
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)

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?

The Story

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

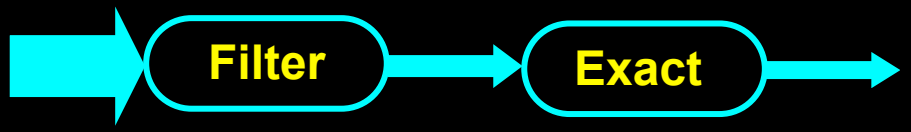
The Story

- 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:

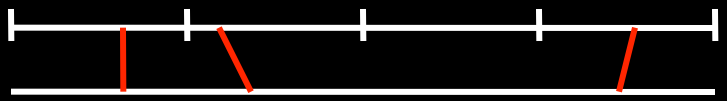
- Look for exact matches to k-mers of the query (in an index)
  (Pearson & Lipman FASTA, Chang & Lawler, O(dn/lg p))
- Instead look for k-mers that are a small distance away, e.g. 1 or 2 diff's, from a k-mer of the query, i.e. the neighborhood

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

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

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)

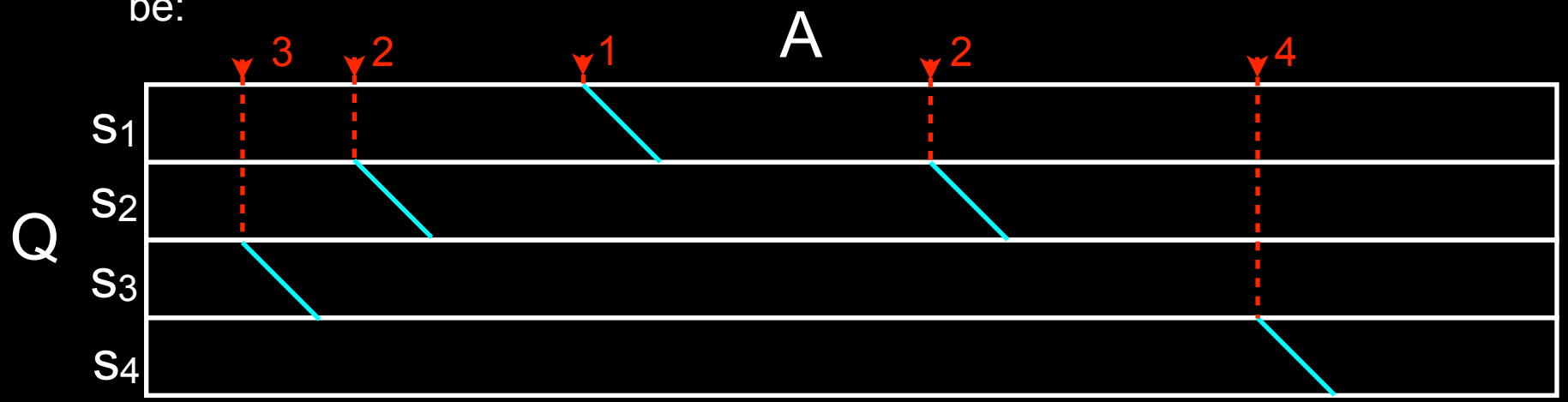
If divide into 2 20-mers then at least one of the $\mathcal{N}_1$ strings must match exactly:

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

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

“Seed & Extend”

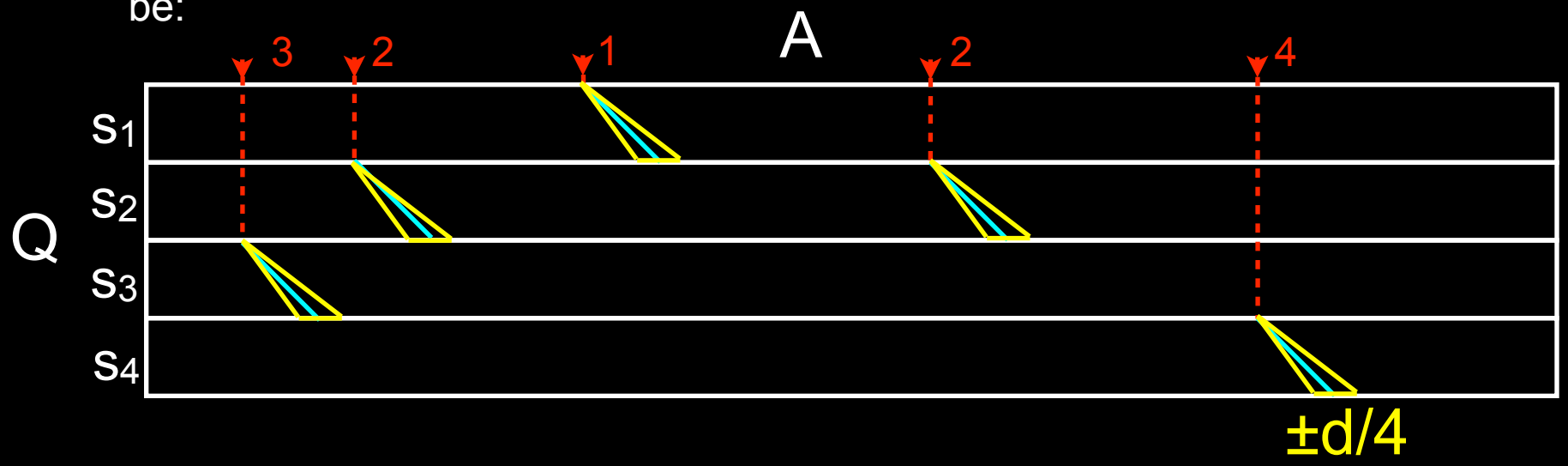
The “seed” matches (either exact or from a neighborhood) are in effect defining areas within the edit graph of Q vs A where the alignment of an ε-match could be:



$Q = s_1 s_2 s_3 s_4$

“Seed & Extend”

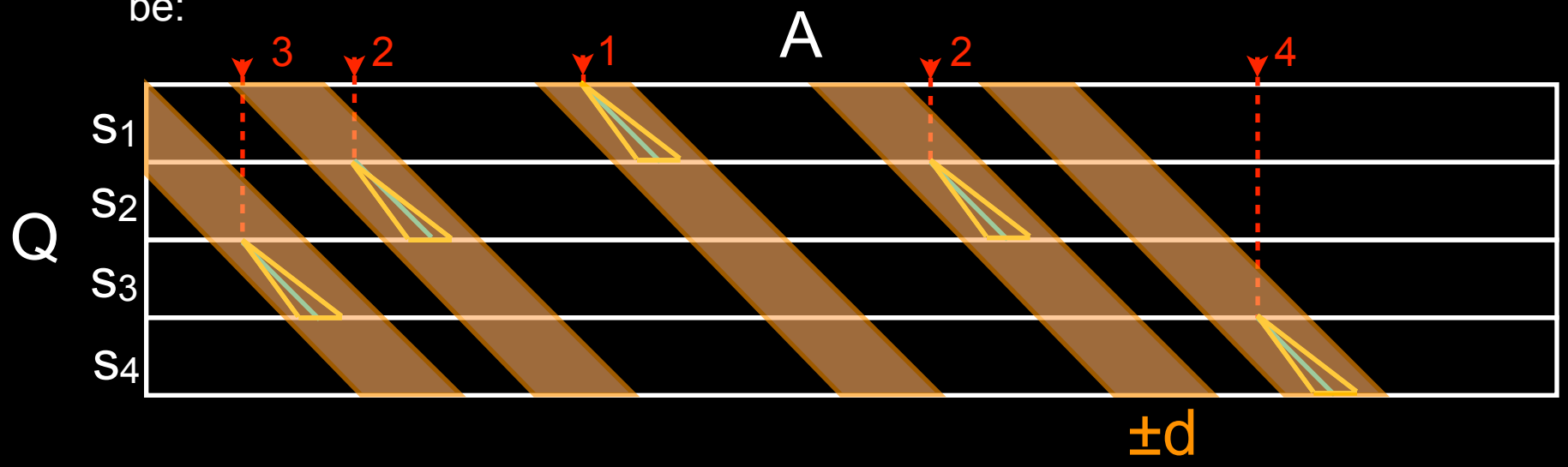
The “seed” matches (either exact or from a neighborhood) are in effect defining areas within the edit graph of Q vs A where the alignment of an ε-match could be:



$Q = s_1 s_2 s_3 s_4$

“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

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

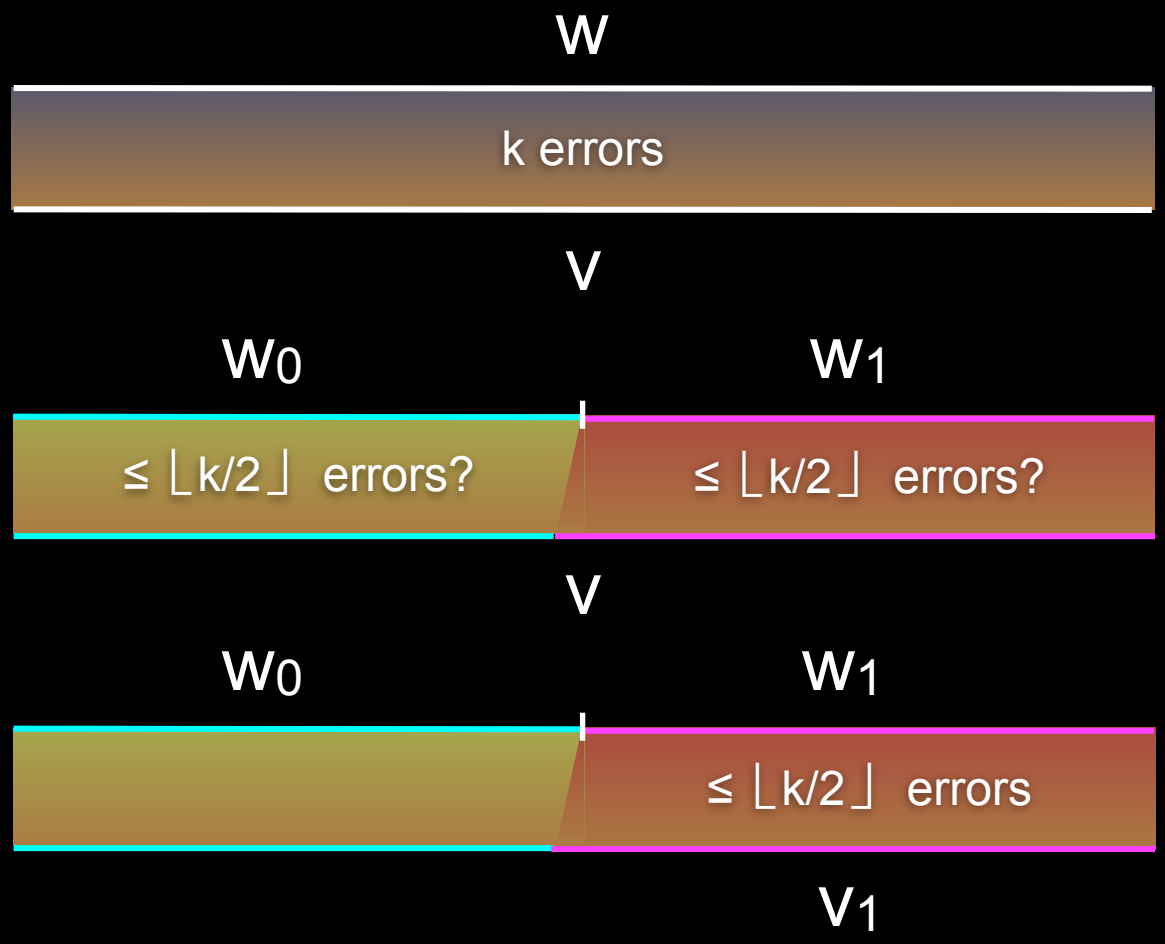
The Splitting Lemma

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

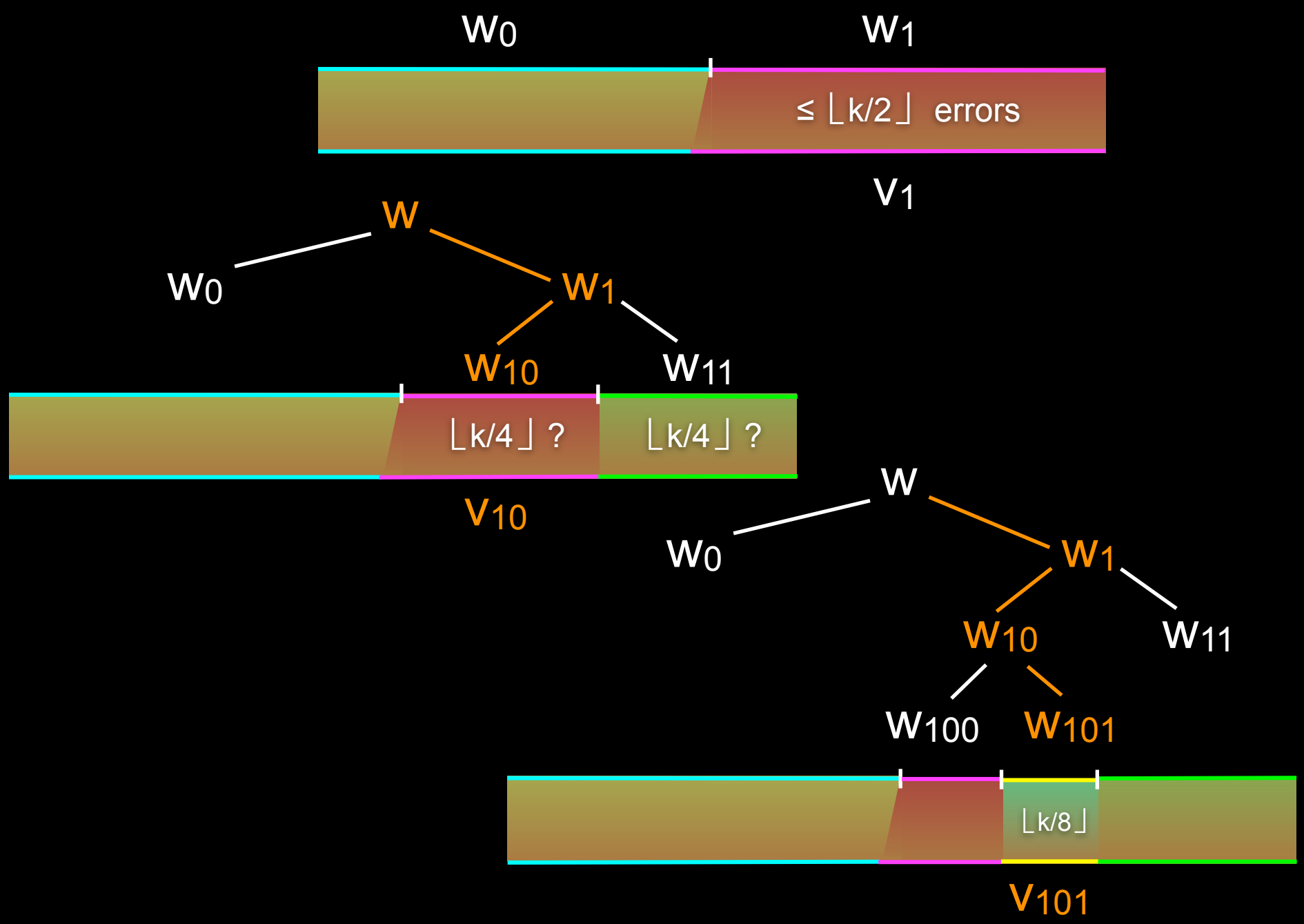
(b) w_1 has an ε-match to a suffix (call it v_1) of v.

Proof:



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

By Pigeon Hole Principle

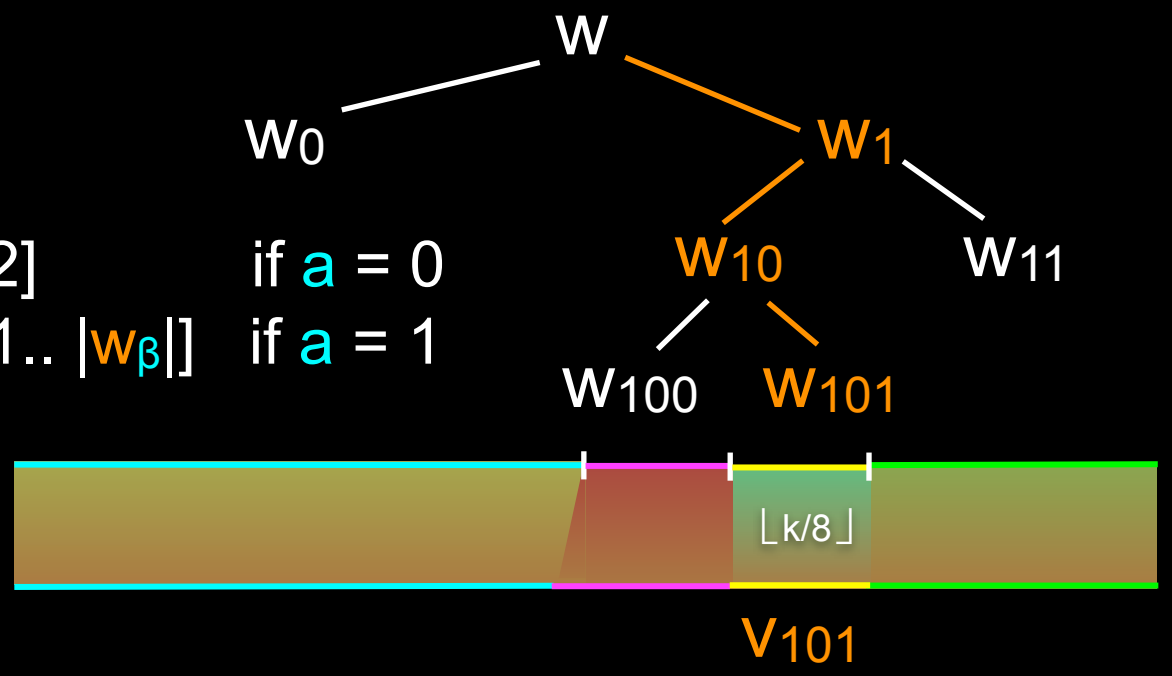

w_0
w_1
$\leq \lfloor k/2 \rfloor$ errors
v_1
w
w_0
w_1
w_{10}
w_{11}
$\lfloor k/4 \rfloor$?
$\lfloor k/4 \rfloor$?
v_{10}
w
w_0
w_1
w_{10}
w_{11}
w_{100}
w_{101}
$\lfloor k/8 \rfloor$
v_{101}

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 β0 is a prefix of α), and

(3) $v_{\beta 1}$ is a suffix of v_β (if β1 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

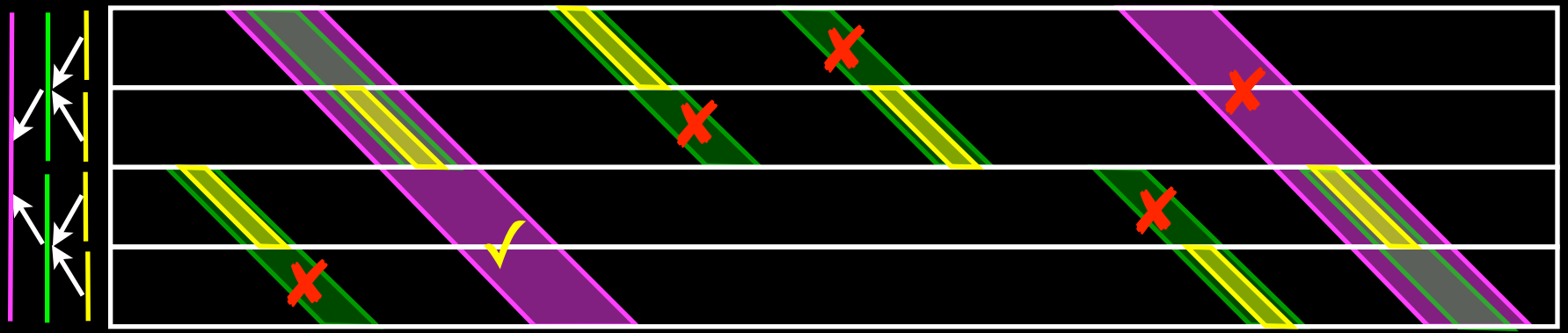
- March ’88: The Lister Hill Meeting & Galil’s 2 questions
- June ‘88: Seed & Extend
- May ’89: The TRW Chip & The Cigarette Break
- Fall ’89: Blast is Born
- Fall ’89: The Splitting Lemma
- Fall ’89: Seed & Extend by Doubling

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 $N_d(w)$ }

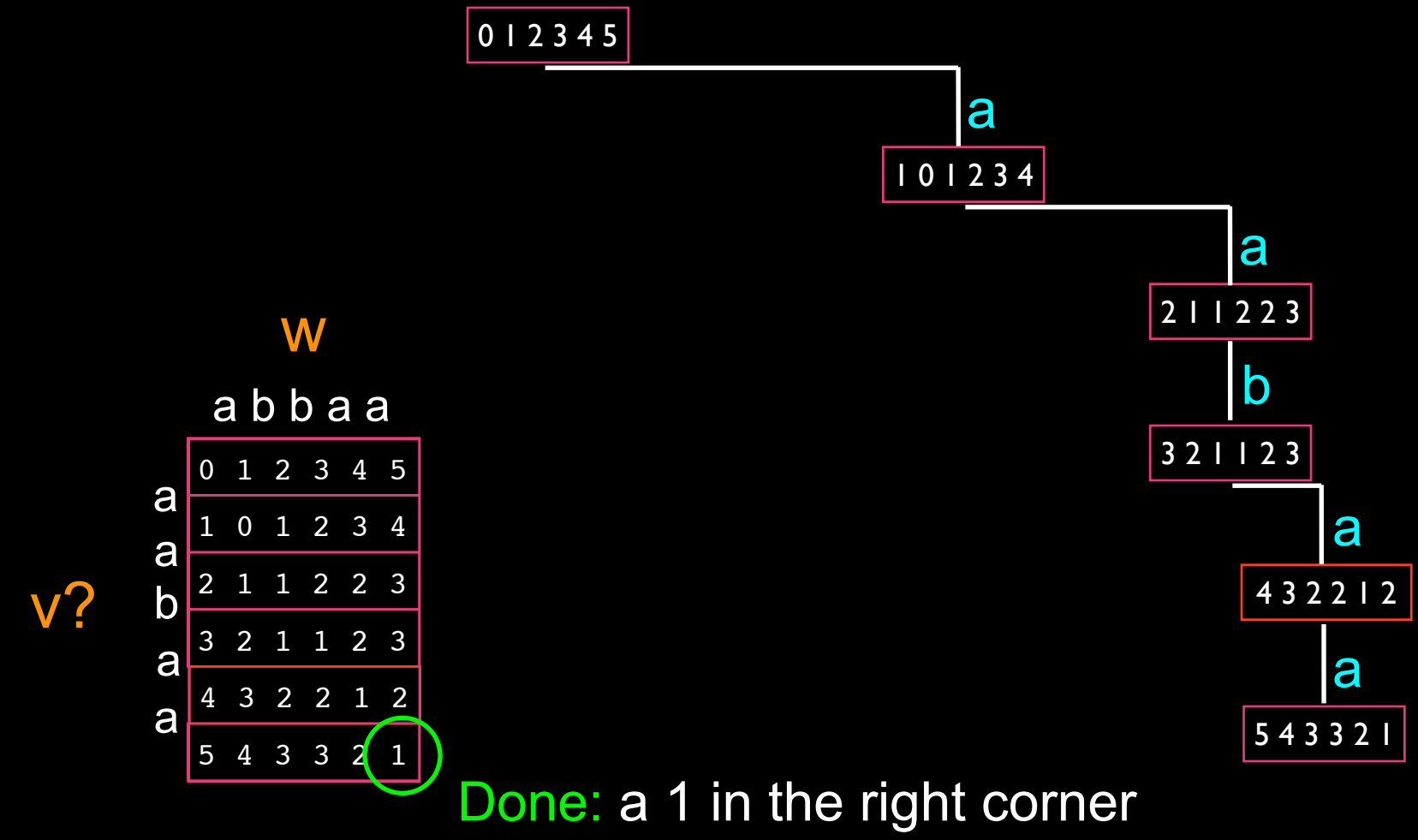
$\overline{\mathcal{N}_1}$(abbaa) = { aabaa, aabbaa, abaa, ~~abaaa~~, ababaa,
abba, ~~abbaa~~, ~~abbab~~, ~~abbaaa~~, ~~abbaba~~,
abbba, ~~abbbaa~~, babbaa, bbaa, bbbaa }

It suffices to find the words in the condensed neighborhood.

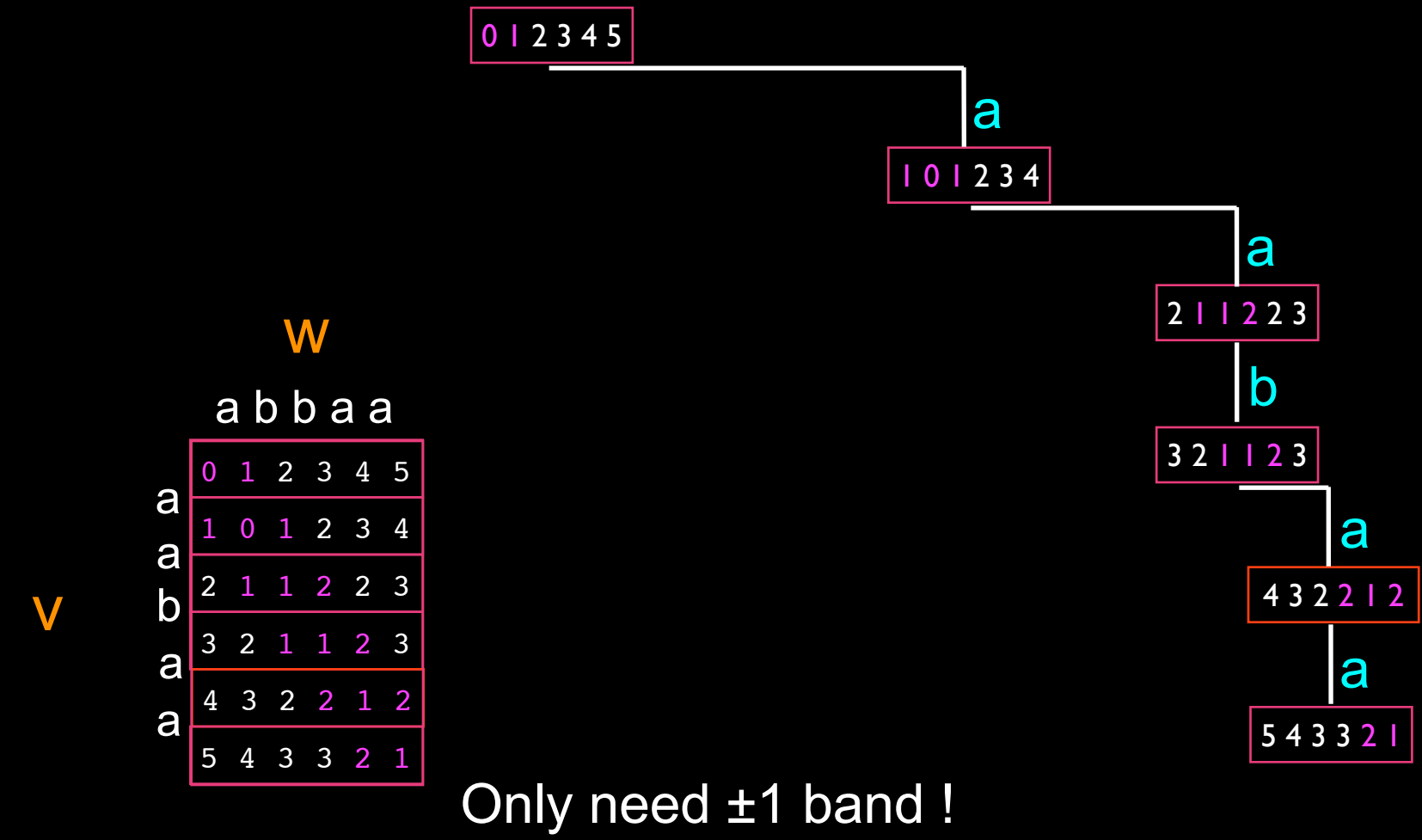
But how do you do that efficiently, including finding them in the index? ...

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

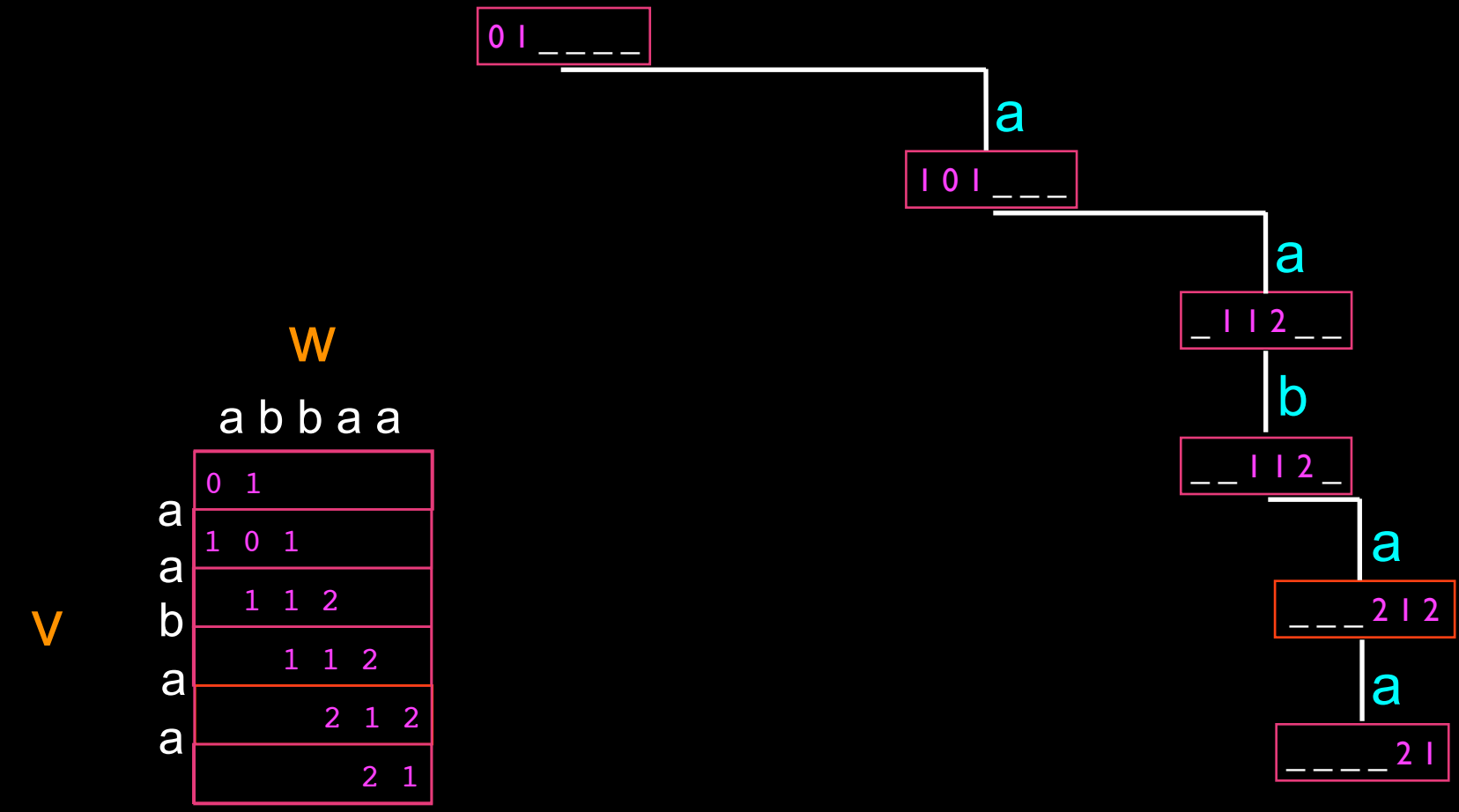
Condensed Neighborhoods



Condensed Neighborhoods

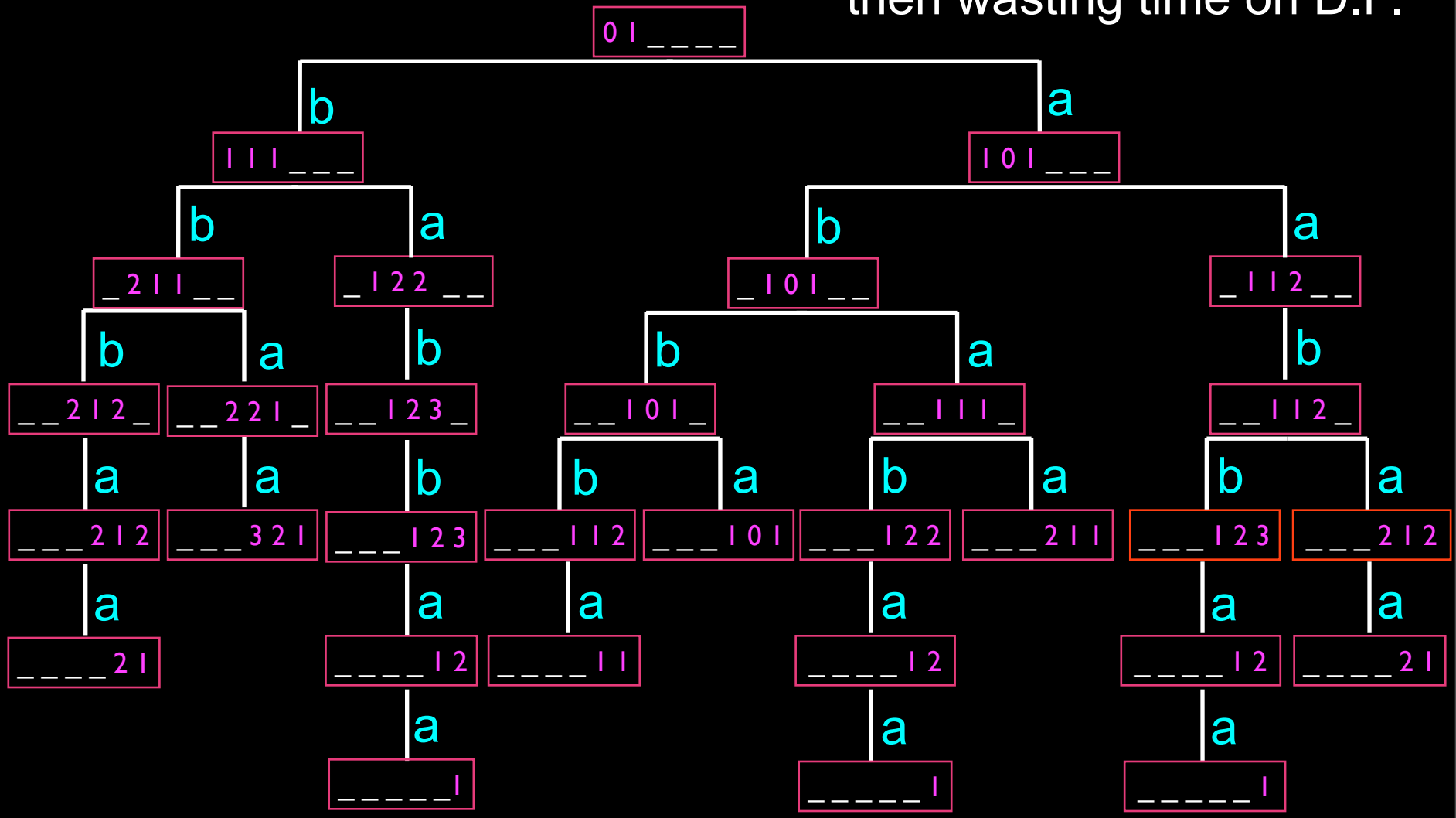


Condensed Neighborhoods



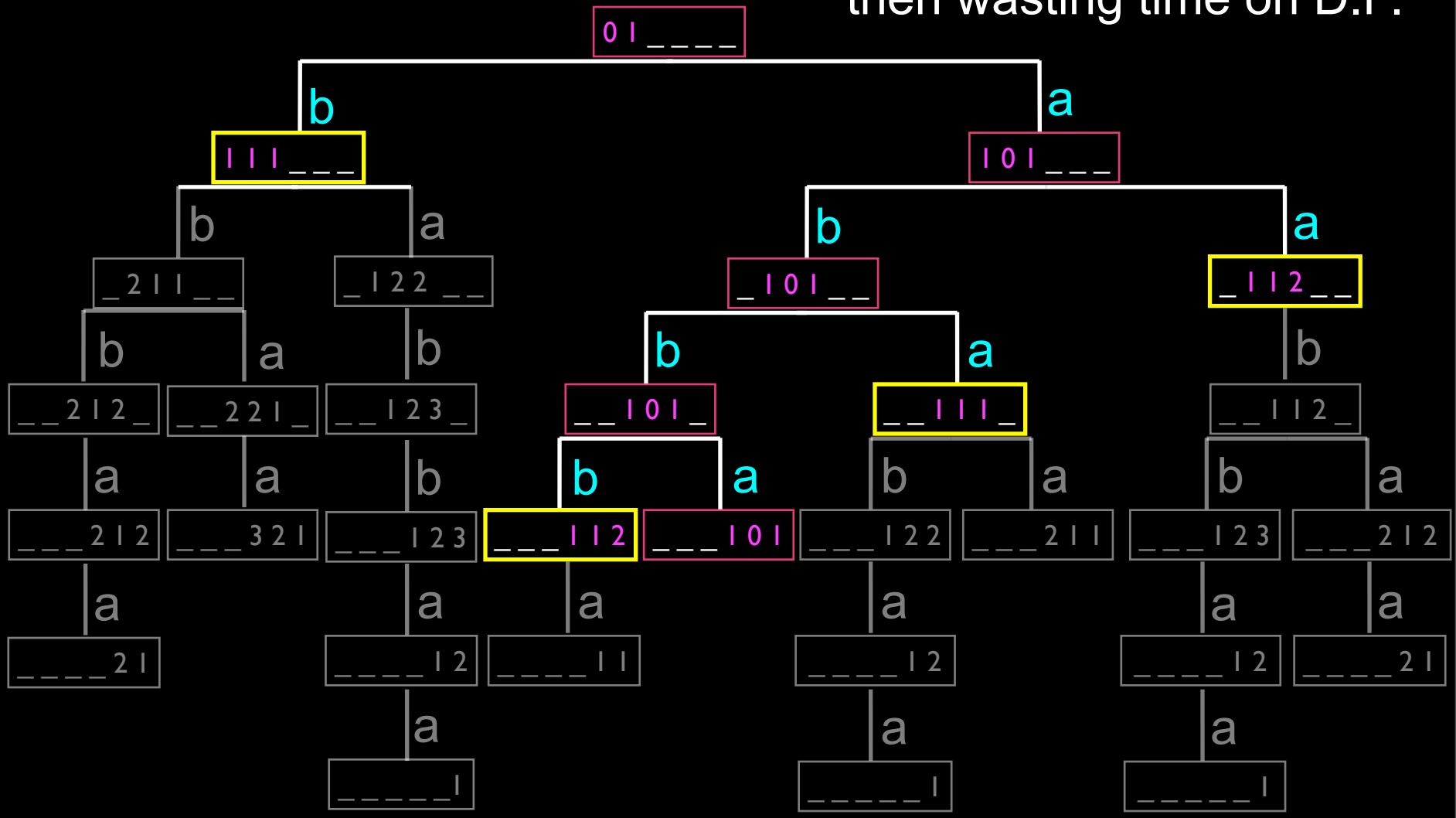
Condensed Neighborhoods

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

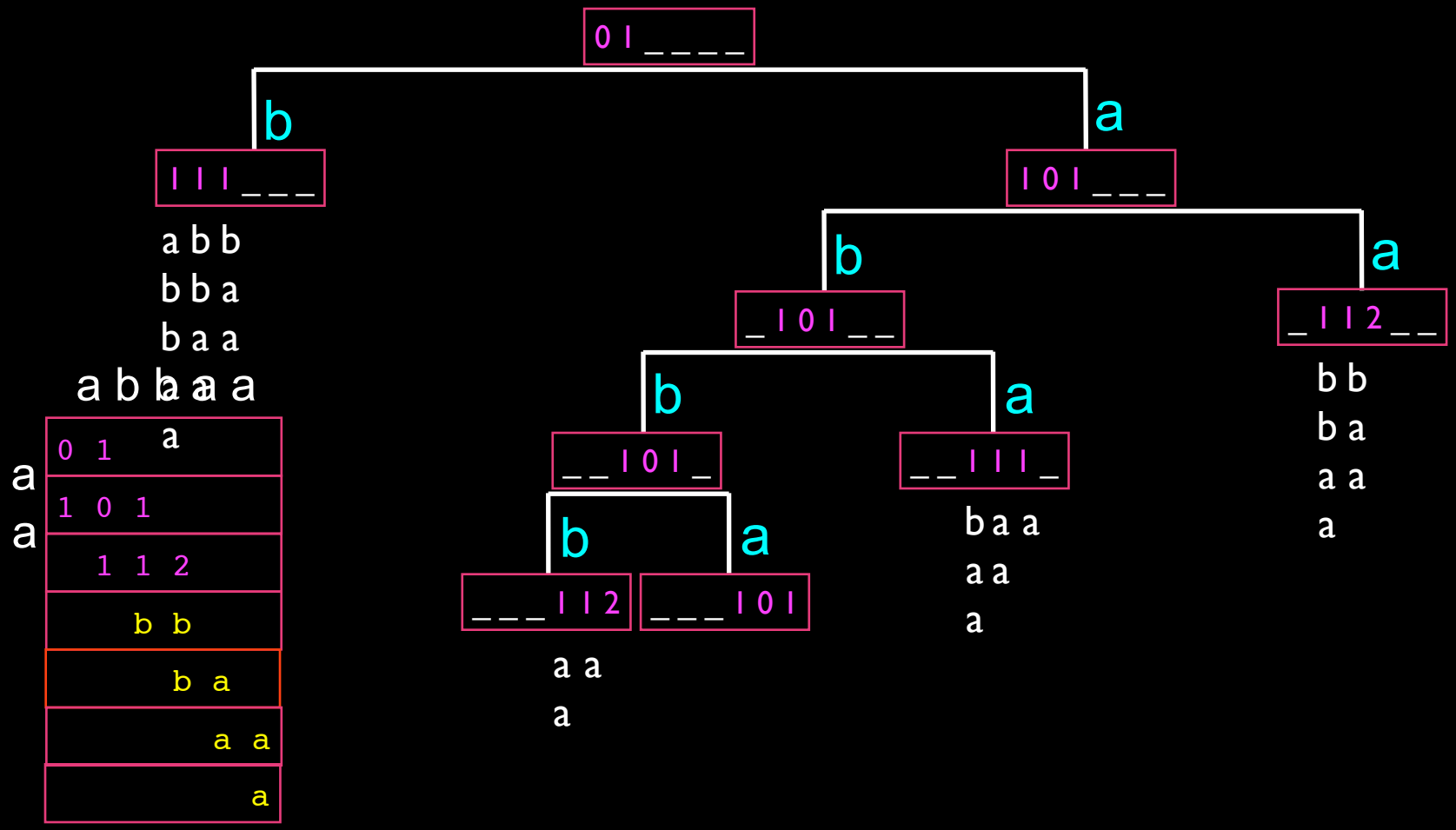


Condensed Neighborhoods

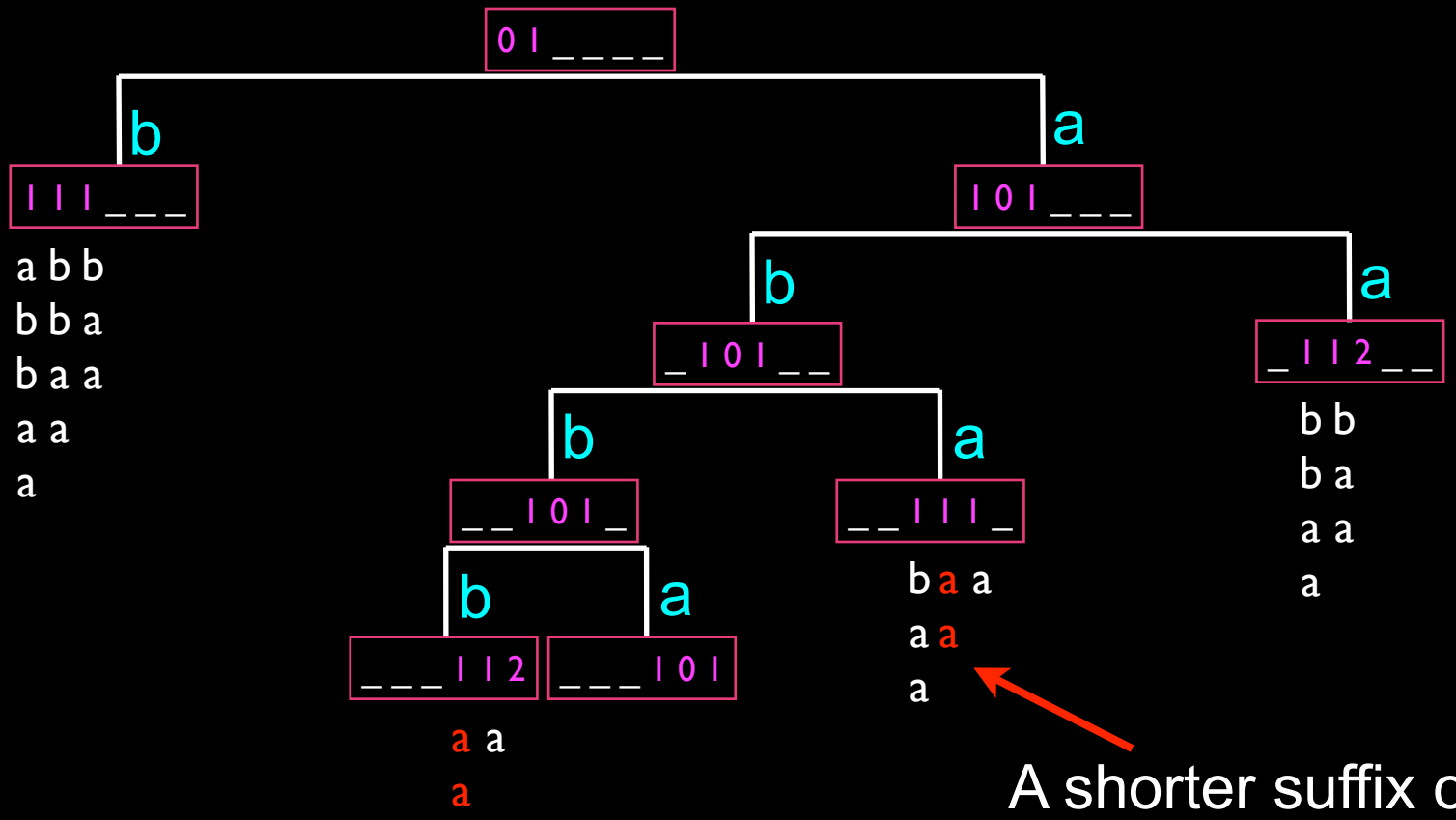
If all entries are ≥ d
then wasting time on D.P.



Condensed Neighborhoods



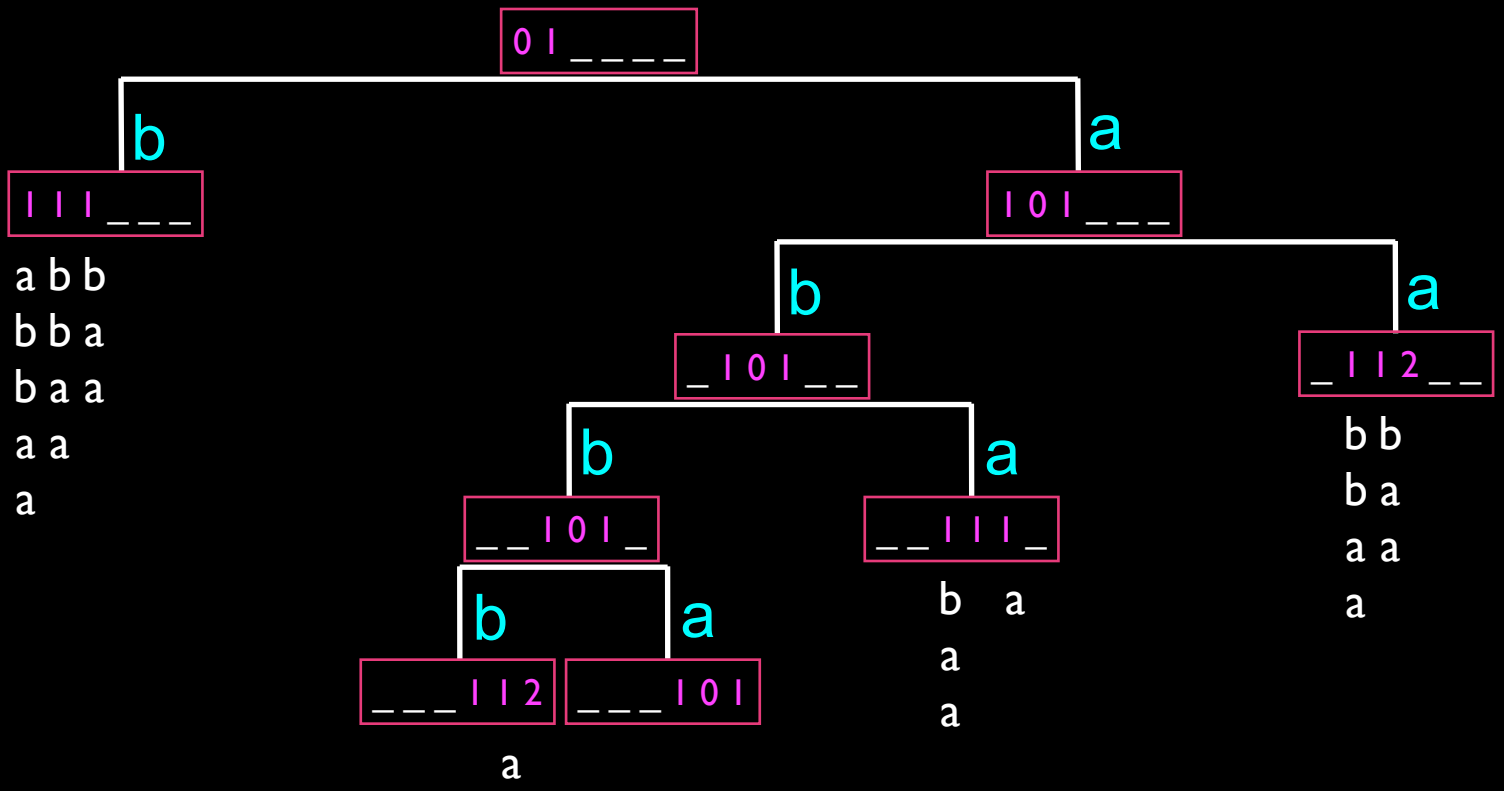
Condensed Neighborhoods



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

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

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 =` Φ

Lemma:

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

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

Complexity

So how big is it?

Lemma:

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

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

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

pow(ε)

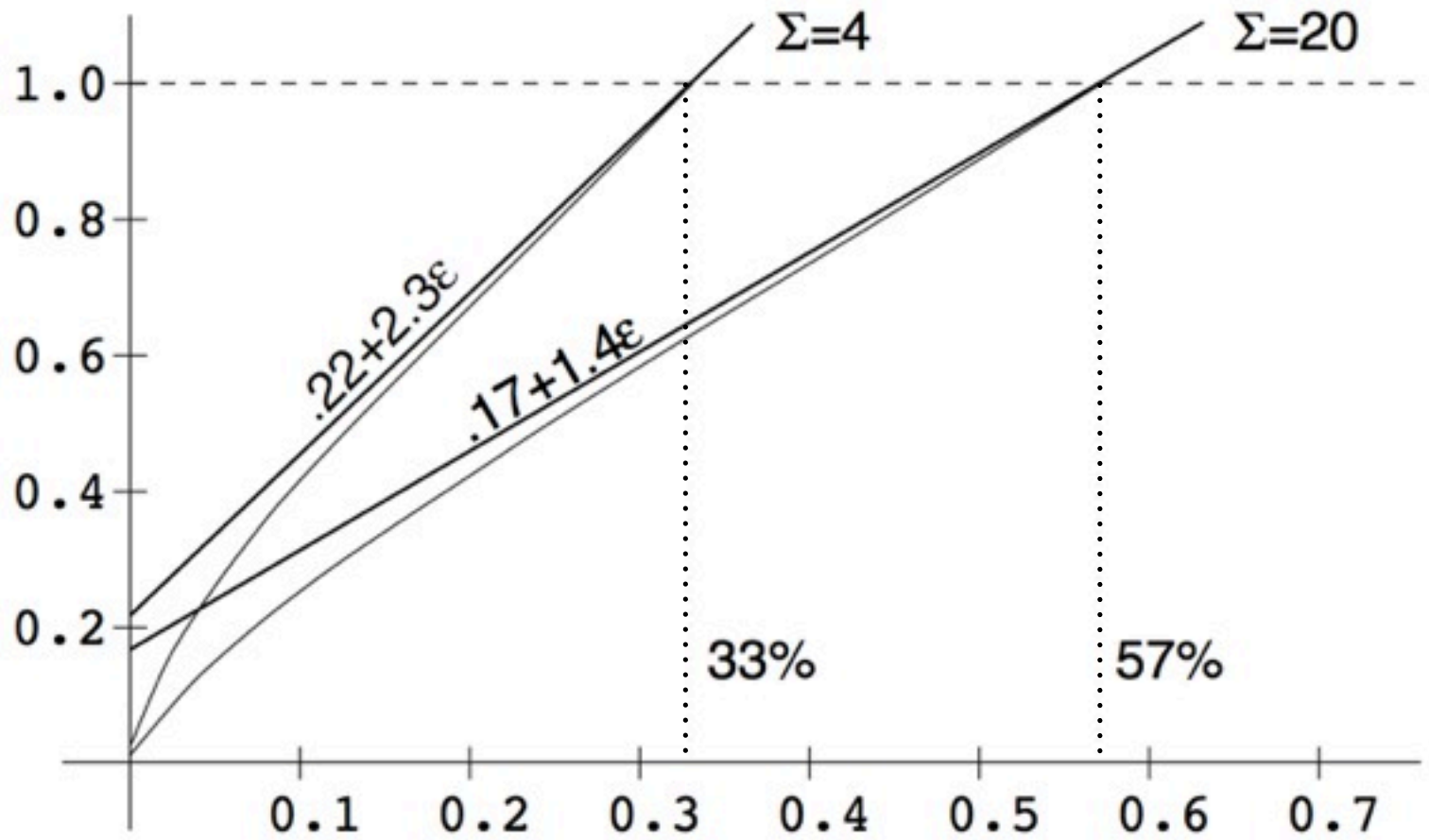


Figure 4: Plot of *pow*(ε) and Sample Bounding Lines

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

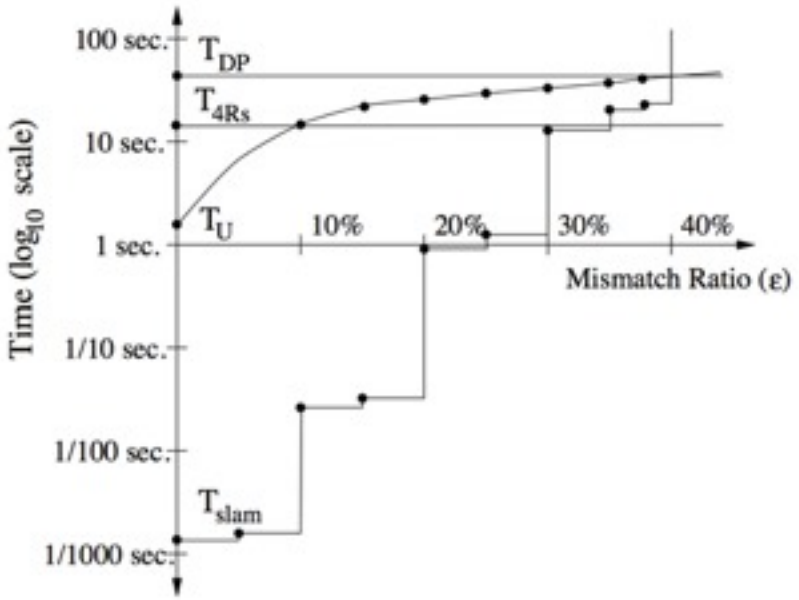
Theorem: Given

(a) A is effectively Bernouilli,

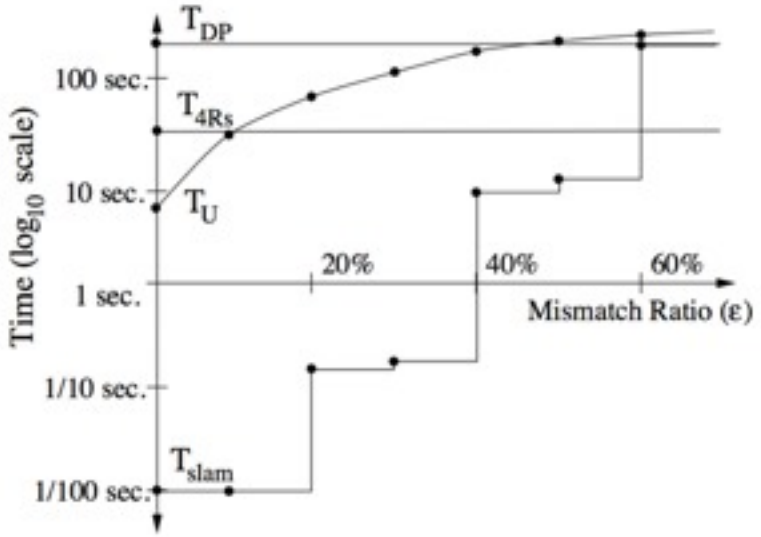
(b) a simple O(n) space, precomputed index of A, and

(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)

(submitted 1991 !)

Les Treilles, May 1990