What's Behind BLAST

Gene Myers, Director

MPI for Cell Biology and Genetics

Dresden, DE

Approximate String Search

Given a string A of length n, a query Q of length $p \ll n$, an alignment scoring function δ, and a threshold d:

Find all substrings of A, say M, s.t. $\delta(Q,M) \leq d$?

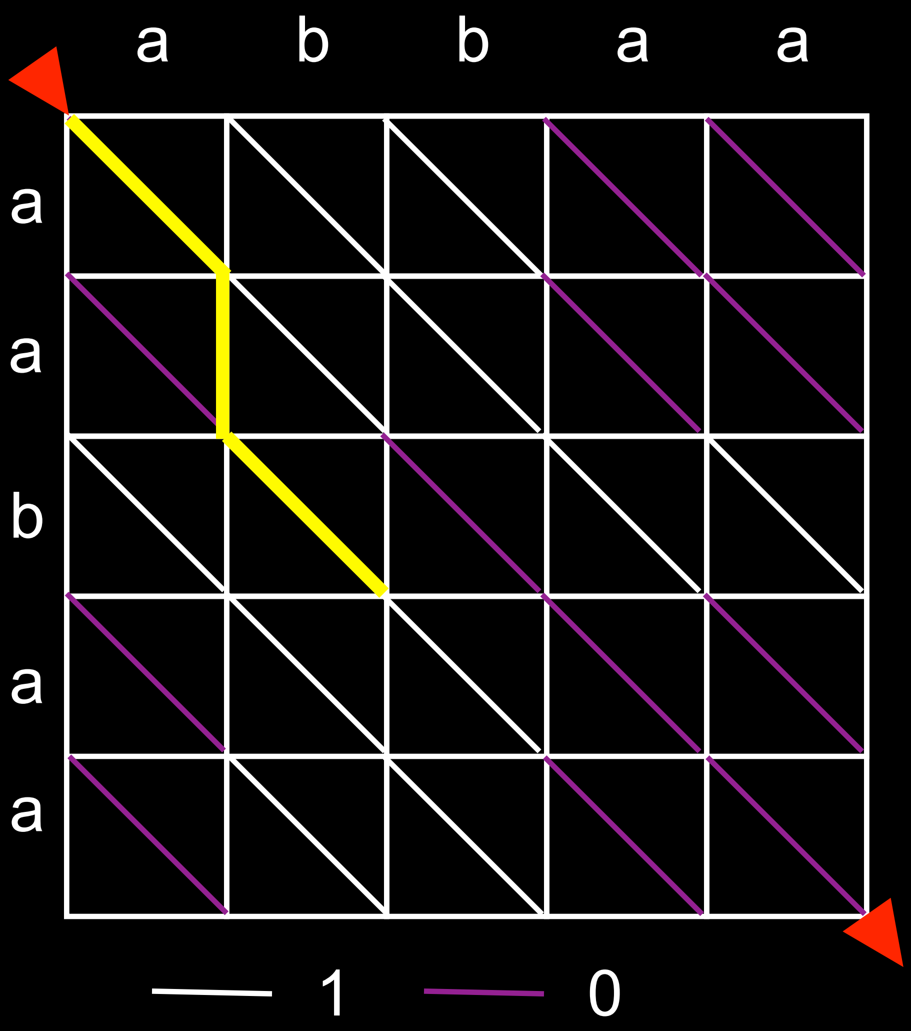
δ here = Simple Levenstein
(unit cost mismatch, insert, & delete)

```
...xxxxxxxxaacgt-gcattacxxxxxxxx...
           aatgtggc-ttac
```

A 3-match (absolute)

A 25%-match (relative)

Edit Graphs

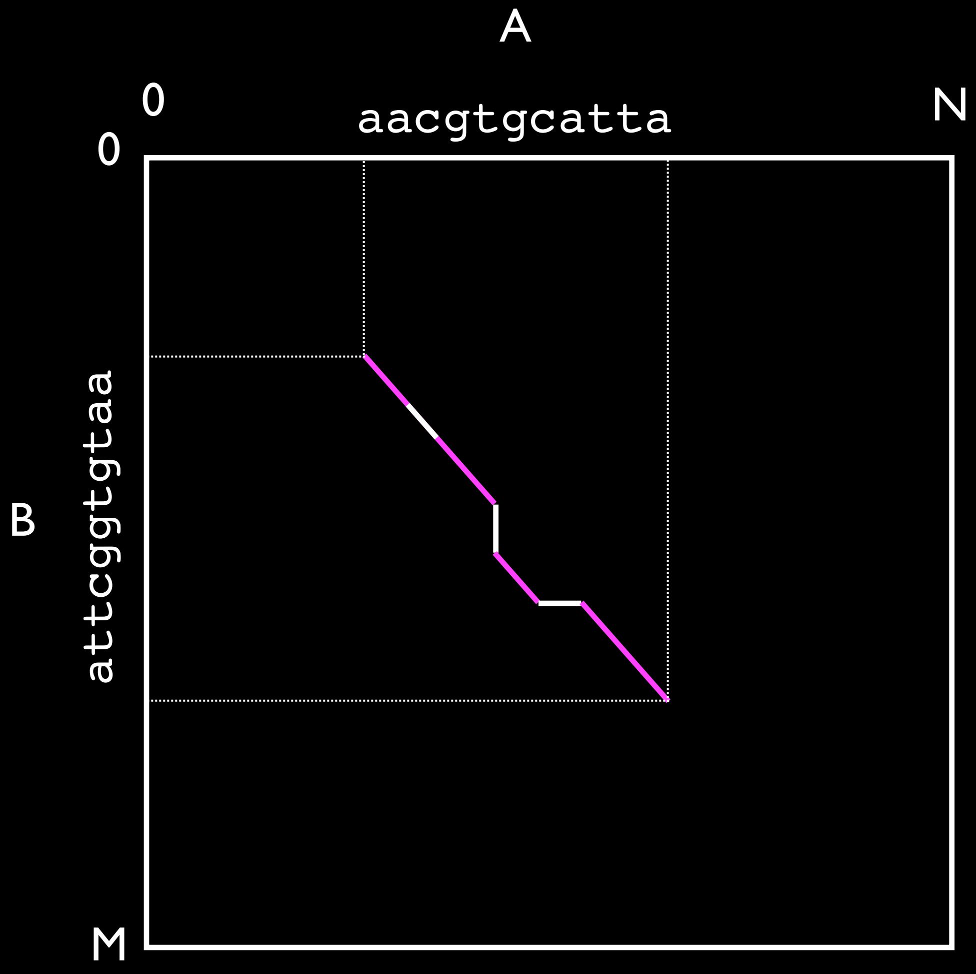


a - b
a a b

Dynamic Programming Matrix

		a	b	b	a	a
	0	1	2	3	4	5
a	1	0	1	2	3	4
a	2	1	1	2	2	3
b	3	2	1	1	2	3
a	4	3	2	2	1	2
a	5	4	3	3	2	1

Alignments



Corresponds to a path in the edit graph of the two sequences.

```
aacgtg-catta
aatgtggc-tta
```

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions

The Beginning

“Workshop for Algorithms in Molecular Genetics”

March 26-28, 1988

S. Altschul
W. Goad
G. Landau
J. Maisel
T. Smith
M. Zuker
D. Sankoff
W. Miller

W. Fitch
T. Hunkapillar
E. Lander
H. Martinez
R. Staden
A. Mukherjee
P. Sellers
G. Myers

Z. Galil
S. Karlin
D. Lipman
C. Sanders
J. Turner
M. Waterman
E. Ukkonen

Galil’s 2 Questions

“Workshop for Algorithms in Molecular Genetics”

March 26-28, 1988

Zvi gave a talk about suffix trees:

Q1: Can one get rid of the annoying dependence on alphabet size Σ?

⇒ Manber & Myers, “Suffix Arrays” 1990

Q2: Can one use an index to get faster approximate search?

Suffix Arrays

- Given subject string of size n over alphabet of size Σ, build an "index" that determines if a query string of length p occurs in the subject efficiently

  Suffix tree: Index is O(nΣ) space, then O(p) time

  Index is O(n) space, then O(plogΣ) time

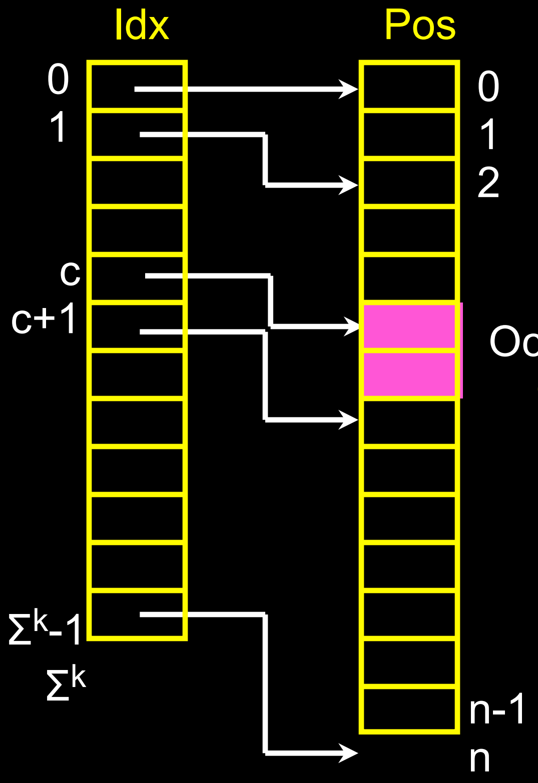
- Galil: Remove annoying dependence on Σ.
- Manber & Myers:

  Suffix array: Index is O(n) space, O(p + logn/logΣ) time.

- Galil says we misunderstood his challenge, sigh. But suffix trees enabled Burroughs-Wheeler Transform that are a sparse index commonly in use today for NGS.

A Simple Index

$\Phi($ "cacgt" $) = 10123_4 = 283_{(10)} \in [0, \Sigma^k\text{-}1]$ for any fixed k-mer size.



Scan 1: Count how big each set Occ(c) will be (in Idx[c+1]), then set Idx[c] += Idx[c-1] to point to proper Pos index.

Scan 2: Fill in each set using Idx[c] as a finger to place the next position, then readjust indices (Idx[c] = Idx[c-1]).

Occurences of k-mers with code c,

$\text{Occ}(c) = \{ p : \Phi(A[p..p+k-1]) = c \}$
$= \{ \text{Pos}[j] : j \in [\text{Idx}[c],\text{Idx}[c+1]-1] \}$

Performance:

- $O(n+ \Sigma^k)$ time and exactly $n+ \Sigma^k$ integers.
- If choose $k \sim \log_\Sigma n$ then $O(n)$.
- $O(p+h)$ expected-time to find any string of length p with h hits.

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions

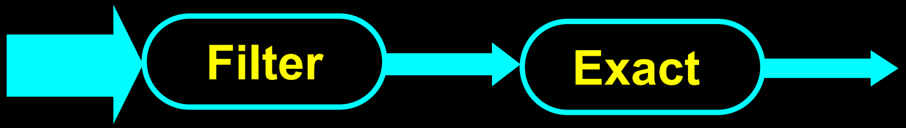
The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend

APM Filters

A filter is an algorithm that eliminates a lot of that which isn't desired.

	100% Sn	< 100% Sn
100% Sp	Exact	
< 100% Sp	Filter	Heuristic



If fast & specific then can improve speed of an exact algorithm.

Approximate match filter ideas:

- Look for exact matches to k-mers of the query (in an index)
  (Pearson & Lipman FASTA, Chang & Lawler, O(dn/lg p))

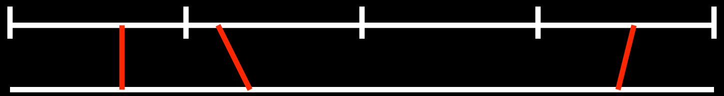
- Instead look for k-mers that are a small distance away, e.g. 1 or 2 diff's, from a k-mer of the query, i.e. the neighborhood

$$\mathcal{N}_d(w) = \{\ v : v \text{ and } w \text{ are} \leq d \text{ differences apart }\}$$

$$|\mathcal{N}_d(k)| \lesssim \binom{k}{d}(2\Sigma)^d$$

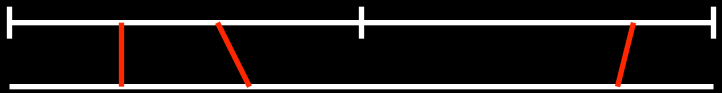
The Power of Neighborhoods

Consider looking for a 9%-match of 40 symbols (⇒ ≤ 3 differences or 3-match):



If divide "query" into 4 10-mers then at least one must match exactly:

⇒ Get a hit every Σ^{10} / 4 symbols (e.g. $2.5 \cdot 10^5$ for DNA)



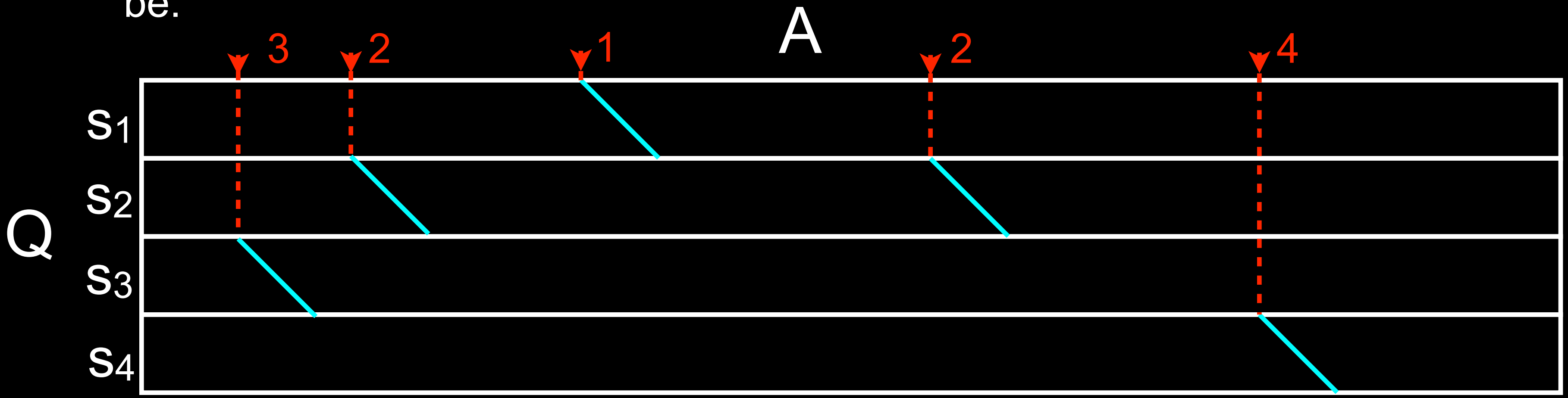
If divide into 2 20-mers then at least one of the $\mathcal{N}_1$ strings must match exactly:

⇒ Get a hit every $\Sigma^{20} / 2\mathcal{N}_1(20)$ symbols (e.g. $10^{12} / 2 \cdot 160 = 3.12 \cdot 10^9$ for DNA)

10,000 times more specific ! (but 80x more lookups)

"Seed & Extend"

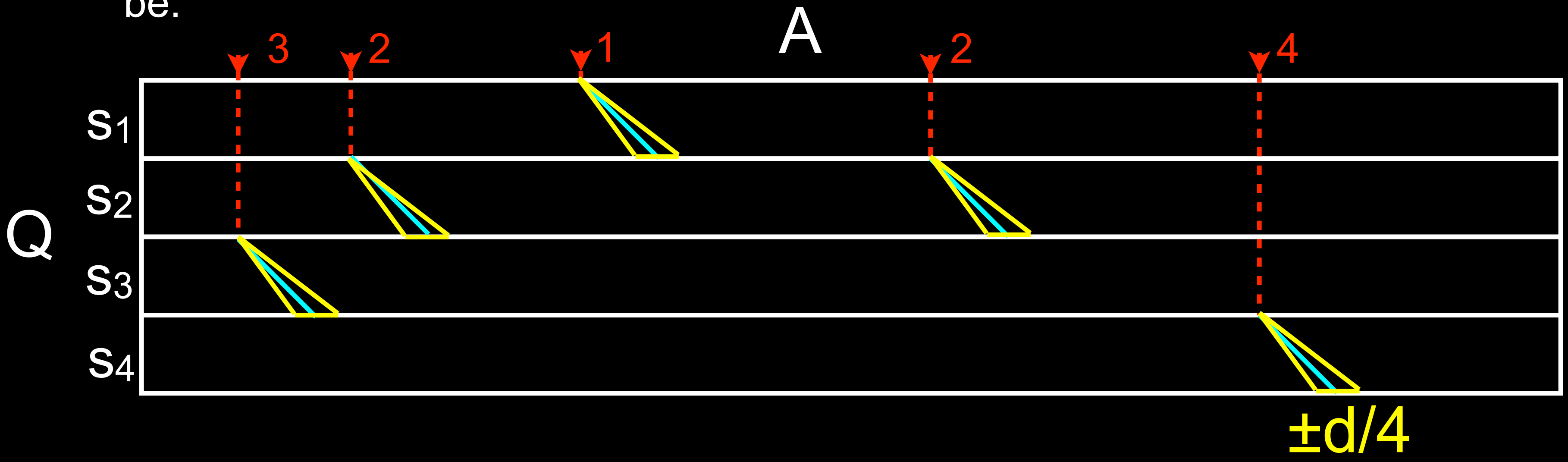
The "seed" matches (either exact or from a neighborhood) are in effect defining areas within the edit graph of Q vs A where the alignment of an ε-match could be:



$Q = s_1 s_2 s_3 s_4$

“Seed & Extend”

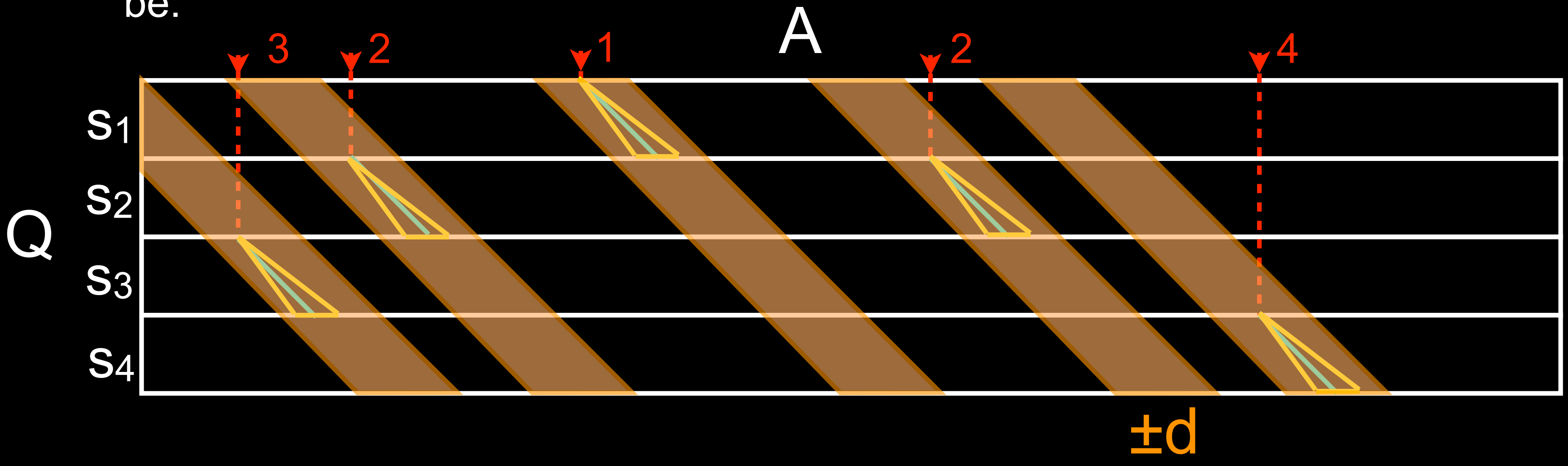
The “seed” matches (either exact or from a neighborhood) are in effect defining areas within the edit graph of Q vs A where the alignment of an ε-match could be:



$Q = s_1 s_2 s_3 s_4$

“Seed & Extend”

The “seed” matches (either exact or from a neighborhood) are in effect defining areas within the edit graph of Q vs A where the alignment of an ε-match could be:



Spend O(pdh + pz) time where

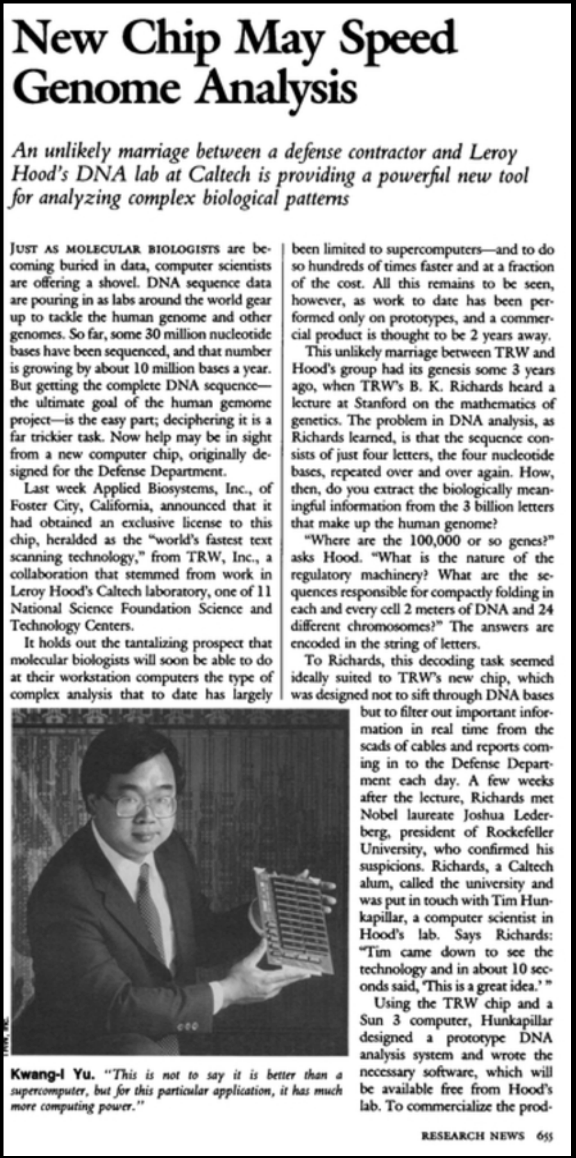
h(k) = the number of seed “k-hits” vs. z(k) = neighborhood size “k-words”

Both z and h are functions of k and the optimal k is “slightly bigger” than $\log_\Sigma n$

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break

The 1st Conversation



New Chip May Speed Genome Analysis

An unlikely marriage between a defense contractor and Leroy Hood's DNA lab at Caltech is providing a powerful new tool for analyzing complex biological patterns

Just as molecular biologists are becoming buried in data, computer scientists are offering a shovel. DNA sequence data are pouring in as labs around the world gear up to tackle the human genome and other genomes. So far, some 30 million nucleotide bases have been sequenced, and that number is growing by about 10 million bases a year. But getting the complete DNA sequence—the ultimate goal of the human gemome project—is the easy part; deciphering it is a far trickier task. Now help may be in sight from a new computer chip, originally designed for the Defense Department.

Last week Applied Biosystems, Inc., of Foster City, California, announced that it had obtained an exclusive license to this chip, heralded as the "world's fastest text scanning technology," from TRW, Inc., a collaboration that stemmed from work in Leroy Hood's Caltech laboratory, one of 11 National Science Foundation Science and Technology Centers.

It holds out the tantalizing prospect that molecular biologists will soon be able to do at their workstation computers the type of complex analysis that to date has largely been limited to supercomputers—and to do so hundreds of times faster and at a fraction of the cost. All this remains to be seen, however, as work to date has been performed only on prototypes, and a commercial product is thought to be 2 years away.

This unlikely marriage between TRW and Hood's group had its genesis some 3 years ago, when TRW's B. K. Richards heard a lecture at Stanford on the mathematics of genetics. The problem in DNA analysis, as Richards learned, is that the sequence consists of just four letters, the four nucleotide bases, repeated over and over again. How, then, do you extract the biologically meaningful information from the 3 billion letters that make up the human genome?

"Where are the 100,000 or so genes?" asks Hood. "What is the nature of the regulatory machinery? What are the sequences responsible for compactly folding in each and every cell 2 meters of DNA and 24 different chromosomes?" The answers are encoded in the string of letters.

To Richards, this decoding task seemed ideally suited to TRW's new chip, which was designed not to sift through DNA bases but to filter out important information in real time from the scads of cables and reports coming in to the Defense Department each day. A few weeks after the lecture, Richards met Nobel laureate Joshua Lederberg, president of Rockefeller University, who confirmed his suspicions. Richards, a Caltech alum, called the university and was put in touch with Tim Hunkapillar, a computer scientist in Hood's lab. Says Richards: "Tim came down to see the technology and in about 10 seconds said, 'This is a great idea.'"

Using the TRW chip and a Sun 3 computer, Hunkapillar designed a prototype DNA analysis system and wrote the necessary software, which will be available free from Hood's lab. To commercialize the prod-

TRW, Inc.

Kwang-I Yu. *"This is not to say it is better than a supercomputer, but for this particular application, it has much more computing power."*

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born

Blast = Seed & Extend

Seeds are neighborhoods of all k-mers of query
under weighted Levenstein (e.g. PAM120)

Find seeds with a deterministic finite automaton
accepting all neighborhood words (⇒O(n))

Extend is just weighted Hamming
but stop when score drops too much

A heuristic

"blast" was inspired by
"slam" = sublinear approximate match

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma

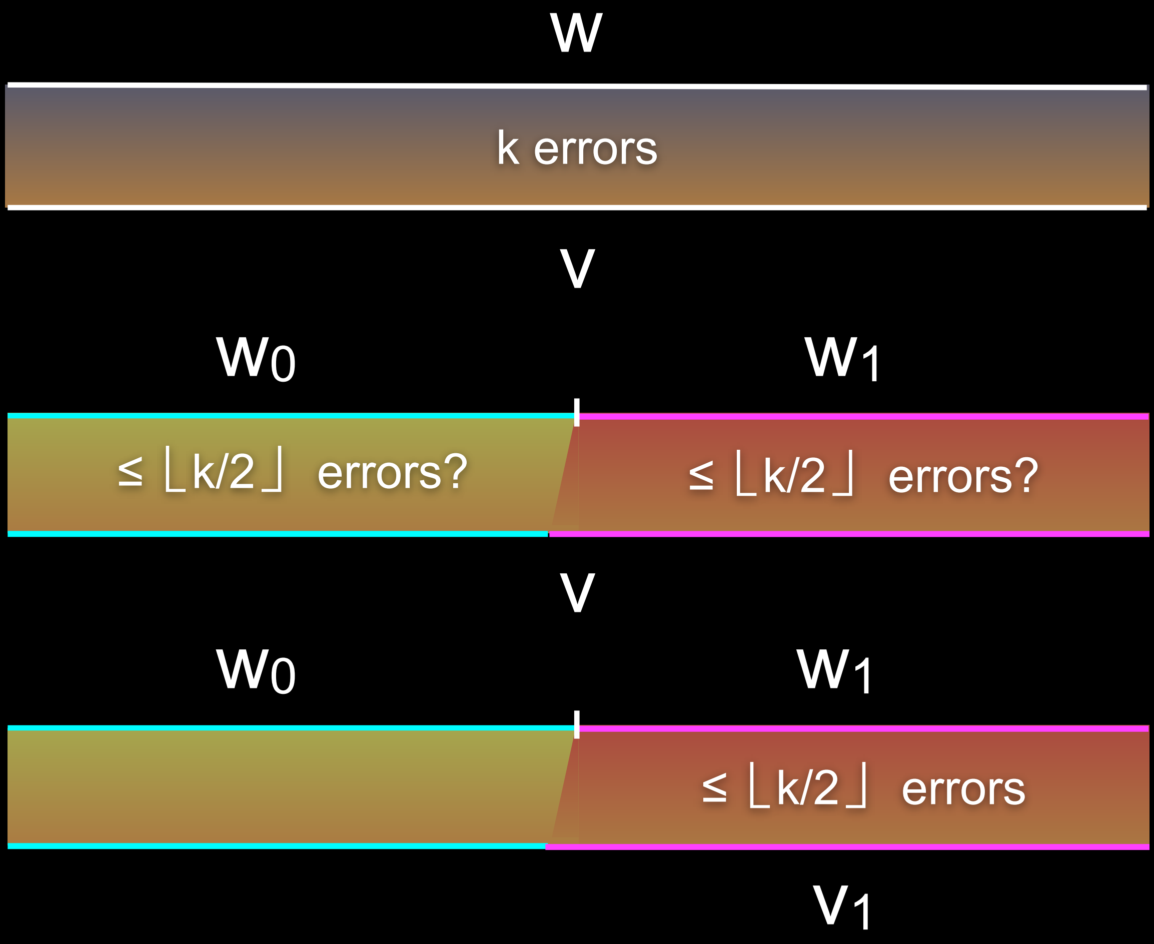
The Splitting Lemma

Lemma: If w ε-matches v then either

(a) w_0 has an ε-match to a prefix (call it v_0) of v, or

(b) w_1 has an ε-match to a suffix (call it v_1) of v.

Proof:



w

k errors

v

w_0 w_1

$\leq \lfloor k/2 \rfloor$ errors? $\leq \lfloor k/2 \rfloor$ errors?

v

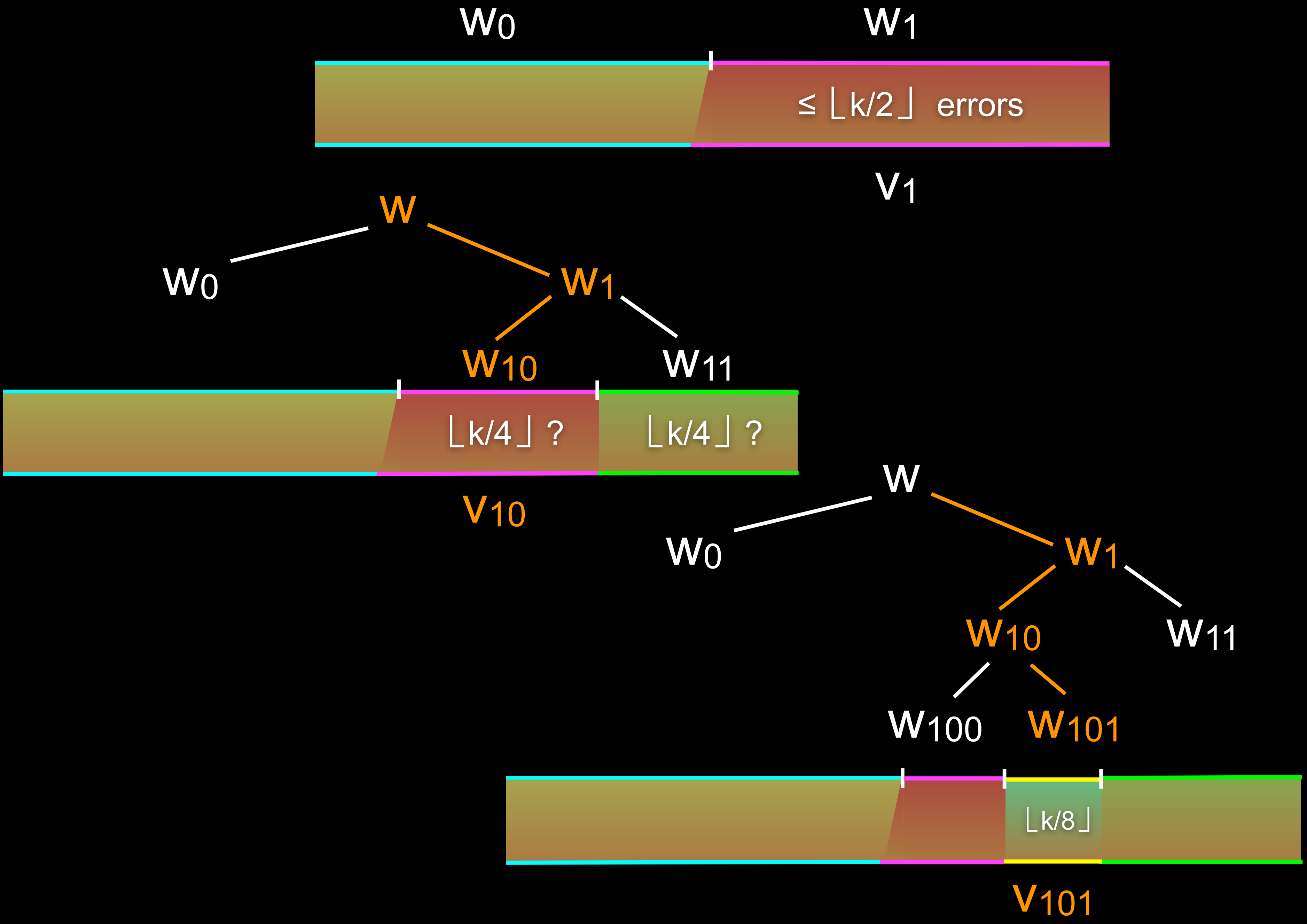
w_0 w_1

$\leq \lfloor k/2 \rfloor$ errors

v_1

$k = \lfloor \varepsilon|w| \rfloor$

By Pigeon Hole Principle

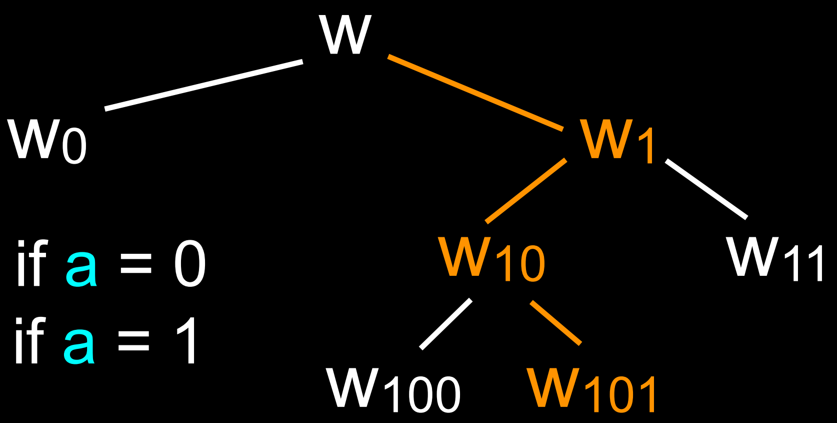

w_0
w_1
≤ ⌊k/2⌋ errors
v_1
w
w_0
w_1
w_{10}
w_{11}
⌊k/4⌋ ?
⌊k/4⌋ ?
v_{10}
w
w_0
w_1
w_{10}
w_{11}
w_{100}
w_{101}
⌊k/8⌋
v_{101}

The Splitting Lemma

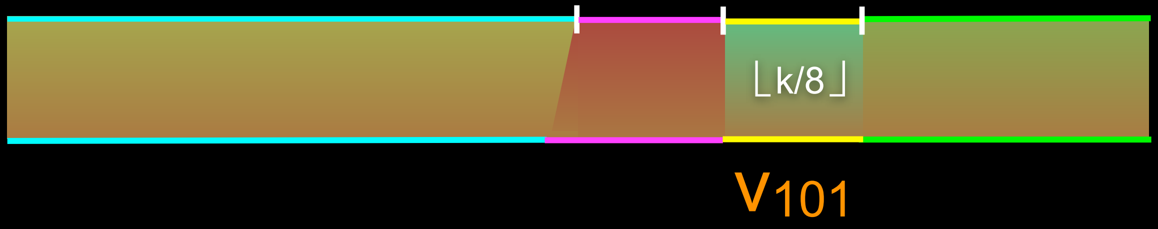
Let $w_\varepsilon = w$

$w_{\beta a} = w_\beta[1..|w_\beta|/2]$ if $a = 0$

$w_\beta[|w_\beta|/2+1..\,|w_\beta|]$ if $a = 1$



e.g. α=101...



Lemma: If w ε-matches v then ∃ α s.t. ∀ prefixes β of α,

(1) w_β has an ε-match to a substring (call it v_β) of v, and

(2) $v_{\beta 0}$ is a prefix of v_β (if $\beta 0$ is a prefix of α), and

(3) $v_{\beta 1}$ is a suffix of v_β (if $\beta 1$ is a prefix of α).

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ’88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma

The Story

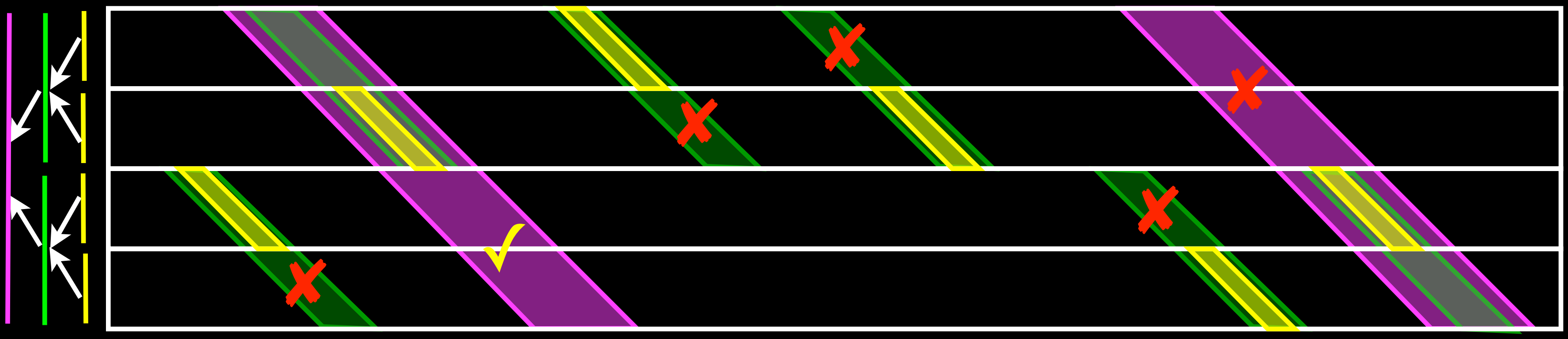
- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling

Doubling Extension

Use $\log_\Sigma n$ as the seed size !

Lemma: Any ε-match of Q has an ε-match to at least one seed segment of size $\log_\Sigma n$

Use the splitting lemma to split Q to seeds of size $\log_\Sigma n$, and instead of extending all at once, extend by doubling using the splitting lemma.



Time for each extension telescopes hyper-geometrically and so is dominated by the first term:

$O(p/\log_\Sigma n \cdot h \cdot \log_\Sigma n \cdot \varepsilon \log_\Sigma n) = O(dh\log_\Sigma n)$

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling
- Spr ’90: Generating Condensed Neighborhoods

Generating (Condensed) Neighborhoods

$\overline{\mathcal{N}}_d(w)$ = { v : v and w are ≤ d differences apart and
v is not a proper prefix of another word in $N_d(w)$ }

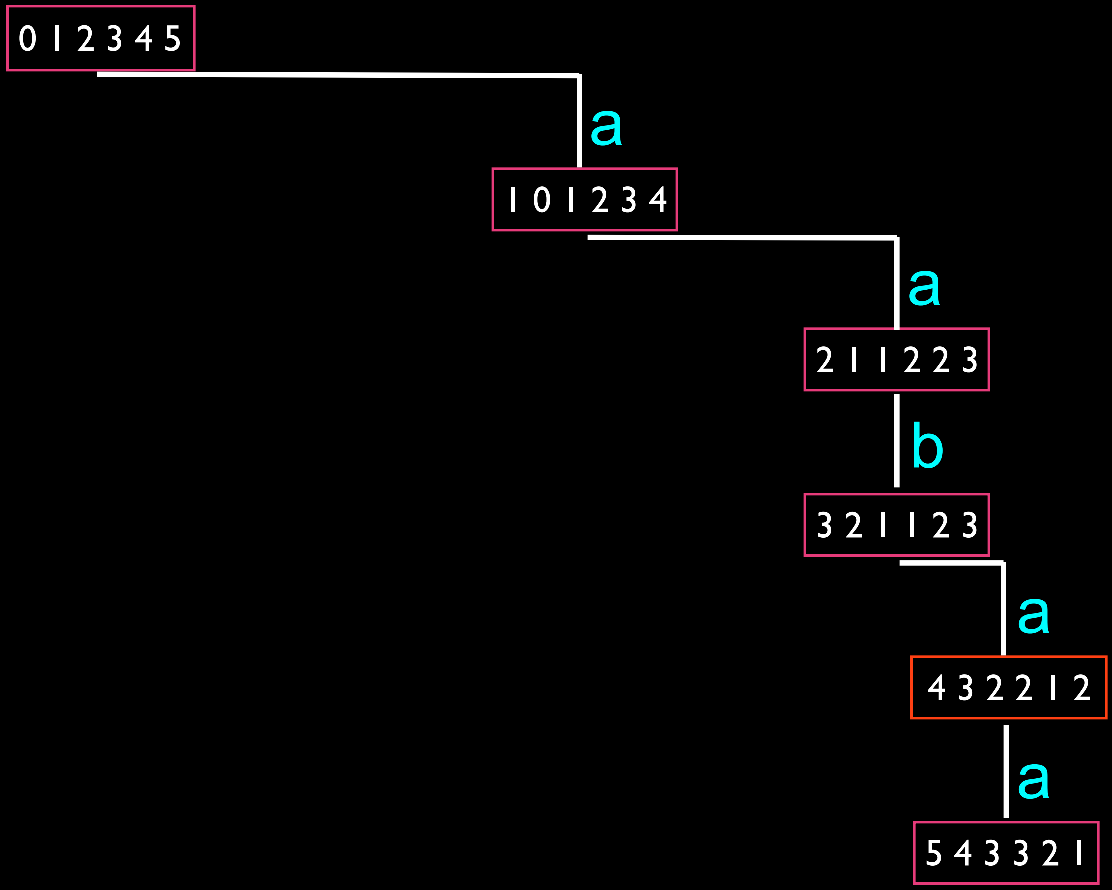
$\overline{\mathcal{N}}_1$(abbaa) = { aabaa, aabbaa, abaa, ~~abaaa~~, ababaa,
abba, ~~abbaa~~, ~~abbab~~, ~~abbaaa~~, ~~abbaba~~,
abbba, ~~abbbaa~~, babbaa, bbaa, bbbaa }

It suffices to find the words in the condensed neighborhood.

But how do you do that efficiently, including finding them in the index? ...

... Compute rows of dynamic programming matrix as one traverses the trie of all strings over Σ

Condensed Neighborhoods



0 1 2 3 4 5

a

1 0 1 2 3 4

a

2 1 1 2 2 3

b

3 2 1 1 2 3

a

4 3 2 2 1 2

a

5 4 3 3 2 1

w

v?

		a	b	b	a	a
	0	1	2	3	4	5
a	1	0	1	2	3	4
a	2	1	1	2	2	3
b	3	2	1	1	2	3
a	4	3	2	2	1	2
a	5	4	3	3	2	1

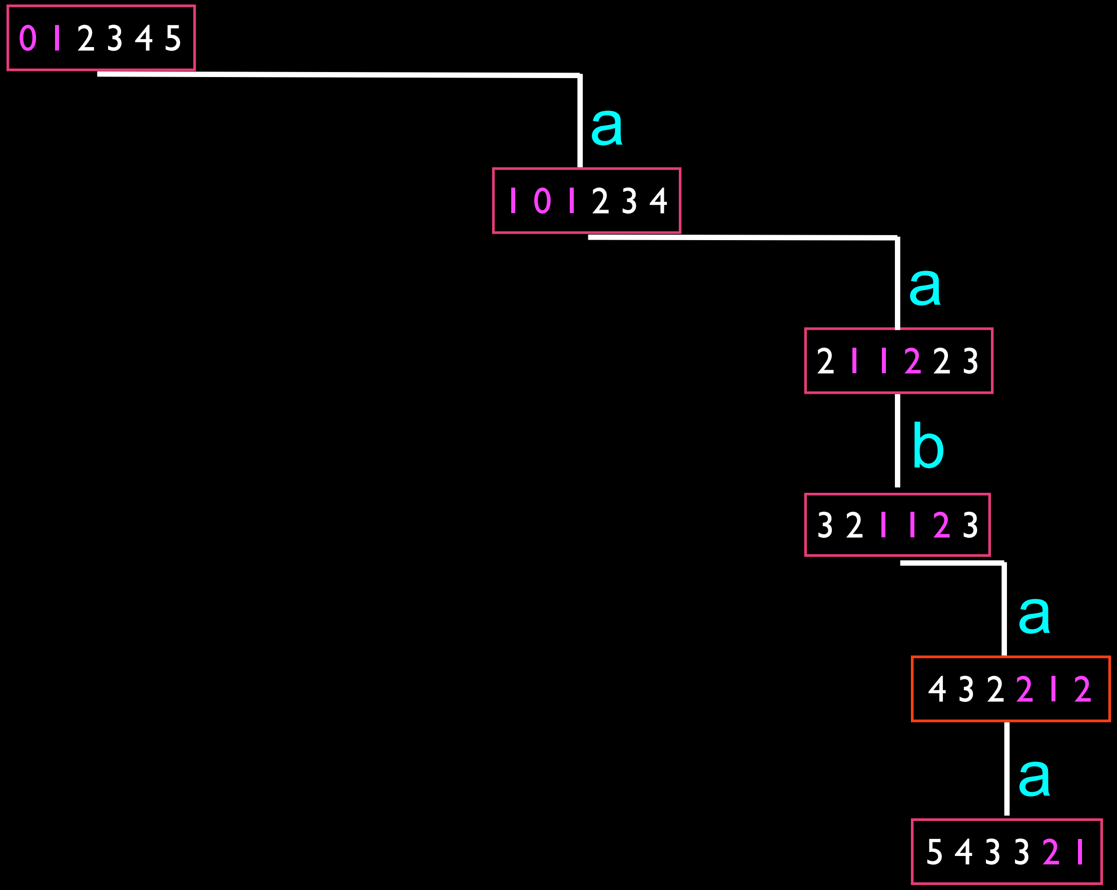
Done: a 1 in the right corner

Condensed Neighborhoods

w

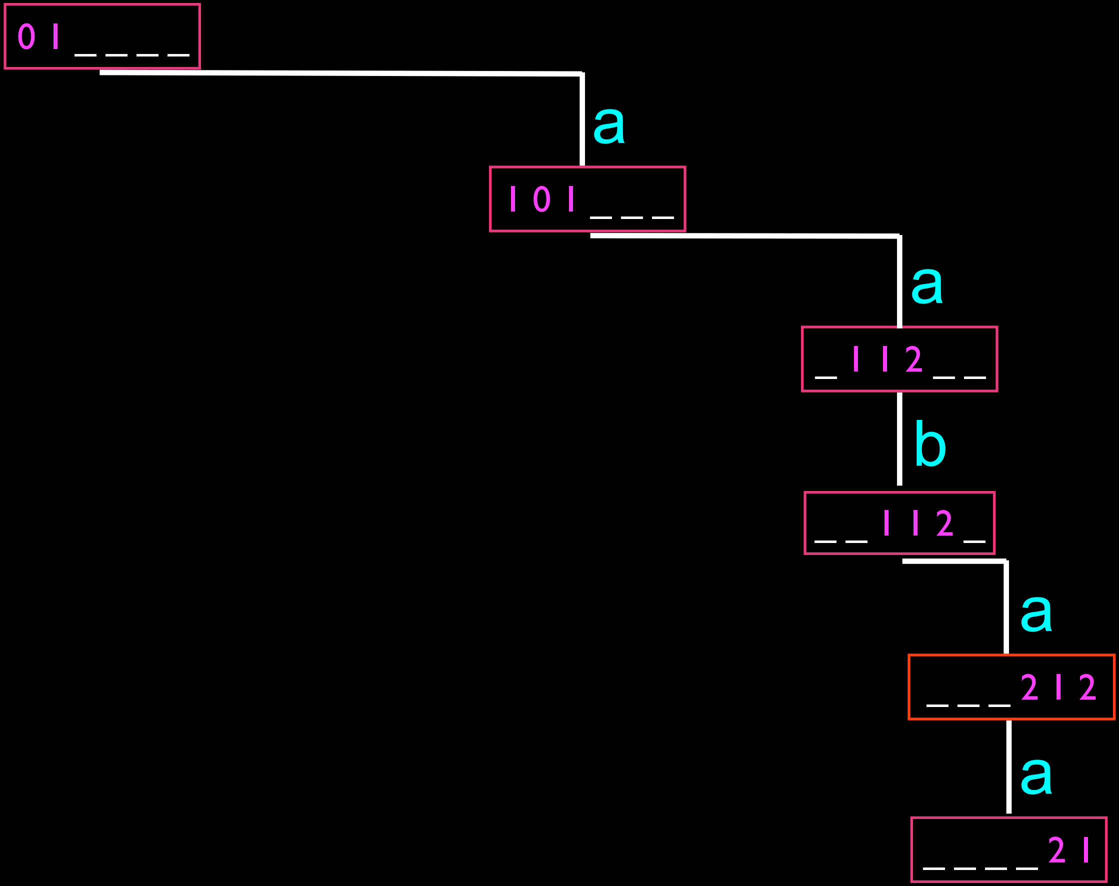
v

	a	b	b	a	a
	0	1	2	3	4	5
a	1	0	1	2	3	4
a	2	1	1	2	2	3
b	3	2	1	1	2	3
a	4	3	2	2	1	2
a	5	4	3	3	2	1



Only need ±1 band !

Condensed Neighborhoods

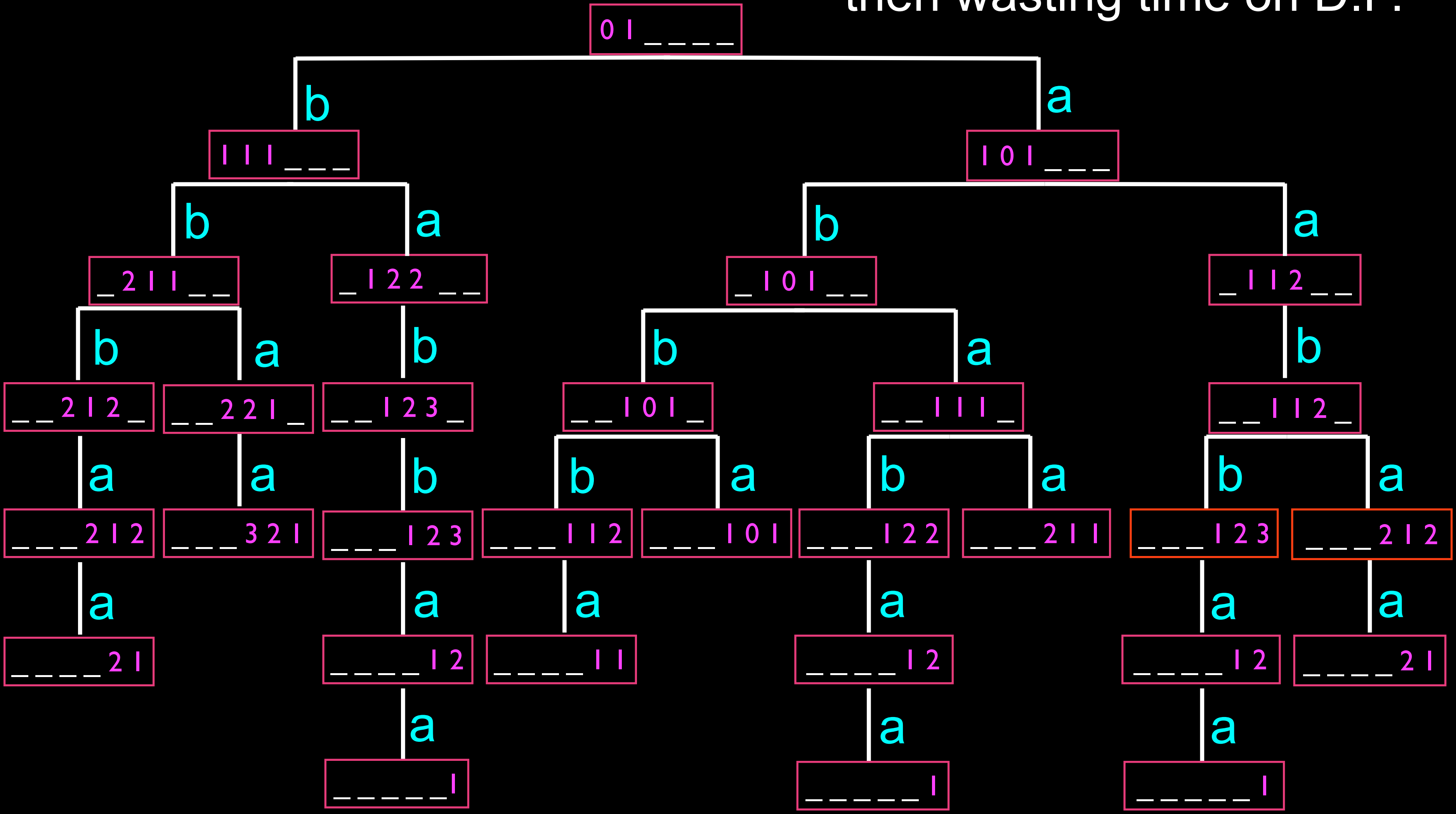


w

			a	b	b	a	a
		0	1				
	a	1	0	1			
	a		1	1	2		
v	b			1	1	2	
	a				2	1	2
	a					2	1

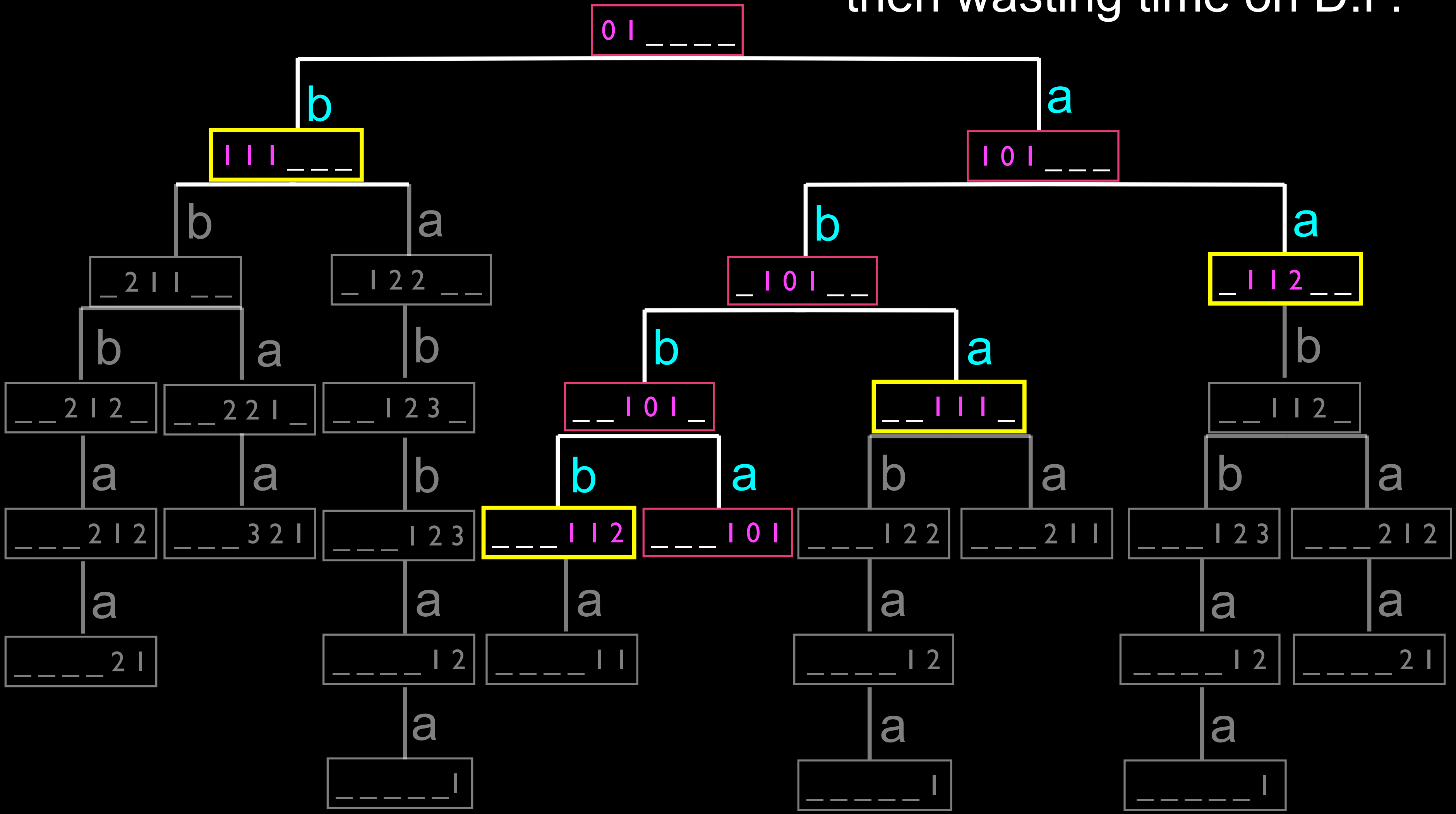
Condensed Neighborhoods

If all entries are ≥ d
then wasting time on D.P.

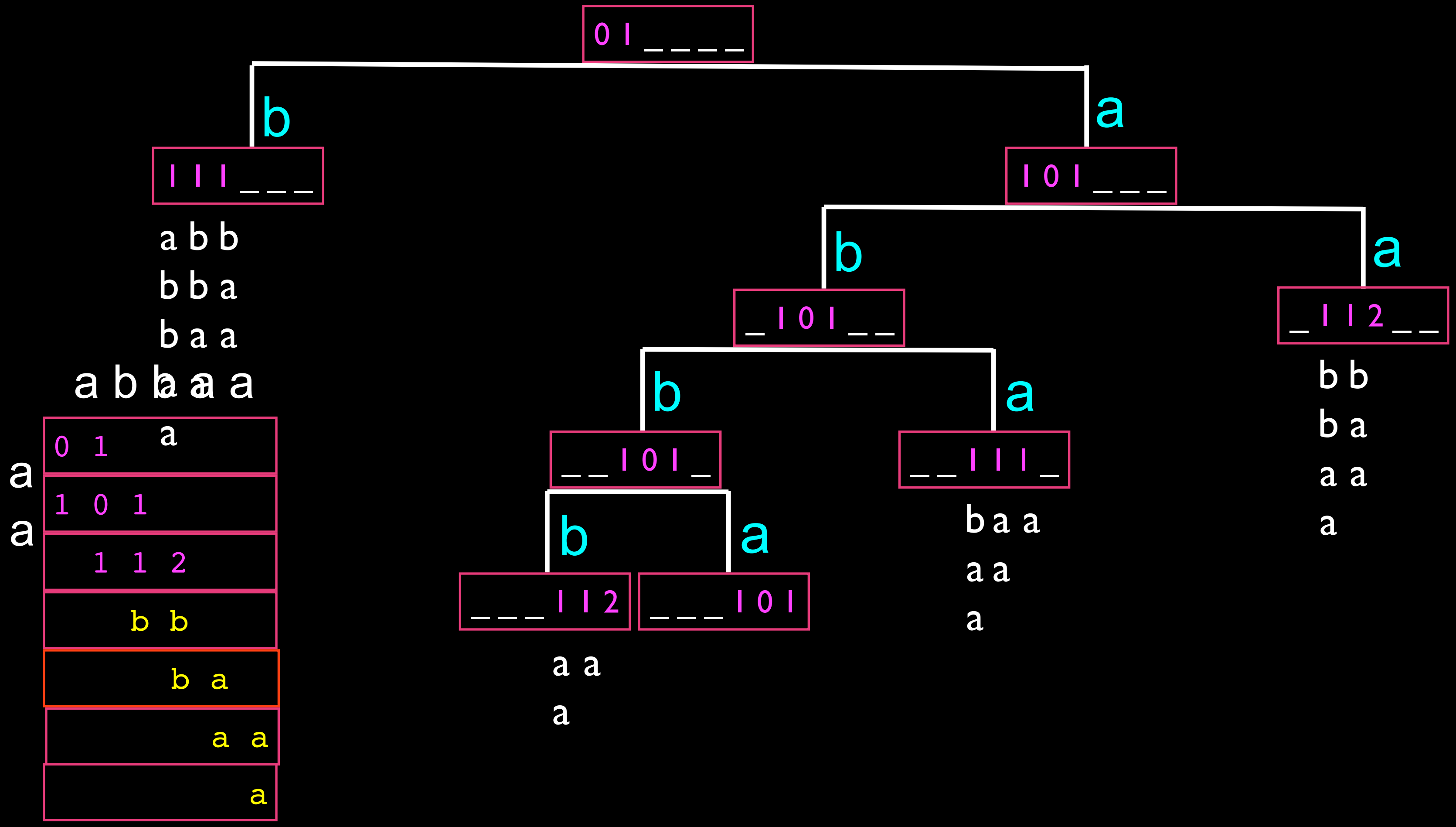


Condensed Neighborhoods

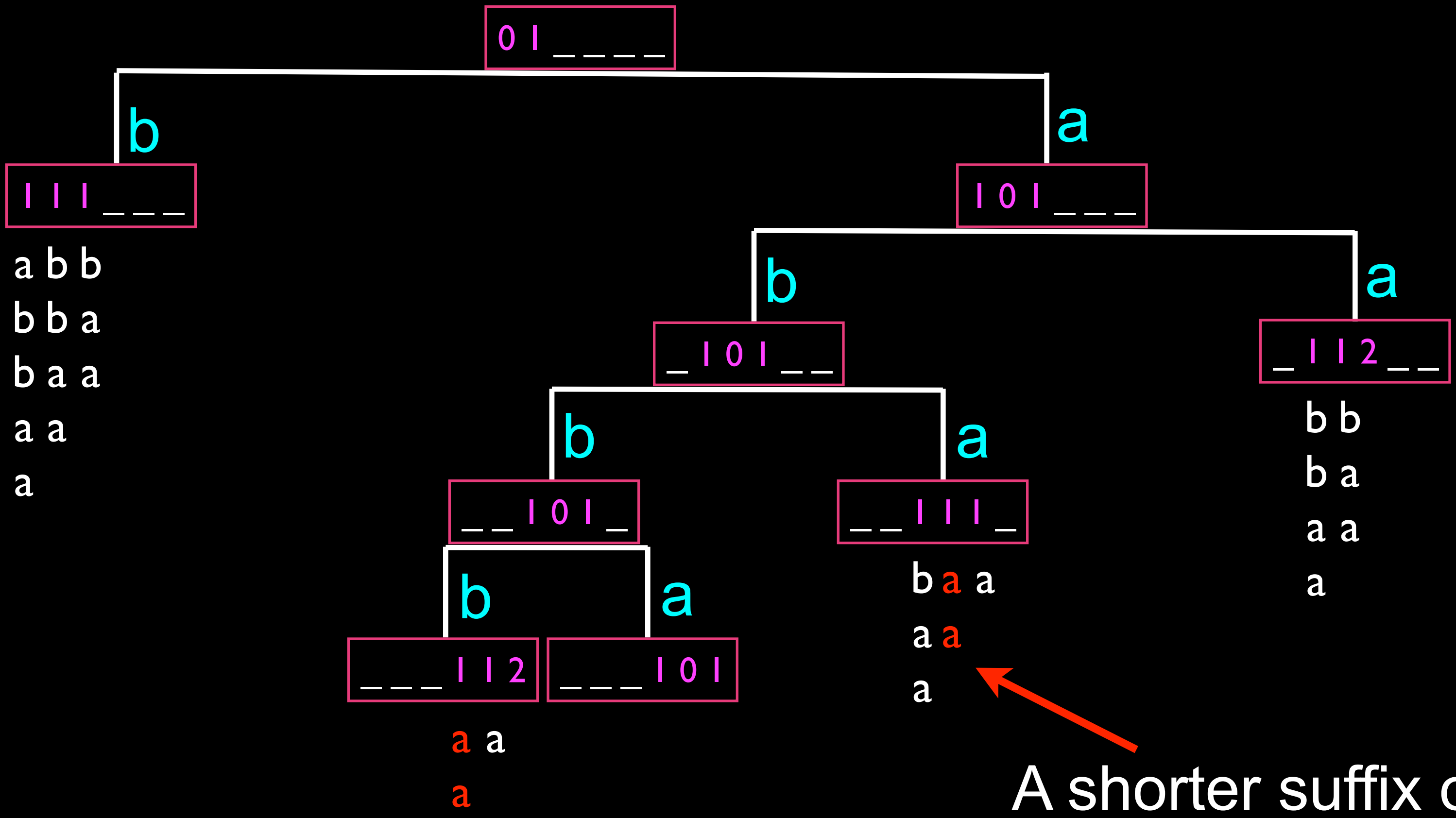
If all entries are ≥ d
then wasting time on D.P.



Condensed Neighborhoods



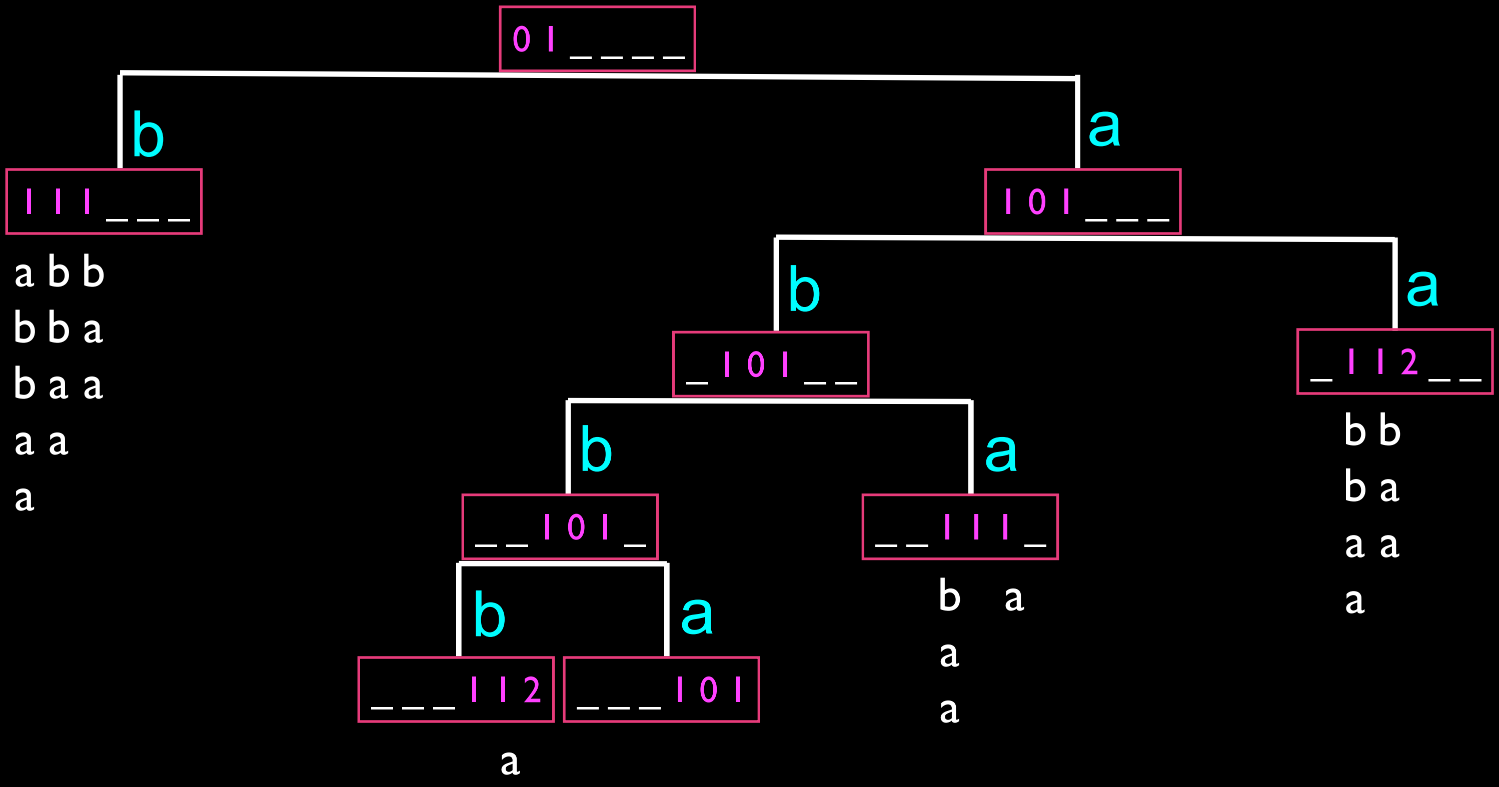
Condensed Neighborhoods



A shorter suffix of w that is a prefix of the extension is also possible

Use “KMP” on reverse of w to efficiently discover these.

Condensed Neighborhoods



Lemma: Neighborhoods and their hits in A can be generated in O(zd+h) time where z = $|\mathcal{N}_d(w)|$

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling
- Spr ’90: Generating Condensed Neighborhoods

The Story

- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling
- Spr ’90: Generating Condensed Neighborhoods
- Fall ’90: Finale: Complexity

Complexity

How big is $\overline{\mathcal{N}_d}(k)$?

Developed recurrence for non-redundant edit scripts:

(a) `DI = S`

(b) `DS = SD`

(c) `IS = SI`

(d) `ID` $= \Phi$

Lemma:

$$S(k,d) = S(k-1,d) + (\Sigma-1)S(k-1,d-1) + (\Sigma-1)\sum_{j=0}^{d-1}\Sigma^j S(k-1,d-1)$$
$$+ (\Sigma-1)^2\sum_{j=0}^{d-2}\Sigma^j S(k-2,d-2-j) + \sum_{j=0}^{d-1} S(k-2-j,d-1-j)$$

$$\overline{\mathcal{N}_d}(k) \leq S(k,d) + \sum_{j=1}^{d}\Sigma^j S(k-1,d-j)$$

Complexity

So how big is it?

Lemma:

$\overline{\mathcal{N}_\varepsilon}(k) \leq 1.708\ \alpha(\varepsilon)^k$

where $\alpha(\varepsilon) = \Sigma^{pow(\varepsilon)}$

and $pow(\varepsilon) = \log_\Sigma \left(\frac{c(\varepsilon)+1}{c(\varepsilon)-1}\right) + \varepsilon \log_\Sigma c(\varepsilon) + \varepsilon$

and $c(\varepsilon) = \varepsilon^{-1} + (1 + \varepsilon^{-2})^{.5}$

Also $Pr(w \text{ in } \overline{\mathcal{N}_\varepsilon}(k)) = O(\,1 / \beta(\varepsilon)^k\,)$

where $\beta(\varepsilon) = \Sigma^{1-pow(\varepsilon)}$

pow(ε)

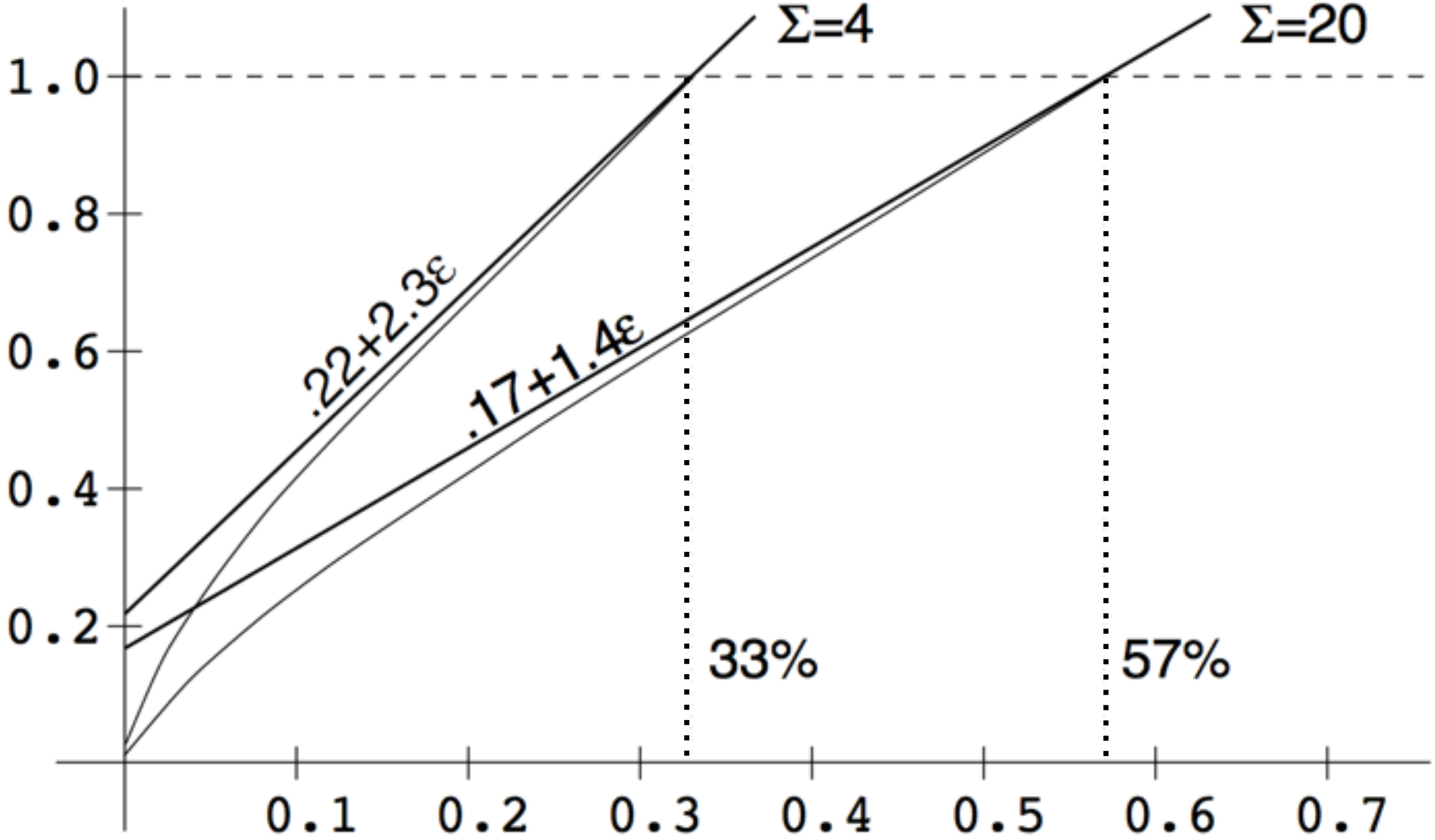


Figure 4: Plot of *pow*(ε) and Sample Bounding Lines

Complexity

So how big is it?

Lemma:

$\overline{\mathcal{N}}_{\varepsilon}(k) = O(\alpha^{k})$

where $\alpha = \Sigma^{pow(\varepsilon)}$ ← Starts at 1 (ε=0) and grows
“Flex factor”

$Pr(w \text{ in } \overline{\mathcal{N}}_{\varepsilon}(k)) = O(1 / \beta^{k})$

where $\beta = \Sigma^{1-pow(\varepsilon)}$ ← Starts at Σ (ε=0) and shrinks
“Effective alphabet size”

And when $k = \log_{\Sigma} n$?

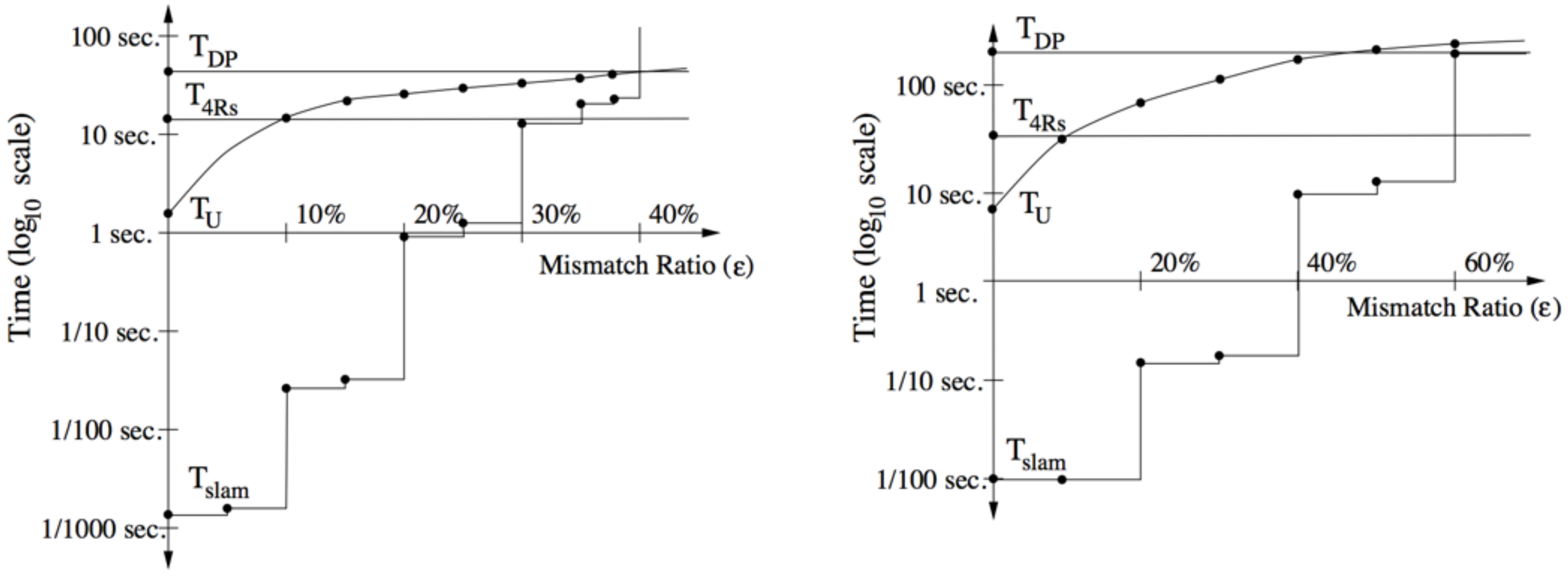
$\overline{\mathcal{N}}_{\varepsilon}(k) = O(n^{pow(\varepsilon)})$ and $Pr(w \text{ in } \overline{\mathcal{N}}_{\varepsilon}(k)) = O(n^{pow(\varepsilon)-1})$

The Result

Theorem: Given

(a) A is effectively Bernouilli,
(b) a simple $O(n)$ space, precomputed index of A, and
(c) there are h d-matches of a query Q to A

then they can be found in $O(d \cdot n^{pow(\varepsilon)} \cdot \log n + pd \cdot h)$ expected-time.



$N = 1{,}000{,}000$ and $|\Sigma| = 4$ $N = 4{,}000{,}000$ and $|\Sigma| = 20$

Figure 10: Timing Plots for Queries of Length $P = 80$

To my knowledge no one has improved on this in the last 20 years ! ?

Algorithmica 12, 4-5 (1994), 345-374

(submitted 1991 !)

A recent retrospective:

Computational Biology 19, (Springer-Verlag 2013), 3-15.