

Indexing Structures and Next Generation Sequencing (NGS)

Thierry Lecoq

Thierry.Lecoq@univ-rouen.fr

Computer Science, Information Processing and Systems Laboratory (LITIS EA4108)
University of Rouen, France

Journées Montoises

10th September 2010 – Amiens – France



Outline

- 1 Molecular Biology
- 2 Indexing Structures
- 3 Mapping Reads
- 4 Perspectives

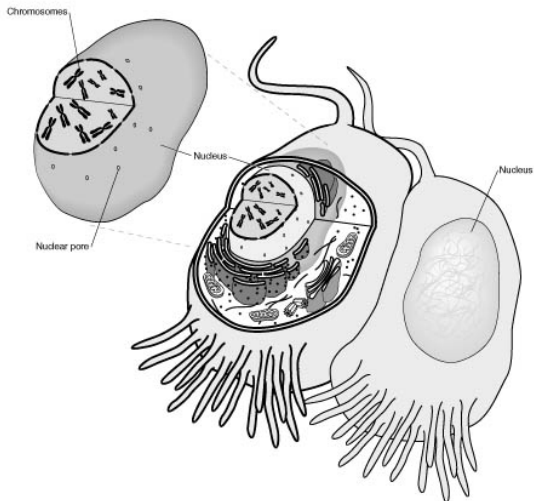
Outline

- 1 **Molecular Biology**
- 2 Indexing Structures
- 3 Mapping Reads
- 4 Perspectives

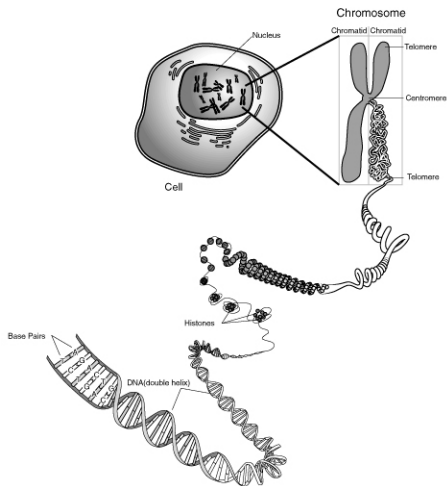
A new era

We can predict that from 10 to 15 years personal DNA sequencing (genome, exome, transcriptome) will be as usual as blood test

Eukaryotic Cell

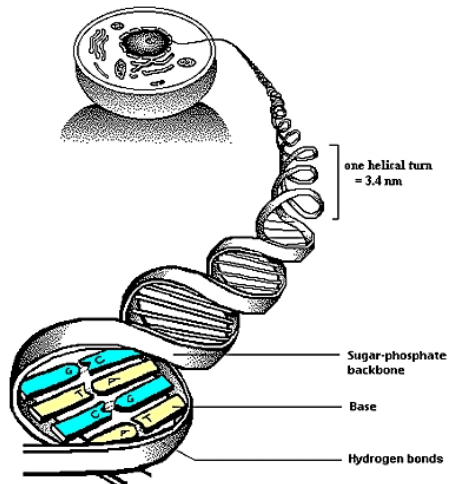


Chromosomes

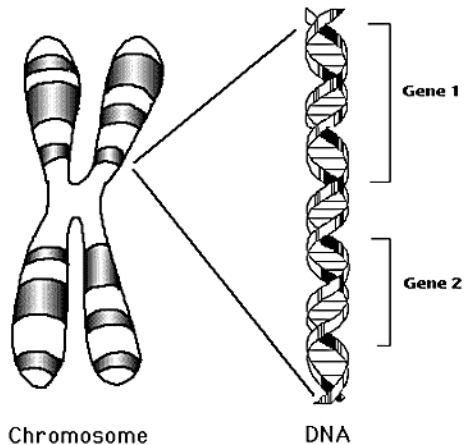


DNA

THE STRUCTURE OF DNA

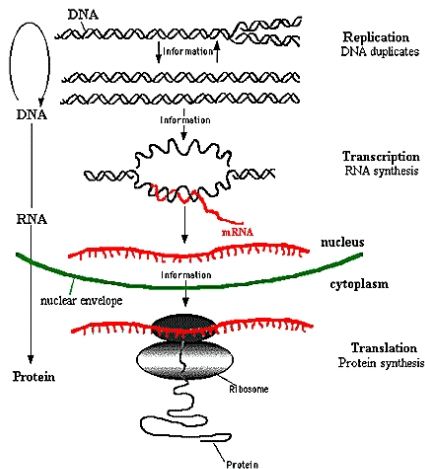


Genes



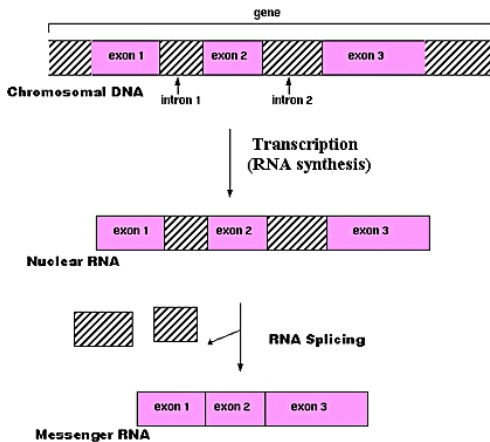
Genes

Synthesis



The Central Dogma of Molecular Biology

RNA Synthesis



RNA synthesis and processing

Size of the Genomes (Mbp)

<i>Escherichia coli</i> (bacteria)	4.6
<i>Saccharomyces cerevisiae</i> (yeast)	13
<i>Caenorhabditis elegans</i> (worm)	100
<i>Arabidopsis thaliana</i> (plant)	125
<i>Drosophila melanogaster</i> (fruit fly)	180
Rice	400
<i>Homo sapiens</i>	3,300
Fern	160,000
<i>Amoeba dubia</i>	670,000

Sequencing techniques

Old Sanger

Long & expensive

Next Generation Sequencing or High Troughput Sequencing techniques

Fast & cheap

Main technologies include: Roche/454, Illumina, SOLiD, Helicos,

...

Main Technologies

Roche/454 - Genome Sequencer FLX

- majority of 500bp reads
- around 1,000,000 reads/run thus 500Mbp/run
- 8h/run



Main Technologies

Illumina/Solexa - Genome Analyzer

- around 100bp reads
- around 240M reads/run
thus 240Gbp/run
- 10 days/run



Main Technologies

SOLiD 4hq System

- 2-base color encoding

	A	C	G	T
A	●	●	●	●
C	●	●	●	●
G	●	●	●	●
T	●	●	●	●
AA	CC	GG	TT	
AC	CA	GT	TG	
AG	CT	GA	TC	
AT	CG	GC	TA	

- around 75bp reads
- around 4G reads/run thus 300Gbp/run
- 14 days/run



NGS

Production of millions of short sequences (around 100 bases) called *tags* or *reads*

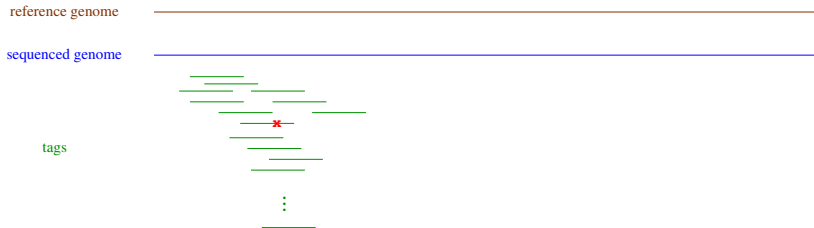
Tags

sequenced genome

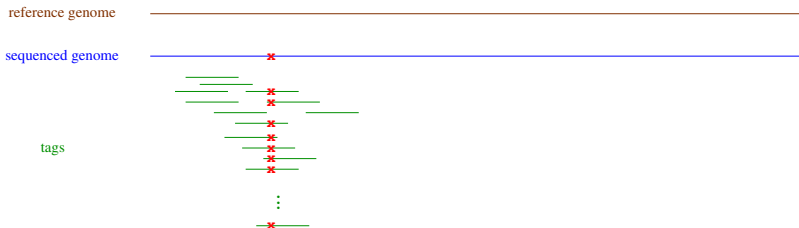
tags



Sequencing Errors



SNP (Single Nucleotide Polymorphism)



Different Projects

Different types

- de novo versus resequencing
- whole genome
- transcriptome (RNA-seq)
- exome
- ChIP-seq
- miRNA
- ...

Different Projects

Ongoing large projects

- www.1000genomes.org: study of human genetic variations
- www.1001genomes.org: study of *Arabidopsis thaliana* genetic variations
- Beijing Genomics Institute will purchase 128 Illumina HiSeq 2000 \Rightarrow 3Tb/day
- metagenomics
- ...

Costs

Less than 10 years ago: the first draft of the Human Genome

A lot of efforts, people, machines, money, ...

Advertisement on the Web (August 2010)

Whole genome	12,500 euros
Exome	3,900 euros
Transcriptome	1,100 euros
miRNA	950 euros

Costs

Less than 10 years ago: the first draft of the Human Genome

A lot of efforts, people, machines, money, ...

Advertisement on the Web (August 2010)

Whole genome	12,500 euros
Exome	3,900 euros
Transcriptome	1,100 euros
miRNA	950 euros

Scaling Problem

Strong need for time-space efficient data structures for manipulating the reads and the genomes

One Basic Task

Pattern matching

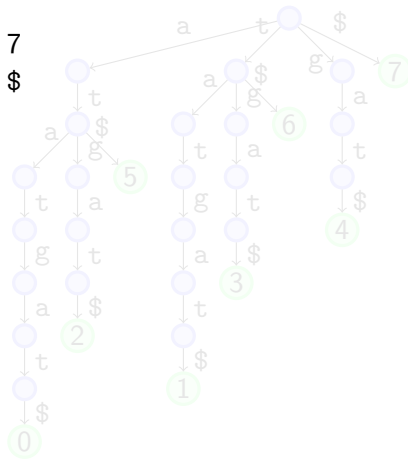
Mapping reads on the genome

Outline

- 1 Molecular Biology
- 2 Indexing Structures**
- 3 Mapping Reads
- 4 Perspectives

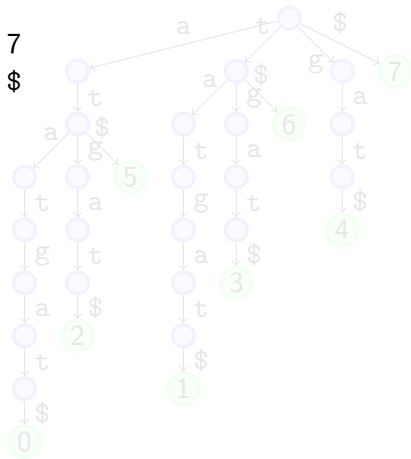
Non Compact Suffix Tree or Suffix Trie

	0	1	2	3	4	5	6	7	
$y =$	a	t	a	t	g	a	t	\$	
0	atatgat\$								
1	tatgat\$								
2	atgat\$								
3	tgat\$								
4	gat\$								
5	at\$								
6	t\$								
7	\$								



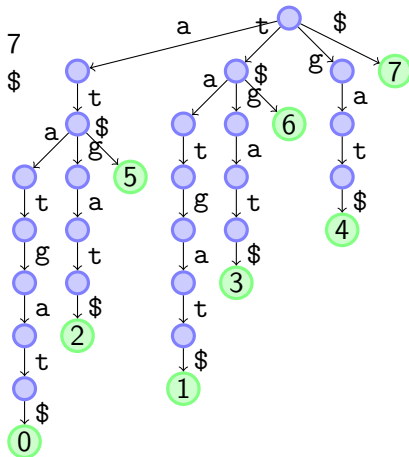
Non Compact Suffix Tree or Suffix Trie

	0	1	2	3	4	5	6	7
<i>y</i> =	a	t	a	t	g	a	t	\$
0	atatgat\$							
1	tatgat\$							
2	atgat\$							
3	tgat\$							
4	gat\$							
5	at\$							
6	t\$							
7	\$							

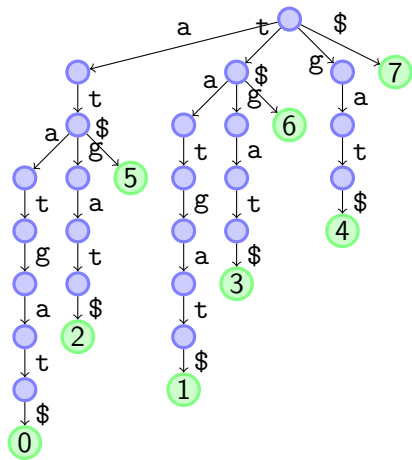


Non Compact Suffix Tree or Suffix Trie

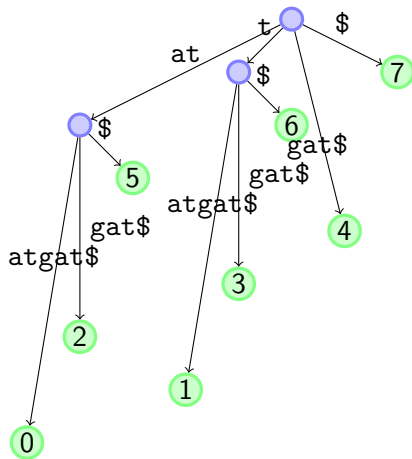
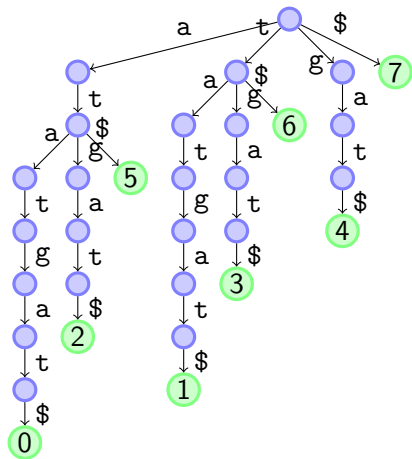
	0	1	2	3	4	5	6	7
$y =$	a	t	a	t	g	a	t	\$
0	atatgat\$							
1	tatgat\$							
2	atgat\$							
3	tgat\$							
4	gat\$							
5	at\$							
6	t\$							
7	\$							



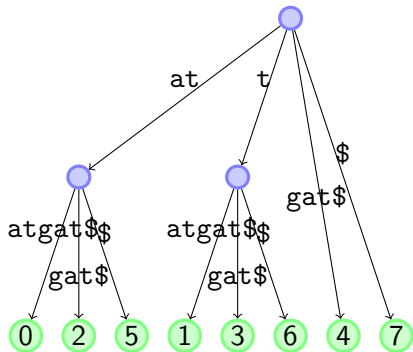
(Compact) Suffix Tree



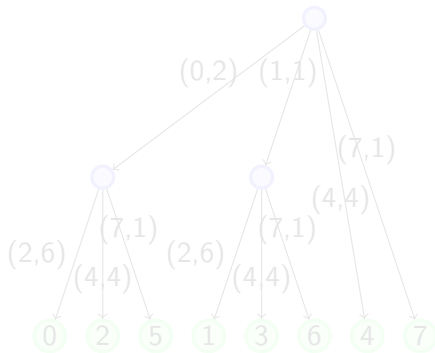
(Compact) Suffix Tree



(Compact) Suffix Tree



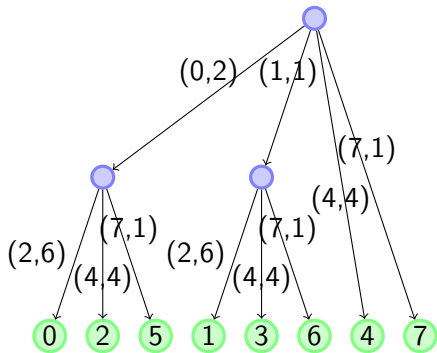
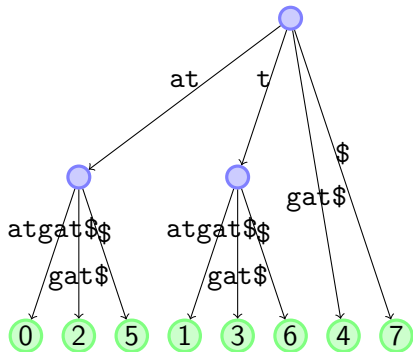
0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$



[Weiner'73, McCreight'76, Ukkonen'92, Farach'97]

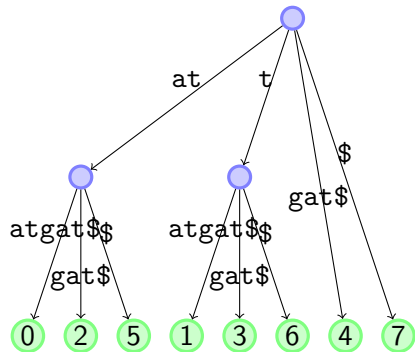
(Compact) Suffix Tree

0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$

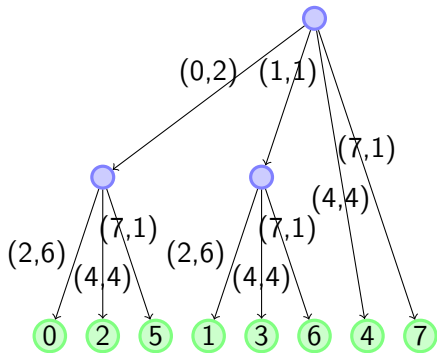


[Weiner'73, McCreight'76, Ukkonen'92, Farach'97]

(Compact) Suffix Tree



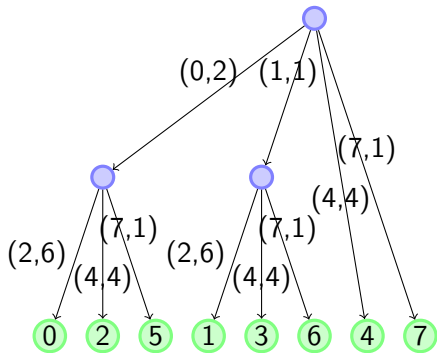
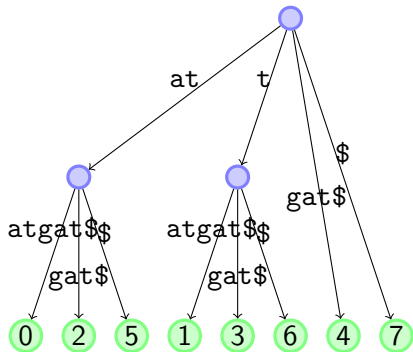
0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$



[Weiner'73, McCreight'76, Ukkonen'92, Farach'97]

(Compact) Suffix Tree

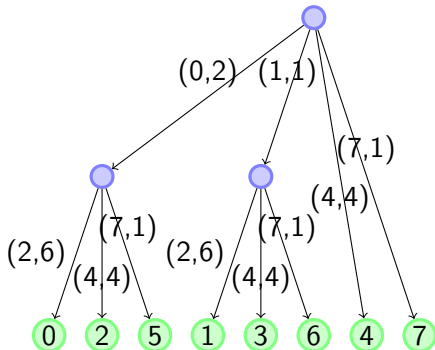
0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$



[Weiner'73, McCreight'76, Ukkonen'92, Farach'97]

(Compact) Suffix Tree

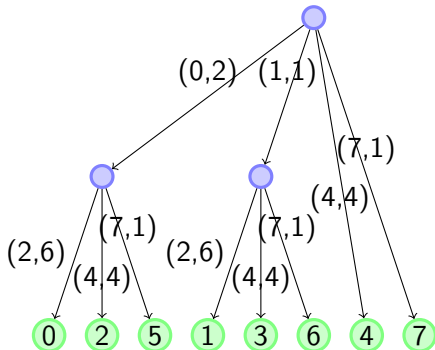
0 1 2 3 4 5 6 7
 a t a t g a t \$



- ta is a factor of y
- tt is not
- at occurs 3 times at positions 0, 2 and 5
- t occurs 3 times at positions 1, 3 and 6

(Compact) Suffix Tree

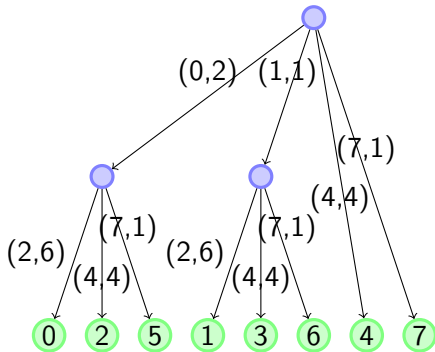
0 1 2 3 4 5 6 7
 a t a t g a t \$



- ta is a factor of y
- tt is not
- at occurs 3 times at positions 0, 2 and 5
- t occurs 3 times at positions 1, 3 and 6

(Compact) Suffix Tree

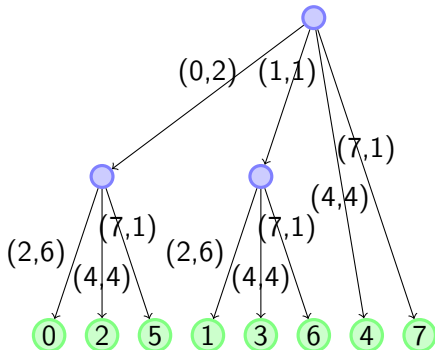
0 1 2 3 4 5 6 7
 a t a t g a t \$



- `ta` is a factor of y
- `tt` is not
- `at` occurs 3 times at positions 0, 2 and 5
- `t` occurs 3 times at positions 1, 3 and 6

(Compact) Suffix Tree

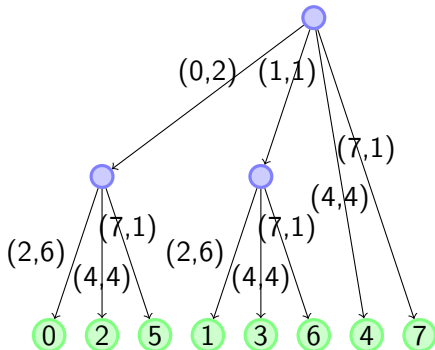
0 1 2 3 4 5 6 7
 a t a t g a t \$



- ta is a factor of y
- tt is not
- at occurs 3 times at positions 0, 2 and 5
- t occurs 3 times at positions 1, 3 and 6

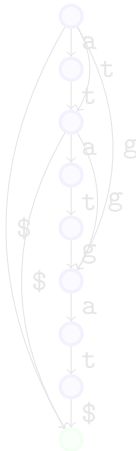
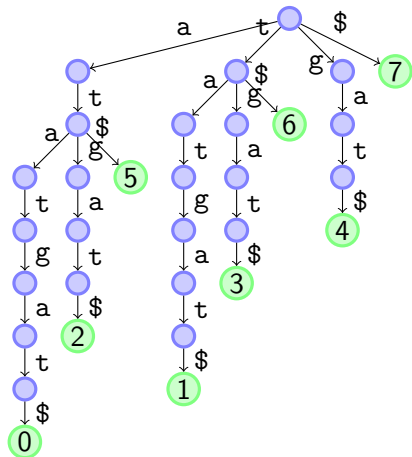
(Compact) Suffix Tree

0 1 2 3 4 5 6 7
a t a t g a t \$



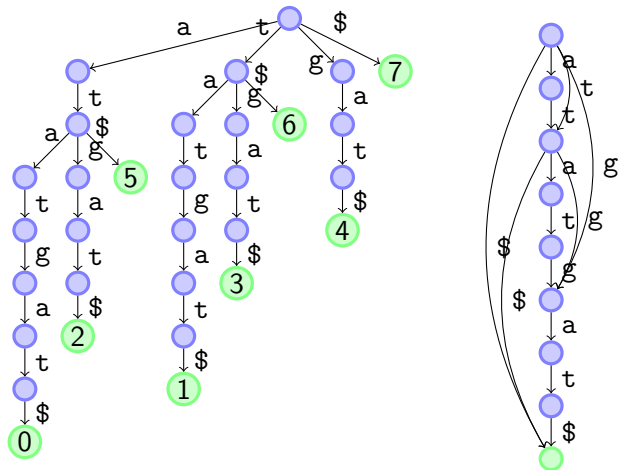
- ta is a factor of y
- tt is not
- at occurs 3 times at positions 0, 2 and 5
- t occurs 3 times at positions 1, 3 and 6

Suffix Automaton or DAWG



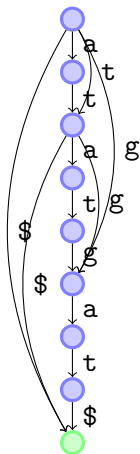
[Blumer *et al*'85, Crochemore'86]

Suffix Automaton or DAWG

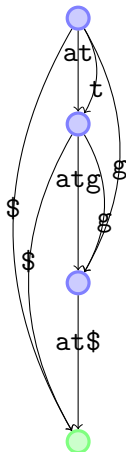
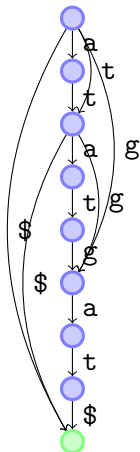


[Blumer *et al*'85, Crochemore'86]

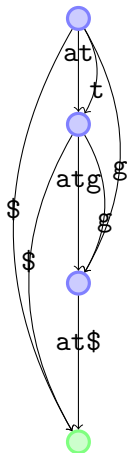
Compact Suffix Automaton



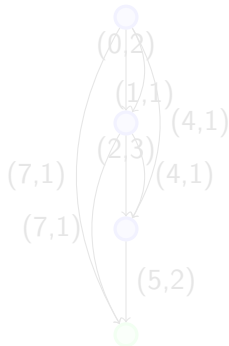
Compact Suffix Automaton



Compact Suffix Automaton

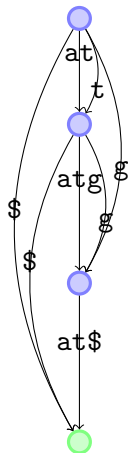


0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$

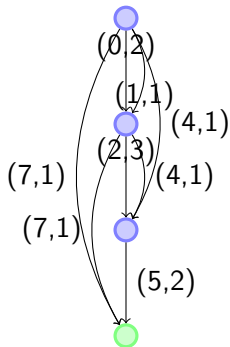


[Crochemore & Verin'97]

Compact Suffix Automaton

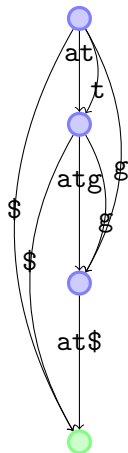


0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$



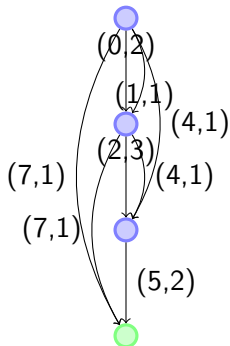
[Crochemore & Verin'97]

Compact Suffix Automaton

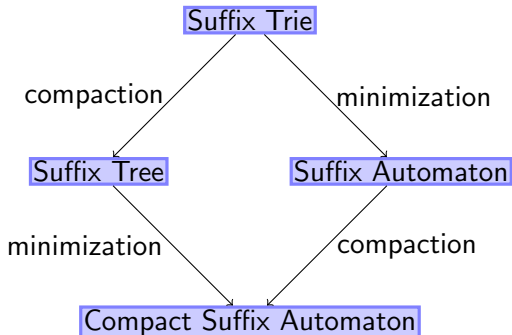


[Crochemore & Verin'97]

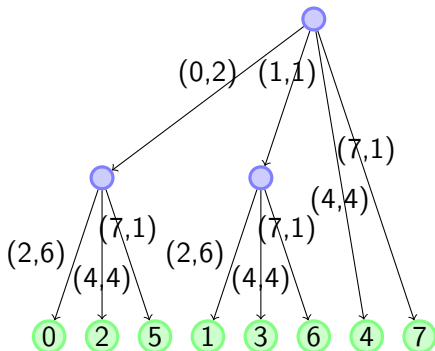
0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$



Classical diagram



Non Compact Suffix Vector



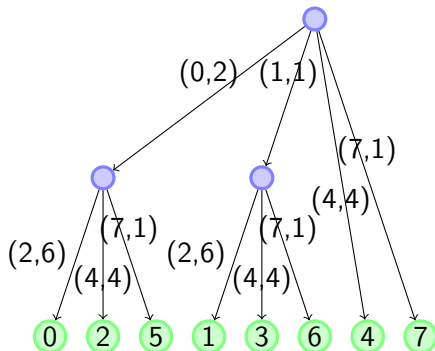
root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

[Monostori et al'01]

Non Compact Suffix Vector



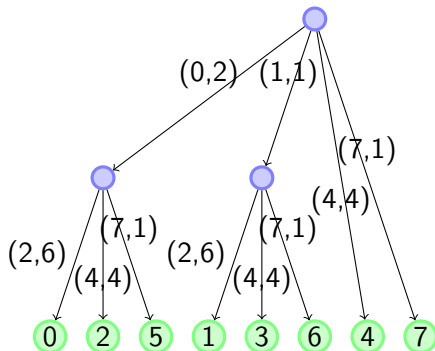
root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

[Monostori et al'01]

Non Compact Suffix Vector



root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

[Monostori *et al*'01]

Compact Suffix Vector

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$

0 1 2 3 4 5 6 7

|

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$

0 1 2 3 4 5 6 7

|

2

2	6	(4,4),(7,1)
---	---	-------------

[Monostori *et al*'01, Prieur & L'08]

Compact Suffix Vector

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

|

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

|

2

2	6	(4,4),(7,1)
---	---	-------------

[Monostori *et al*'01, Prieur & L'08]

Compact Suffix Vector

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

|

2	6	(4,4),(7,1)
1	6	(4,4),(7,1)

root: (0,2),(1,1),(4,4),(7,1)

a t a t g a t \$
0 1 2 3 4 5 6 7

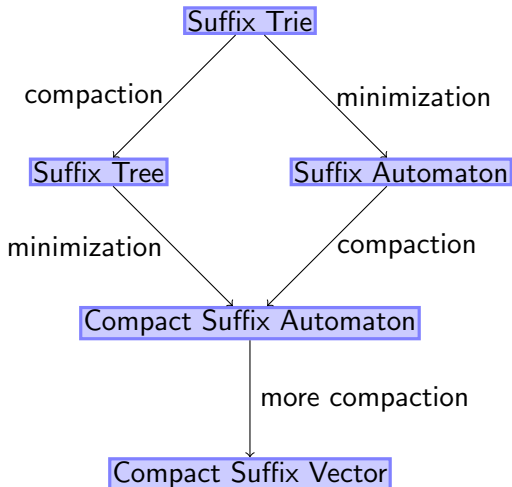
|

2

2	6	(4,4),(7,1)
---	---	-------------

[Monostori *et al*'01,Prieur & L'08]

Augmented classical diagram



Complexities

Algorithms for building these structures are

- online
- linear in time
- linear in space

Practical considerations

Bytes per symbol

	suffix trees	compact suffix vectors
<i>Ecoli</i> (Canterbury corpus)	12.56	12.51

Human genome 3.3Gbp

41.25 Gb

Practical considerations

Bytes per symbol

	suffix trees	compact suffix vectors
<i>Ecoli</i> (Canterbury corpus)	12.56	12.51

Human genome 3.3Gbp

41.25 Gb

Suffix Array

$y =$ 0 1 2 3 4 5 6 7
 a t a t g a t \$

		SA	sort of the suffixes
0	atatgat\$	5	at\$
1	tatgat\$	0	atatgat\$
2	atgat\$	2	atgat\$
3	tgat\$	4	gat\$
4	gat\$	6	t\$
5	at\$	1	tatgat\$
6	t\$	3	tgat\$
7	\$	7	\$

$n \log n$ bits

[Manber & Myers'90, Kärkkäinen & Saunders'03, Kim et al'03, Ko & Aluru'03]

Suffix Array

$y =$ 0 1 2 3 4 5 6 7
 a t a t g a t \$

		SA	sort of the suffixes
0	atatgat\$	5	at\$
1	tatgat\$	0	atatgat\$
2	atgat\$	2	atgat\$
3	tgat\$	4	gat\$
4	gat\$	6	t\$
5	at\$	1	tatgat\$
6	t\$	3	tgat\$
7	\$	7	\$

$n \log n$ bits

[Manber & Myers'90, Kärkkäinen & Saunders'03, Kim et al'03, Ko & Aluru'03]

Suffix Array

$y =$ 0 1 2 3 4 5 6 7
 a t a t g a t \$

		SA	sort of the suffixes
0	atatgat\$	5	at\$
1	tatgat\$	0	atatgat\$
2	atgat\$	2	atgat\$
3	tgat\$	4	gat\$
4	gat\$	6	t\$
5	at\$	1	tatgat\$
6	t\$	3	tgat\$
7	\$	7	\$

$n \log n$ bits

[Manber & Myers'90, Kärkkäinen & Saunders'03, Kim *et al*'03, Ko & Aluru'03]

Suffix Array

$y =$ 0 1 2 3 4 5 6 7
 a t a t g a t \$

		SA	sort of the suffixes
0	atatgat\$	5	at\$
1	tatgat\$	0	atatgat\$
2	atgat\$	2	atgat\$
3	tgat\$	4	gat\$
4	gat\$	6	t\$
5	at\$	1	tatgat\$
6	t\$	3	tgat\$
7	\$	7	\$

$n \log n$ bits

[Manber & Myers'90, Kärkkäinen & Saunders'03, Kim et al'03, Ko & Aluru'03]

Suffix Array

$y =$ 0 1 2 3 4 5 6 7
 a t a t g a t \$

		SA	sort of the suffixes
0	atatgat\$	5	at\$
1	tatgat\$	0	atatgat\$
2	atgat\$	2	atgat\$
3	tgat\$	4	gat\$
4	gat\$	6	t\$
5	at\$	1	tatgat\$
6	t\$	3	tgat\$
7	\$	7	\$

$n \log n$ bits

[Manber & Myers'90, Kärkkäinen & Saunders'03, Kim *et al*'03, Ko & Aluru'03]

Longest Common Prefix

$y =$

0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$

SA	sort of the suffixes	SA	LCP	sort of the suffixes
0	atatgat\$	0		atatgat\$
2	atgat\$	2	2	atgat\$
5	at\$	5	2	at\$
4	gat\$	4	0	gat\$
1	tatgat\$	1	0	tatgat\$
3	tgat\$	3	1	tgat\$
6	t\$	6	1	t\$
7	\$	7	0	\$

[Kasai *et al*'01]

Longest Common Prefix

$y =$

0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$

SA	sort of the suffixes	SA	LCP	sort of the suffixes
0	atatgat\$	0		atatgat\$
2	atgat\$	2	2	atgat\$
5	at\$	5	2	at\$
4	gat\$	4	0	gat\$
1	tatgat\$	1	0	tatgat\$
3	tgat\$	3	1	tgat\$
6	t\$	6	1	t\$
7	\$	7	0	\$

[Kasai *et al*'01]

Longest Common Prefix

$y =$

0	1	2	3	4	5	6	7
a	t	a	t	g	a	t	\$

SA	sort of the suffixes	SA	LCP	sort of the suffixes
0	atatgat\$	0		atatgat\$
2	atgat\$	2	2	atgat\$
5	at\$	5	2	at\$
4	gat\$	4	0	gat\$
1	tatgat\$	1	0	tatgat\$
3	tgat\$	3	1	tgat\$
6	t\$	6	1	t\$
7	\$	7	0	\$

[Kasai *et al*'01]

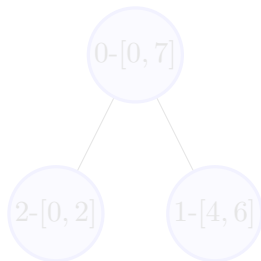
LCP-interval

An interval $[i, j]$, $0 \leq i < j \leq n$, is an **lcp-interval** of **lcp value** ℓ (denoted by ℓ - $[i, j]$) if

- 1 $LCP[i] < \ell$,
- 2 $LCP[k] \geq \ell$ for $i + 1 \leq k \leq j$,
- 3 $LCP[k] = \ell$ for at least one k with $i + 1 \leq k \leq j$,
- 4 $LCP[j + 1] < \ell$.

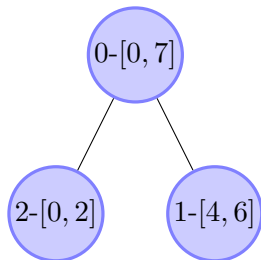
LCP tree

y	=	0	1	2	3	4	5	6	7
		a	t	a	t	g	a	t	\$
	SA	LCP	sort of the suffixes						
0	0		atatgat\$						
1	2	2	atgat\$						
2	5	2	at\$						
3	4	0	gat\$						
4	1	0	tatgat\$						
5	3	1	tgat\$						
6	6	1	t\$						
7	7	0	\$						

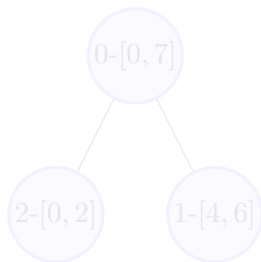
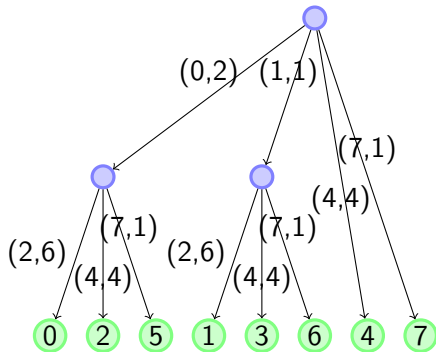


LCP tree

		0	1	2	3	4	5	6	7
y	=	a	t	a	t	g	a	t	\$
	SA	sort of the suffixes							
0	0	atatgat\$							
1	2	2	atgat\$						
2	5	2	at\$						
3	4	0	gat\$						
4	1	0	tatgat\$						
5	3	1	tгат\$						
6	6	1	t\$						
7	7	0	\$						

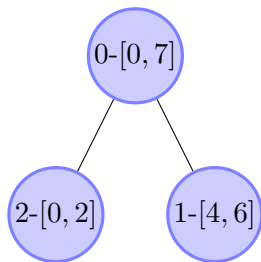
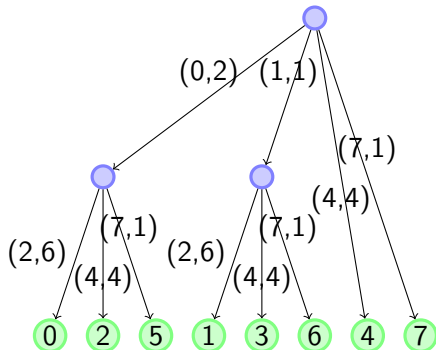


LCP tree



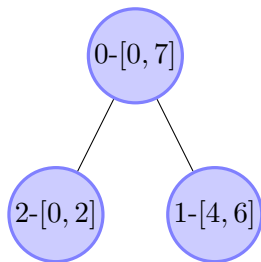
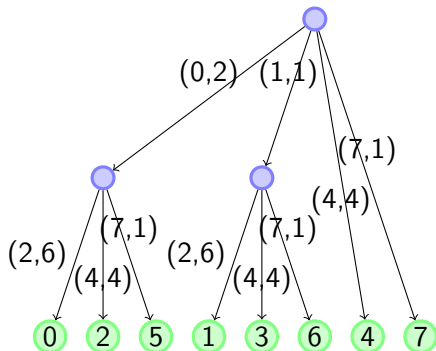
[Abouelhoda *et al*'04]

LCP tree



[Abouelhoda *et al*'04]

LCP tree



[Abouelhoda *et al*'04]

Compact Suffix Array

SA	sort of the suffixes
0	atatgat\$
2	atgat\$
5	at\$
4	gat\$
1	tatgat\$
3	tgat\$
6	t\$
7	\$

[Mäkinen'02]

Compact Suffix Array

SA	sort of the suffixes
0	atatgat\$
2	atgat\$
5	at\$
4	gat\$
1	tatgat\$
3	tgat\$
6	t\$
7	\$

[Mäkinen'02]

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀		SA ₁	B ₁	rank ₁	Ψ ₁
0	0	0	0	4	0	2	0	0	3
1	2	0	0	5	1	0	0	0	2
2	5	1	1	2	2	1	1	1	2
3	4	0	1	2	3	3	1	2	3
4	1	1	2	4					
5	3	1	3	5		SA ₂	B ₂	rank ₂	Ψ ₂
6	6	0	3	7	0	0	0	0	1
7	7	1	4	7	1	1	1	1	1

 $B_0[i] = 1$ if $SA_0[i]$ is odd

 $rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$
 $\Psi_0[i] =$

$$\begin{cases} i & \text{if } SA_0[i] \text{ is odd} \\ j & \text{st } SA_0[j] = SA_0[i] + 1 \text{ otherwise} \end{cases}$$
 $SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$
 $SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$
 $SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$
 $SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$
 $SA_0[4] = 2 \times 0 + B_0[4]$
 $SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀		SA ₁	B ₁	rank ₁	Ψ ₁
0	0	0	0	4	0	2	0	0	3
1	2	0	0	5	1	0	0	0	2
2	5	1	1	2	2	1	1	1	2
3	4	0	1	2	3	3	1	2	3
4	1	1	2	4					
5	3	1	3	5		SA ₂	B ₂	rank ₂	Ψ ₂
6	6	0	3	7	0	0	0	0	1
7	7	1	4	7	1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀		SA ₁	B ₁	rank ₁	Ψ ₁
0	0	0	0	4	0	2	0	0	3
1	2	0	0	5	1	0	0	0	2
2	5	1	1	2	2	1	1	1	2
3	4	0	1	2	3	3	1	2	3
4	1	1	2	4					
5	3	1	3	5		SA ₂	B ₂	rank ₂	Ψ ₂
6	6	0	3	7	0	0	0	0	1
7	7	1	4	7	1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

	SA ₀	B ₀	rank ₀	Ψ ₀
0	0	0	0	4
1	2	0	0	5
2	5	1	1	2
3	4	0	1	2
4	1	1	2	4
5	3	1	3	5
6	6	0	3	7
7	7	1	4	7

	SA ₁	B ₁	rank ₁	Ψ ₁
0	2	0	0	3
1	0	0	0	2
2	1	1	1	2
3	3	1	2	3

	SA ₂	B ₂	rank ₂	Ψ ₂
0	0	0	0	1
1	1	1	1	1

$B_0[i] = 1$ if SA₀[i] is odd

$rank_0[i] = \sum_{\ell=0}^i B_0[\ell]$

$\Psi_0[i] =$

$\begin{cases} i & \text{if SA}_0[i] \text{ is odd} \\ j & \text{st SA}_0[j] = \text{SA}_0[i] + 1 \text{ otherwise} \end{cases}$

$SA_k[i] = 2 \times SA_{k+1}[rank_k(\Psi_k(i)) - 1] + B_k[i]$

$SA_0[4] = 2 \times SA_1[rank_0(\Psi_0(4)) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[rank_0(4) - 1] + B_0[4]$

$SA_0[4] = 2 \times SA_1[2 - 1] + B_0[4]$

$SA_0[4] = 2 \times 0 + B_0[4]$

$SA_0[4] = 1$

Compressed Suffix Array

To be stored

- last SA_ℓ
- every B_k , $rank_k$ and Ψ_k for $0 \leq k \leq \ell - 1$ in a compressed form

[Grossi & Vitter'00]

Compressed Suffix Array

To be stored

- last SA_ℓ
- every B_k , $rank_k$ and Ψ_k for $0 \leq k \leq \ell - 1$ in a compressed form

[Grossi & Vitter'00]

Burrows-Wheeler Transform

$y =$ 0 1 2 3 4 5 6 7
a t a t g a t \$

SA sort of the suffixes

0 atatgat\$

2 atgat\$

5 at\$

4 gat\$

1 tatgat\$

3 t\$

6 t\$

7 \$

sort of the conjugates

F *L*

a tatgat \$

a t\$at\$ t

a t\$atat g

g at\$aata t

t atgat\$ a

t gat\$at a

t \$atatg a

\$ atatga t

BWT = \$tgtaa

$n \log \sigma$ bits

Burrows-Wheeler Transform

$y =$ 0 1 2 3 4 5 6 7
a t a t g a t \$

SA sort of the suffixes

0 atatgat\$

2 atgat\$

5 at\$

4 gat\$

1 tatgat\$

3 tgat\$

6 t\$

7 \$

sort of the conjugates

F L

a tatgat \$

a tgat\$a t

a t\$atat g

g at\$aata t

t atgat\$ a

t gat\$at a

t \$atatg a

\$ atatga t

BWT = \$tgtaa

$n \log \sigma$ bits

BWT

L

\$

t

g

t

a

a

a

t

BWT

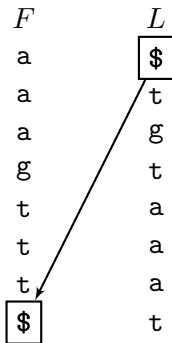
<i>F</i>	<i>L</i>
a	\$
a	t
a	g
g	t
t	a
t	a
t	a
\$	t

BWT

<i>F</i>	<i>L</i>
a	\$
a	t
a	g
g	t
t	a
t	a
t	a
\$	t

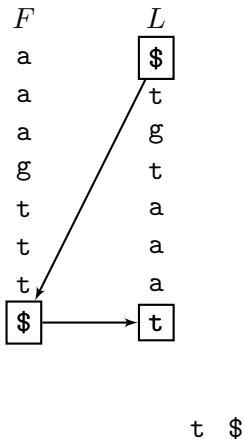
\$

BWT

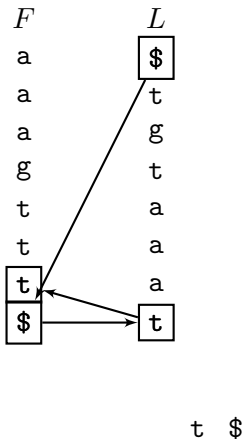


\$

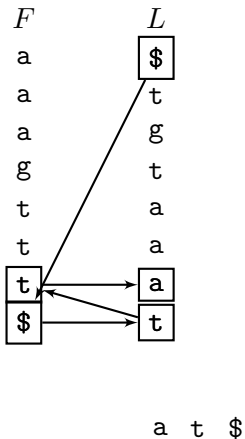
BWT



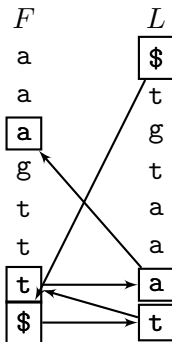
BWT



BWT

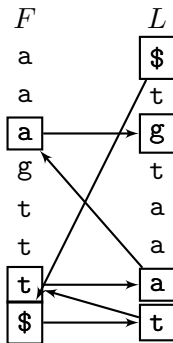


BWT



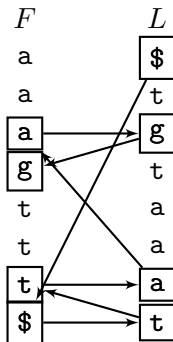
a t \$

BWT



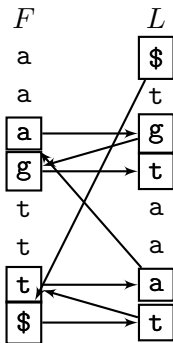
g a t \$

BWT



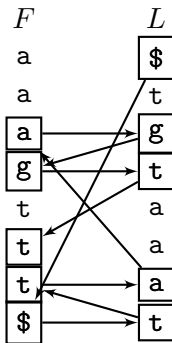
g a t \$

BWT



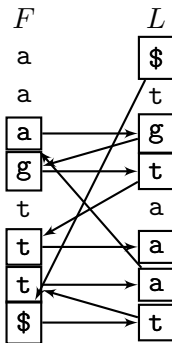
t g a t \$

BWT



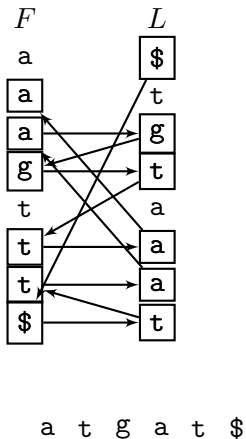
t g a t \$

BWT

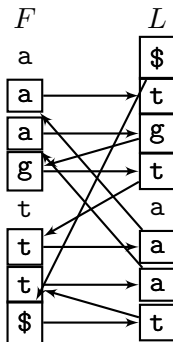


a t g a t \$

BWT

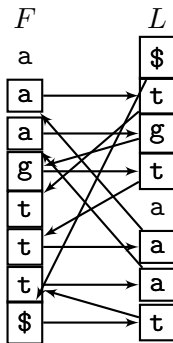


BWT



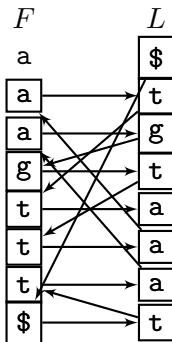
t a t g a t \$

BWT



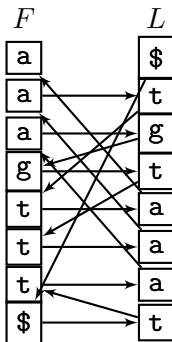
t a t g a t \$

BWT



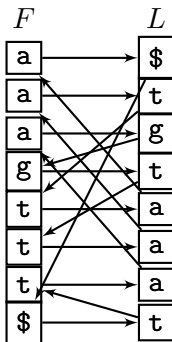
a t a t g a t \$

BWT



a t a t g a t \$

BWT



a t a t g a t \$

LF Function

atatgat\$

	a	g	t	\$
<i>C</i>	0	3	4	7

$$LF[i] = C[L[i]] + \text{rank}_{L[i]}(L, i)$$

LF Function

atatgat\$

	a	g	t	\$
<i>C</i>	0	3	4	7

$$LF[i] = C[L[i]] + \text{rank}_{L[i]}(L, i)$$

	<i>F</i>	<i>L</i>
0	a	\$
1	a	t
2	a	g
3	g	t
4	t	a
5	t	a
6	t	a
7	\$	t

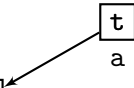
LF Function

atatgat\$

	a	g	t	\$
<i>C</i>	0	3	4	7

$$LF[i] = C[L[i]] + \text{rank}_{L[i]}(L, i)$$

	<i>F</i>	<i>L</i>
0	a	\$
1	a	t
2	a	g
3	g	t
4	t	a
5	t	a
6	t	a
7	\$	t



FM-index

Self-index

Compressed suffix array + BWT

Human Genome

Less than 2Gb

[Ferragina & Manzini'00]

FM-index

Self-index

Compressed suffix array + BWT

Human Genome

Less than 2Gb

[Ferragina & Manzini'00]

FM-index

Self-index

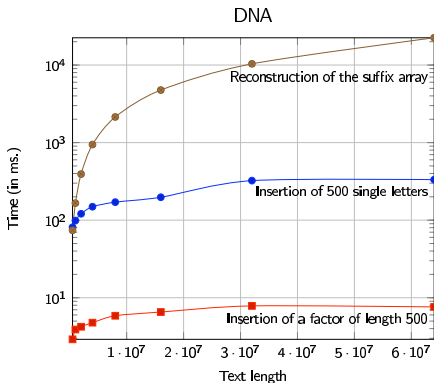
Compressed suffix array + BWT

Human Genome

Less than 2Gb

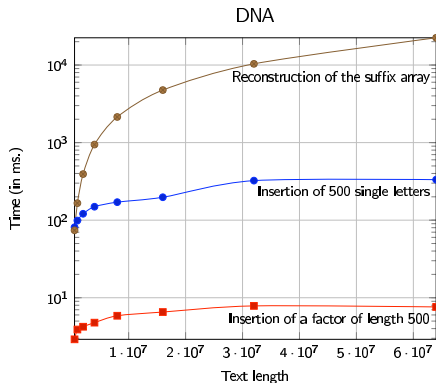
[Ferragina & Manzini'00]

Dynamic structures



[Salson et al'09 and '10]

Dynamic structures



[Salson *et al*'09 and '10]

Indexing structures

Other structures include

- PATRICIA tries
- Factor oracles
- Bonsai trees
- Suffix sequoias
- Compressed suffix trees
- LZ-indexes
- Wavelet trees
- Contracted suffix trees
- ...

Most of the times they can be generalized to several strings

Outline

- 1 Molecular Biology
- 2 Indexing Structures
- 3 Mapping Reads**
- 4 Perspectives

Mapping Reads

Indexing Structures

- MUMmer
- OASIS
- Vmatch
- Segemehl
- Bowtie
- BWA
- SOAP2
- BWT-SW
- BWA-SW
- ...

Bowtie

Exact search for gat in atatgat

L
\$
t
g
t
a
a
a
t

[Ferragina & Manzini'00]

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	\$
a	t
a	g
g	t
t	a
t	a
\$	t

[Ferragina & Manzini'01]

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	tgat\$a t
a	t\$atat g
g	at\$ata t
t	atgat\$ a
t	gat\$at a
t	\$atatg a
\$	atatga t

[Ferragina & Manzini'01]

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	tgat\$a t
a	t\$atat g
g	at\$aata t
t	atgat\$ a
[Ferragina & Manzini'0]	t gat\$at a
t	\$atatg a
\$	atatga t

Bowtie

Exact search for gat in atatgat

<i>F</i>		<i>L</i>	
a	tatgat	\$	
a	tgat\$a	t	
a	t\$atat	g	
g	at\$aata	t	
t	atgat\$	a	
[Ferragina & Manzini'0]	t	gat\$at	a
t	\$atatg	a	
\$	atatga	t	

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	tgat\$a t
a	t\$atat g
g	at\$aata t
t	atgat\$ a
t	gat\$at a
t	\$atatg a
\$	atatga t

[Ferragina & Manzini'01]

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	t gat\$ a t
a	t\$ at a t g
g	a t\$ a t a t
t	a t g a t\$ a
[Ferragina & Manzini'01]	g a t\$ a t a
t	\$ a t a t g a
\$	a t a t g a t

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	tgat\$a t
a	t\$atat g
g	at \$ata t
t	atgat\$ a
[Ferragina & Manzini'01]	gat\$at a
t	\$atatg a
\$	atatga t

Bowtie

Exact search for gat in atatgat

<i>F</i>	<i>L</i>
a	tatgat \$
a	tgat\$a t
a	t\$atat g
g	at \$ata t
t	atgat\$ a
[Ferragina & Manzini'00]	gat\$at a
t	\$atatg a
\$	atatga t

Bowtie

Approximate search for tta in atatgat: backtracking

L
\$
t
g
t
a
a
a
t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>	<i>L</i>
a	\$
a	t
a	g
g	t
t	a
t	a
t	a
\$	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$aata	t
t	atgat\$a	a
t	gat\$a	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$ata	t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$ata	t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$aata	t
t	atgat	\$ a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$ata	t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	a t\$a	t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at \$ata	t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g	at\$ata	t
t	atgat\$	a
	t gat\$a	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Bowtie

Approximate search for tta in atatgat: backtracking

<i>F</i>		<i>L</i>
a	tatgat	\$
a	tgat\$a	t
a	t\$atat	g
g		t
t	atgat\$	a
t	gat\$at	a
t	\$atatg	a
\$	atatga	t

[Langmead *et al*'09]

Mapping Reads

Hash Tables

- ELAND, ELANDv2
- SOAP
- SeqMap
- MAQ
- BFAST
- SOCS
- RMAP
- ZOOM
- CloudBurst
- PerM
- GNUMAP
- SHRiMP
- RazerS
- Novoalign
- ...

Hash tables

- store the positions of every factors (called q -grams or *seeds*) of a given length
- spaced seeds (partial words, strings with holes, string with don't cares ...): more sensitive
- if seeds of different lengths or different shapes: several hash tables

Approximate matching

Search for abcdefgh with 2 substitutions: at least of factor of length 2 will match exactly:

	a**defgh	abc**fgh
	a*c*efgh	abc*e*gh
**cdefgh	a*cd*fgh	abc*ef*h
*b*defgh	a*cde*gh	abc*efg*
*bc*efgh	a*cdef*h	abcd**gh
*bcd*fgh	a*cdefg*	abcd*f*h
*bcde*gh	ab**efgh	abcd*fg*
*bcdef*h	ab*d*fgh	abcde**h
*bcdefgh	ab*de*gh	abcde*g*
	ab*def*h	abcdef**
	ab*defg*	

Approximate matching

Search for x of length m with k substitutions: at least one factor of length $\lfloor m/k \rfloor$ will match exactly

Use of spaced seeds

Weight of a seed: number of solid (non don't care) symbols

Goal

Design spaced seeds with maximal weight and minimize the number of hash tables

Approximate matching

Search for x of length m with k substitutions: at least one factor of length $\lfloor m/k \rfloor$ will match exactly

Use of spaced seeds

Weight of a seed: number of solid (non don't care) symbols

Goal

Design spaced seeds with maximal weight and minimize the number of hash tables

Approximate matching

Search for x of length m with k substitutions: at least one factor of length $\lfloor m/k \rfloor$ will match exactly

Use of spaced seeds

Weight of a seed: number of solid (non don't care) symbols

Goal

Design spaced seeds with maximal weight and minimize the number of hash tables

Space seeds

Approximate search with 2 substitutions

c a c g t a t g	g a g c t a a t	c g g t c c t g	t a g g a c g t
-----------------	-----------------	-----------------	-----------------

c a c g t a t g	g a g c t a a t	*****	*****
-----------------	-----------------	-------	-------

*****	g a g c t a a t	c g g t c c t g	*****
-------	-----------------	-----------------	-------

*****	*****	c g g t c c t g	t a g g a c g t
-------	-------	-----------------	-----------------

c a c g t a t g	*****	c g g t c c t g	*****
-----------------	-------	-----------------	-------

*****	g a g c t a a t	*****	t a g g a c g t
-------	-----------------	-------	-----------------

c a c g t a t g	*****	*****	t a g g a c g t
-----------------	-------	-------	-----------------

Seeds used by ELAND, SOAP and MAQ

PerM (Periodic Seed Mapping)

0	1	2	3	4	5	6	
#	#	#	-	#	-	-	(3,5),(3,6),(5,6)
-	#	#	#	-	#	-	(0,4),(0,6),(4,6)
-	-	#	#	#	-	#	(0,1),(0,5),(1,5)
#	-	-	#	#	#	-	(1,2),(1,6),(2,6)
-	#	-	-	#	#	#	(0,2),(0,3),(2,3)
#	-	#	-	-	#	#	(1,3),(1,4),(3,4)
#	#	-	#	-	-	#	(2,4),(2,5),(4,5)

[Chen et al'09]

PerM (Periodic Seed Mapping)

0	1	2	3	4	5	6	
#	#	#	-	#	-	-	(3,5),(3,6),(5,6)
-	#	#	#	-	#	-	(0,4),(0,6),(4,6)
-	-	#	#	#	-	#	(0,1),(0,5),(1,5)
#	-	-	#	#	#	-	(1,2),(1,6),(2,6)
-	#	-	-	#	#	#	(0,2),(0,3),(2,3)
#	-	#	-	-	#	#	(1,3),(1,4),(3,4)
#	#	-	#	-	-	#	(2,4),(2,5),(4,5)

[Chen *et al*'09]

Outline

- 1 Molecular Biology
- 2 Indexing Structures
- 3 Mapping Reads
- 4 Perspectives**

Perspectives

- develop methods for all materials and all type of experiments
- representation of consensus and variations (SNPs, CNVs, ...)
- parallelize the algorithms
- anticipate on the next technologies (longer reads)
- ...

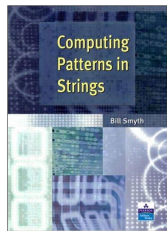
References – Books



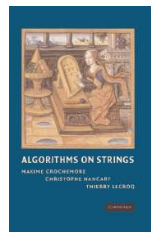
D. Gusfield,
*Algorithms on
Strings, Trees and
Sequences*,
Cambridge
University Press,
1997



D. Adjeroh, T. Bell
& A. Mukherjee,
*The
Burrows-Wheeler
Transform*,
Springer, 2008



W.F. Smyth,
*Computing Pattern
in Strings*,
Addison-Wesley,
2003



M. Crochemore, C.
Hancart & T. L.
*Algorithms on
Strings*,
Cambridge
University Press,
2007

References



M.I. Abouelhoda, S. Kurtz and E. Ohlebusch
Replacing suffix trees with enhanced suffix arrays
Journal of Discrete Algorithms 2(1) (2004) 53–86



A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M. T. Chen and J. Seiferas
The smallest automaton recognizing the subwords of a text
Theoretical Computer Science 40(1) (1985) 31–55



M. Burrows and D.J. Wheeler
A block-sorting lossless data compression algorithm
Technical Report 124, DEC, Palo Alto, California



Y. Chen, T. Souaiaia and T. Chen
PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds
Bioinformatics 25(19) (2009) 2514–2521



M. Crochemore
Transducers and repetitions
Theoretical Computer Science 45(1) (1986) 63–86

References



M. Crochemore and R. V erin

Direct construction of compact directed acyclic word graphs
8th CPM, 116–129, 1997



M. Farach

Optimal Suffix Tree Construction with Large Alphabets,
38th FOCS, 137–143, 1997



P. Ferragina and G. Manzini

Opportunistic data structures with applications
41st FOCS, 390–398, 2000



R. Grossi and J.S. Vitter






Compressed suffix arrays and suffix trees with applications to text indexing and string matching
SIAM Journal of Computing 35(2) (2005) 378–407



J. K arkkainen and P. Saunders

Simple linear work suffix array construction
ICALP, 943–955, 2003

References

-  T. Kasai, G. Lee, H. Arimura, S. Arikawa and K. Park
Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications.
CPM, 181–192, 2001
-  D.K. Kim, J.S. Sim, H. Park and K. Park
Linear-time construction of suffix arrays
CPM, 186–199, 2003
-  P. Ko and S. Aluru
Space efficient linear time construction of suffix arrays
CPM, 200–210, 2003
-  B. Langmead, C. Trapnell, M. Pop and S.L. Salzberg
Ultrafast and memory-efficient alignment of short DNA sequences to the human genome
Genome Biology **10** (2009) R25
-  V. Mäkinen
Compact suffix array – a space-efficient full-text index
Fundamenta informaticae **56**(1-2) (2003) 191–210

References



U. Manber and G. Myers

Suffix arrays: a new method for on-line string searches
SIAM Journal of Computing 22(5) (1993) 935–948



E. M. McCreight

A space-economical suffix tree construction algorithm
Journal of Algorithms 23(2) (1976) 262–272



K. Monostori, A. Zaslavsky and I. Vajk

Suffix Vector: A Space-efficient Suffix Tree Representation
12th International Symposium on Algorithms and Computation, 707–718, 2001



É. Prieur and T. Lecroq

Direct construction of suffix vectors and maximal repeats
Theoretical Computer Science 407(1-3) (2008) 290–301



M. Salson, T. Lecroq, M. Léonard and L. Mouchard

A four-stage algorithm for updating a Burrows-Wheeler Transform
Theoretical Computer Science 410(43) (2009) 4350–4359



M. Salson, T. Lecroq, M. Léonard and L. Mouchard

Dynamic extended suffix array
Journal of Discrete Algorithms 8(2) (2010) 241–257

References



E. Ukkonen

On-line construction of suffix trees

Algorithmica 14(3) (1995) 249–260



P. Weiner

Linear pattern matching algorithm

14th Annual IEEE Symposium on Switching and Automata Theory, 1–11, 1973