LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Space Efficient Modifications to Structator

Benjamin Albrecht and Volker Heun

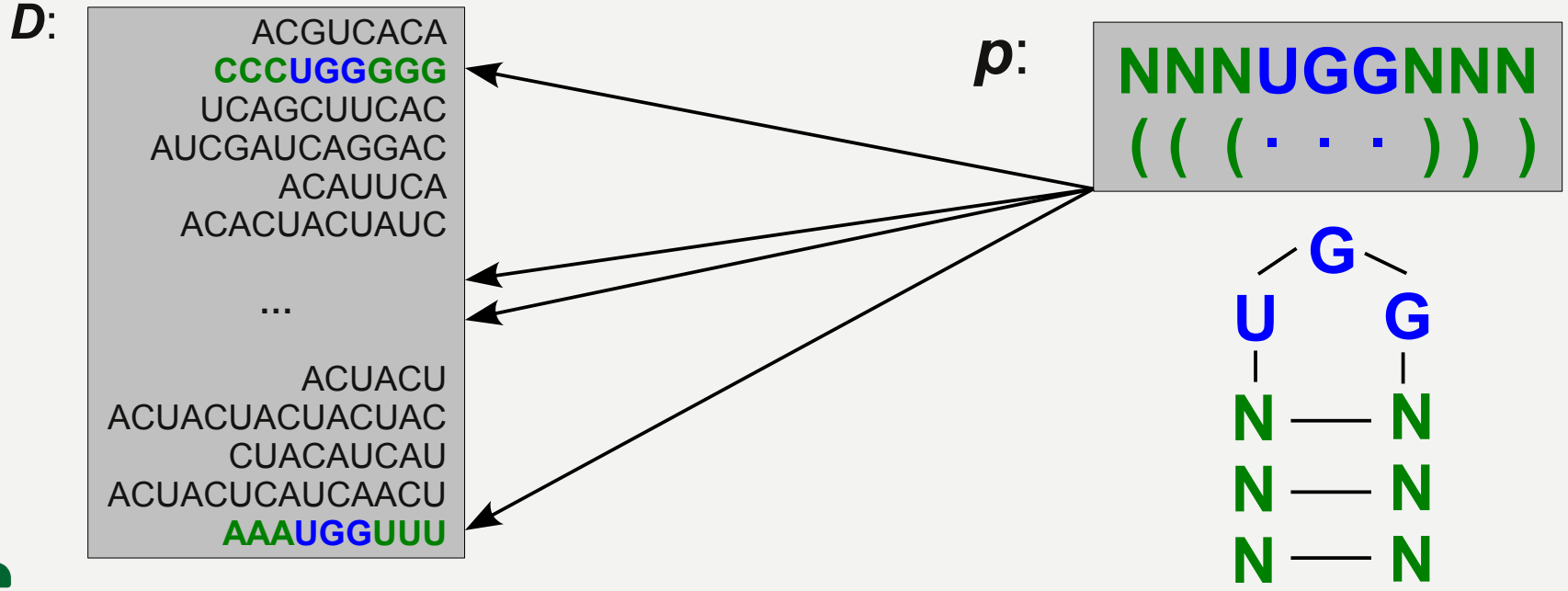
Institut für Informatik, LFE Bioinformatik, LMU München

Problem

Problem:

Given: **(1)** Database ***D*** containing sequences,
(2) Sequence-structure Pattern ***p***

Output: All sequences in ***D*** matching ***p***



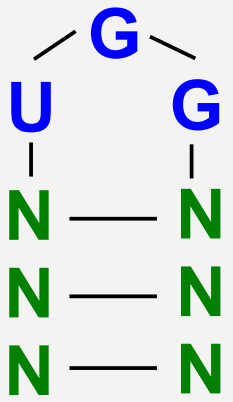
Matching Criteria

Sequence *s* **matches** pattern *p...*

- unbounded characters are **identical**
- all **complementary constraints** are fulfilled:
  (A-U), (G-C), (U-A), (C-G), (N-N)

	Match?
s_1: ACGUGUCGU	
s_2: ACGUGGAGU	
s_3: ACGUGGCGU	

***p*:** NNNUGGNNN
(((. . .)))



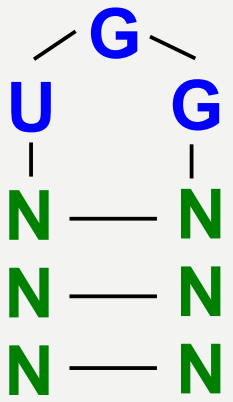
Matching Criteria

Sequence *s* **matches** pattern *p*...

- unbounded characters are **identical**
- all **complementary constraints** are fulfilled:
  (A-U), (G-C), (U-A), (C-G), (N-N)

	Match?
s_1: ACGUGUCAG	No!
s_2: ACGUGGAGU	
s_3: ACGUGGCGU	

***p*:** NNNUGGNNN
(((. . .)))



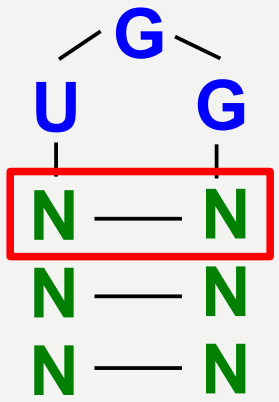
Matching Criteria

Sequence *s* **matches** pattern *p*...

- unbounded characters are **identical**
- all **complementary constraints** are fulfilled:
  (A-U), (G-C), (U-A), (C-G), (N-N)

	Match?
s_1: ACGUGUCAG	No!
s_2: ACGUGGAGU	No!
s_3: ACGUGGCGU	

***p*:** NNNUGGNNN
(((. . .)))



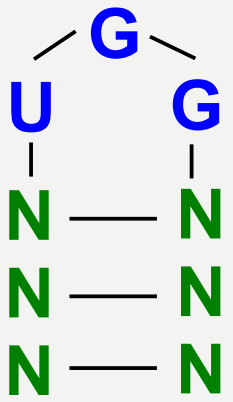
Matching Criteria

Sequence *s* **matches** pattern *p*...

- unbounded characters are **identical**
- all **complementary constraints** are fulfilled:
  (A-U), (G-C), (U-A), (C-G), (N-N)

	Match?
s_1: ACGUGUCAG	No!
s_2: ACGUGGAGU	No!
s_3: ACGUGGCGU	Yes!

***p*:** NNNUGGNNN
(((. . .)))



Solutions

Approach	Advantage +	Disadvantage -
Naive Approach	less space (+++)	bad runtime (---)
Index based Approach	good runtime (++)	more space (-)

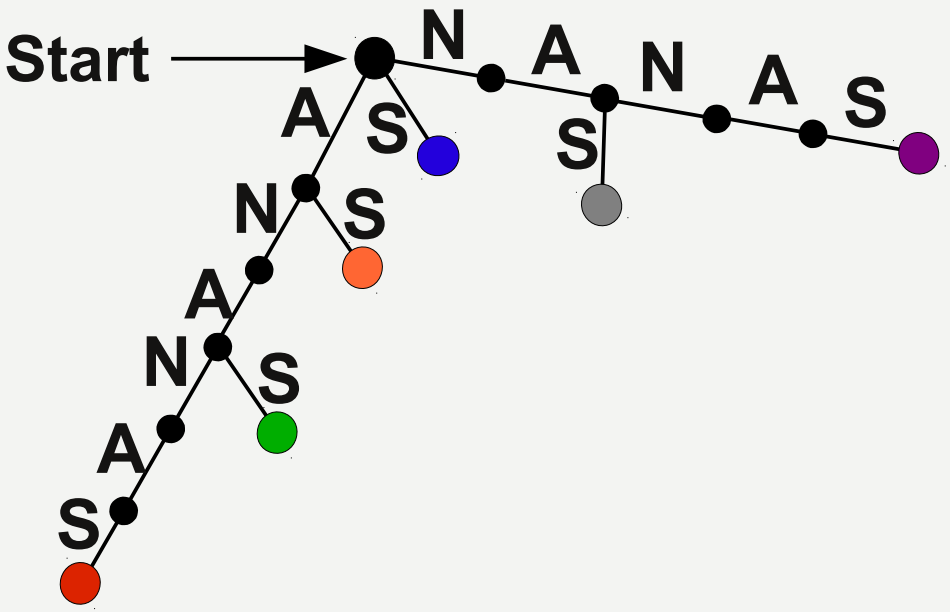
Suffix Tree

Text:

ANANAS

Suffixes:

S, **AS**, **NAS**, **ANAS**, **NANAS**, **ANANAS**



Suffix Trees	fast Search (+)	much space (--)
Suffix Arrays	fast Search (+) less space (++)	only unidirectional Search (-)
Affix Arrays	rapid Search (++) → bidirectional Search	more space (-)

Suffix Array

Text:

ANANAS

Suffixes:

S, **AS**, **NAS**, **ANAS**, **NANAS**, **ANANAS**

Suffix Array

0	**0** ANANAS
1	**2** ANAS
2	**4** AS
3	**1** NANAS
4	**3** NAS
5	**5** S

Suffix Trees	fast Search (+)	much space (--)
Suffix Arrays	fast Search (+) less space (++)	only unidirectional Search (-)
Affix Arrays	rapid Search (++) → bidirectional Search	more space (-)

Affix Array

- Further Development of a Suffix Array
- **Uni-** and **Bidirectional Search**
- Details will be explained later...

Suffix Trees	fast Search (+)	much space (--)
Suffix Arrays	fast Search (+) less space (++)	only unidirectional Search (-)
Affix Arrays	rapid Search (++) → bidirectional Search	more space (-)

Uni- vs. Bidirectional Search

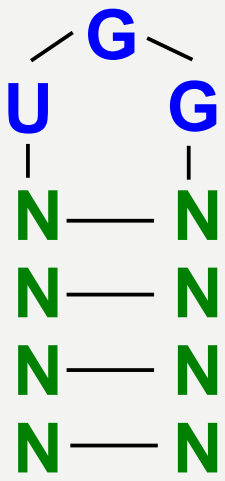
Given pattern ***p***:

NNNNUGGNNNN

((((· · ·))))

Sequence ***s***:

AGACUGGGGCU

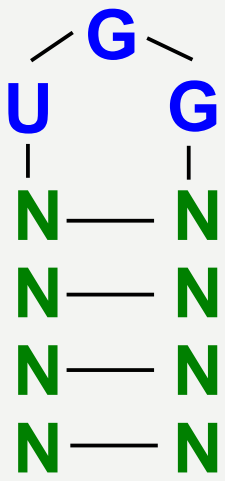


Uni- vs. Bidirectional Search

Given pattern ***p***:

NNNNUGGNNNN

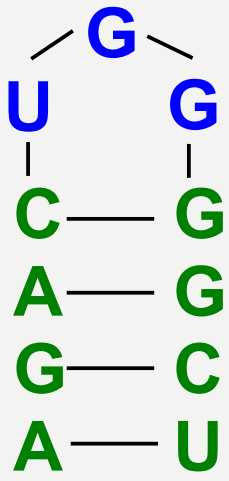
((((· · ·))))



Sequence ***s***:

AGACUGGGGCU

((((· · ·))))



Uni- vs. Bidirectional Search

Given pattern ***p***:

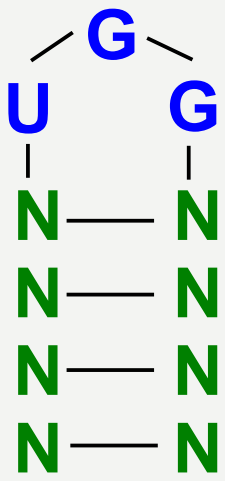
NNNNUGGNNNN

((((· · ·))))

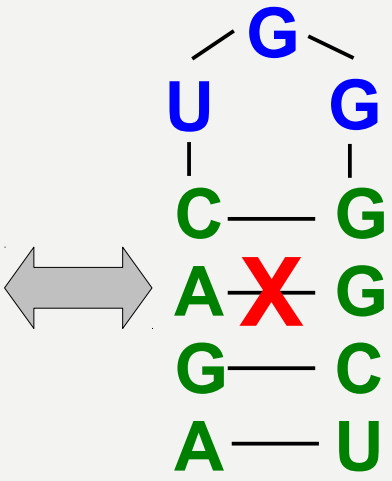
Sequence ***s***:

AGACUGGGGCU

((((· · ·))))



Inconsistent due to the pair **A,G**

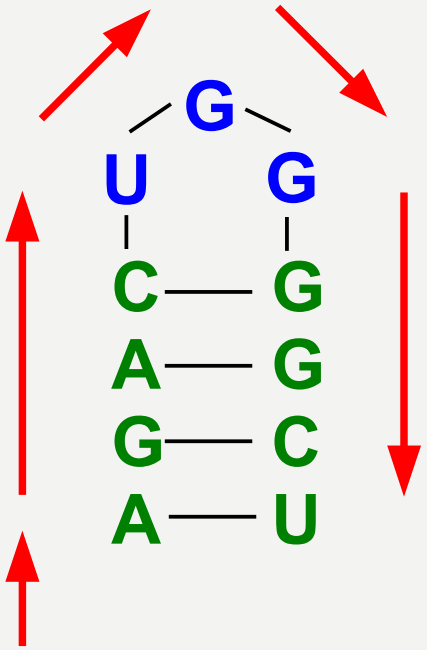


Unidirectional Search

Sequence ***s***:

AGACUGGGGCU

((((· · ·))))

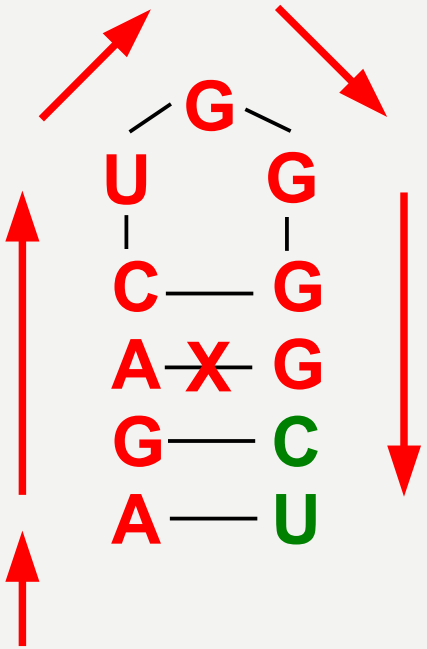


1. visit each character from **left to right**
2. check unbounded characters
3. check each pair as early as possible

Unidirectional Search

Sequence ***s***:

AGACUGGGGCU
((((· · ·))))



1. visit each character from **left to right**
2. check unbounded characters
3. check each pair as early as possible

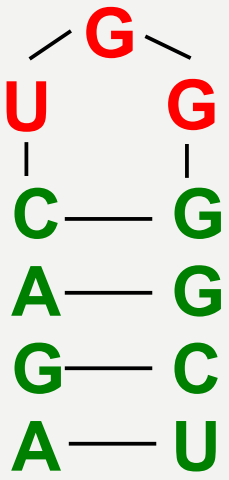
→ need **9 steps** to find inconsistent pair **A,G**

Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
((((. . .))))

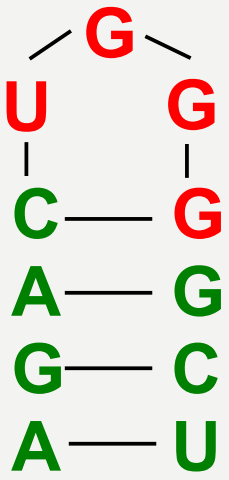
1. check unbounded characters



Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
((((. . .))))



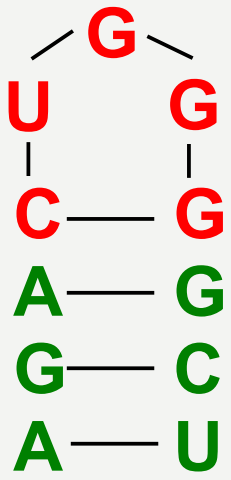
1. check unbounded characters
2. check each pair **consecutively** from top to bottom

Right Extension

Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
((((· · ·))))



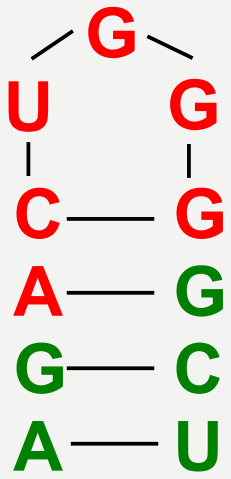
1. check unbounded characters
2. check each pair **consecutively** from top to bottom

Left Extension

Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
(((((· · ·)))))



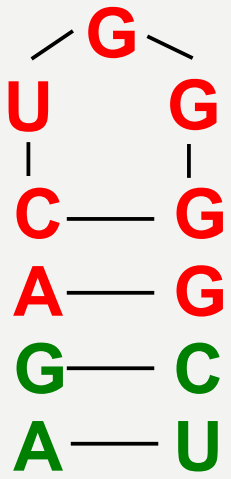
1. check unbounded characters
2. check each pair **consecutively** from top to bottom

Left Extension

Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
(((((· · ·)))))



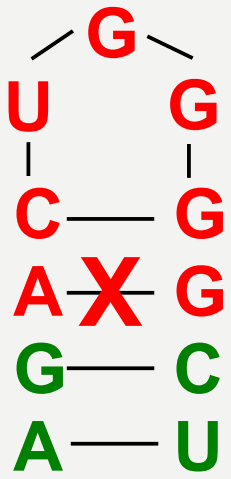
1. check unbounded characters
2. check each pair **consecutively** from top to bottom

Right Extension

Bidirectional Search

Sequence ***s***:

AGACUGGGGCU
(((((· · ·)))))



1. check unbounded characters
2. check each pair **consecutively** from top to bottom

→ need **7 steps** to find inconsistent pair **A,G**

Affix Array

Suffix Array suf_F

0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

T=AUAGCUGCUGCUGCA

Affix Array

LCP Array $\mathbf{lcp}_F$

	$\mathbf{suf}_F$			
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

T=AUAGCUGCUGCUGCA

Affix Array

Affix-Link Array $\mathbf{aflk}_F$

	$\mathbf{suf}_F$	$\mathbf{lcp}_F$		
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

T=AUAGCUGCUGCUGCA

Affix Array

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG	7	0	11	7
AUAGCUG	8	1	8	8
AUAGCUGCUG	9	4	5	9
AUAGCUGCUGCUG		7	2	10
AU	11	0	13	11
AUAGCU	8	1	9	12
AUAGCUGCU	9	3	6	13
AUAGCUGCUGCU		6	3	14
			15	15

$T=AUAGCUGCUGCUGCA$ $T^{-1}=ACGUCGUCGUCGAUA$

Affix Link

p = NNNUGCUNNN

(((. . . .)))

T=AUAGCUGCUGCUGCA

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG	7	0	11	7
AUAGCUG	8	1	8	8
AUAGCUGCUG	9	4	5	9
AUAGCUGCUGCUG		7	2	10
AU	11	0	13	11
AUAGCU	8	1	9	12
AUAGCUGCU	9	3	6	13
AUAGCUGCUGCU		6	3	14
			15	15

Affix Link

p = NNNUGCUNNN

((((....))))

T=AUAGCUGCUGCUGCA

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG	7	0	11	7
AUAGCUG	8	1	8	8
AUAGCUGCUG	9	4	5	9
AUAGCUGCUGCUG		7	2	10
AU	11	0	13	11
AUAGCU	8	1	9	12
AUAGCUGCU	9	3	6	13
AUAGCUGCUGCU		6	3	14
			15	15

Affix Array

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG	7	0	11	7
AUAGCUG	8	1	8	8
AUAGCUGCUG	9	4	5	9
AUAGCUGCUGCUG		7	2	10
AU	11	0	13	11
AUAGCU	8	1	9	12
AUAGCUGCU	9	3	6	13
AUAGCUGCUGCU		6	3	14
			15	15

(4n+1n+4n) x 2= 9n x 2 = 18n Bytes

Affix Array

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0	3	GCA
8	9	2	4	GCUGCA
9	6	5	5	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG	7	0	11	7
AUAGCUG	8	1	8	8
AUAGCUGCUG	9	4	5	9

Modification:

reduce Space
keep bidirectional search

(4n+1n+4n) x 2= 9n x 2 = 18n Bytes

Structator - Overview

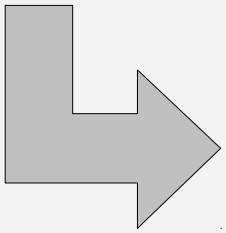
(1) Index Creation: Compute Index for a Database ***D***

Input: Database ***D***
Output: Index *on* ***D***

(2) Pattern Search: find each occurrence of ***p*** in ***D***

Input: Pattern ***p,*** Index on ***D***
Output: each occurrence of ***p*** in ***D***

- search after ***p*** within the index
  → via bidirectional search



Focus of our Modifications

1st Modification

1st Modification:

- **Omit** Affix Links Array
  → Space reduced from **18n to 10n Bytes**

- Compute each Affix Link via ***Binary Search***
  → **Less space** but higher runtime.

1st Modification

	suf_F	lcp_F	$aflk_F$	
0	2	0	0	AGCUGCUGCUGCA
1	0	1		AUAGCUGCUGCUGCA
2	14	1		A
3	13	0	3	CA
4	10	1	4	CUGCA
5	7	4	5	CUGCUGCA
6	4	7		CUGCUGCUGCA
7	12	0		GCA
8	9	2		GCUGCA
9	6	5	6	GCUGCUGCA
10	3	8		GCUGCUGCUGCA
11	1	0	11	UAGCUGCUGCUGCA
12	11	1	4	UGCA
13	8	3	5	UGCUGCA
14	5	6		UGCUGCUGCA
15	15	0		

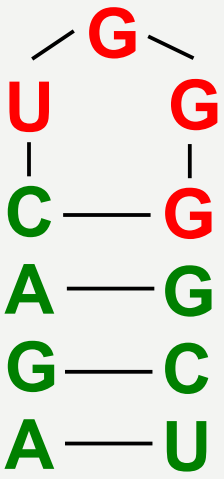
	$aflk_R$	lcp_R	suf_R	
AUAGCUGCUGCUGCA	0	0	0	0
AUA		1	12	1
A		1	14	2
AUAGC	7	0	10	3
AUAGCUGC	8	2	7	4
AUAGCUGCUGC	9	5	4	5
AUAGCUGCUGCUGC		8	1	6
AUAG		0	11	7
AUAGCUG		1	8	8
AUAGCUGCUG	9	4	5	9
AUAGCUGCUGCUG		7	2	10
AU	11	0	13	11
AUAGCU	8	1	9	12
AUAGCUGCU	9	3	6	13
AUAGCUGCUGCU		6	3	14
			15	15

(4n+1n+~~4n~~) x 2= 5n x 2 = 10n Bytes

1st Modification

Sequence ***s***:

AGACUGGGGCU
((((. . .))))

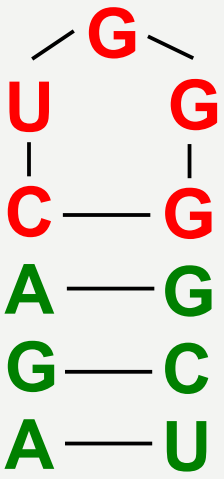


- **reverse** already matched pattern:
  UGGG → GGGU
- search for **GGGU** in the **reversed** suffix array

1st Modification

Sequence ***s***:

AGACUGGGGCU
((((. . .))))



- **reverse** already matched pattern:
  UGGG → GGGU
- search for **GGGU** in the **reversed** suffix array
- continue search into **other** direction

2nd Modification

2nd Modification:

- Replace Affix Links Array by a ***LCP-Tree Array***
  → Space reduced from **18n to 12n Bytes**

- Compute each Affix Link via ***Improved Binary Search***
  → **Less space** but a (reduced) higher runtime.

2nd Modification

Given: Pattern *p* with size m,
Text *t* with size n, Suffix Array on Text *t*
Problem: Find all occurrences of *p* in *t.*

- **Binary Search** (1st Modification):
  Runtime: ***m*log(n)***

- **Improved Binary Search*** (2nd Modification)
  Runtime: ***m+log(n)***
  → **less** character comparisons but a little bit more space

* Manber, U.; Myers, G.: Suffix arrays: a new method for on-line string searches

Experiments

2 Kind of Experiments:

(1) **Index Construction**
→ **USER** Runtime
→ Max Memory Usage

(2) **Pattern Search**
→ **USER** Runtime
→ **REAL** Runtime

System

Database: **Rfam** release **10.0**

Pattern: **397 RNA-Sequence-Structure-Patterns**
→ available from www.zbh.uni-hamburg.de/Structator

System: two **2.4** GHz dual cores and **4** GB RAM
→ max memory limit: **6,5** GB

Swapping Effects !!!

Experimental Results

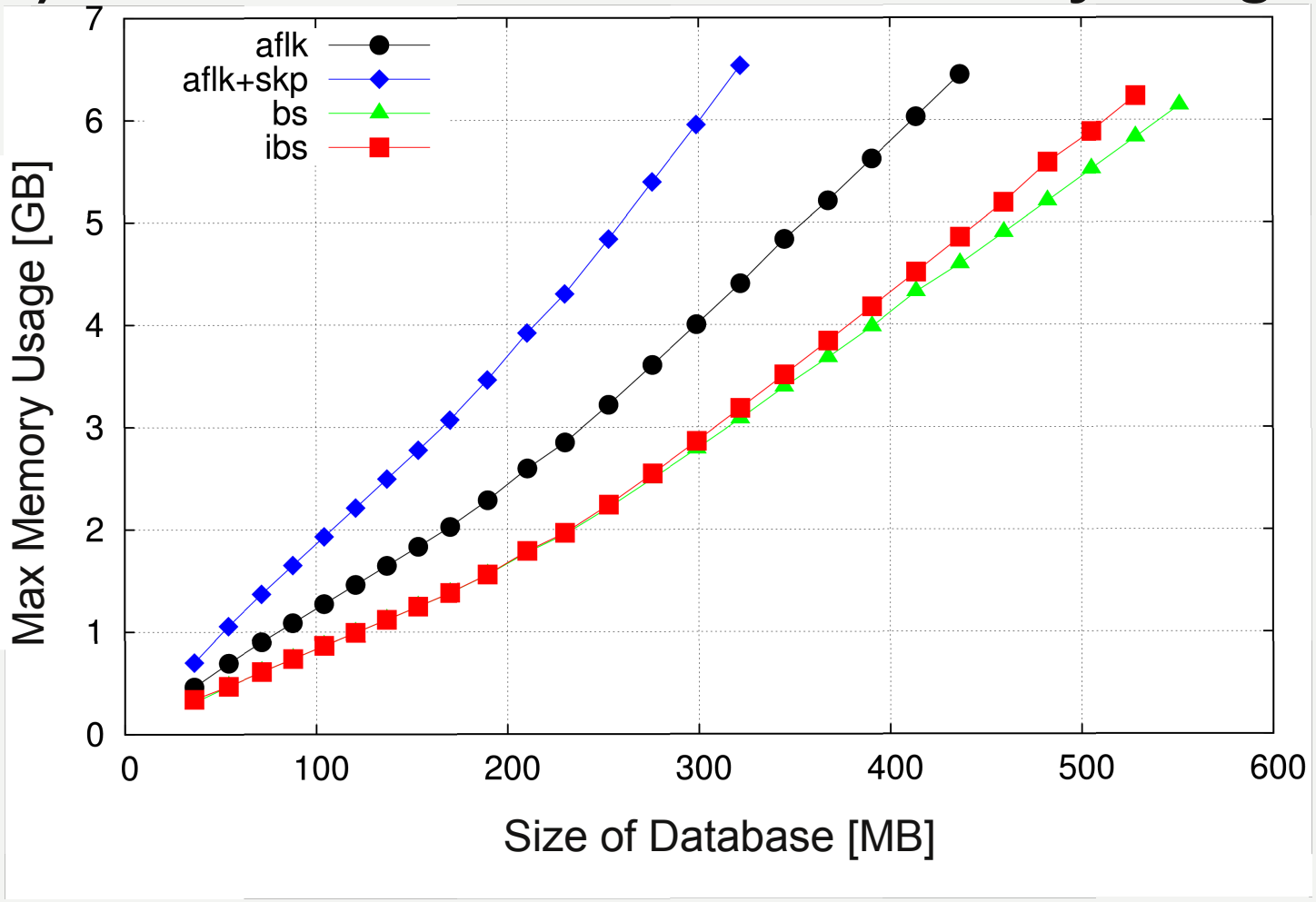
* accelerates computation of ***aflk*** *table*

4 Different Modes of Structator:

Mode	*suf*	*lcp*	*aflk*	*skp**	*lcp-Tree*	Space
aflk	x	x	x			18n
aflk+skp	x	x	x	x		26n
bs	x	x				10n
ibs	x	x			x	12n

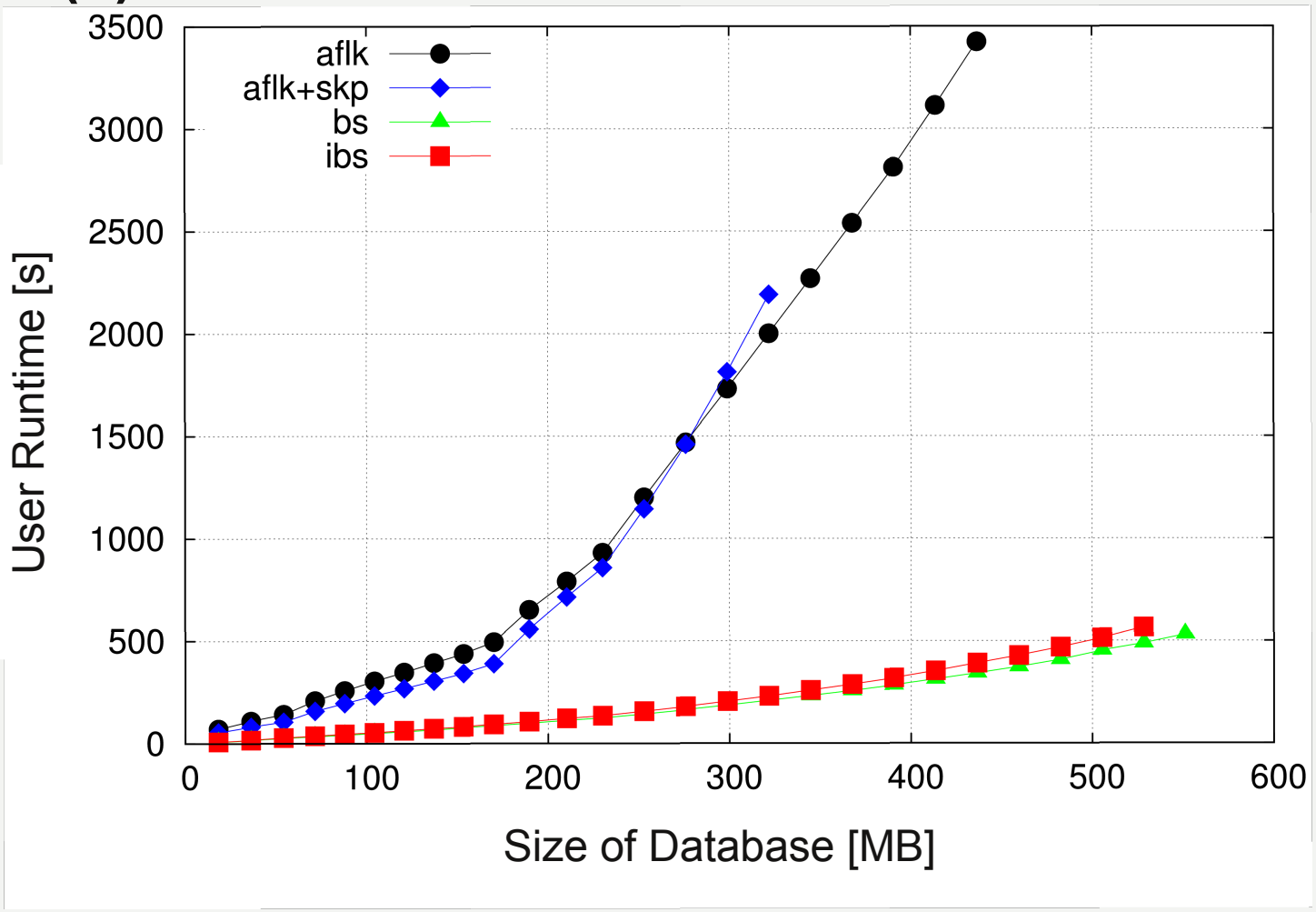
Experimental Results

(1) Index Construction – Max Memory Usage



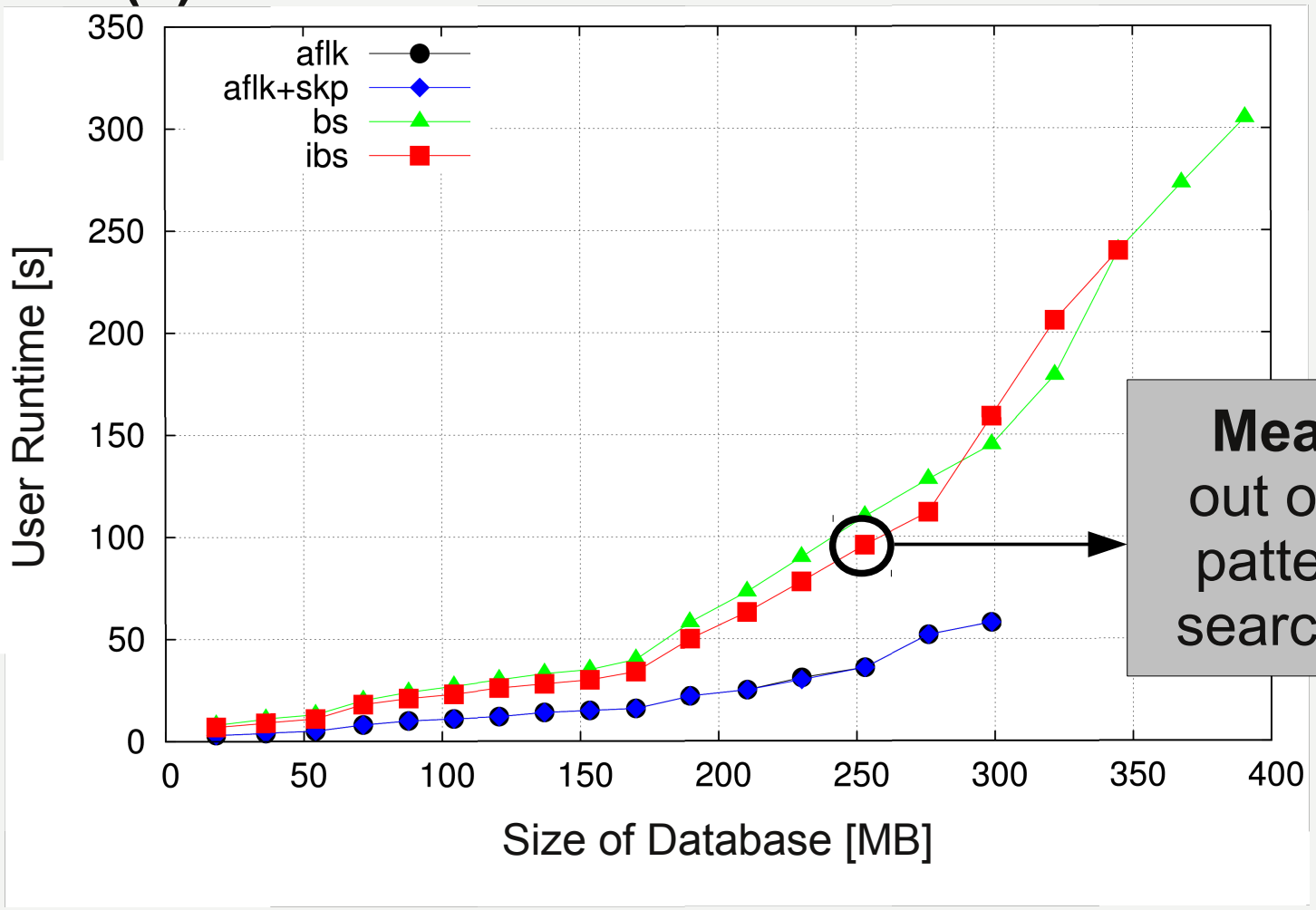
Experimental Results

(1) Index Construction – USER Runtime



Experimental Results

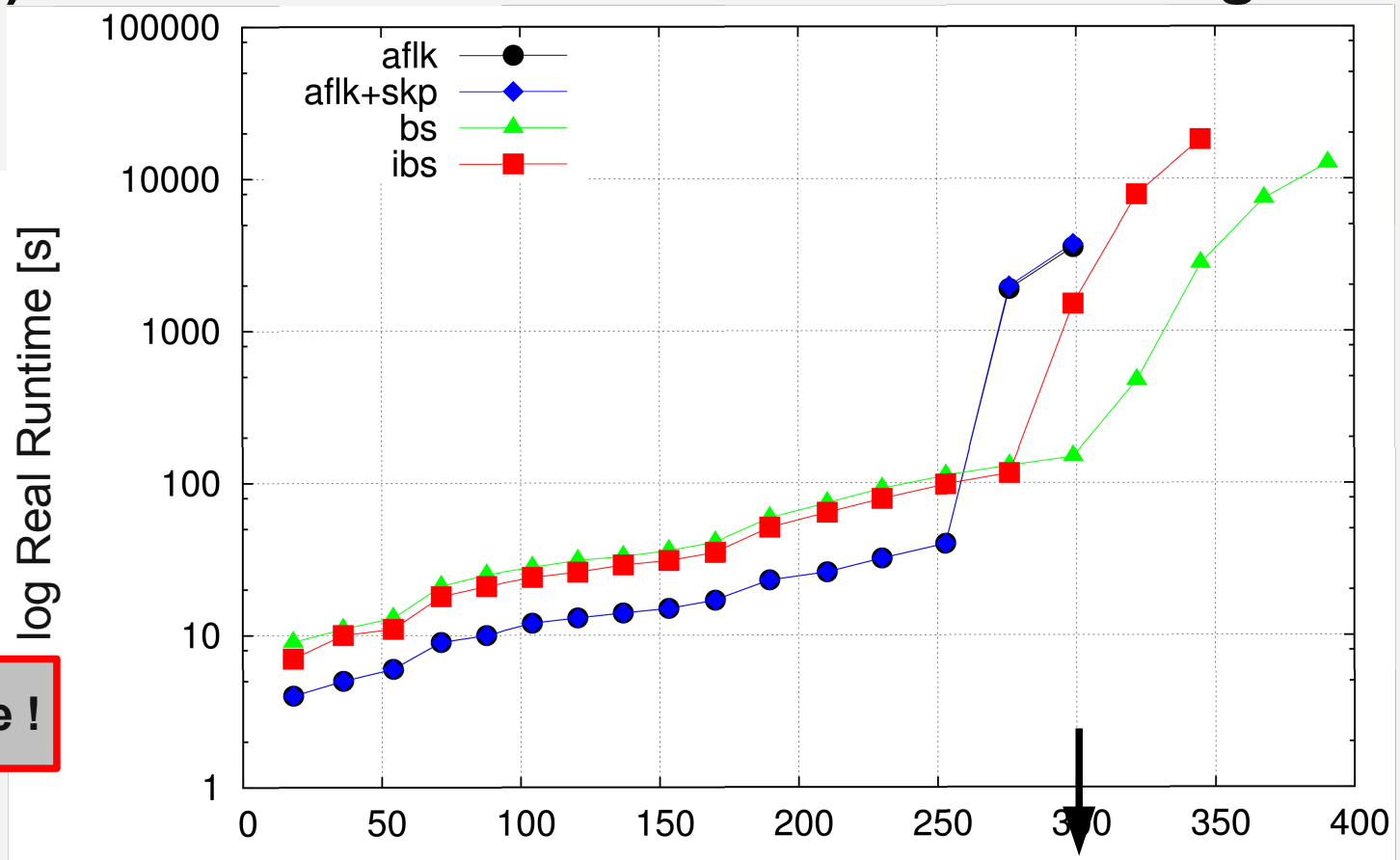
(2) Pattern Search – USER Runtime



Mean out of **9** pattern searches

Experimental Results

(2) Pattern Search – REAL Runtime in log scale



log scale !

Affix Array ~ 4GB

Summary

2 Modifications to Structator:

(1) Affix Links via **Binary Search**
(2) Affix Links via **Improved Binary Search**

→ **reduced** Space but higher runtime

More efficient if affix array gets close to system's memory!

Our two Modifications are **available**
at the official **Structator** website:

http://www.zbh.uni-hamburg.de/Structator

Thanks

Structator Team

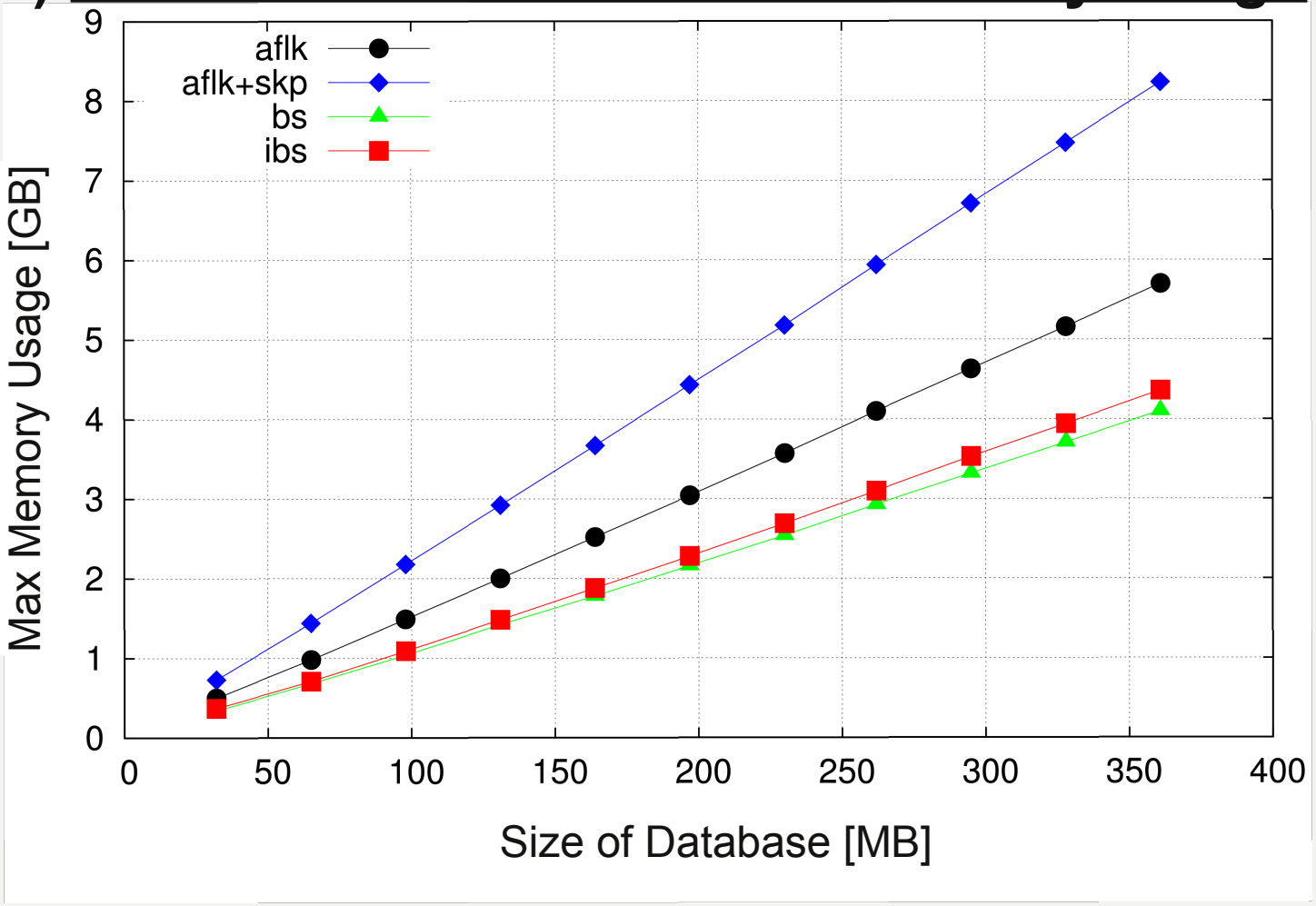
Fernando Meyer[1], Stefan Kurtz[1], Rolf Backofen[2], Sebastian Will[2,3], and Michael Beckstette[1]

[1]Center for Bioinformatics, University of Hamburg

[2]Chair for Bioinformatics, University of Freiburg

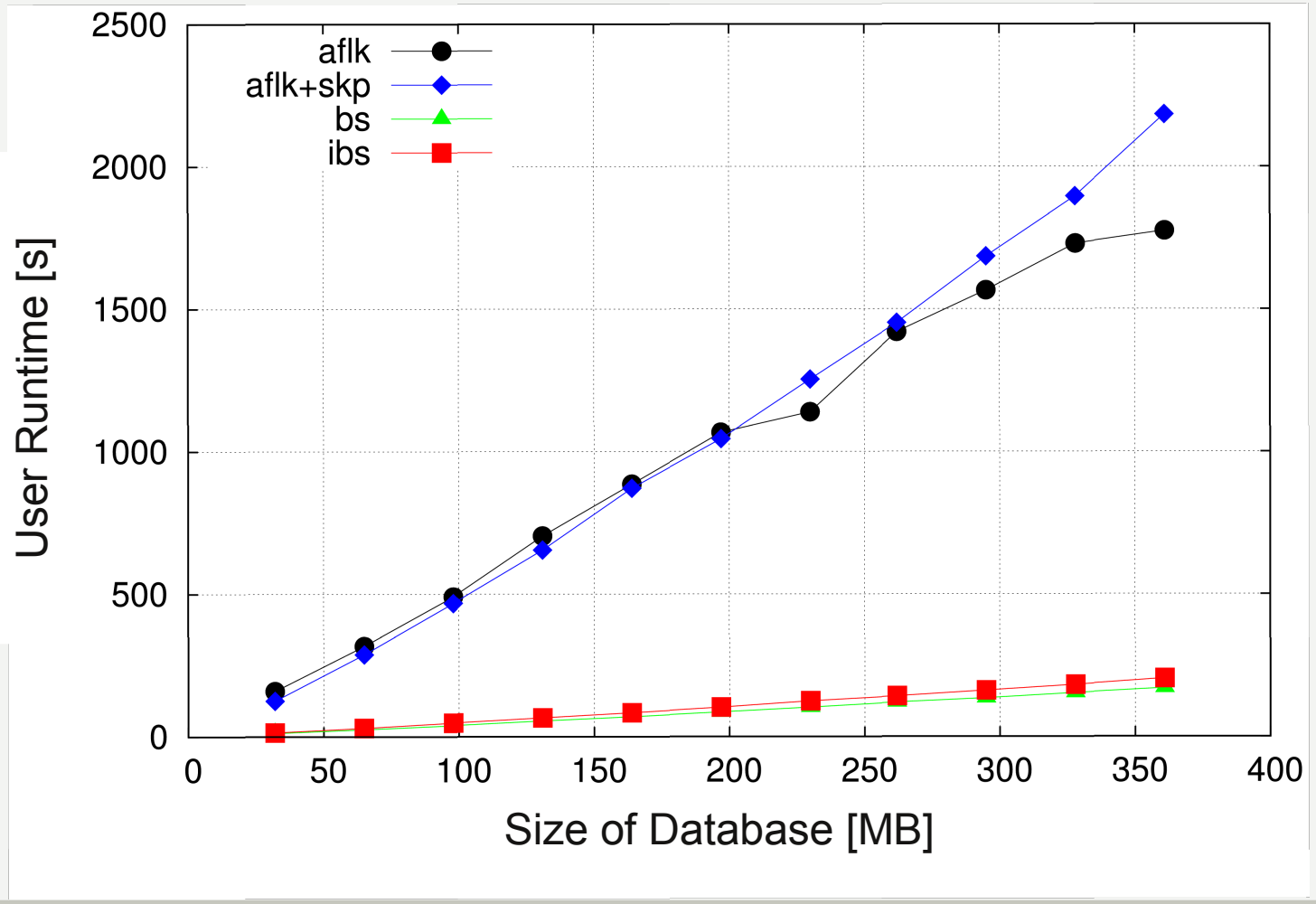
[3]Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology

(1) Index Construction – Max Memory Usage



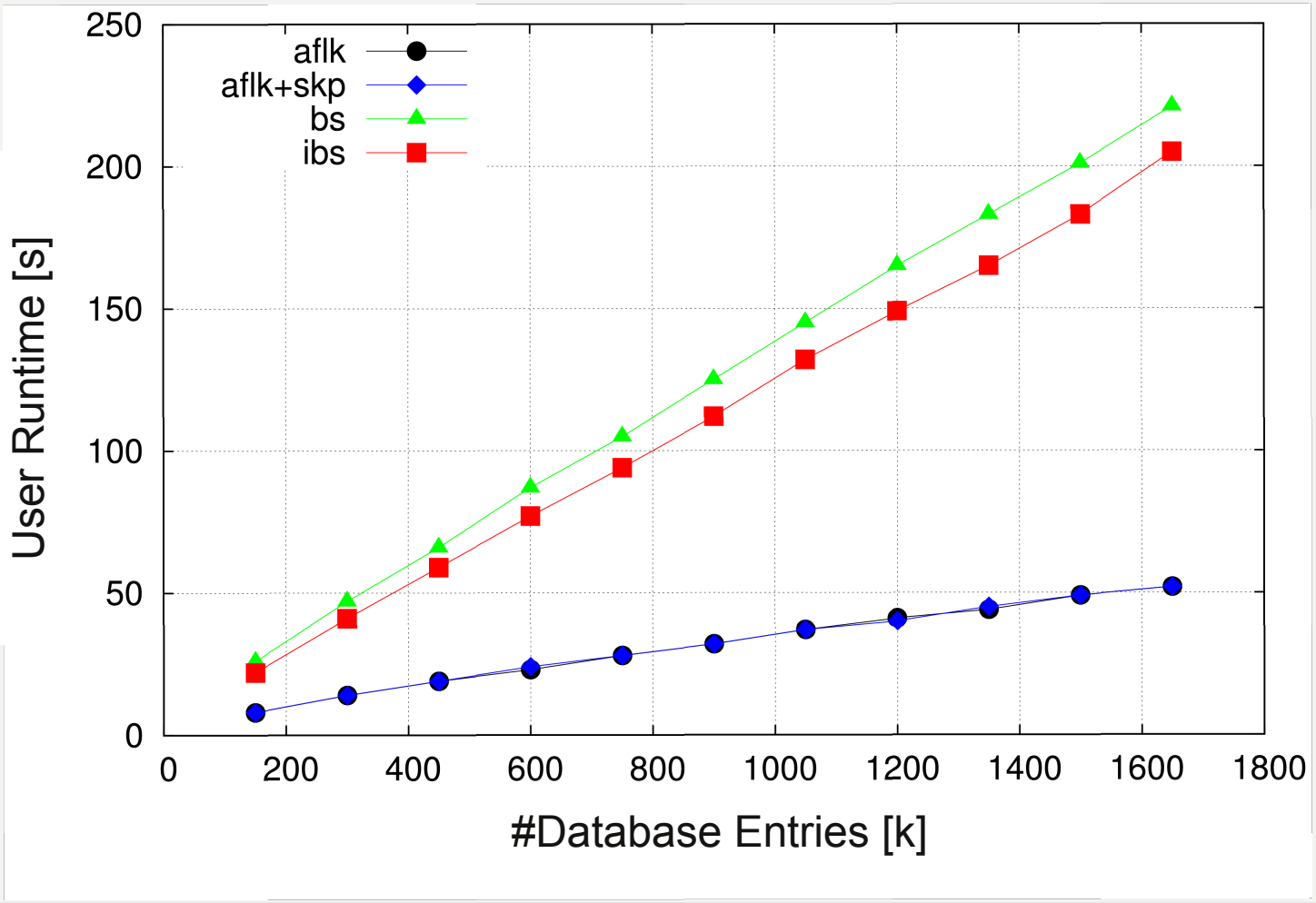
Experimental Results

(1) Index Construction – Runtime



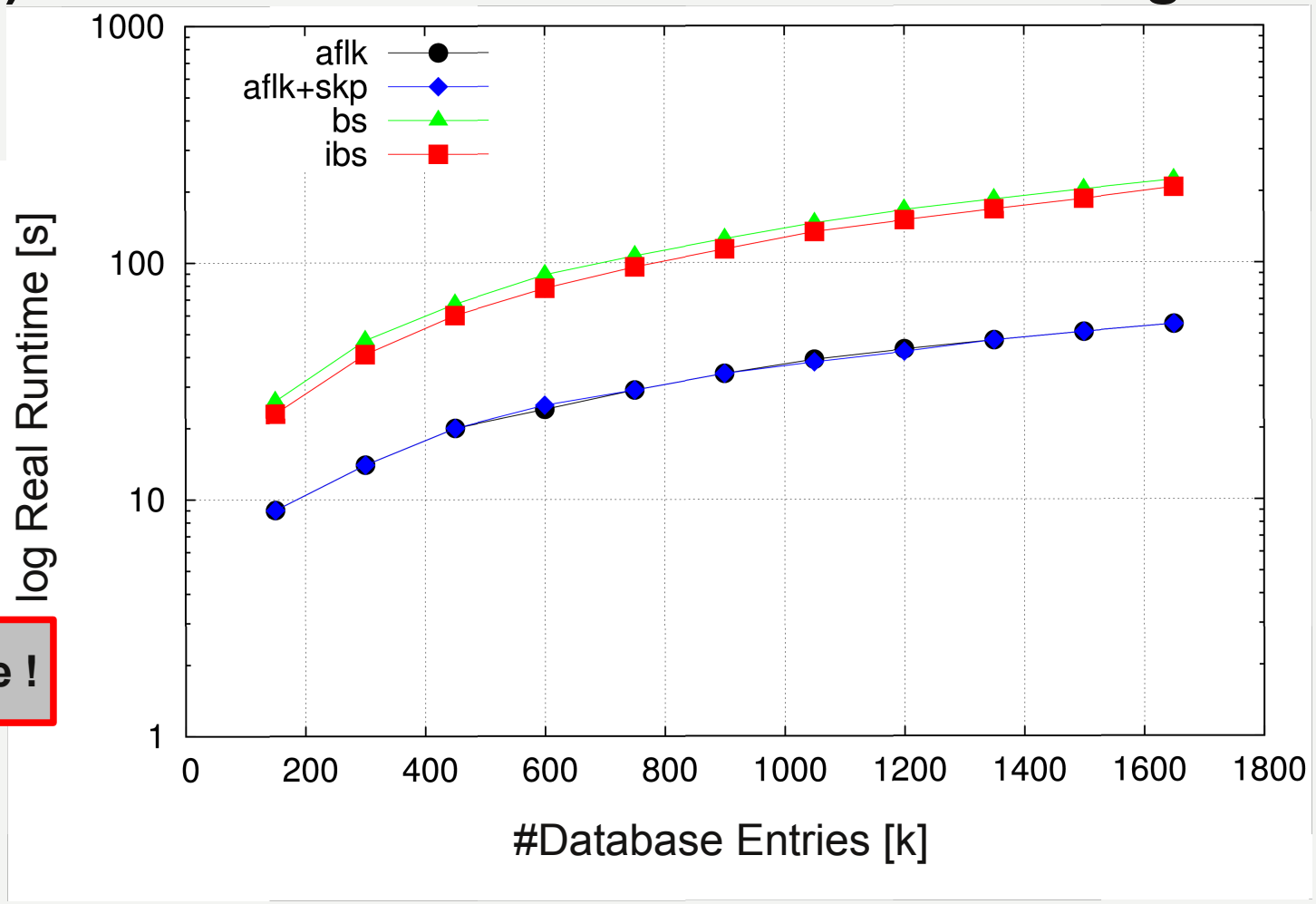
Experimental Results

(2) Pattern Search – USER Runtime



Experimental Results

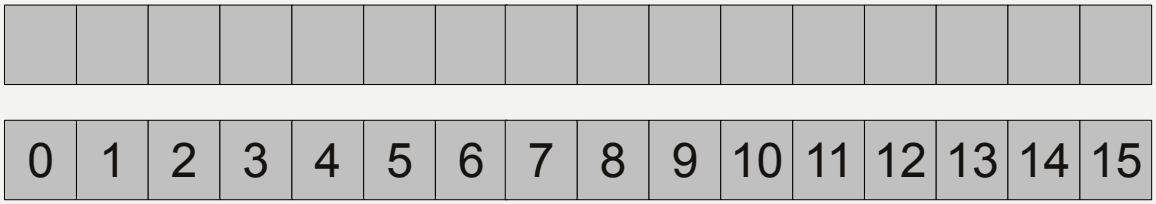
(2) Pattern Search – REAL Runtime in log scale



log scale !

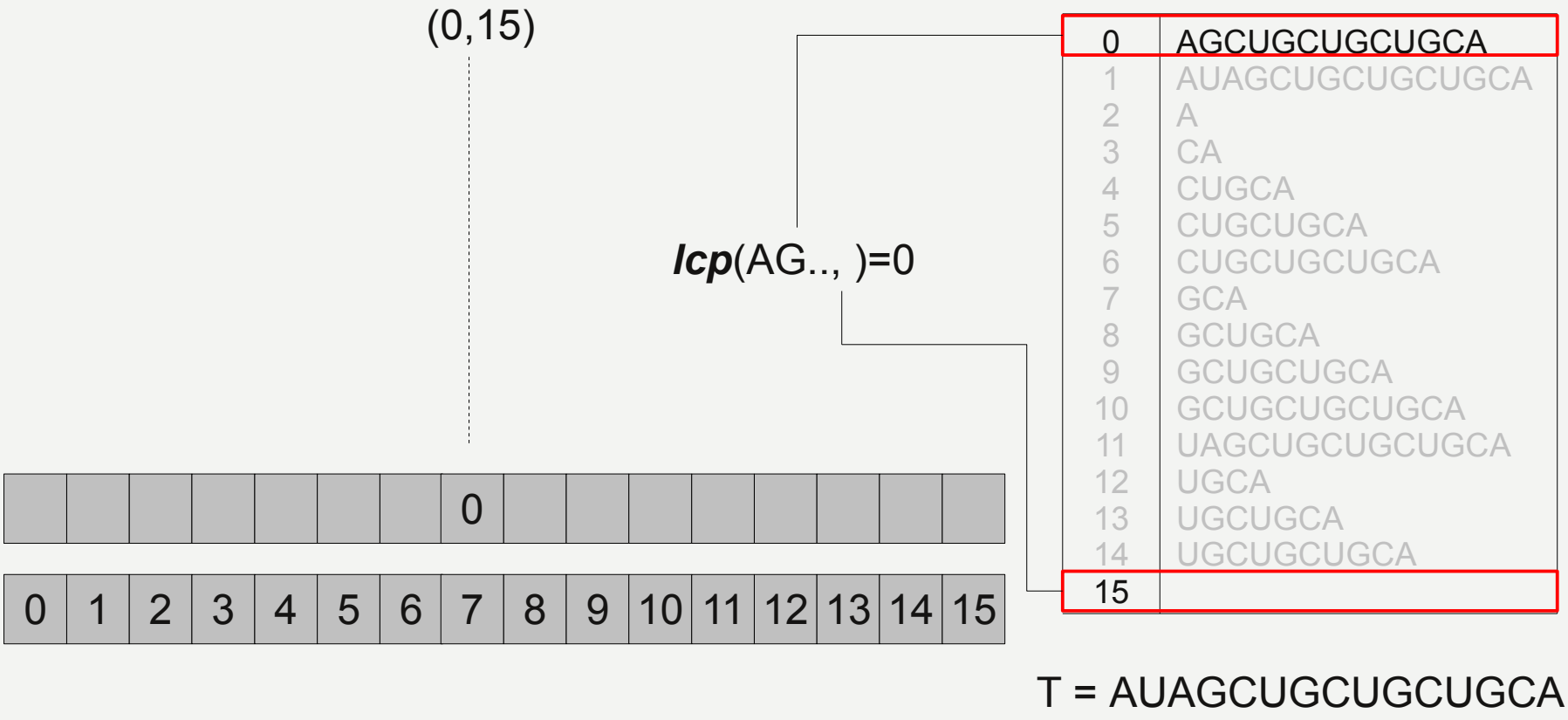
LCP Tree

0	AGCUGCUGCUGCA
1	AUAGCUGCUGCUGCA
2	A
3	CA
4	CUGCA
5	CUGCUGCA
6	CUGCUGCUGCA
7	GCA
8	GCUGCA
9	GCUGCUGCA
10	GCUGCUGCUGCA
11	UAGCUGCUGCUGCA
12	UGCA
13	UGCUGCA
14	UGCUGCUGCA
15	

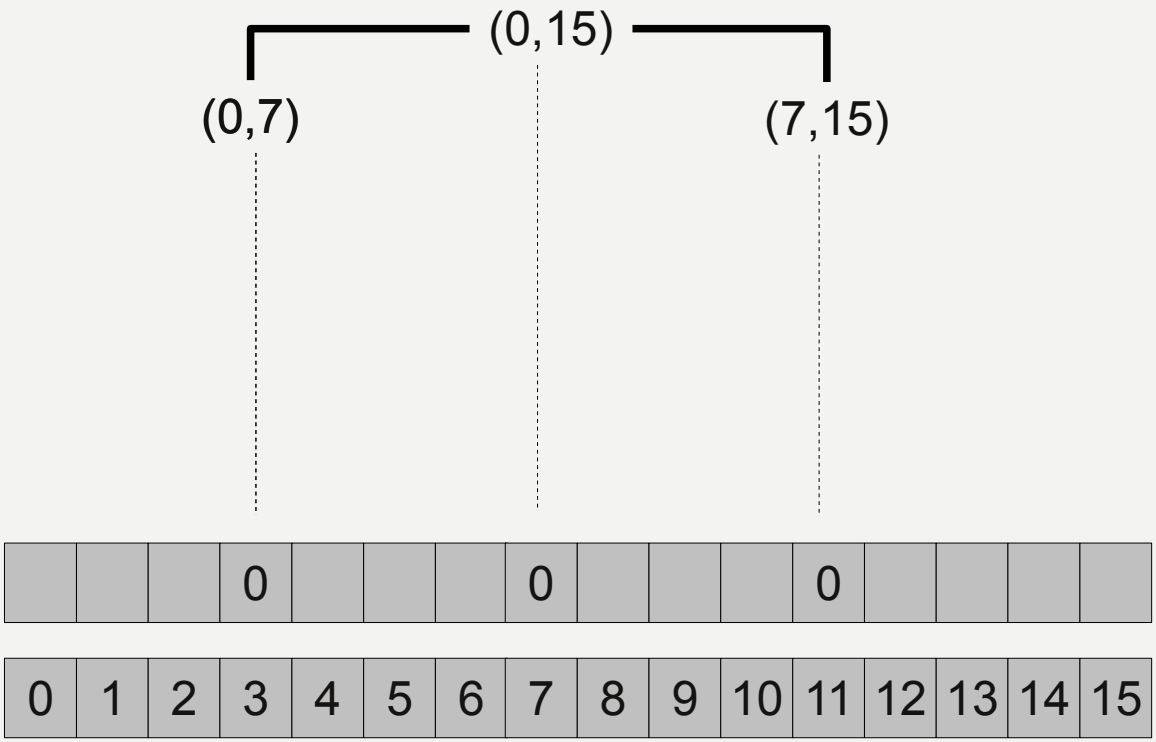


T = AUAGCUGCUGCUGCA

LCP Tree


(0,15)
lcp(AG..,)=0
0 AGCUGCUGCUGCA
1 AUAGCUGCUGCUGCA
2 A
3 CA
4 CUGCA
5 CUGCUGCA
6 CUGCUGCUGCA
7 GCA
8 GCUGCA
9 GCUGCUGCA
10 GCUGCUGCUGCA
11 UAGCUGCUGCUGCA
12 UGCA
13 UGCUGCA
14 UGCUGCUGCA
15
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
T = AUAGCUGCUGCUGCA

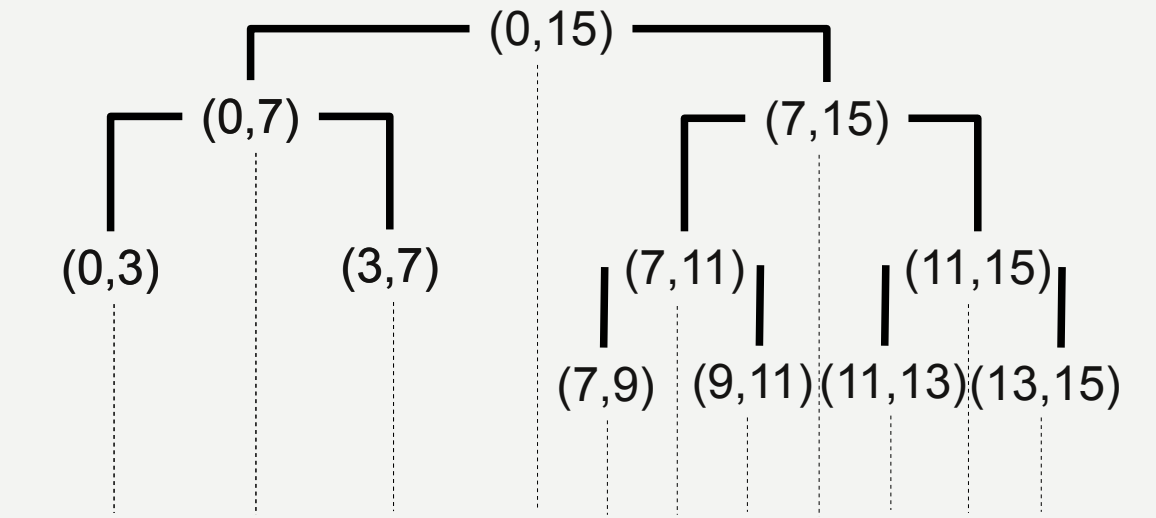
LCP Tree



0	AGCUGCUGCUGCA
1	AUAGCUGCUGCUGCA
2	A
3	CA
4	CUGCA
5	CUGCUGCA
6	CUGCUGCUGCA
7	GCA
8	GCUGCA
9	GCUGCUGCA
10	GCUGCUGCUGCA
11	UAGCUGCUGCUGCA
12	UGCA
13	UGCUGCA
14	UGCUGCUGCA
15	

T = AUAGCUGCUGCUGCA

LCP Tree



0	AGCUGCUGCUGCA
1	AUAGCUGCUGCUGCA
2	A
3	CA
4	CUGCA
5	CUGCUGCA
6	CUGCUGCUGCA
7	GCA
8	GCUGCA
9	GCUGCUGCA
10	GCUGCUGCUGCA
11	UAGCUGCUGCUGCA
12	UGCA
13	UGCUGCA
14	UGCUGCUGCA
15	

T = AUAGCUGCUGCUGCA