2nd Workshop on Compression, Text, and Algorithms 2007

Santiago de Chile. Nov 1st, 2007

A Compressed Self-Index on Words

Databases Laboratory
University of A Coruña (Spain)
+ Gonzalo Navarro

Outline

- Introduction
- Some previous Concepts
- **W**ord **C**ompressed **S**uffix **A**rray
  - Experimental results
- Presentation Layer + WiCSA
  - Motivation
  - Structure
  - Preliminary Results

Introduction

- Very large text collections of data
  - Need of IR
  - Aiming at:
    - Reducing space needs: compression
    - Improving efficiency of retrieval

T = M**IS**S**IS**SIPPI

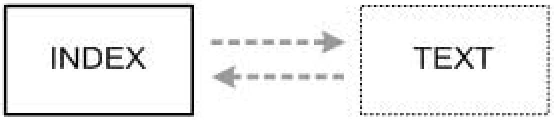
P = **IS**

- String matching
    - Searches that imply a sequential scan.
      - Over compressed text? → **0,3** · |T|
    - Indexed searches.
      - Inverted indexes, suffix arrays, …

Introduction

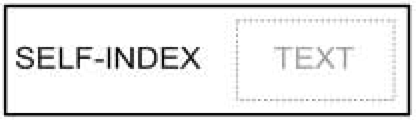
A Word-Oriented ***Self-index***

- Traditional index (with or without compression)



COUNT & **LOCATE** operations

- Self-index



COUNT, **LOCATE** & **DISPLAY** operations

Introduction

- Full-text indexes for Nat. Lang (English) text…
  - Are they actually needed?
    - Flexible and powerful
    - Reasonable small AF-index, LZ-index … (pizza&chili)
  - But…
    - a word-oriented index has less symbols
      - Smaller ?
      - Faster ?
      - Poses enough flexibility (i.e. inverted indexes)
  - So lets try a word-oriented approach

Introduction

WCSA?   Presentation layer + WCSA ?

- WCSA?
  - Self-Index based on Sadakane's CSA
    - Word-based Suffix array vs char-based.

- Presentation layer + W_iCSA?
  - How to retrieve documents containing …

    "time", "Time", "times", "TIME", "tiME" …

1. Indexing canonical words (*Separators & stopwords are not indexed*)
2. Adding a presentation layer that permits to recover the source text

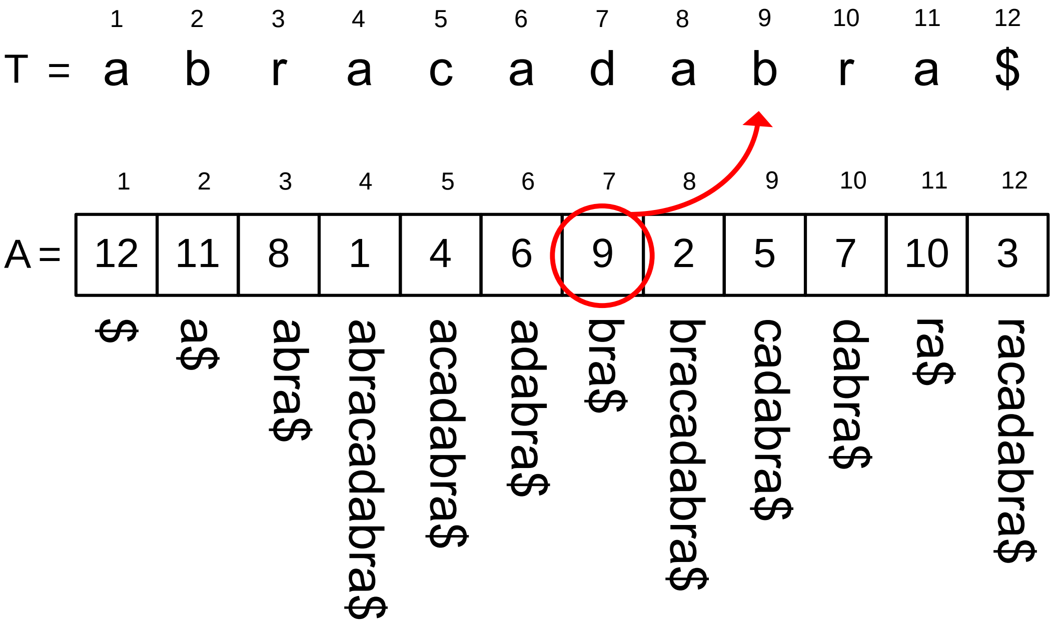
Outline

- Introduction
- Some previous Concepts
- **W**ord **C**ompressed **S**uffix **A**rray
  - Experimental results
- Presentation Layer + WiCSA
  - Motivation
  - Structure
  - Preliminary Results

Previous concepts

Suffix Array

- Suffixes of the text sorted alphabetically



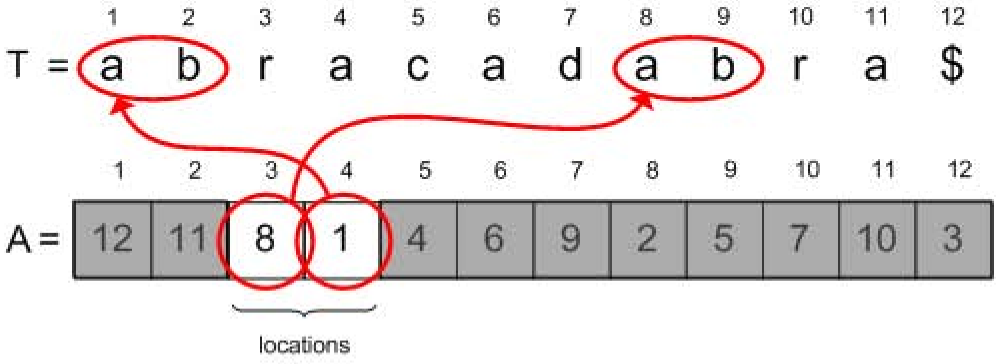
	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

	1	2	3	4	5	6	7	8	9	10	11	12
A =	12	11	8	1	4	6	9	2	5	7	10	3
	$	a$	abra$	abracadabra$	acadabra$	adabra$	bra$	bracadabra$	cadabra$	dabra$	ra$	racadabra$

Previous concepts

Suffix Array

– Binary search: i.e Pattern "ab"



	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

	1	2	3	4	5	6	7	8	9	10	11	12
A =	12	11	8	1	4	6	9	2	5	7	10	3

⇧ Fast

⇩ Space

Conceptos Previos

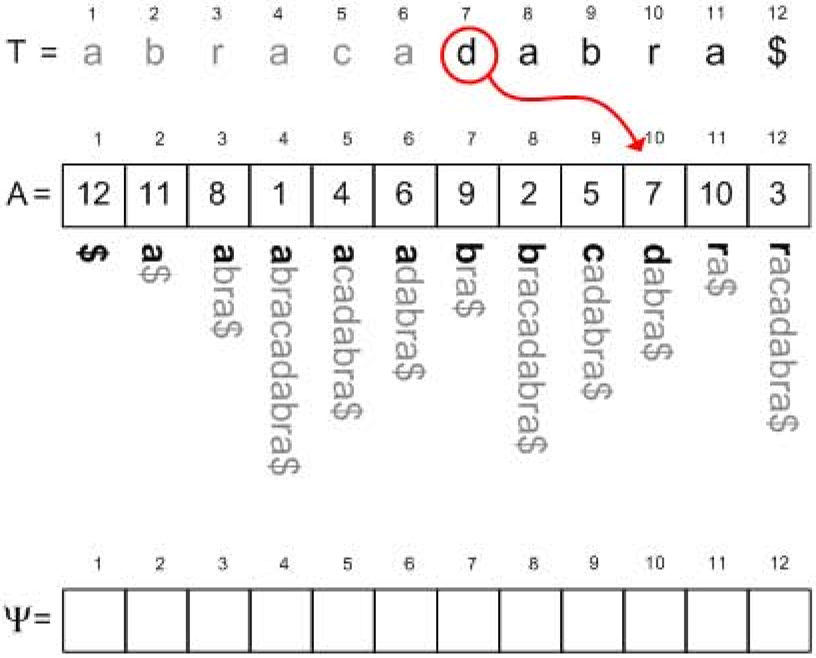
Sadakane's CSA

	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

	1	2	3	4	5	6	7	8	9	10	11	12
A =	12	11	8	1	4	6	9	2	5	7	10	3
	$	**a**$	**a**bra$	**a**bracadabra$	**a**cadabra$	**a**dabra$	**b**ra$	**b**racadabra$	**c**adabra$	**d**abra$	**r**a$	**r**acadabra$

Previous concepts

Sadakane's CSA: Ψ function

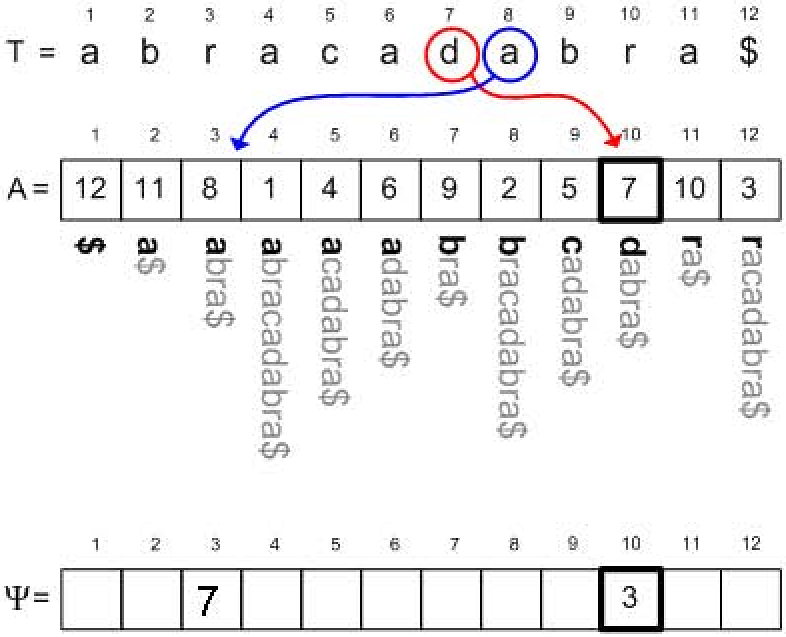


	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$
A =	12	11	8	1	4	6	9	2	5	7	10	3
	$	**a**$	**a**bra$	**a**bracadabra$	**a**cadabra$	**a**dabra$	**b**ra$	**b**racadabra$	**c**adabra$	**d**abra$	**r**a$	**r**acadabra$
Ψ=												

- ***$A[\Psi(i)] = A[i] + 1$***

Previous concepts

Sadakane's CSA: Ψ function



	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

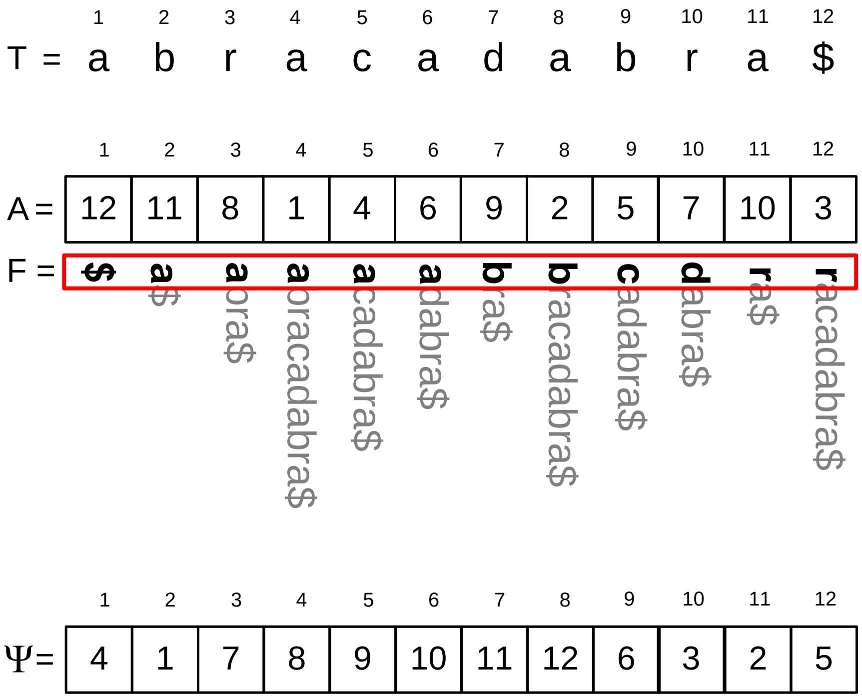
	1	2	3	4	5	6	7	8	9	10	11	12
A =	12	11	8	1	4	6	9	2	5	7	10	3
	$	a$	abra$	abracadabra$	acadabra$	adabra$	bra$	bracadabra$	cadabra$	dabra$	ra$	racadabra$

	1	2	3	4	5	6	7	8	9	10	11	12
Ψ=			7							3		

– ***$A[\Psi(10)] = A[3] = A[10] + 1 = 8$***

Previous concepts

Sadakane's CSA: Ψ & F



	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

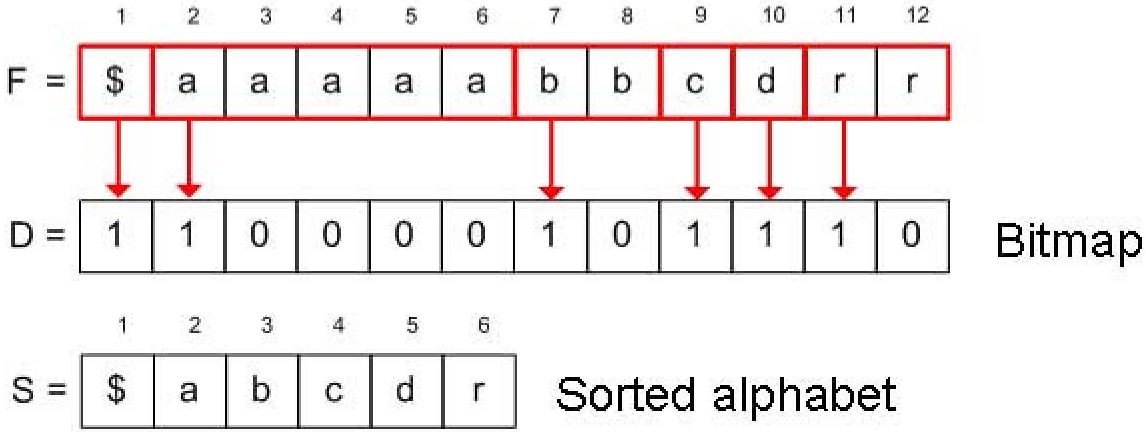
	1	2	3	4	5	6	7	8	9	10	11	12
A =	12	11	8	1	4	6	9	2	5	7	10	3
F =	**$**	**a**	**a**	**a**	**a**	**a**	**b**	**b**	**c**	**d**	**r**	**r**
	$	**a**$	**a**bra$	**a**bracadabra$	**a**cadabra$	**a**dabra$	**b**ra$	**b**racadabra$	**c**adabra$	**d**abra$	**r**a$	**r**acadabra$

	1	2	3	4	5	6	7	8	9	10	11	12
Ψ=	4	1	7	8	9	10	11	12	6	3	2	5

- ***Ψ and F suffices for performing binary search***

Previous concepts

Sadakane's CSA: Ψ & F



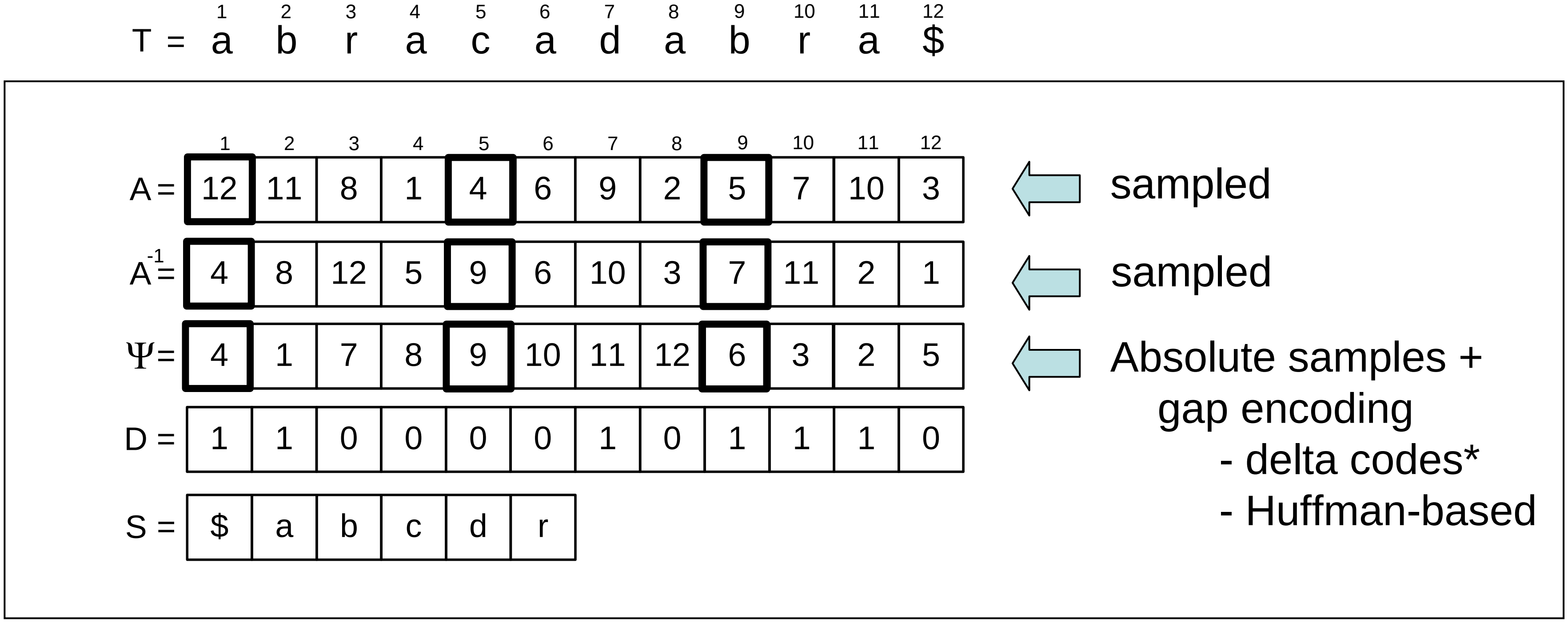
	1	2	3	4	5	6	7	8	9	10	11	12	
F =	$	a	a	a	a	a	b	b	c	d	r	r	
D =	1	1	0	0	0	0	1	0	1	1	1	0	Bitmap

	1	2	3	4	5	6	
S =	$	a	b	c	d	r	Sorted alphabet

- *Example:* $F[8] = S[rank_1(D, 8)] = S[3] = b$

Previous concepts

Sadakane's CSA: Summary



	1	2	3	4	5	6	7	8	9	10	11	12
T =	a	b	r	a	c	a	d	a	b	r	a	$

	1	2	3	4	5	6	7	8	9	10	11	12
A =	**12**	11	8	1	**4**	6	9	2	**5**	7	10	3
A^{-1} =	**4**	8	12	5	**9**	6	10	3	**7**	11	2	1
Ψ =	**4**	1	7	8	**9**	10	11	12	**6**	3	2	5
D =	1	1	0	0	0	0	1	0	1	1	1	0
S =	$	a	b	c	d	r						

A: sampled

A^{-1}: sampled

Ψ: Absolute samples +
gap encoding
- delta codes*
- Huffman-based

- *Ψ, D, S* → *count*
- *A* → *locate*
- *A^{-1}* → *display*

Outline

- Introduction
- Some previous Concepts
- **W**ord **C**ompressed **S**uffix **A**rray
    - Experimental results
- Presentation Layer + WiCSA
    - Motivation
    - Structure
    - Preliminary Results

Word Compressed Suffix Array

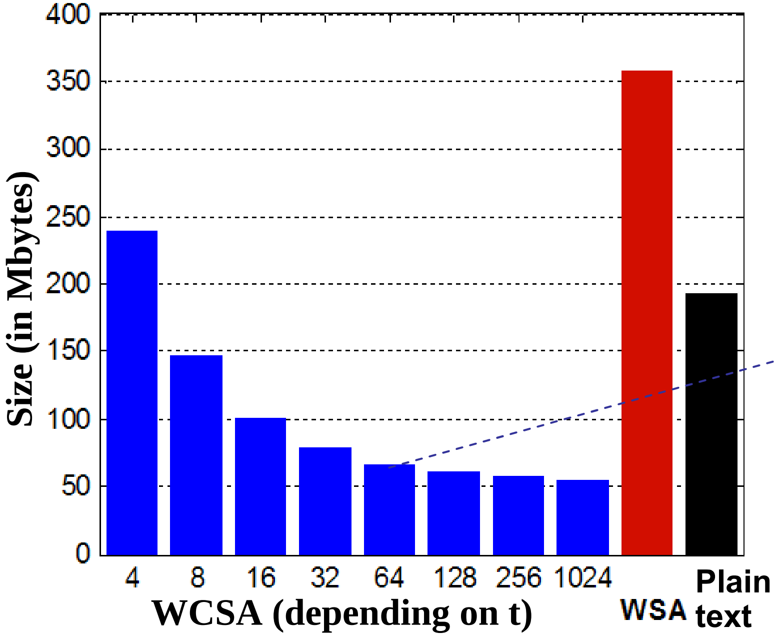
Word-based CSA (WCSA)

sA	Samples of A
sA^{-1}	Samples of A^{-1}
S	**Vocabulary of words** sorted alphabetically
D	Bitmap
Ψ	psi array (with absolute samples & gap encoding)

Word Compressed Suffix Array

Some experimental results: Space

- **WCSA – WSA**
  - *Financial Times 1994*, from *TREC-4 (200Mb English text)*
  - Pentium®IV@3.0GHz, 1GB, Debian (kernel 2.4.27), gcc 3.3.5 (-O9)

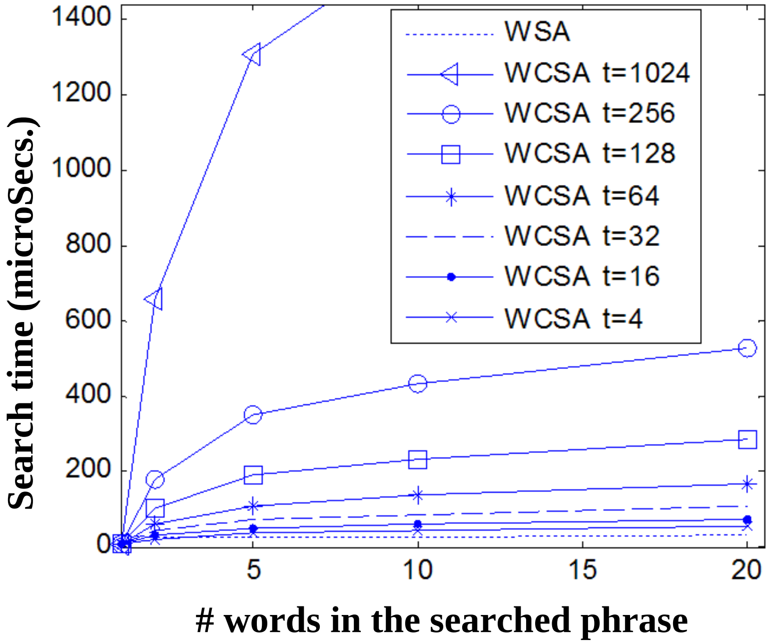


	WCSA (t = sample period)							WSA
	t=4	t=16	t=32	t=64	t=128	t=256	t=1024	
Mbytes	240,6	101,4	78,2	66,6	60,8	57,5	55,7	358,5
% cr	123,8	52,2	40,2	34,3	31,3	29,6	28,7	184,5

Word **C**ompressed **S**uffix **A**rray

Some preliminary experimental results: count time

- **Counting** occurrences of phrases with #words each
  - user-time in microseconds.



# words	WCSA (t = sample period)							WSA
	t=4	t=16	t=32	t=64	t=128	t=256	t=1024	
1	7	7	7	7	7	7	7	31
2	22	30	40	60	100	180	660	27
5	36	51	71	110	191	351	1308	27
10	43	62	85	136	234	435	1617	28
20	52	75	105	164	284	526	1975	29

- 30 - 40% space
- Not optimized but still good results

Outline

- Introduction
- Some previous Concepts
- **W**ord **C**ompressed **S**uffix **A**rray
  - Some experimental results
- Presentation Layer + WiCSA
  - Structures
  - Preliminary Results

Presentation Layer + WiCSA

Motivation

- Self-indexes on words (i.e. WCSA)
  - Good compression
  - Enough flexibility

- Indexing only “valid-words” (neither stopwords nor separators)
  - Number of indexed words much smaller
  - Searches refer only to meaningful words (in a phrase)
    - “time series” ← “time series” “time-series” “time of series”

- Indexing a **canonical** form of a word
  - “time” ← “Time”, “TIME”, “Times”,…
  - Searches obtain all the **variants** at once

- A sort of **presentation layer** permit us to recover the source text