.

# Multiple allocation strategies

*boundary* :

- + Good : page edited in the end.
- Good : page edited in front.



## Intuition

The allocation strategy *boundary* is **not sufficient** to be employed as a safe allocation strategy. However, by using its antagonist strategy, each strategy **cancels** the other's **deficiency**.

# Random strategy choice

- Unique strategy : not sufficient

⇒ Strategy choice : When ? Which ?

## Intuition : When

The **opening** of a new space has a major meaning : Either the allocation strategy went wrong, or, on the opposite, a high number of insertions saturated the previous depths, meaning that it requires more space.

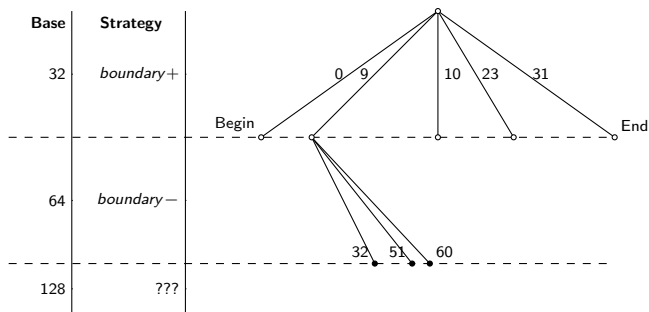
Therefore, the space opening is an ideal moment to decide which strategy to employ.

## Intuition : Which

Since it is impossible to *a priori* know the editing behaviour, the strategy choice should **not favorize any behaviour**. Consequently, the frequency of appearance of each strategies must be equal.

# Synthesis : example

- Exponential tree
- Two allocation strategies : *boundary+* and *boundary-*
- Random strategy choice



# Experimentations

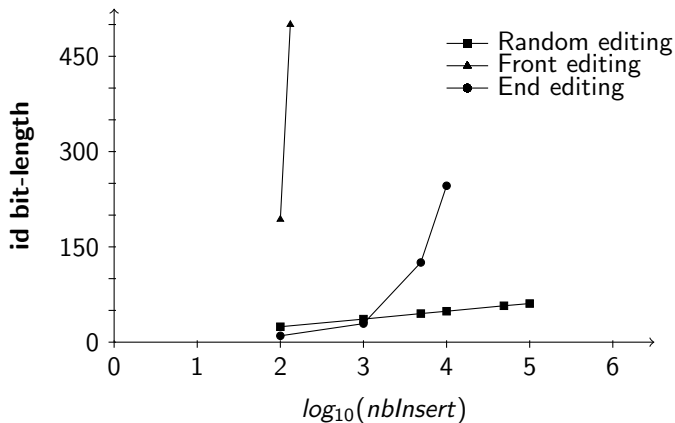
## 1 Influence of each LSEQ's component

- ⇒ Synthetic documents.
- ⇒ High amount of insertions.
- ⇒ 3 editing behaviour : in the beginning, in the end, random.

## 2 Comparison with variable-size CRDT.

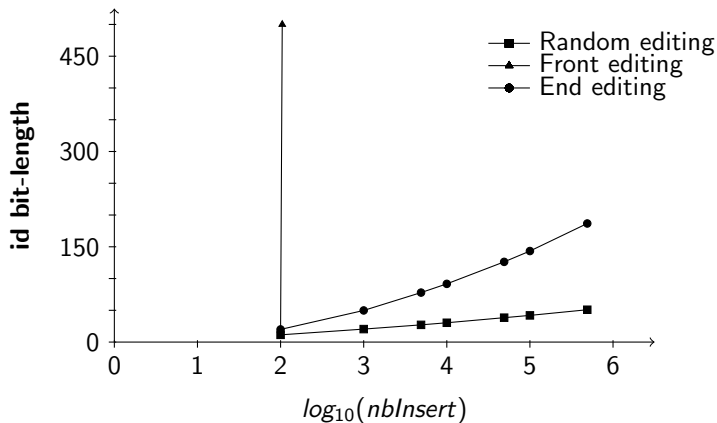
- ⇒ Real documents : Wikipedia.
- ⇒ 2 editing behaviour : in the beginning, in the end.

# Boundary



Simple *boundary+* setup with  $base = 2^{10}$  and  $boundary = 10$

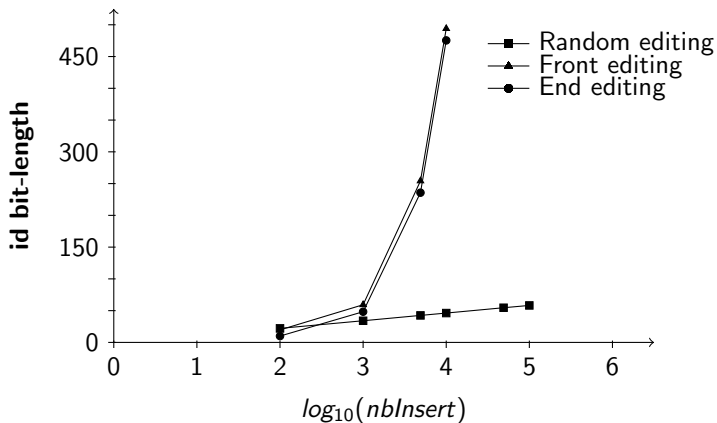
# Exponential tree



Base doubling setup with  $base = 2^{4+id.size}$  and  $boundary = 10$

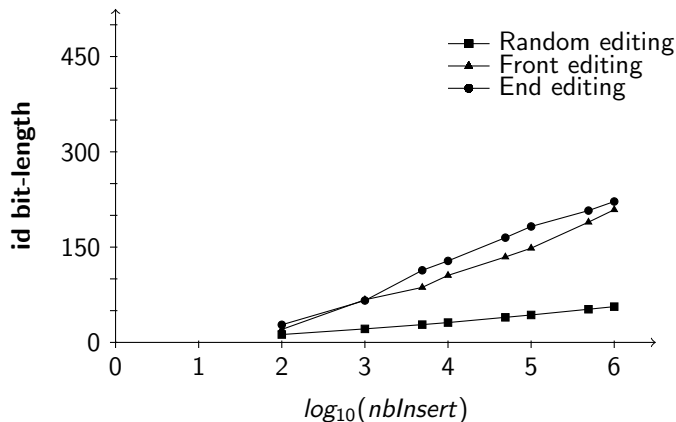


## Strategy choice



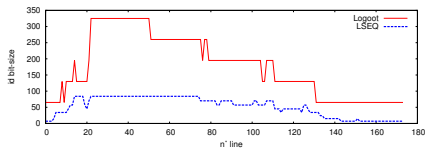
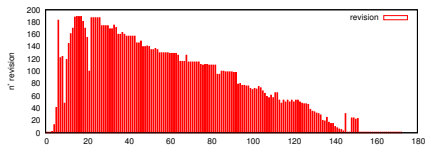
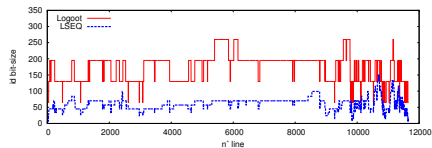
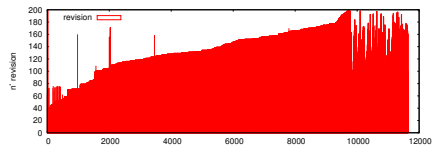
Round-Robin (**RR**) alternation of strategies *boundary+* and *boundary-*  
(*base* =  $2^{10}$ ; *boundary* = 10)

# LSEQ



LSEQ randomly alternating *boundary+* and *boundary-* and using the base doubling ( $base = 2^{4+id.size}$ ;  $boundary = 10$ )

# Comparison with Logoot I



## Comparison with Logoot II

		<b>L</b>	<b>LSEQ</b>
id-length	avg	2.65	6.25
	max	4	12
id-bit-length	avg	169.7	<b>61.24</b>
	max	256	150

Numerical values of a page edited in the end.

		<b>L</b>	<b>LSEQ</b>
id-length	avg	2.69	5.29
	max	5	8
id-bit-length	avg	172.25	<b>51.99</b>
	max	320	84

Numerical values on front edited page.

# Synthesis : experiments

- 1 Each component contributes to LSEQ :
  - Exponential tree : sub-linear behaviour
  - Multiple strategies + choice : generic
- 2 Better than Logoot :
  - On documents edited in the end
  - On documents edited in the beginning

# Conclusion and Future Works

- Proof : sub-linear space complexity.
  - $n$  operations : uniform distribution  $\Rightarrow O(\log n)$
  - $n$  operations : monotononic  $\Rightarrow O((\log n)^2)$
  - $n$  operations : worst-case  $\Rightarrow O(n^2) ???$
- Proof : worst-case happens with a negligible probability
- Concurrency effect