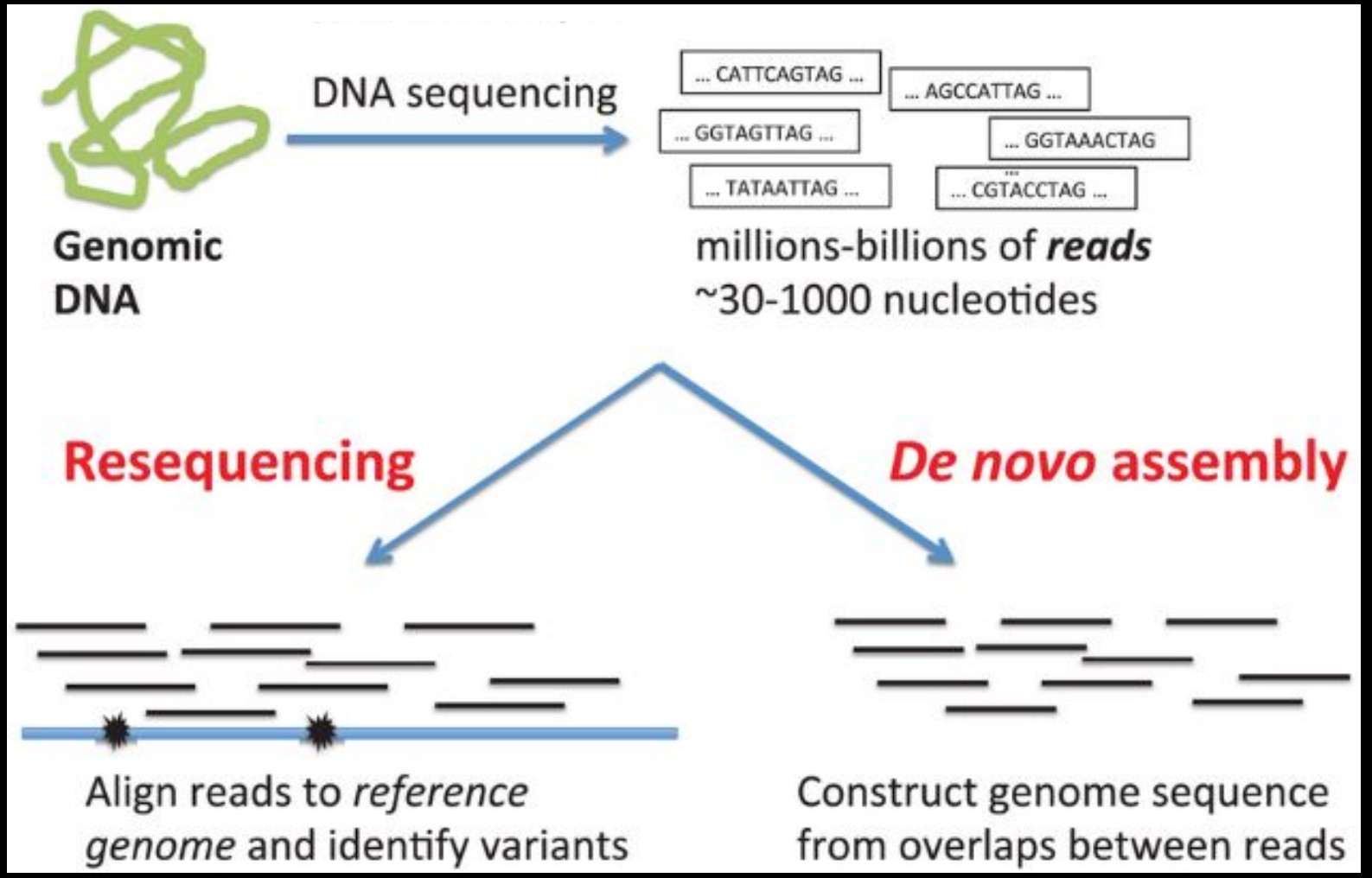
Comparing Highly Similar Sequences: Suffix Trees, Suffix Arrays and the Burrows-Wheeler Transform

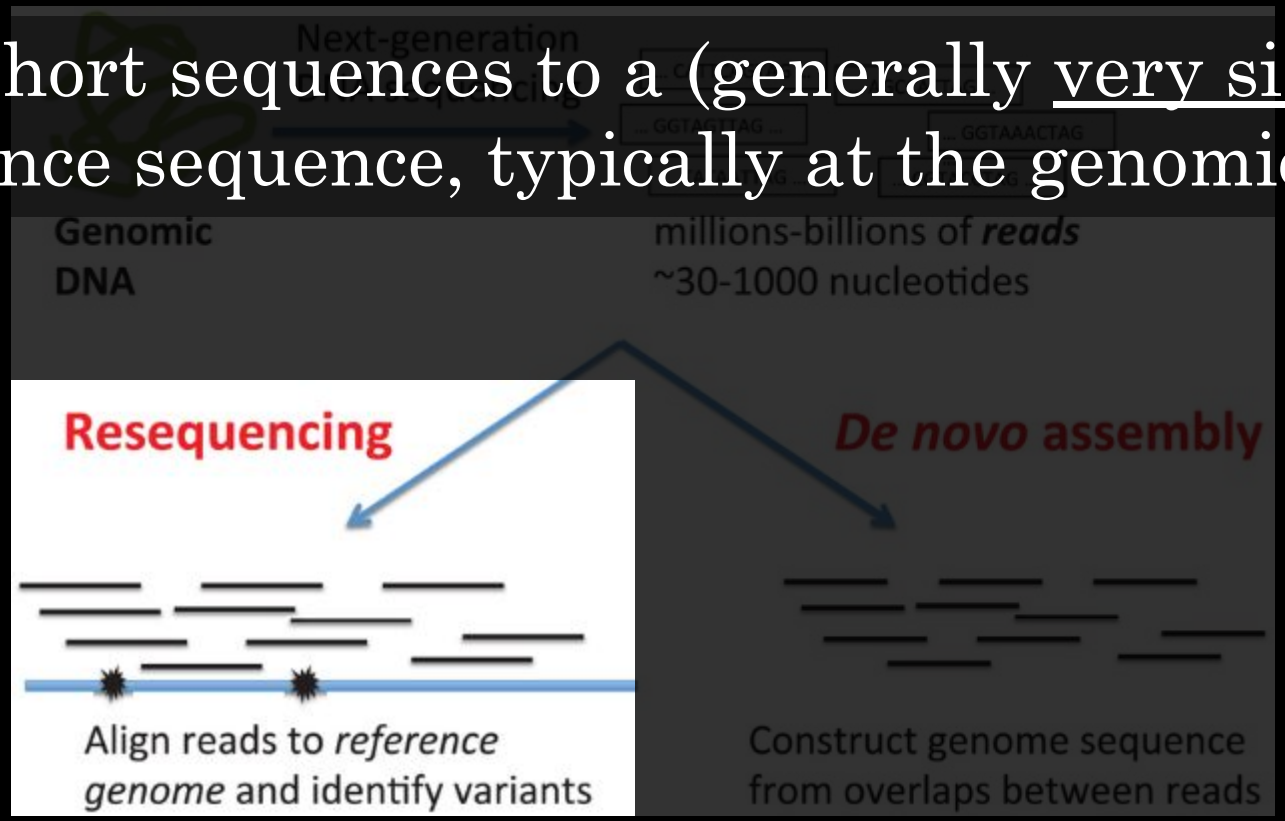
Resequencing: A Different(-ish) Use Case



Raphael (2012) *PLoS Comp Biol*

Resequencing:
A Different(-ish) Use Case

Map short sequences to a (generally very similar) reference sequence, typically at the genomic level



Raphael (2012) *PLoS Comp Biol*

Contrasting Cases

- MEGABLAST: Somewhat similar sequences
- BLAST: Remote homology – lots of sequence differences, possibly a lot of gaps too
- PSI-BLAST / hidden Markov models: Super remote homology

- Mapping: Very similar, few to no differences / gaps

Resequencing

UK

100K

RARE GENETIC VARIANTS IN HEALTH AND DISEASE

2001

articles

Initial sequencing and analysis of the human genome

International Human Genome Sequencing Consortium*

* A partial list of authors appears on the opposite page. Affiliations are listed at the end of the paper.

The human genome holds an extraordinary trove of information about human development, physiology, medicine and evolution. Here we report the results of an international collaboration to produce and make freely available a draft sequence of the human genome. We also present an initial analysis of the data, describing some of the insights that can be gleaned from the sequence.

THE HUMAN GENOME

The Sequence of the Human Genome

J. Craig Venter,[1]* Mark D. Adams,[1] Eugene W. Myers,[1] Peter W. Li,[1] Richard J. Mural,[1] Granger G. Sutton,[1] Hamilton O. Smith,[1] Mark Yandell,[1] Cheryl A. Evans,[1] Robert A. Holt,[1] Jeannine D. Gocayne,[1] Peter Amanatides,[1] Richard M. Ballew,[1] Daniel H. Huson,[1] Jennifer Russo Wortman,[1] Qing Zhang,[1] Chinnappa D. Kodira,[1] Xiangqun H. Zheng,[1] Lin Chen,[1] Marian Skupski,[1] Gangadharan Subramanian,[1] Paul D. Thomas,[1] Jinghui Zhang,[1] George L. Gabor Miklos,[2] Catherine Nelson,[3] Samuel Broder,[1] Andrew G. Clark,[4] Joe Nadeau,[5] Victor A. McKusick,[6] Norton Zinder,[7] Arnold J. Levine,[7] Richard J. Roberts,[8] Mel Simon,[9] Carolyn Slayman,[10] Michael Hunkapiller,[11] Randall Bolanos,[1] Arthur Delcher,[1] Ian Dew,[1] Daniel Fasulo,[1] Michael Flanigan,[1] Liliana Florea,[1] Aaron Halpern,[1] Sridhar Hannenhalli,[1] Saul Kravitz,[1] Samuel Levy,[1] Clark Mobarry,[1] Knut Reinert,[1] Karin Remington,[1] Jane Abu-Threideh,[1] Ellen Beasley,[1] Kendra Biddick,[1] Vivien Bonazzi,[1] Rhonda Brandon,[1] Michele Cargill,[1] Ishwar Chandramouliswaran,[1] Rosane Charlab,[1] Kabir Chaturvedi,[1] Zuoming Deng,[1] Valentina Di Francesco,[1] Patrick Dunn,[1] Karen Eilbeck,[1] Carlos Evangelista,[1] Andrei E. Gabrielian,[1] Weiniu Gan,[1] Wangmao Ge,[1] Fangcheng Gong,[1] Zhiping Gu,[1] Ping Guan,[1] Thomas J. Heiman,[1] Maureen E. Higgins,[1] Rui-Ru Ji,[1] Zhaoxi Ke,[1] Karen A. Ketchum,[1] Zhongwu Lai,[1] Yiding Lei,[1] Zhenya Li,[1] Jiayin Li,[1] Yong Liang,[1] Xiaoying Lin,[1] Fu Lu,[1]

2022

HUMAN GENOMICS

The complete sequence of a human genome

Sergey Nurk[1]†, Sergey Koren[1]†, Arang Rhie[1]†, Mikko Rautiainen[1]†, Andrey V. Bzikadze[2], Alla Mikheenko[3], Mitchell R. Vollger[4], Nicolas Altemose[5], Lev Uralsky[6,7], Ariel Gershman[8], Sergey Aganezov[9]‡, Savannah J. Hoyt[10], Mark Diekhans[11], Glennis A. Logsdon[4], Michael Alonge[9], Stylianos E. Antonarakis[12], Matthew Borchers[13], Gerard G. Bouffard[14], Shelise Y. Brooks[14], Gina V. Caldas[15], Nae-Chyun Chen[9], Haoyu Cheng[16,17], Chen-Shan Chin[18], William Chow[19], Leonardo G. de Lima[13], Philip C. Dishuck[4], Richard Durbin[19,20], Tatiana Dvorkina[3], Ian T. Fiddes[21], Giulio Formenti[22,23], Robert S. Fulton[24], Arkarachai Fungtammasan[18], Erik Garrison[11,25], Patrick G. S. Grady[10], Tina A. Graves-Lindsay[26], Ira M. Hall[27], Nancy F. Hansen[28], Gabrielle A. Hartley[10], Marina Haukness[11], Kerstin Howe[19], Michael W. Hunkapiller[29], Chirag Jain[1,30], Miten Jain[11], Erich D. Jarvis[22,23], Peter Kerpedjiev[31], Melanie Kirsche[9], Mikhail Kolmogorov[32], Jonas Korlach[29], Milinn Kremitzki[26], Heng Li[16,17], Valerie V. Maduro[33], Tobias Marschall[34], Ann M. McCartney[1], Jennifer McDaniel[35], Danny E. Miller[4,36], James C. Mullikin[14,28], Eugene W. Myers[37], Nathan D. Olson[35], Benedict Paten[11], Paul Peluso[29], Pavel A. Pevzner[32], David Porubsky[4], Tamara Potapova[13], Evgeny I. Rogaev[6,7,38,39], Jeffrey A. Rosenfeld[40], Steven L. Salzberg[9,41], Valerie A. Schneider[42], Fritz J. Sedlazeck[43], Kishwar Shafin[11], Colin J. Shew[44], Alaina Shumate[41], Ying Sims[19], Arian F. A. Smit[45], Daniela C. Soto[44], Ivan Sović[29,46], Jessica M. Storer[45], Aaron Streets[5,47], Beth A. Sullivan[48], Françoise Thibaud-Nissen[42], James Torrance[19], Justin Wagner[35], Brian P. Walenz[1], Aaron Wenger[29], Jonathan M. D. Wood[19], Chunlin Xiao[42], Stephanie M. Yan[49], Alice C. Young[14], Samantha Zarate[9], Urvashi Surti[50], Rajiv C. McCoy[49], Megan Y. Dennis[44], Ivan A. Alexandrov[3,7,51], Jennifer L. Gerton[13,52], Rachel J. O'Neill[10], Winston Timp[8,41], Justin M. Zook[35], Michael C. Schatz[9,49], Evan E. Eichler[4,53]*, Karen H. Miga[11,54]*, Adam M. Phillippy[1]*

Since its initial release in 2000, the human reference genome has covered only the euchromatic fraction of the genome, leaving important heterochromatic regions unfinished. Addressing the remaining 8% of the genome, the Telomere-to-Telomere (T2T) Consortium presents a complete 3.055 billion–base pair sequence of a human genome, T2T-CHM13, that includes gapless assemblies for all chromosomes except Y, corrects errors in the prior references, and introduces nearly 200 million base pairs of sequence containing 1956 gene predictions, 99 of which are predicted to be protein coding. The completed regions include all centromeric satellite arrays, recent segmental duplications, and the short arms of all five acrocentric chromosomes, unlocking these complex regions of the genome to variational and functional studies.

Lander et al. (2001) *Nature*
Venter et al. (2001) *Science*
Nurk et al. (2022) *Science*
"Landmark human genome papers"
https://doe-humangenomeproject.ornl.gov/landmark-hgp-papers/

The Critical Question

- How can we exploit the high expected degree of similarity to accelerate the search?

- Key considerations:
  - DNA is a four-letter alphabet, there will be lots of redundant sequences in the genome
  - We can use approaches that treat mismatches and gaps less efficiently than dynamic programming

Preliminaries: strings

Offsets

```
S<-AGGCCTA
  0123456

 S[3]=C
S[5:]=TA
```

Substring

```
AGCGCTA
AGCGCTA
 GCGC
  CTA
   A
GC
 GC
```

Terminator

```
AGCGCTA$

$<A<C<G<T

AA$<AAA$

BEE<BEER
```

Preliminaries: strings

Prefixes	Suffixes
AGCGCTA$	AGCGCTA$
AGCGCTA	AGCGCTA$
AGCGCT	GCGCTA$
AGCGC	CGCTA$
AGCG	GCTA$
AGC	CTA$
AG	TA$
A	A$
	$

Naive Searching

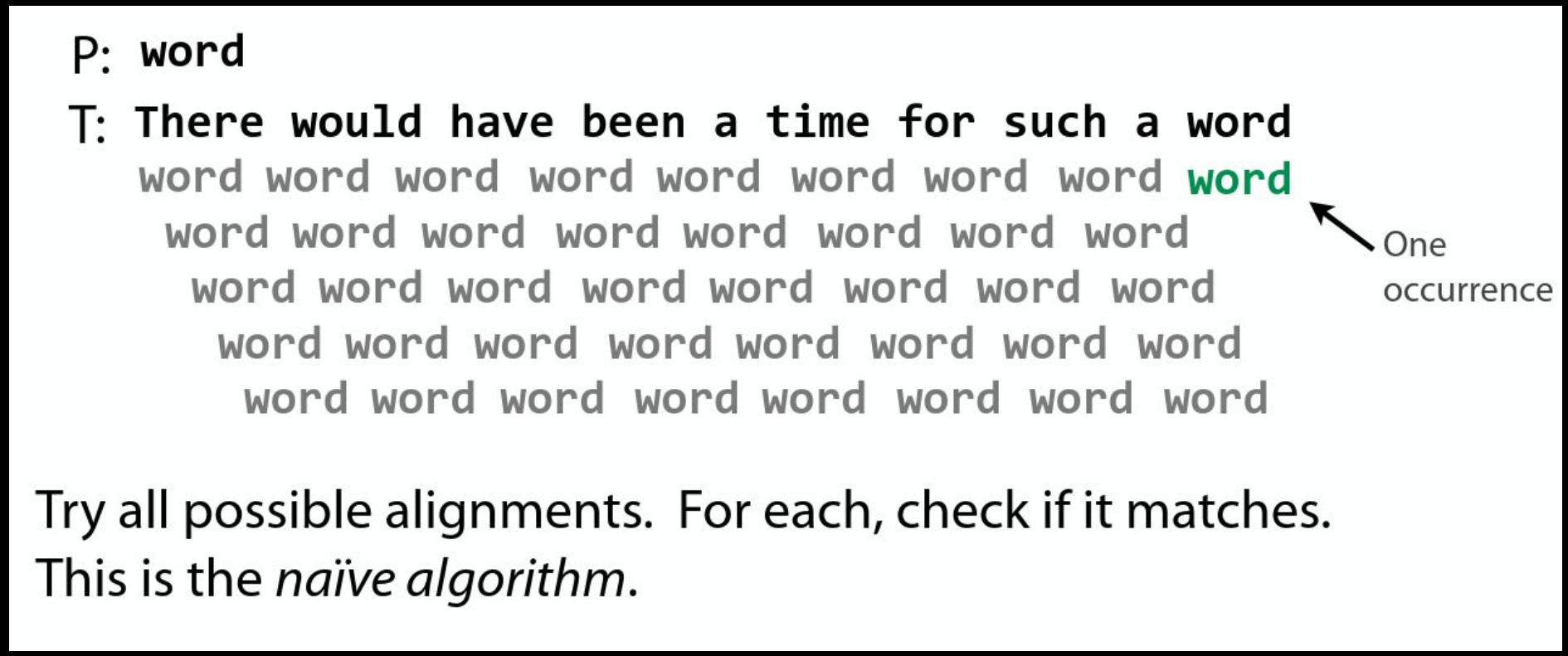
Find places where *pattern P* occurs as a substring of *text T*.
Each such place is an *occurrence* or *match*.

Let $n = |P|$, and let $m = |T|$ Assume $n \leq m$

Alignment: a way of putting *P's* characters opposite *T's*.
May or may not correspond to an match.

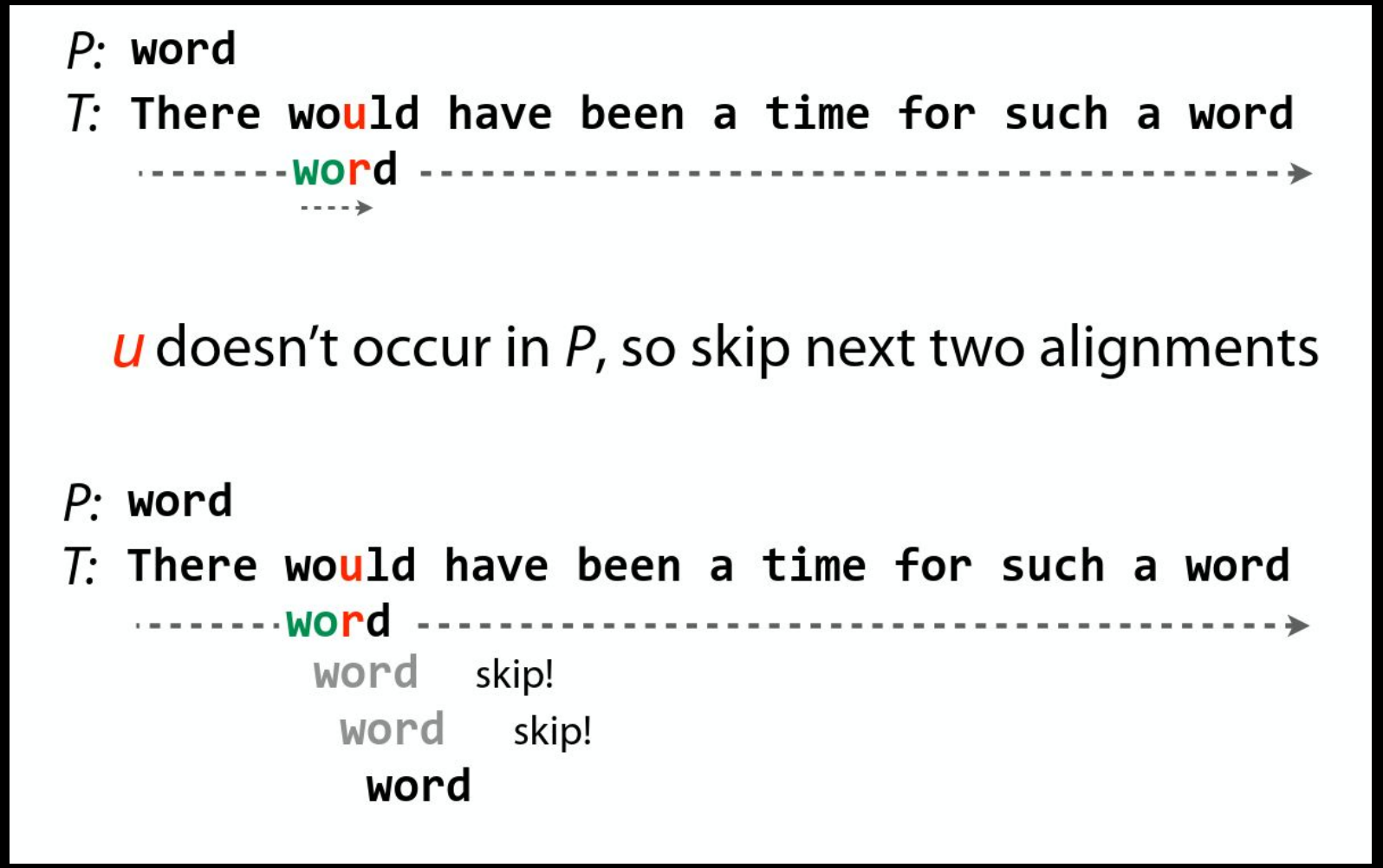
```
P: word
T: There would have been a time for such a word
Alignment 1: word
                                  Alignment 2: word
```

Naive Searching



Not good: back to m - n + 1 possible matches or ~4^n with gaps!

Skip fruitless alignments: Boyer-Moore



extending to approximate matching is awkward.

Scaling in genome size will not work!

P: word

T: There would have been a time for such a word

word

As *m* & *n* grow, # characters comparisons grows with...

$|P| = n \quad |T| = m$

	Naïve matching	Boyer-Moore
Worst case	m·n	m
Best case	m	m / n

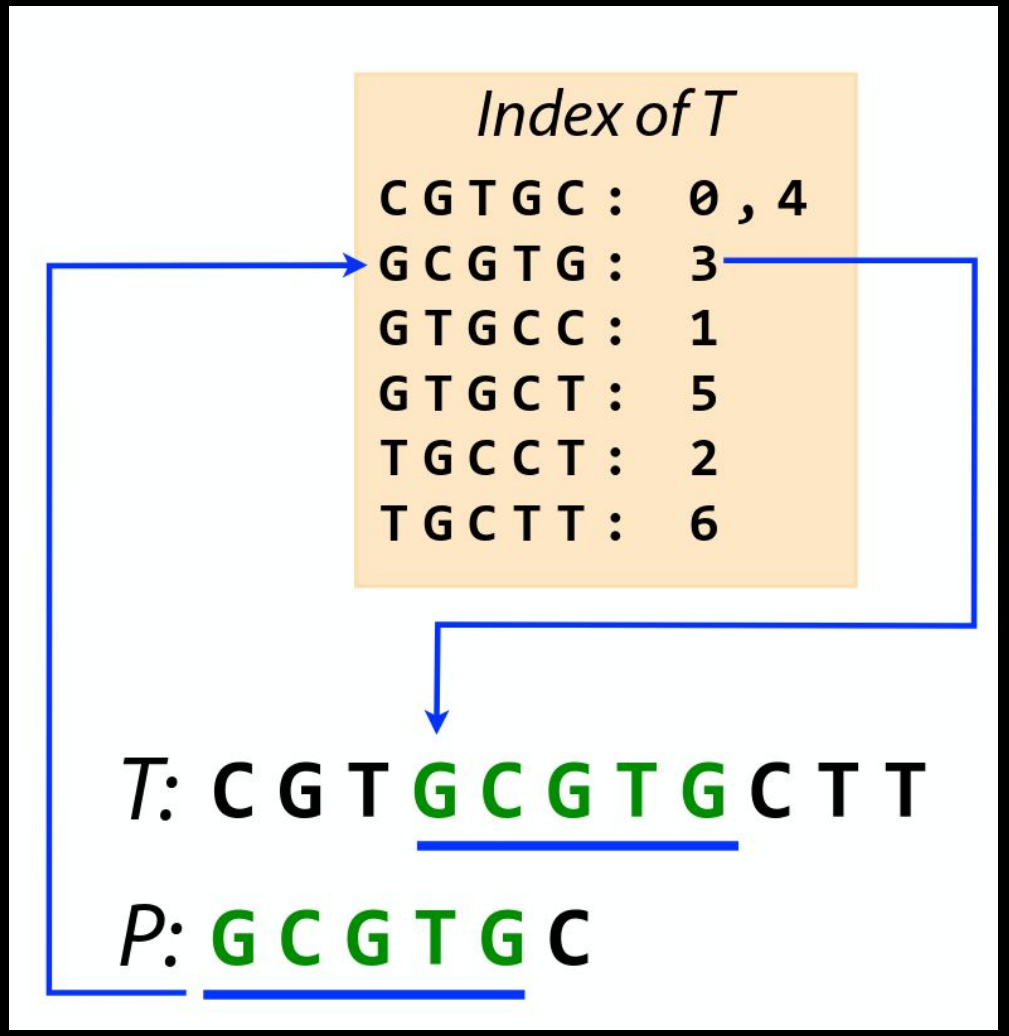
Read mapping:

- Many short reads (small n) vs.
- Large genome (big m).
- Multiple matches possible

BLAST has same issue

Can we use some indexing tricks to scale in **n** rather than **m**?

K-mer Indexing



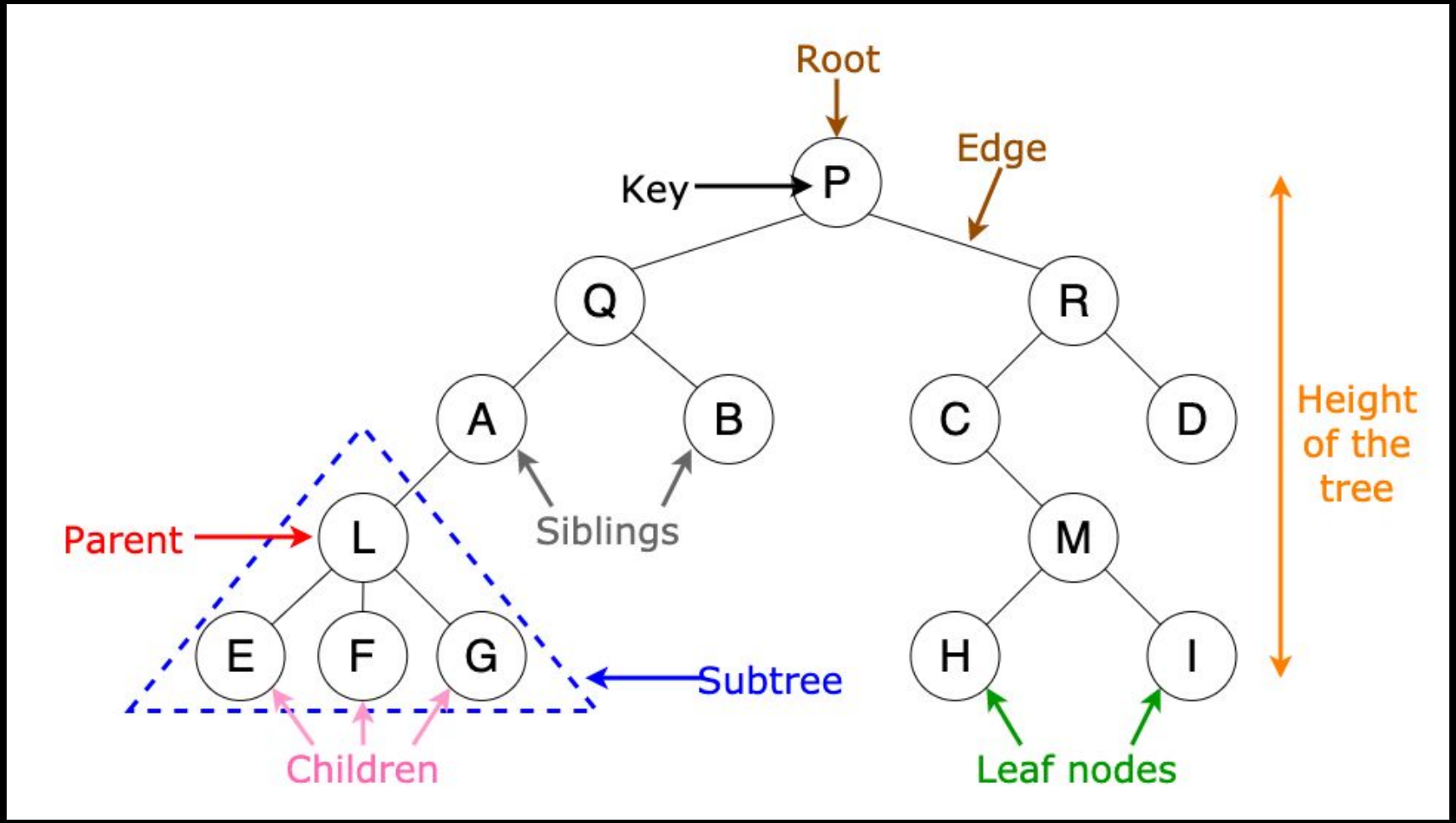
Reference indexed once and O(1) lookup

BUT:

- K-mers shorter than reads
- Extensions requires many lookups (neighbourhood enumeration) or expensive DP
- Repetitive K-mers mean many candidates
- Index doesn't easily capture context (adjacent K-mers)

Let’s trie a different way of indexing a genome

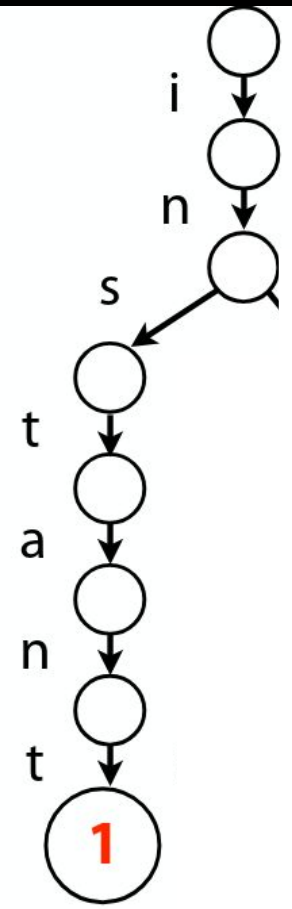
Trees in Computer Science



A trie is a special type of tree

Keys: instant, internal, internet

Key	Value
instant	1
internal	2
internet	3



Smallest tree such that:

- Each edge is labeled with a character $c \in \Sigma$
- For given node, at most one child edge has label c, for any $c \in \Sigma$
- Each key is "spelled out" along some path starting at root

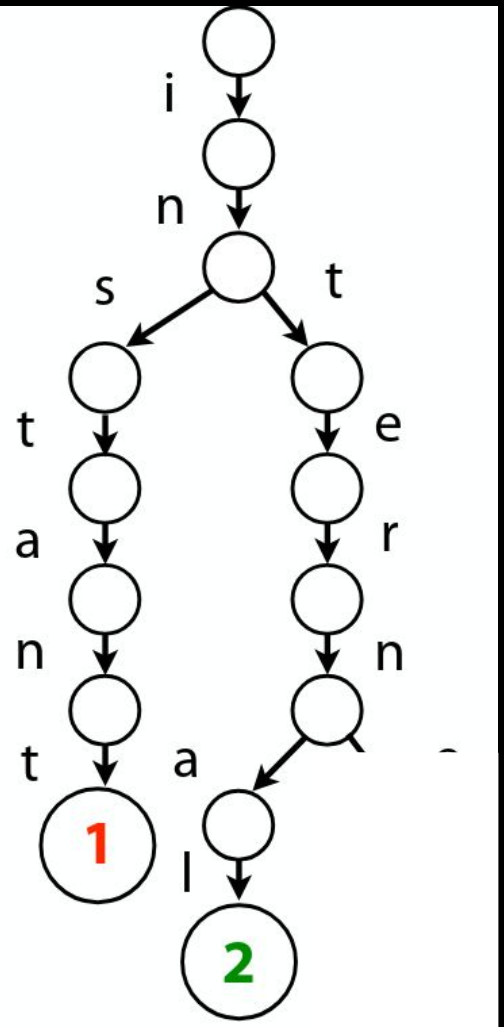
A trie is a special type of tree

Keys: instant, internal, internet

Key	Value
instant	1
internal	2
internet	3

Smallest tree such that:

- Each edge is labeled with a character $c \in \Sigma$
- For given node, at most one child edge has label c, for any $c \in \Sigma$
- Each key is "spelled out" along some path starting at root



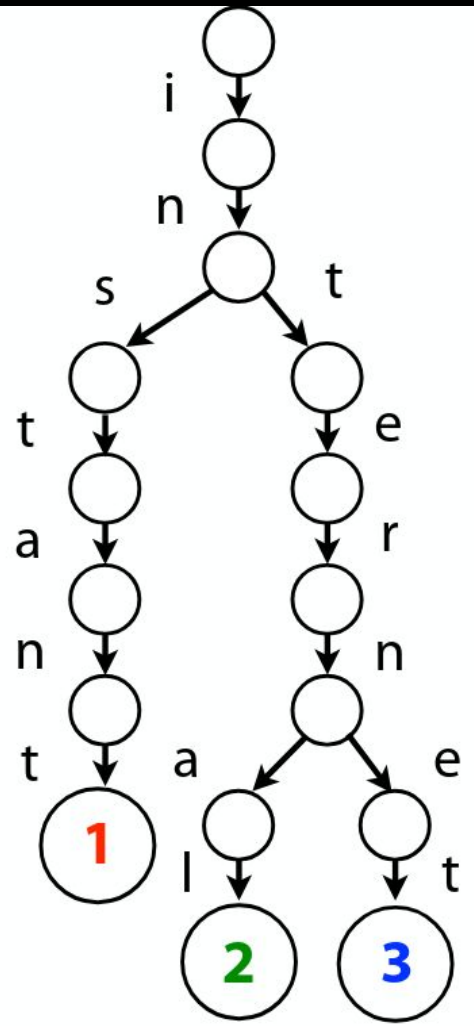
A trie is a special type of tree

Keys: instant, internal, internet

Key	Value
instant	1
internal	2
internet	3

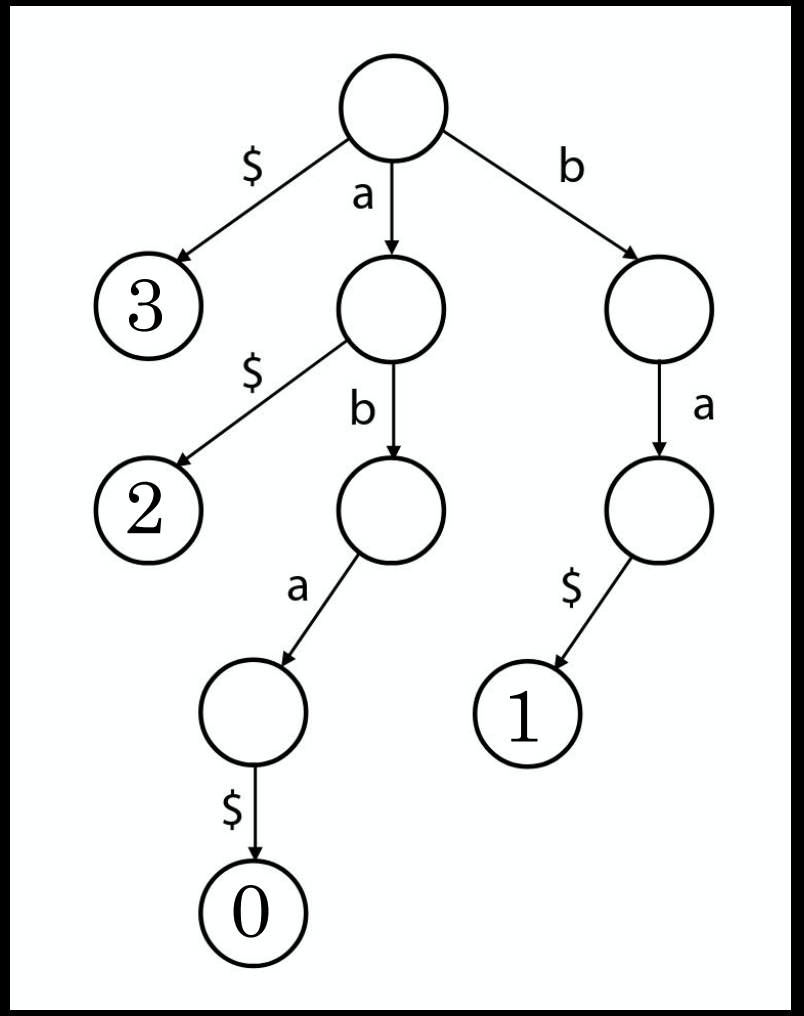
Smallest tree such that:

- Each edge is labeled with a character $c \in \Sigma$
- For given node, at most one child edge has label c, for any $c \in \Sigma$
- Each key is "spelled out" along some path starting at root



Suffix trie: trie containing all suffixes

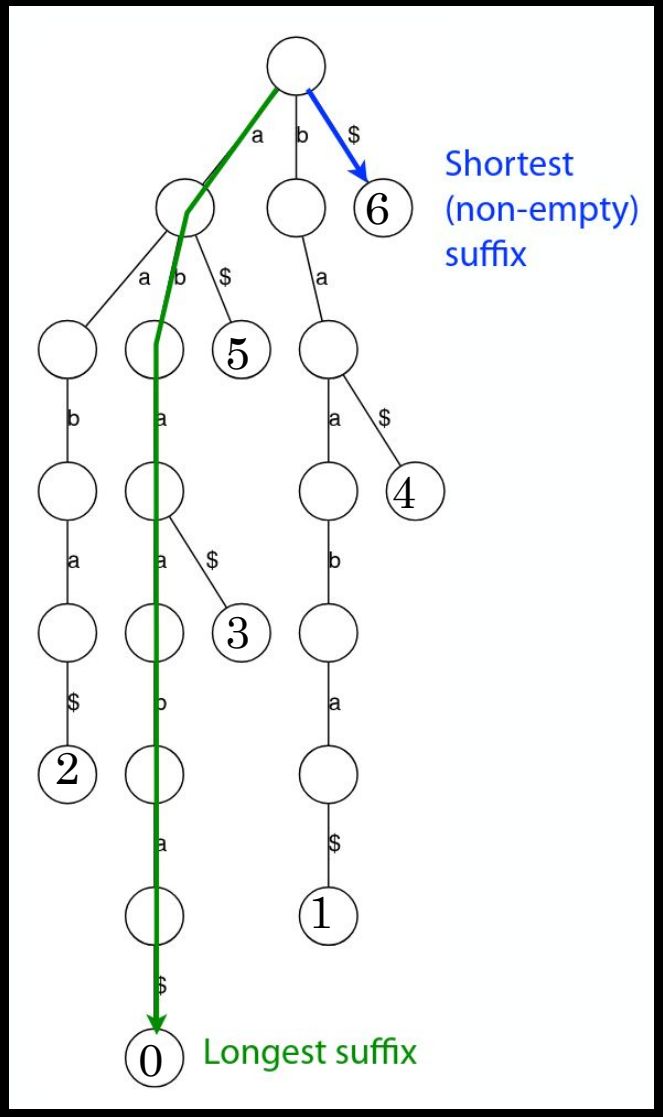
```
T = aba$
    aba$
     ba$
      a$
       $
```



Each path from root to leaf represents a suffix; each suffix is represented by some path from root to leaf

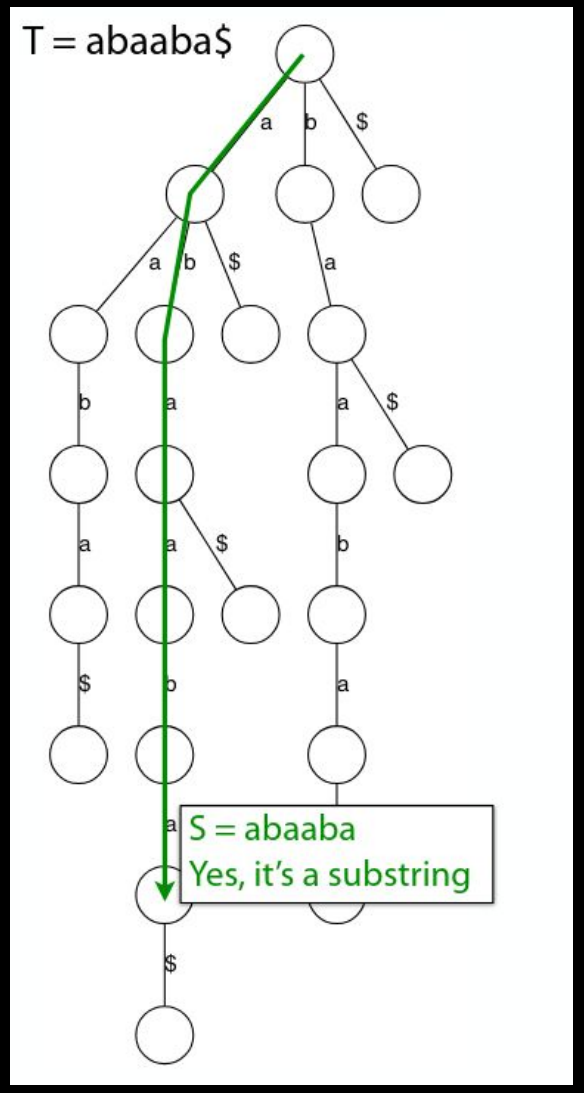
Suffix trie: trie containing all suffixes

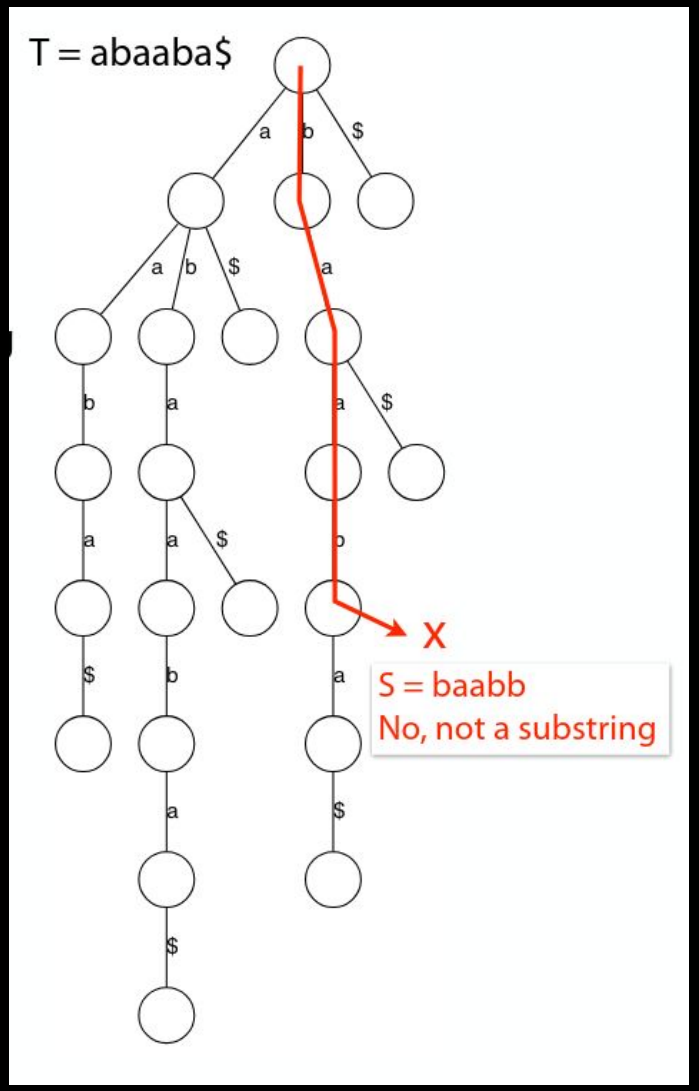
```
T = abaaba$
    abaaba$
     baaba$
      aaba$
       aba$
        ba$
         a$
          $
```

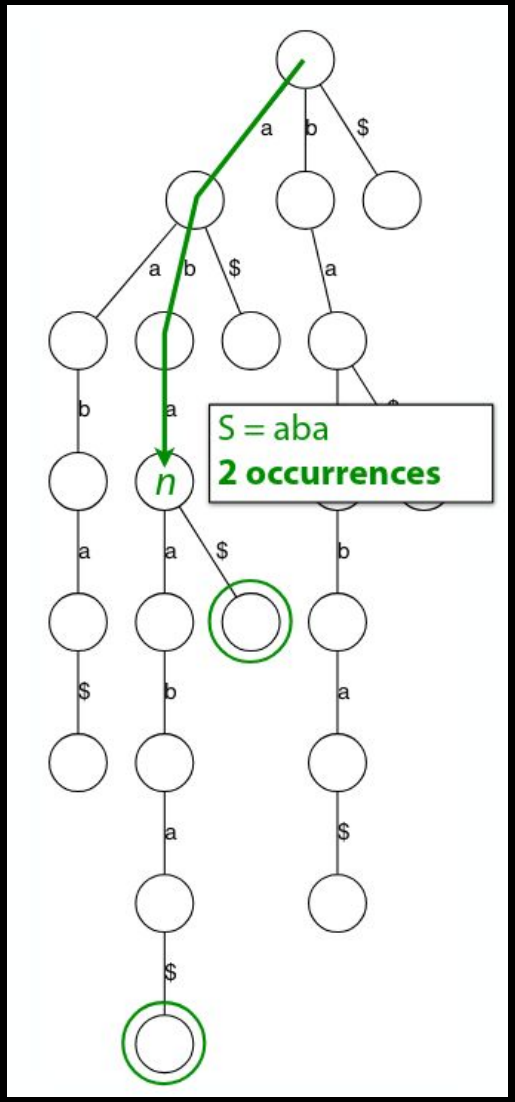


Each path from root to leaf represents a suffix; each suffix is represented by some path from root to leaf

Why? Search and count in O(|S|)

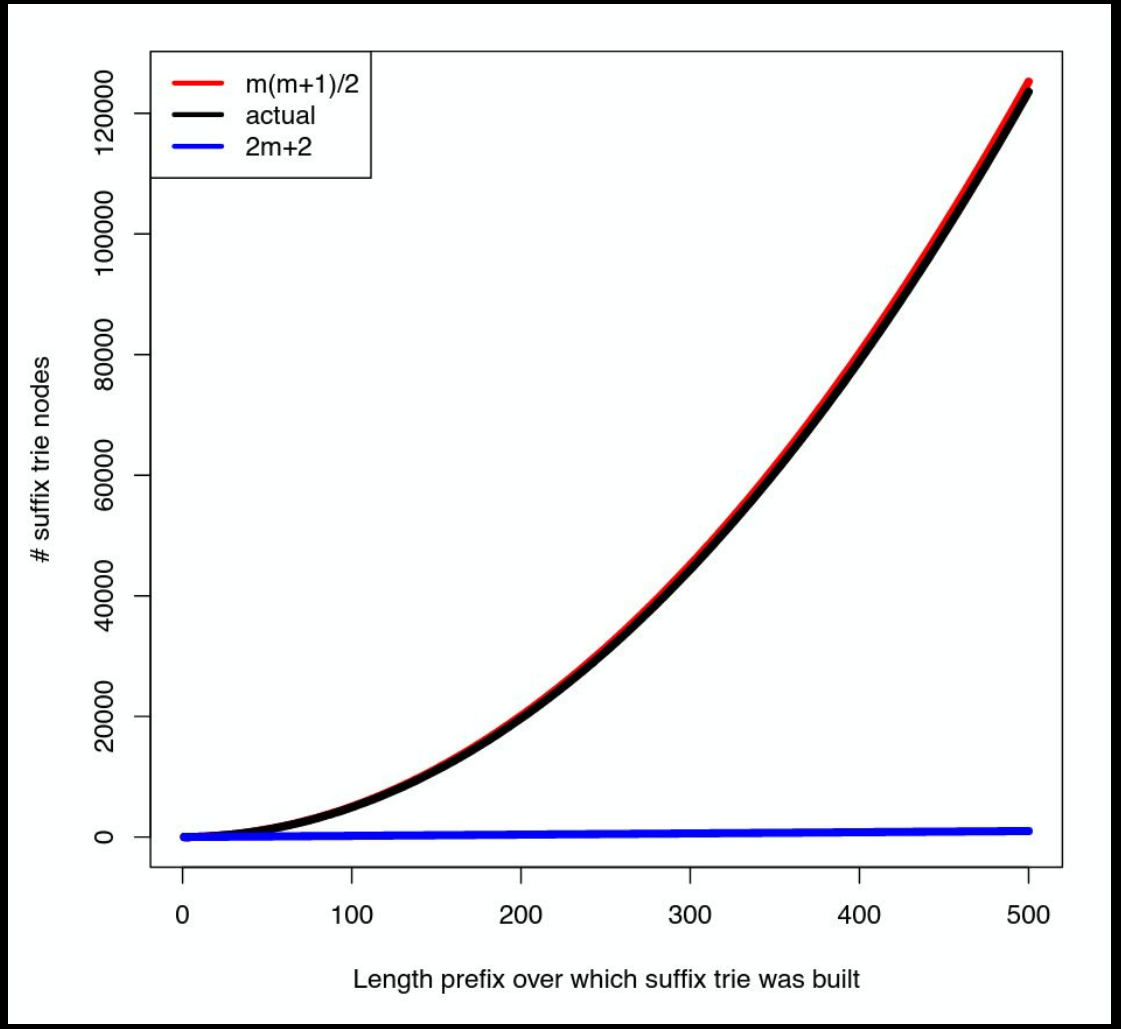




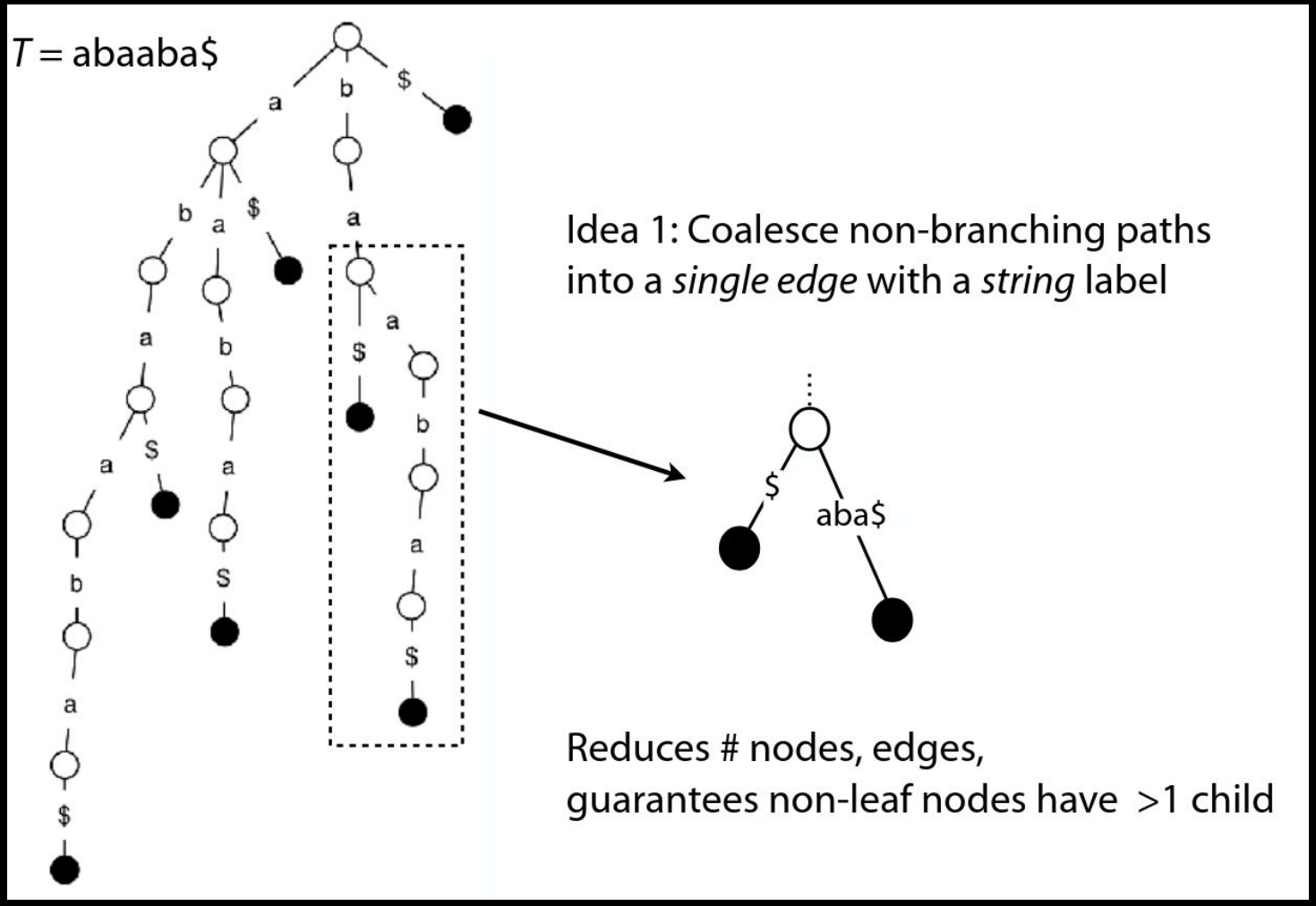


But Suffix Trie ~O(m^2 / 2) space

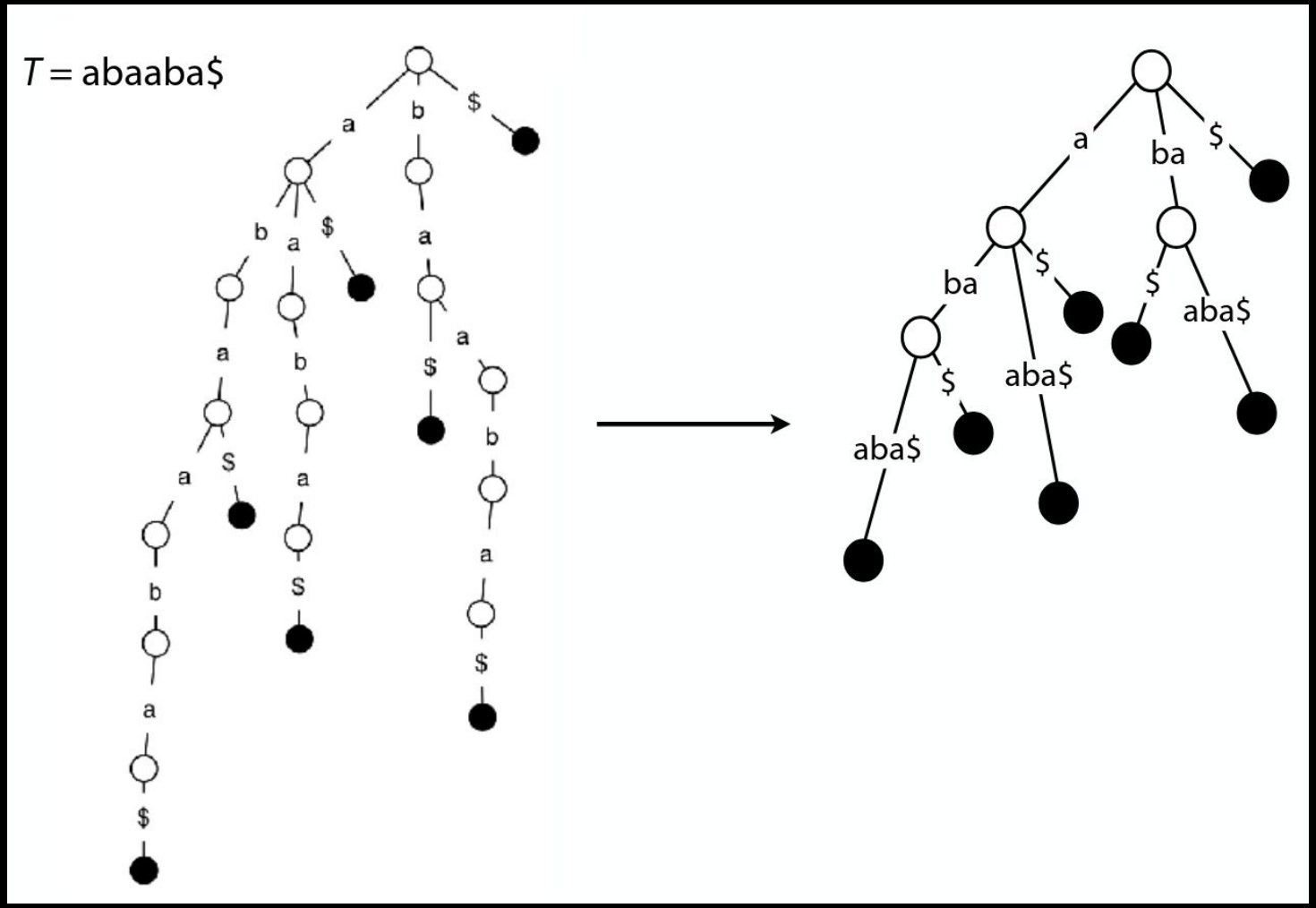
```
T = abaaba$
    abaaba$
     baaba$
      aaba$
       aba$
        ba$
         a$
          $
```



Compressing a Suffix Trie into a Tree



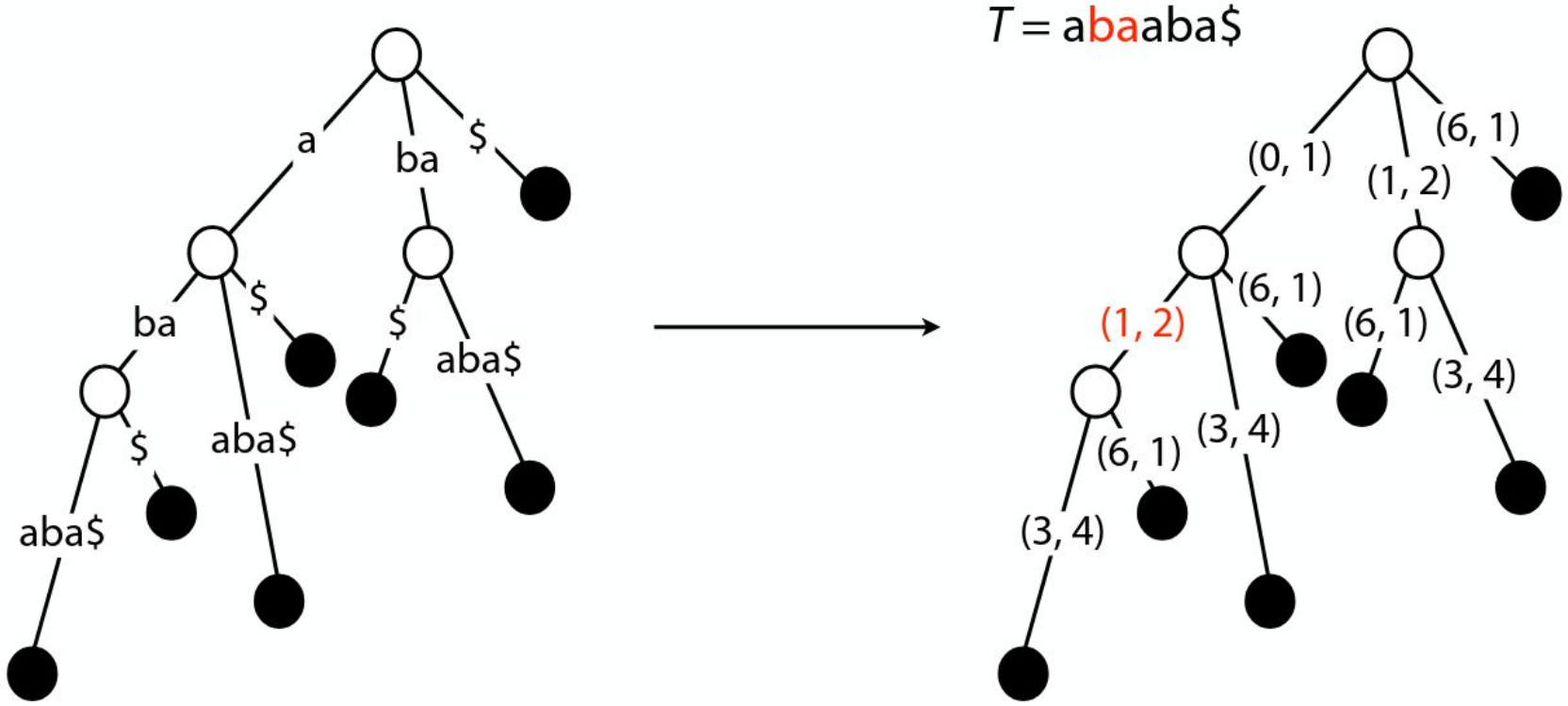
Compressing a Suffix Trie into a Tree



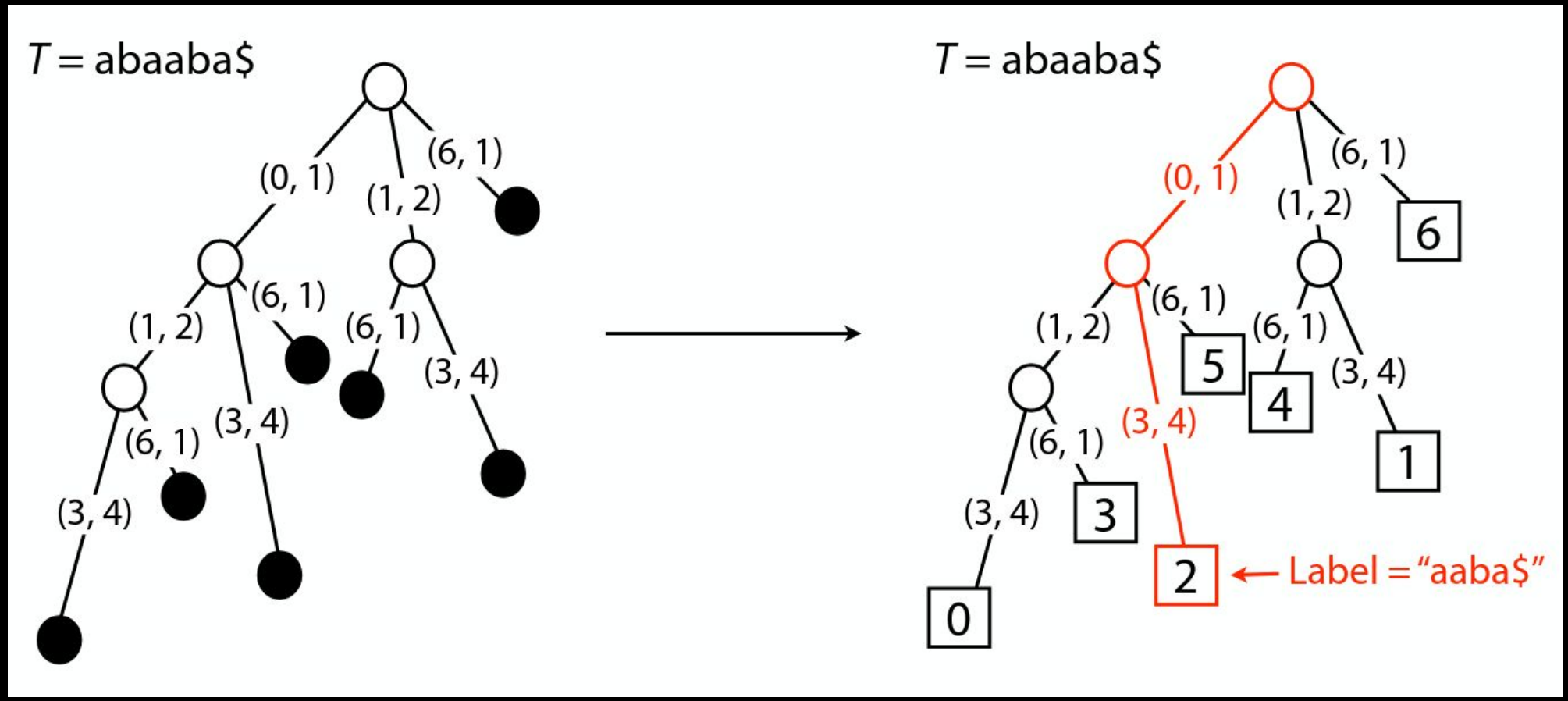
Storing suffix strings still memory intensive

Storing locations instead of strings

Idea 2: Store *T* itself in addition to the tree. Convert tree's edge labels to (offset, length) pairs with respect to *T*.



Label leaves with offsets



Retain the same operations

Suffix Trie ~ $O(m^2/2)$ space
Suffix Tree ~ O(m) space!

BUT… don't forget constant factors!

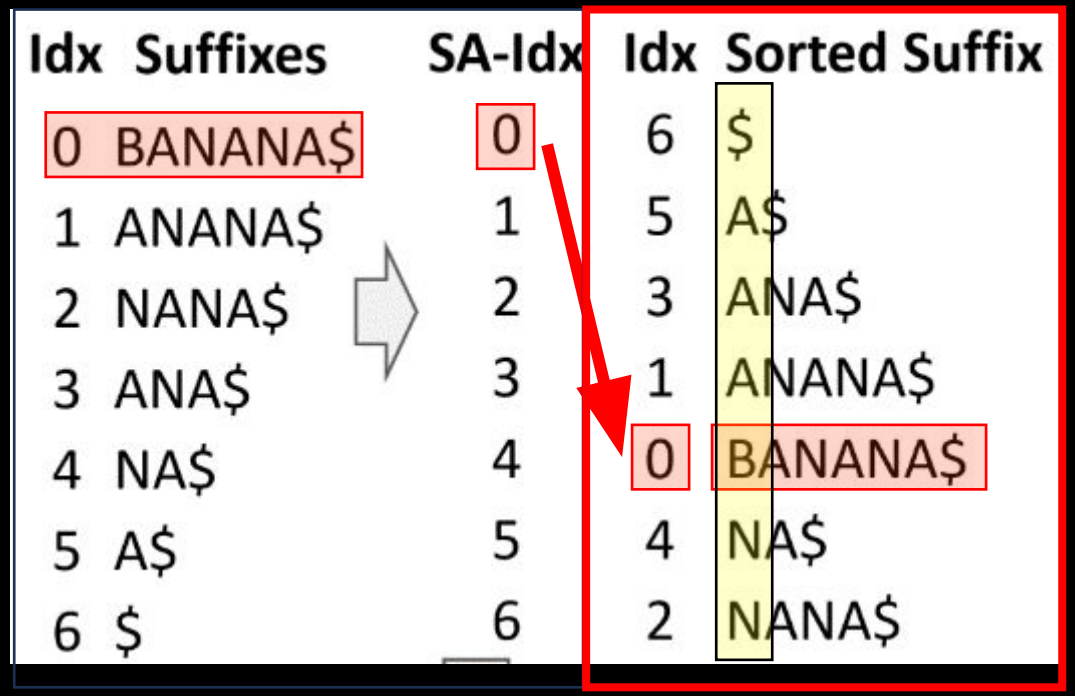
MUMmer ≈ 15.76 bytes per nt

Suffix tree of human genome will still be >45GB (or larger depending on exact data structures underlying suffix tree nodes/edges)

	Suffix tree
Time: Does P occur?	$O(n)$
Time: Count k occurrences of P	$O(n + k)$
Time: Report k locations of P	$O(n + k)$
Space	$O(m)$

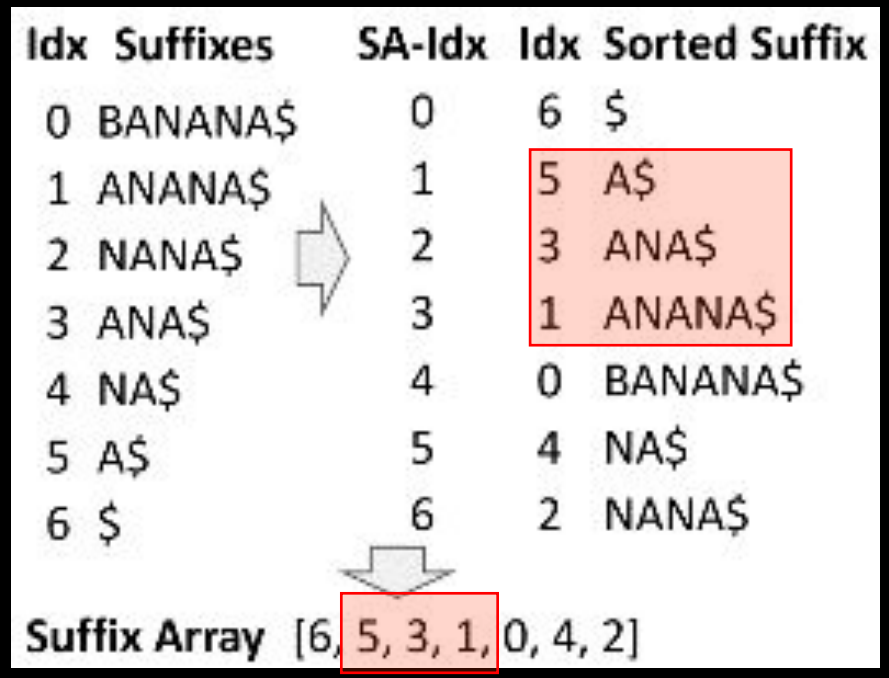
$m = |T|,\ n = |P|,\ k =$ # occurrences of P in T

From suffix trees to suffix arrays



Every suffix, sorted lexicographically
($ is smallest)
Index value remains the same!

What is the point of *this*?



We now have a compressed representation where lexicographically similar strings are adjacent to one another

All suffixes that start with "A" are at positions 1 to 3 (because the first position is numbered 0) in the suffix array

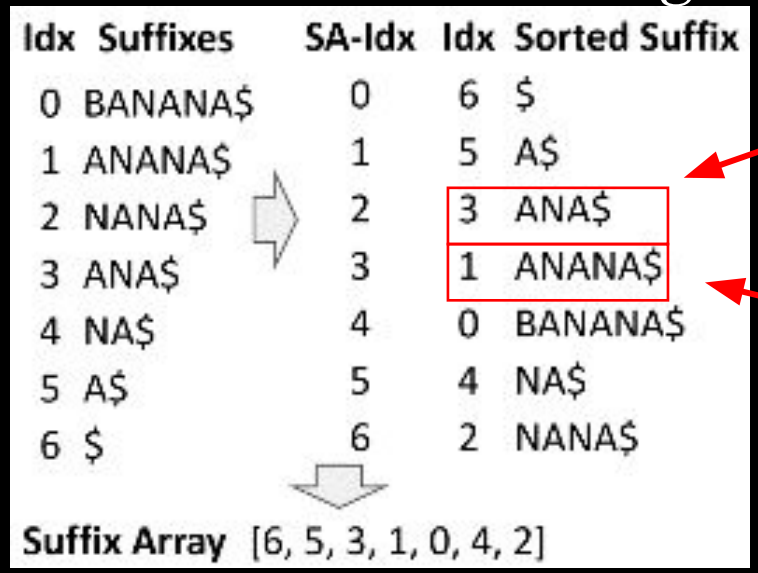
So if we're searching for "A" we only need to look at this consecutive block of rows!

Still O(m) space BUT much reduced constant factor ~4 bytes/base * 3bn bp ≈ 12 GB.

Suffix tree is >45 GB.

Does my sequence match?

Search “ANA” against the suffix array



Binary search to find first “ANA”

Progress to last “ANA”

There are two matches of “ANA” in “BANANA”
They start at positions 3 and 1

Naive: O(n log m), Skipping: O(n + log m) - smaller but slower than suffix array!

BUT… still a large index and inexact matching (not covered) is a bit painful!

Want something smaller than suffix array but with query time of suffix tree (and can handle inexact matches)

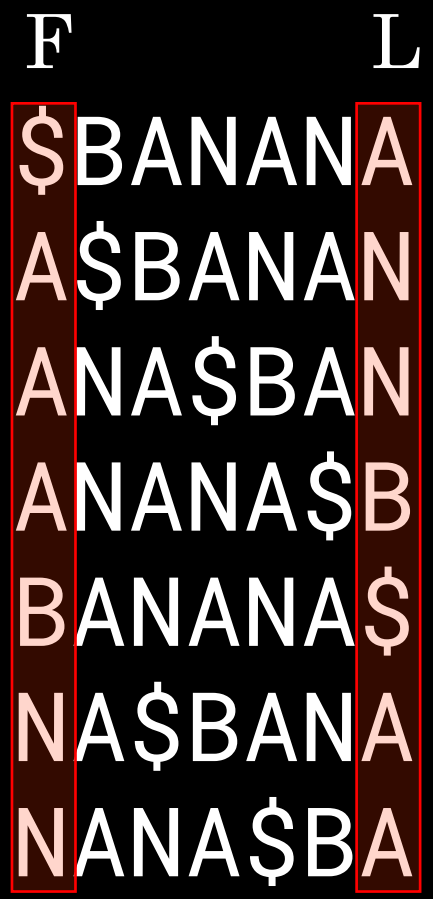
Burrows-Wheeler Transform

All Rotations	Sort Rotations	Take last column
B A N A N A $	$ B A N A N A	$ B A N A N A
A N A N A $ B	A $ B A N A N	A $ B A N A N
N A N A $ B A	A N A $ B A N	A N A $ B A N
A N A $ B A N	A N A N A $ B	A N A N A $ B
N A $ B A N A	B A N A N A $	B A N A N A $
A $ B A N A N	N A $ B A N A	N A $ B A N A
$ B A N A N A	N A N A $ B A	N A N A $ B A

$ < a < b < n

BWT(BANANA$) = ANNB$AA

Why Do Such a Ridiculous Thing?



If we're doing string searching, we need to explore different rows of the matrix

The end of one row should point to the row that starts with the same exact character in the string. This is LF mapping

Why Do Such a Ridiculous Thing?

F	L
$BANAN	A
A$BANAN	
ANA$BAN	
ANANA$B	
BANANA$	
NA$BANA	
NANA$BA	

The first “A” in L is also the first “A” in F

Why Do Such a Ridiculous Thing?

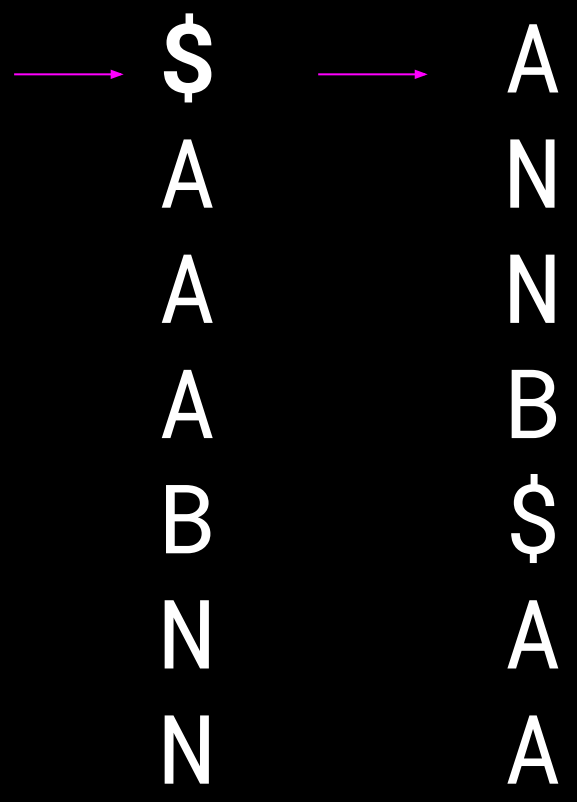
F	L
$BANAN	A
A$BANA	N
ANA$BAN	
ANANA$B	
BANANA$	
NA$BANA	
NANA$BA	

The first “N” in L is also the first “N” in F

LF-Mapping allows reversal of BWT

$BANANA
A$BANAN
ANA$BAN
ANANA$B
BANANA$
NA$BANA
NANA$BA

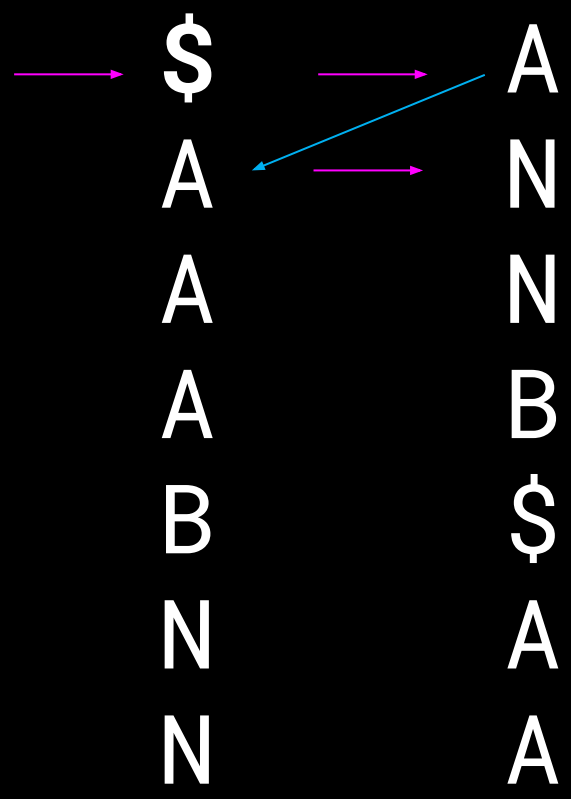
LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

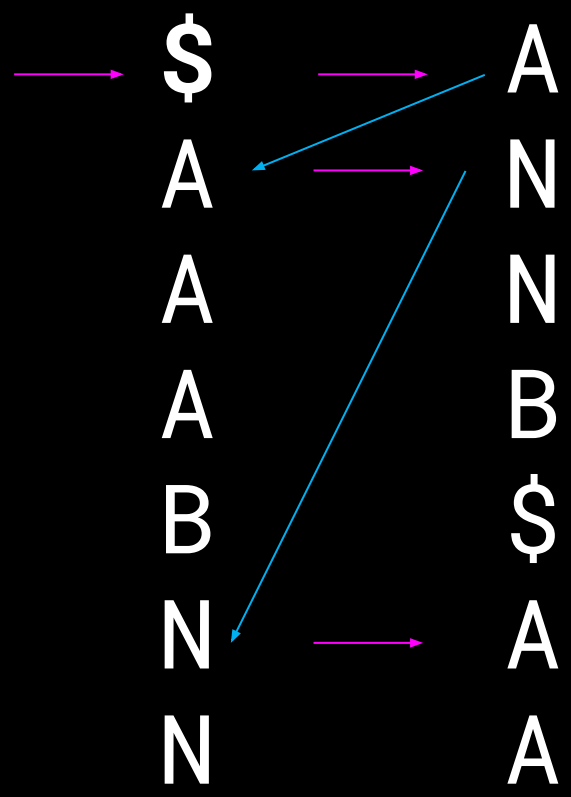
LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

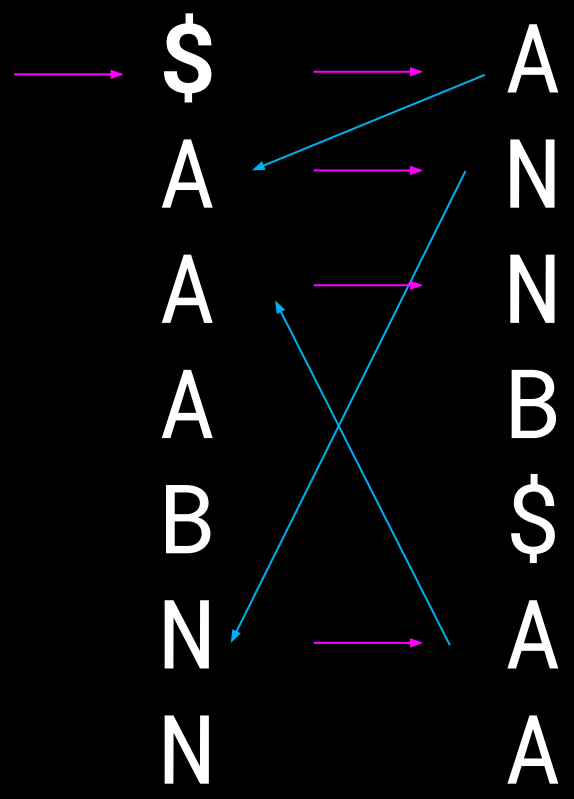
LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

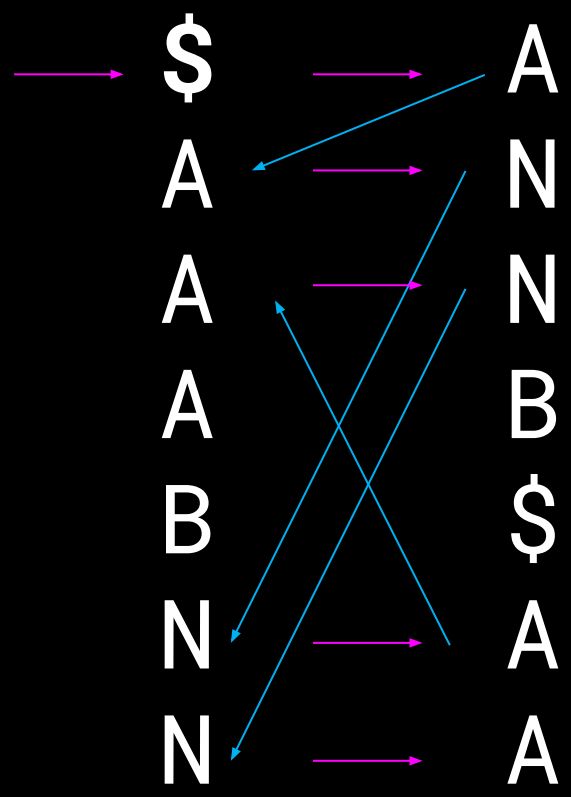
LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

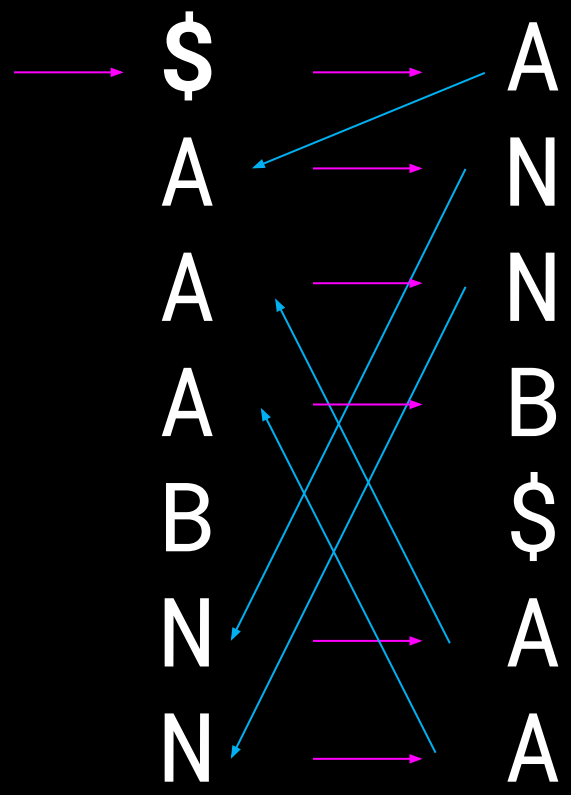
LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

LF-Mapping allows reversal of BWT



T=B A N A N A $

BWT(T)=ANNB$AA

Searching BWM for a string

Look for range of rows of BWM(T) with *P* as prefix

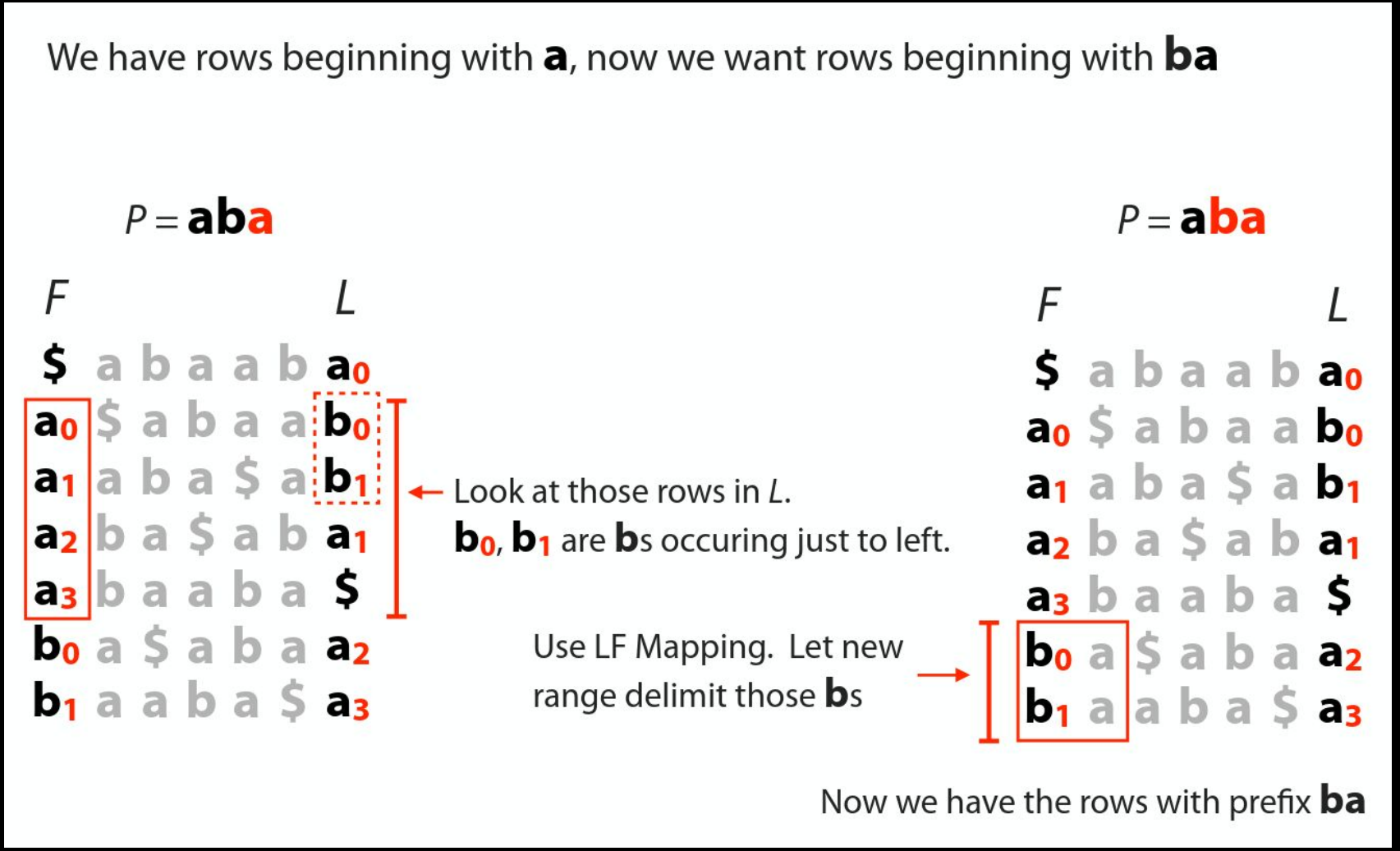
Start with shortest suffix, then match successively longer suffixes

$P = \mathbf{aba}$

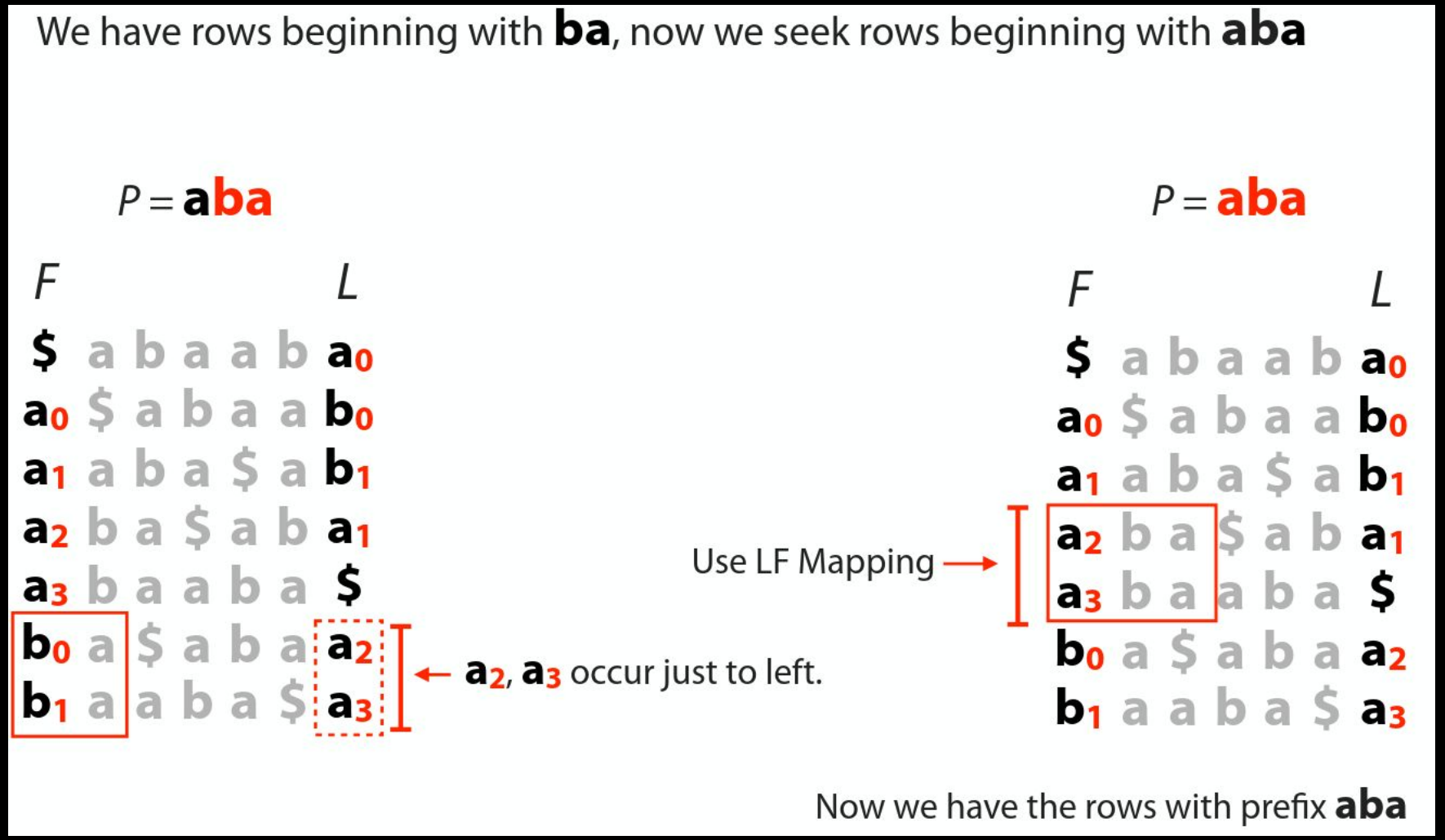
F						*L*
$	a	b	a	a	b	a_0
a_0	$	a	b	a	a	b_0
a_1	a	b	a	$	a	b_1
a_2	b	a	$	a	b	a_1
a_3	b	a	a	b	a	$
b_0	a	$	a	b	a	a_2
b_1	a	a	b	a	$	a_3

Easy to find all the rows beginning with **a**

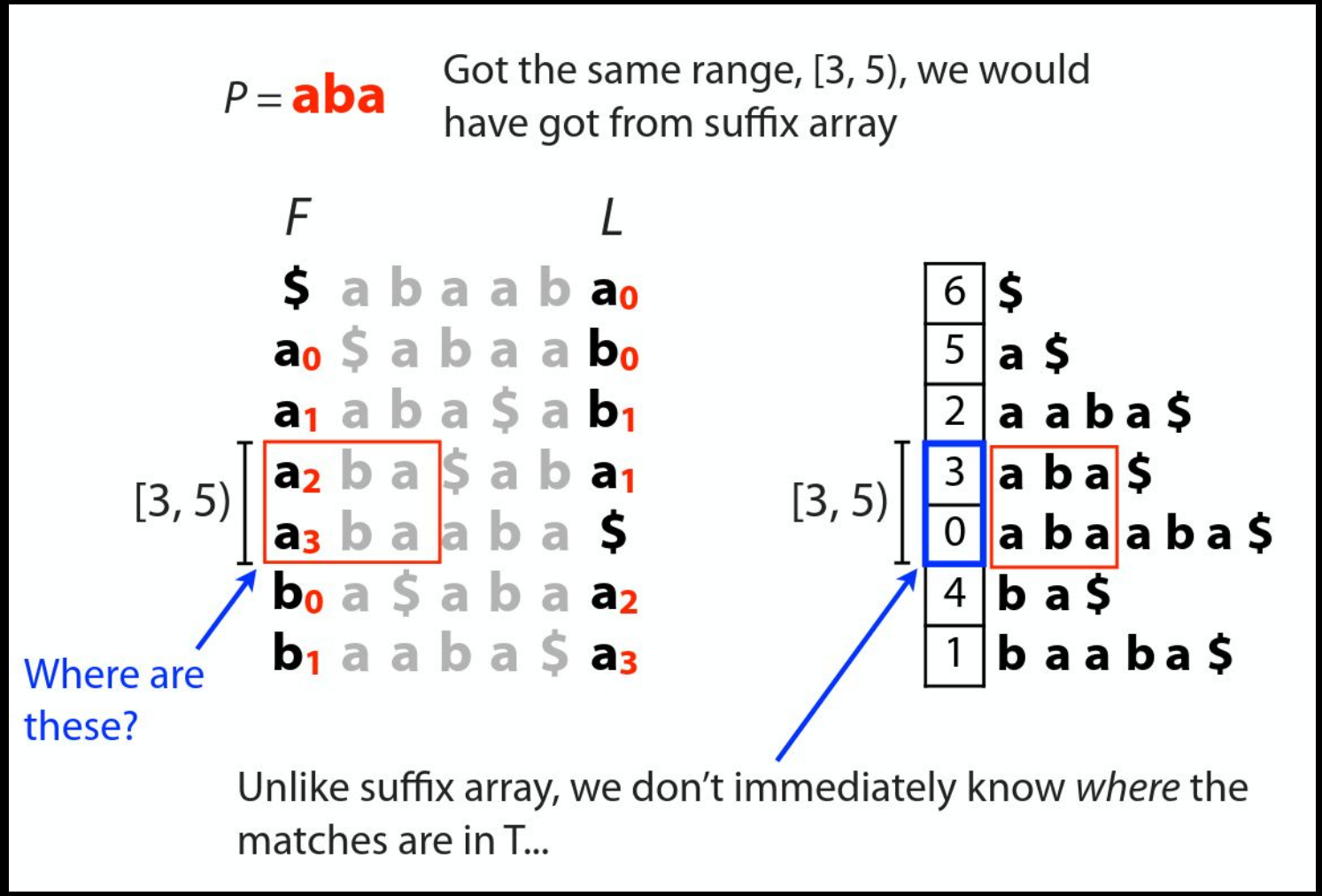
Searching BWM for a string



Searching BWM for a string

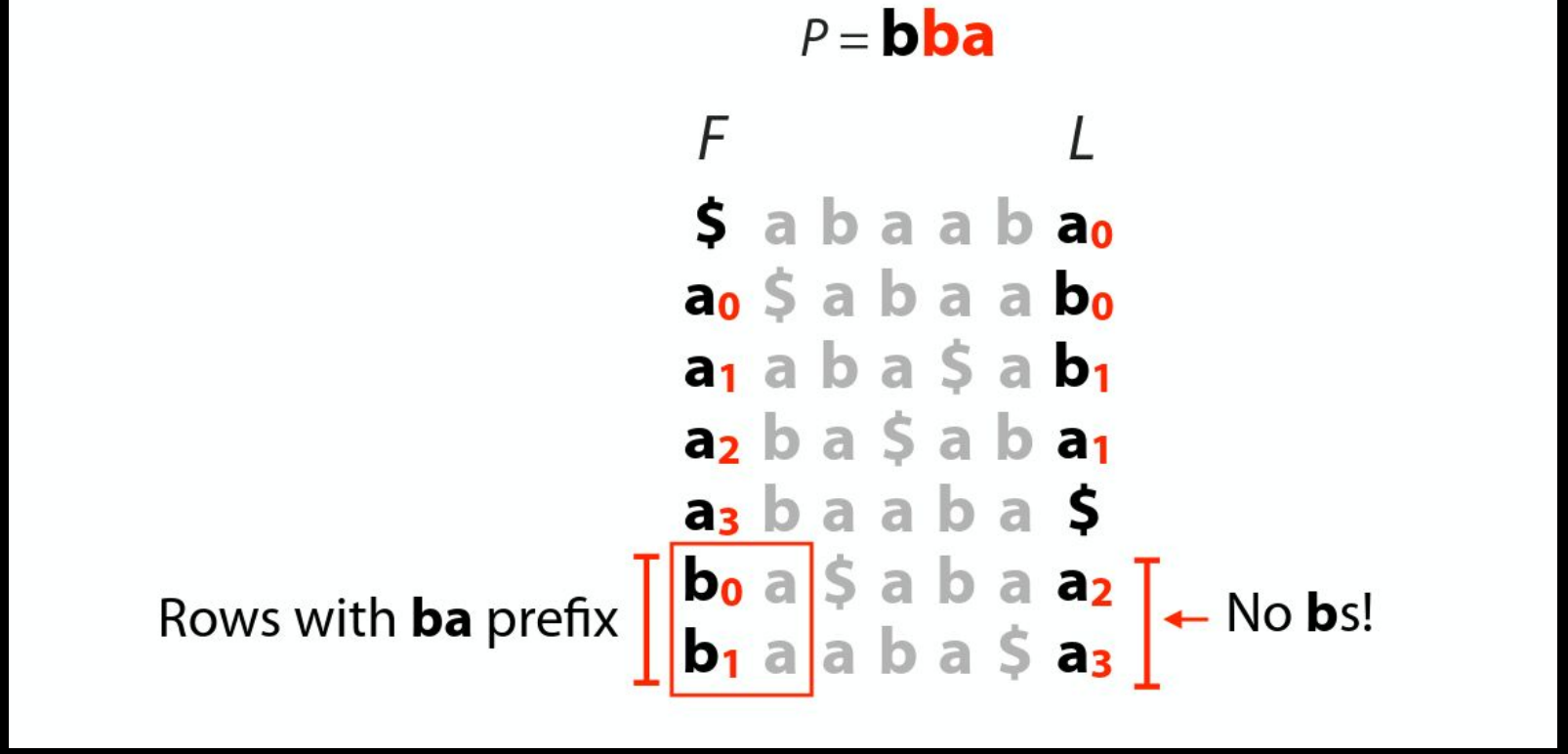


Searching BWM for a string



Searching BWM for a string

When *P* does not occur in *T*, we eventually fail to find next character in *L*:



BWA:
The Burrows-Wheeler Aligner

Idx	Sorted Suffix
6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

$BANAN	A
A$BANA	N
ANA$BA	N
ANANA$	B
BANANA	$
NA$BAN	A
NANA$B	A

Rapid string search using the sorted index
(equivalent to F) and the BWT string (equal to L)

BWA:
The Burrows-Wheeler Aligner

Idx	Sorted Suffix
6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

$BANAN	A
A$BANA	N
ANA$BA	N
ANANA$	B
BANANA	$
NA$BAN	A
NANA$B	A

Key parts of an FM-Index

Rapid string search using the sorted index
(equivalent to F) and the BWT string (equal to L)
And nothing else

Why this is awesome: Sequence reads are effectively searched against different parts of the reference genome at the same time

Using the L-F mapping allows us to remove redundancy relative to the suffix array

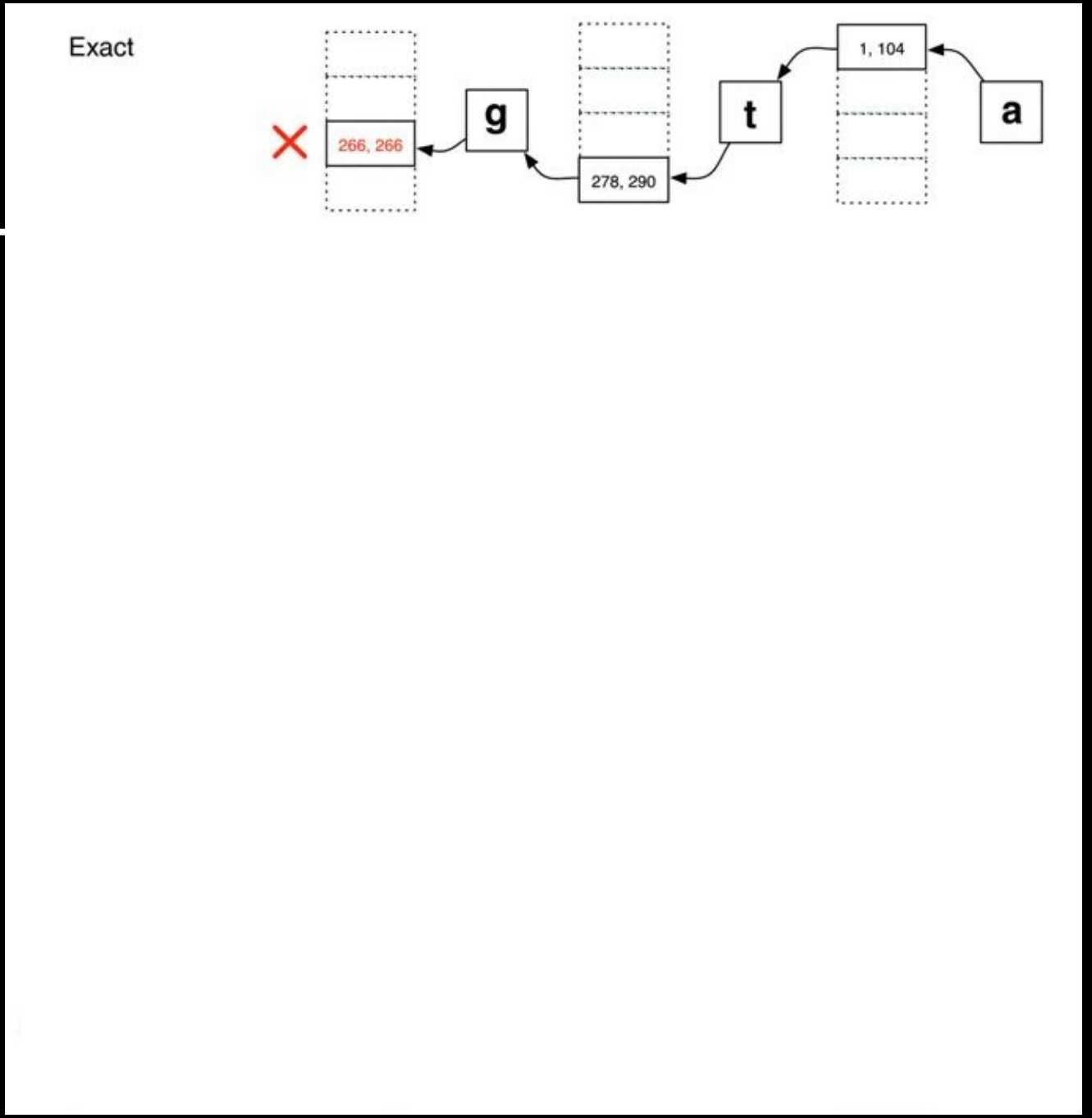
- **Why this is slightly less awesome**: Preprocessing requires many GB of memory

- What about mismatches?

Backtracking

Search string: GGTA

“Genome” contains no GGTA, but contains GGTG



Langmead et al. (2009) *Genome Biol*

BWA refinements

Allowing mismatches: maximum deviation from the search string

- Store searches in a heap to prioritize the lowest mismatches in the search so far

Custom penalties for mismatches, insertion and deletions

Double indexing: BWTs from both ends meet in the middle (avoids massive amounts of futile backtracing)

Memory refinements: store only parts of the BWT and supplementary data structures, calculate the rest on the fly

Suffix tree vs Suffix array vs FM-index

	Suffix tree	Suffix array	FM Index
Time: Does P occur?	$O(n)$	$O(n \log m)$	$O(n)$
Time: Count k occurrences of P	$O(n + k)$	$O(n \log m)$	$O(n)$
Time: Report k locations of P	$O(n + k)$	$O(n \log m + k)$	$O(n + k)$
Space	$O(m)$	$O(m)$	$O(m)$
Needs T?	*yes*	*yes*	*no*
Bytes per input character	>15	~4	~0.5

$m = |T|, n = |P|, k =$ # occurrences of P in T

	Single-end			Paired-end		
Program	Time (s)	Conf (%)	Err (%)	Time (s)	Conf (%)	Err (%)
bowtie-125	1966	88.0	0.07	1701	91.0	0.37
BWA-125	3021	93.0	0.05	3059	97.6	0.04
MAQ-125	17506	92.7	0.08	19388	96.3	0.02
SOAP2-125	555	91.5	0.17	1187	90.8	0.14

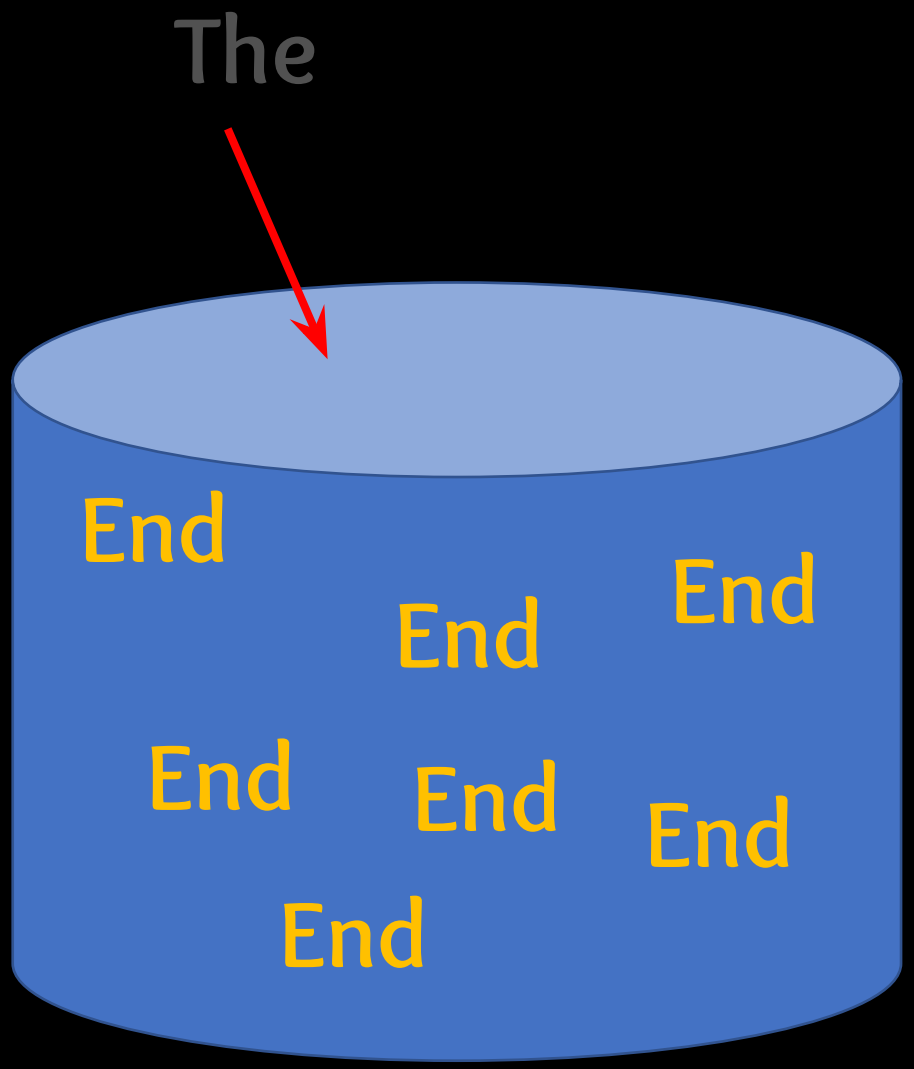
One million pairs of 32, 70 and 125 bp reads, respectively, were simulated from the human genome with 0.09% SNP mutation rate, 0.01% indel mutation rate and 2% uniform sequencing base error rate. The insert size of 32 bp reads is drawn from a normal distribution $N(170,25)$, and of 70 and 125 bp reads from $N(500,50)$. CPU time in seconds on a single core of a 2.5 GHz Xeon E5420 processor (Time), percent confidently mapped reads (Conf) and percent erroneous alignments out of confident mappings (Err) are shown in the table.

SOAP2: VERY approximately 1000x faster than BLAST

Table 2. Evaluation on real data

Program	Time (h)	Conf (%)	Paired (%)
Bowtie	5.2	84.4	96.3
BWA	4.0	88.9	98.8
MAQ	94.9	86.1	98.7
SOAP2	3.4	88.3	97.5

The 12.2 million read pairs were mapped to the human genome. CPU time in hours on a single core of a 2.5 GHz Xeon E5420 processor (Time), percent confidently mapped reads (Conf) and percent confident mappings with the mates mapped in the correct orientation and within 300 bp (Paired), are shown in the table.


The
End
End
End
End
End
End
End