CSE 549

Sael Lee

WHOLE GENOME SEQ. ALIGNMENT

Slides Courtesy of Michael Schatz
Quantitative Biology Class @ CSHL

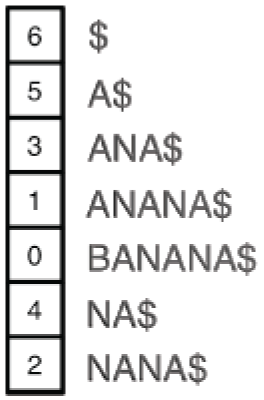
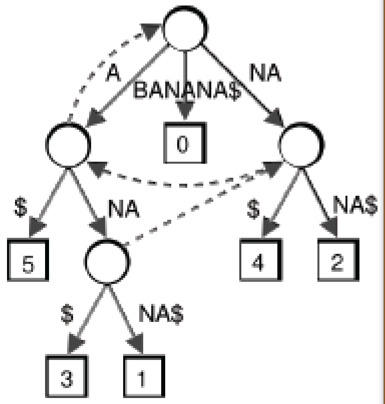
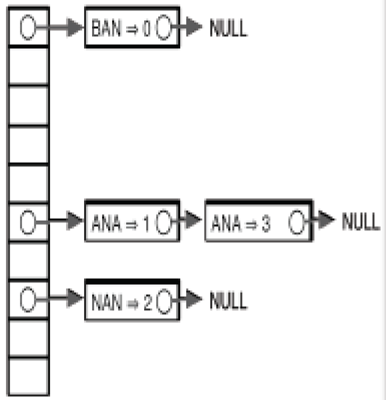
EXACT MATCHING

Slide extracts from Michael Schatz's Quantitative Biology Class @ CSHL
http://schatzlab.cshl.edu/teaching/2010

EXACT MATCHING OVERVIEW

Where is GATTACA in the human genome?

Brute Force (3 GB)	Suffix Array (>15 GB)	Suffix Tree (>51 GB)	Hash Table (>15 GB)

`BANANA`
`BAN`
`  ANA`
`    NAN`
`      ANA`
			

Naive	Vmatch, PacBio Aligner	MUMmer, MUMmerGPU	BLAST, MAQ, ZOOM, RMAP, CloudBurst
Slow & Easy	Binary Search	Tree Searching	Seed-and-extend

BRUTE FORCE ANALYSIS

- Brute Force:
  - At every possible offset in the genome:
    - Do all of the characters of the query match?
- Analysis
  - Simple, easy to understand
  - Genome length = n
  - Query length = m
  - Comparisons: (n-m+1) * m
- Overall runtime: O(nm)
  - If we double genome or query size, takes twice as long
  - If we double both, takes 4 times as long

SEARCHING FOR GATTACA

- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
G	A	T	T	A	C	A									

No match at offset 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
	G	A	T	T	A	C	A								

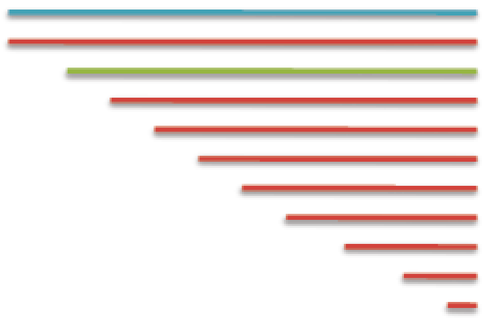
Match at offset 2

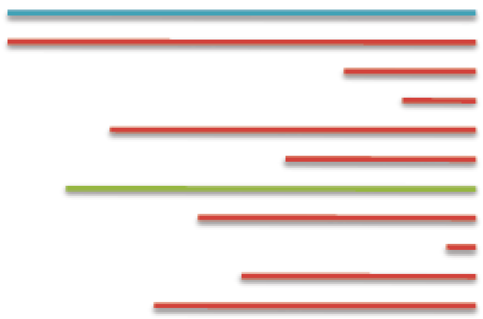
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

No match at offset 9 <- Checking each possible position takes time

SUFFIX ARRAYS

- What if we need to check many queries?
  - Sorting alphabetically lets us immediately skip through the data *without any loss in accuracy*
- Sorting the genome: Suffix Array (Manber & Myers, 1991)
  - Sort every suffix of the genome


Split into n suffixes


Sort suffixes alphabetically

SEARCHING THE INDEX

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or l ower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9; Mid = (9+9)/2 = 9
  - Middle = Suffix[9] = GATTACA…
    => Match at position 2!

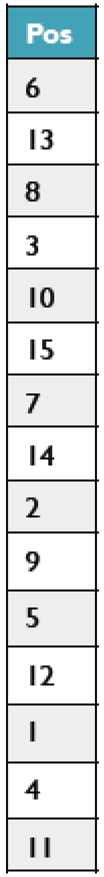
Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

SUFFIX ARRAY CONSTRUCTION

- Searching the array is very fast, but it takes time to construct
  - This time will be amortized over many, many searches
  - Run it once "overnight" and save it away for all future queries
- How do we store the suffix array?
  - Explicitly storing all n strings is not feasible $O(n^2)$
- Instead use implicit representation
  - Keep 1 copy of the genome, and a list of sorted offsets
  - Storing 3 billion offsets requires a big server (12GB)
    - Build a separate index for each chromosome

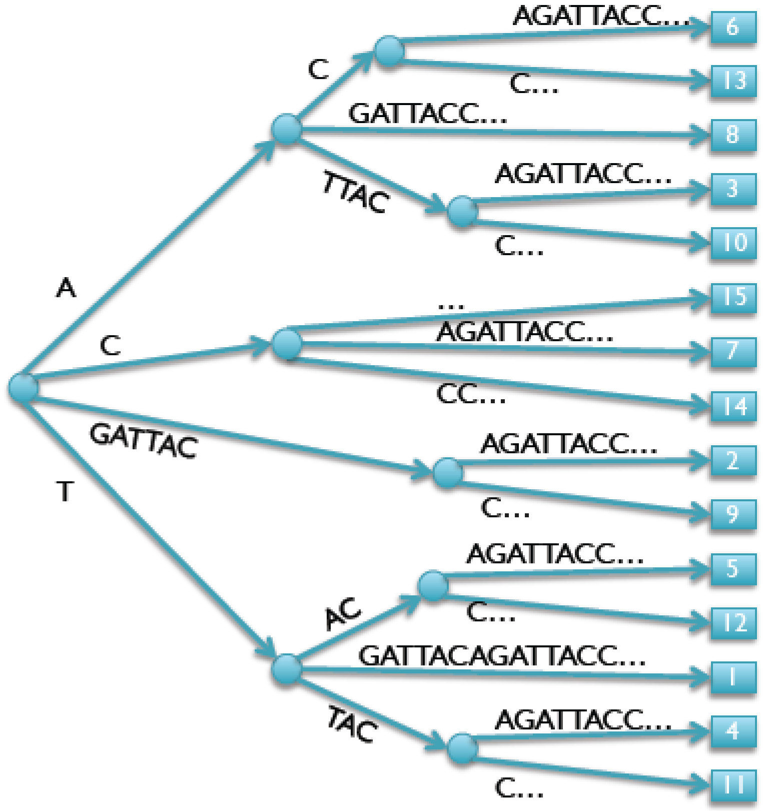


Pos
6
13
8
3
10
15
7
14
2
9
5
12
1
4
11

TGATTACAGATTACC

SUFFIX TREES

#	Sequence	Pos
1	ACAGATTACC...	6
2	ACC...	13
3	AGATTACC...	8
4	ATTACAGATTACC...	3
5	ATTACC...	10
6	C...	15
7	CAGATTACC...	7
8	CC...	14
9	GATTACAGATTACC...	2
10	GATTACC...	9
11	TACAGATTACC...	5
12	TACC...	12
13	TGATTACAGATTACC...	1
14	TTACAGATTACC...	4
15	TTACC...	11



Suffix Tree = Tree of suffixes (indexes **all** substrings of a sequence)

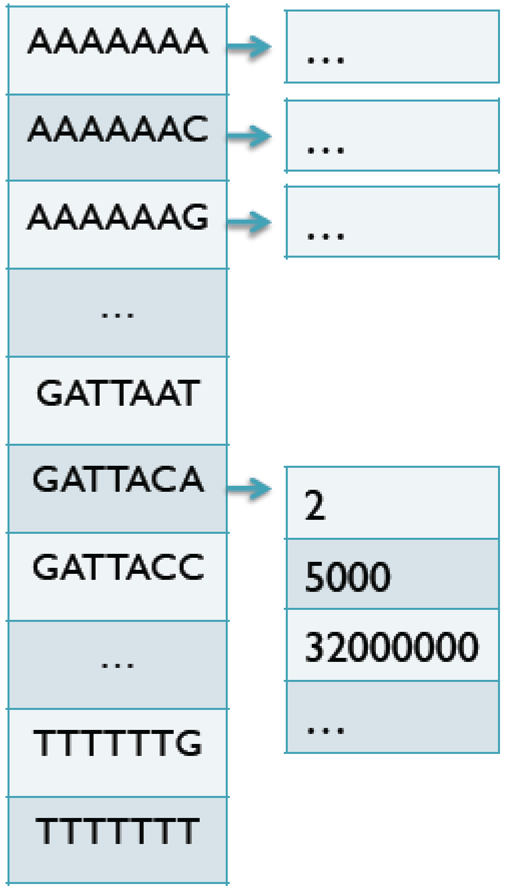
- 1 Leaf ($) for each suffix, path-label to leaf spells the suffix
- Nodes have at least 2 and at most 5 children (A,C,G,T,$)

SUFFIX TREE PROPERTIES & APPLICATIONS

- Properties
  - Number of Nodes/Edges: O(n)
  - Tree Size: O(n)
  - Max Depth: O(n)
  - Construction Time: O(n)
    - Uses suffix links to jump between nodes without rechecking
    - Tricky to implement, prove efficiency
- Applications
  - Sorting all suffixes: O(n)
  - Check for query: O(m)
  - Find all z occurrences of a query O(m + z)
  - Find maximal exact matches O(m)
  - Longest common substring O(m)
- Used for many string algorithms in linear time
  - Many can be implemented on suffix arrays using a little extra work

HASHING

- Where is GATTACA in the human genome?
  - Build an inverted index (table) of every k-mer in the genome
- How do we access the table?
  - We can only use numbers to index
    - table[GATTACA] <- error, does not compute
  - Encode sequences as numbers
    - Easy: A = 110, C = 210, G = 310, T = 410
      - GATTACA = 314412110
    - Smart: A = 002, C = 012, G = 102, T = 112
      - GATTACA = 100011110001002 = 915610
  - Running time
    - Construction: O(n)
    - Lookup: O(1) + O(z)
    - Sorts the genome mers in linear time



IN-EXACT ALIGNMENT

Slide extracts from Michael Schatz’s Quantitative Biology Class @ CSHL
http://schatzlab.cshl.edu/teaching/2010

IN-EXACT ALIGNMENT

- Where is GATTACA *approximately* in the human genome?
  - And how do we efficiently find them?
- It depends…
  - Define 'approximately'
    - Hamming Distance, Edit distance, or Sequence Similarity
    - Ungapped vs Gapped vs Affine Gaps
    - Global vs Local
    - All positions or the single 'best'?
- Efficiency depends on the data characteristics & goals
  - Smith-Waterman: Exhaustive search for optimal alignments
  - BLAST: Hash based homology searches
  - MUMmer: Suffix Tree based whole genome alignment
  - Bowtie: BWT alignment for short read mapping

SEED-AND-EXTEND ALIGNMENT

- Theorem: An alignment of a sequence of length *m* with at most *k* differences ***must*** contain an exact match at least $s=m/(k+1)$ bp long (*Baeza-Yates* and Perleberg, 1996)
  - Proof: Pigeon hole principle

- Search Algorithm
  - Use an index to rapidly find short exact alignments to seed longer in-exact alignments
    - RMAP, CloudBurst, …
  - Specificity of the seed depends on length
  - Length s seeds can also seed some lower quality alignments
    - Won't have perfect sensitivity, but avoids very short seeds

HAMMING DISTANCE LIMITATIONS

- Hamming distance measures the number of substitutions (SNPs)
  - Appropriate if that's all we expect/want to find
    - Illumina sequencing error model
    - Other highly constrained sequences
- What about insertions and deletions?
  - At best the **indel** will only slightly lower the score
  - At worst highly similar sequences will fail to align

```
ACGTCTAG
||*****^
ACTCTAG-
```

Hamming distance=5 : 2 matches, 5 mismatches, 1 not aligned

```
ACGTCTAG
||^|||||
AC-TCTAG
```

Edit Distance = 1 : 7 matches, 0 mismatches, 1 not aligned

EDIT DISTANCE EXAMPLE

TGCATAT → ATCCGAT in 4 steps

TGCATAT → (insert A at front)

ATGCATAT → (delete 6th T)

ATGCATA → (substitute G for 5th A)

ATGCGTA → (substitute C for 3rd G)

ATCCGAT (Done)

Can it be done in 3 steps???

BASIC LOCAL ALIGNMENT SEARCH TOOL (BLAST)

- Rapidly compare a sequence Q to a database to find all sequences in the database with an score above some cutoff S.
  - Which protein is most similar to a newly sequenced one?
  - Where does this sequence of DNA originate?
- Speed achieved by using a procedure that typically finds “most” matches with scores > S.
  - Tradeoff between sensitivity and specificity/speed
    - Sensitivity – ability to find all related sequences
    - Specificity – ability to reject unrelated sequences

(Altschul et al. 1990)

BLAST: SEED AND EXTEND

```
FAKDFLAGGVAAAISKTAVAPIERVKLLLQVQHASKQITADKQYKGIIDCVVRIPKEQGV
F  D  +GG AAA+SKTAVAPIERVKLLLQVQ ASK I  DK+YKGI+D ++R+PKEQGV
FLIDLASGGTAAAVSKTAVAPIERVKLLLQVQDASKAIAVDKRYKGIMDVLIRVPKEQGV
```

- Homologous sequence are likely to contain a short high scoring word pair, a seed.
  - BLAST *doesn't* make explicit guarantees
- BLAST then tries to extend high scoring word pairs to compute maximal high scoring segment pairs (HSPs).
  - Heuristic algorithm but evaluates the result statistically.

BLAST - ALGORITHM

- Step 1: Preprocess Query
  - Compile the short-high scoring word list from query. The length of query word, w, is 3 for protein scoring Threshold T is 13
- Step 2: Construct Query Word Hash Table
- Step 3: Scanning DB
  - Identify all exact matches with DB sequences
- Step 4: Search optimal alignment
  - For each hit-word, extend ungapped alignments in both directions.
  - Let S be a score of hit-word
- Step 5: Evaluate the alignment statistically
  - Stop extension when E-value (depending on score S) become less than threshold. The extended match is called High Scoring Segment Pair.

WHOLE GENOME ALIGNMENT WITH MUMMER

- Maximal Unique Matcher (MUM)er
  - match
    - exact match of a minimum length
  - maximal
    - cannot be extended in either direction without a mismatch
  - *unique*
    - occurs only once in both sequences (MUM)
    - occurs only once in a single sequence (MAM)
    - occurs one or more times in either sequence (MEM)

Slides Courtesy of Adam M. Phillippy
amp@umics.umd.edu

MUMMER

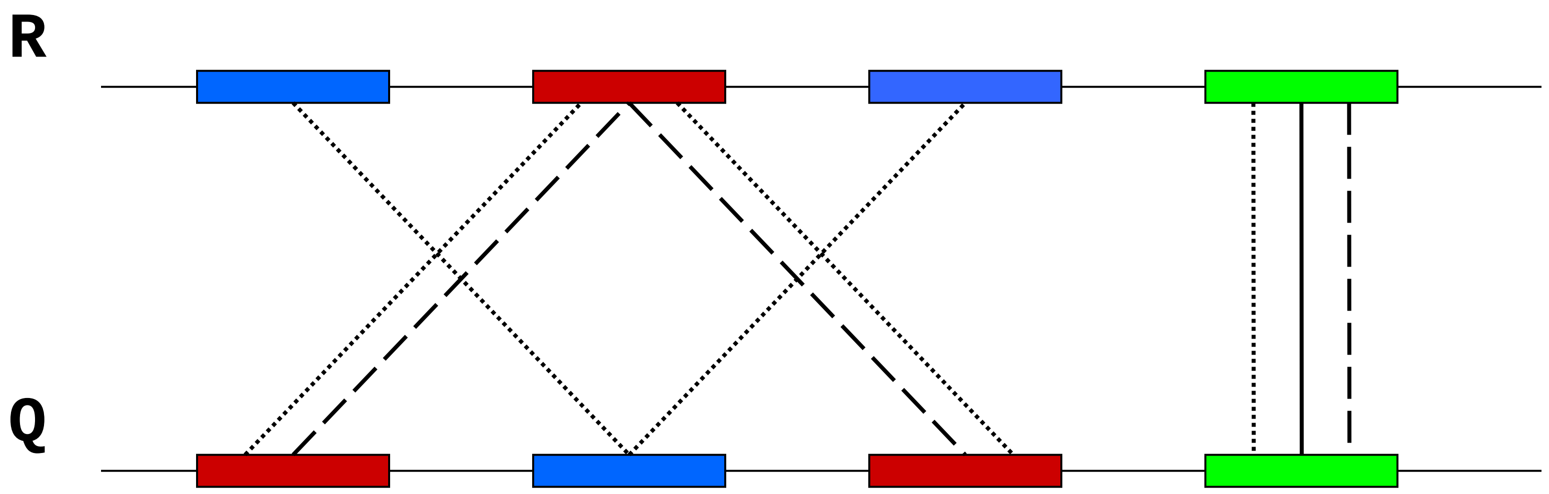
- Primary uses
  - exact matching (seeding)
  - dot plotting
- Pros
  - very efficient *O*(n) time and space
    - ~17 bytes per bp of reference sequence
    - *E. coli K12* vs. *E. coli O157:H7* (~5Mbp each)
      - 17 seconds using 77 MB RAM
  - multi-FastA input
- Cons
  - exact matches only

IS IT A MAM, MEM OR MUM?

MUM : maximal unique match ———

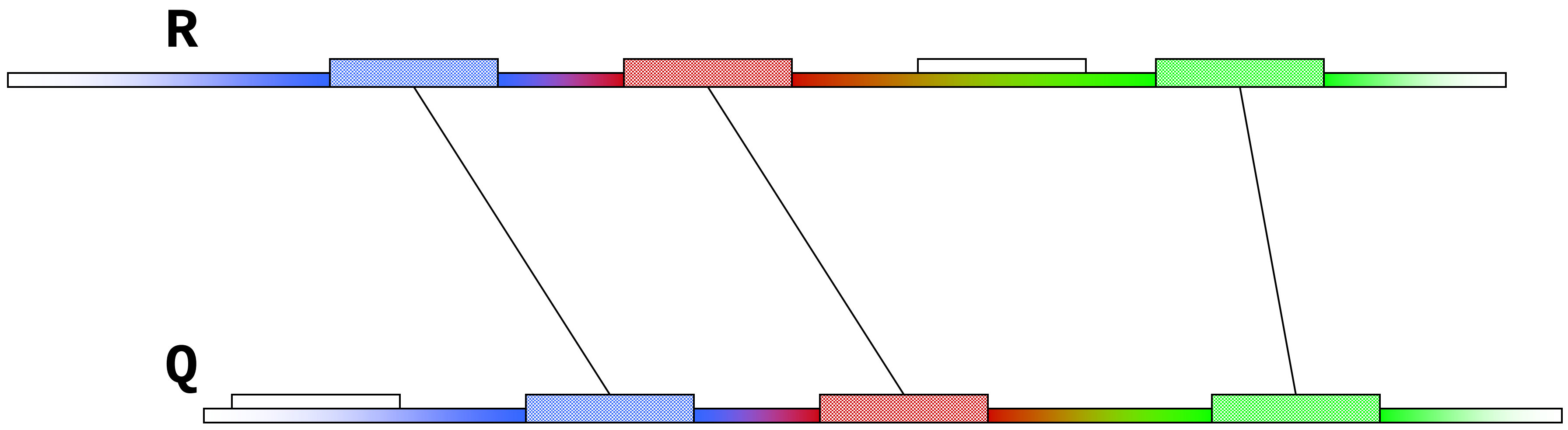
MAM : maximal almost-unique match – – – – –

MEM : maximal exact match ⋯⋯⋯

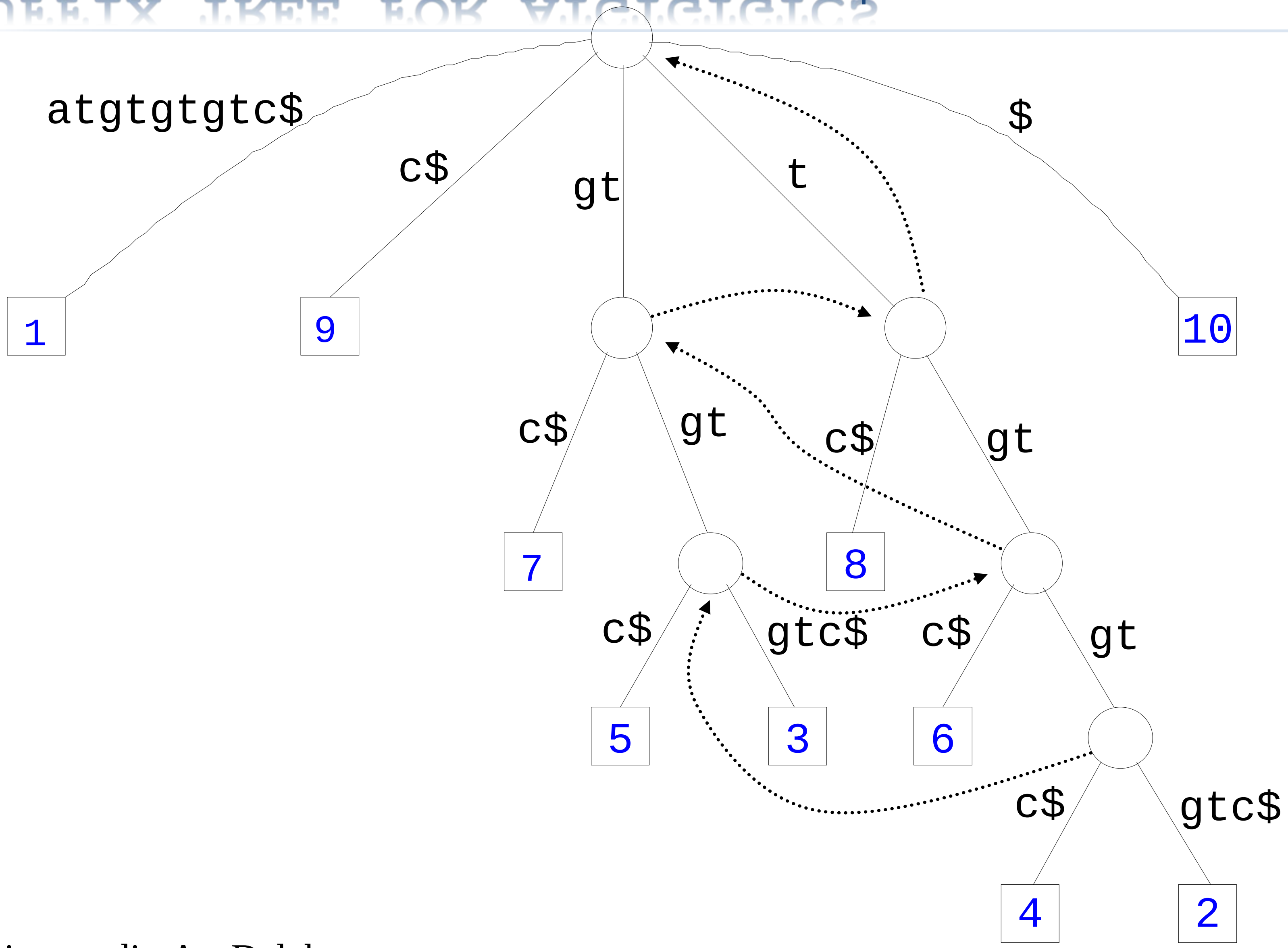


SEED AND EXTEND

- How can we make MUMs **BIGGER?**
  - Find MUMs
    - using a suffix tree
  - Cluster MUMs
    - using size, gap and distance parameters
  - Extend clusters
    - using modified Smith-Waterman algorithm



SUFFIX TREE FOR ATGTGTGTC$



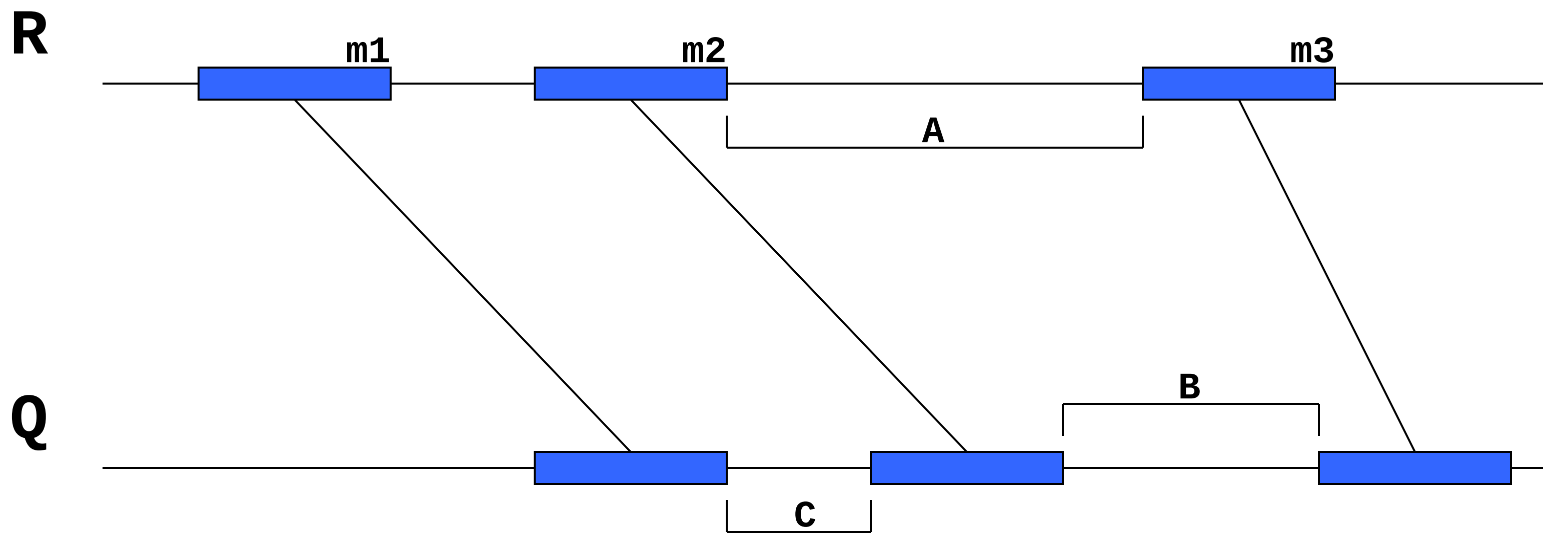
Drawing credit: Art Delcher

CLUSTERING

cluster length = Σm_i

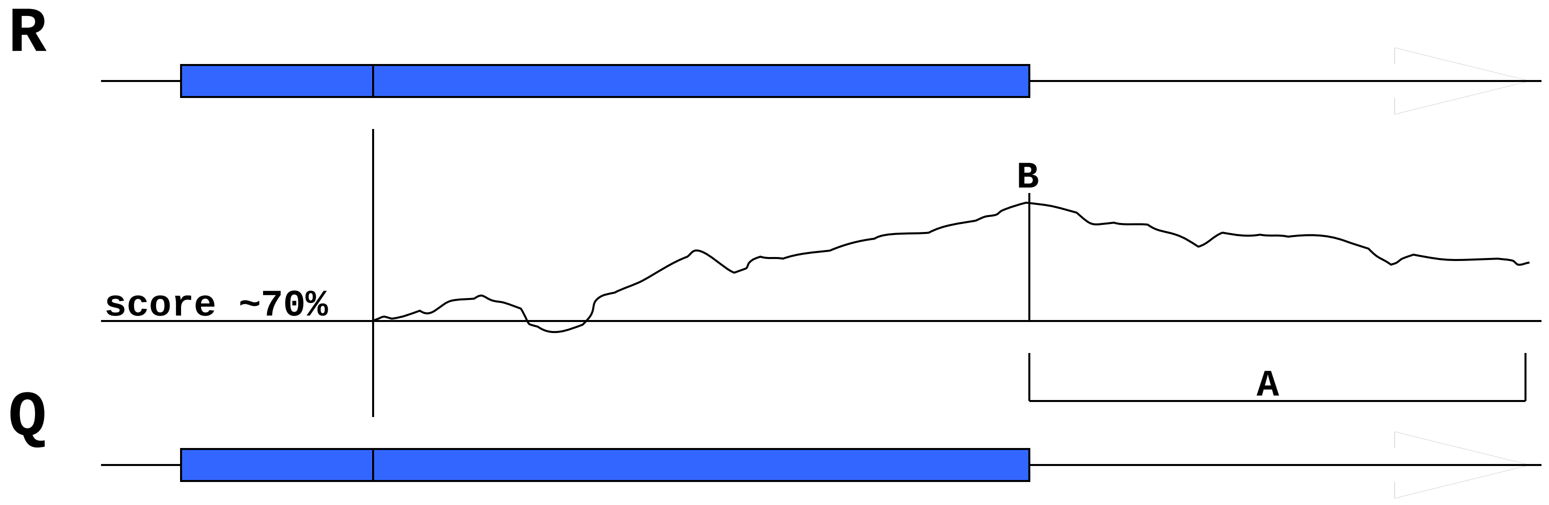
gap distance = c

indel factor = |B – A| / B *or* |B – A|



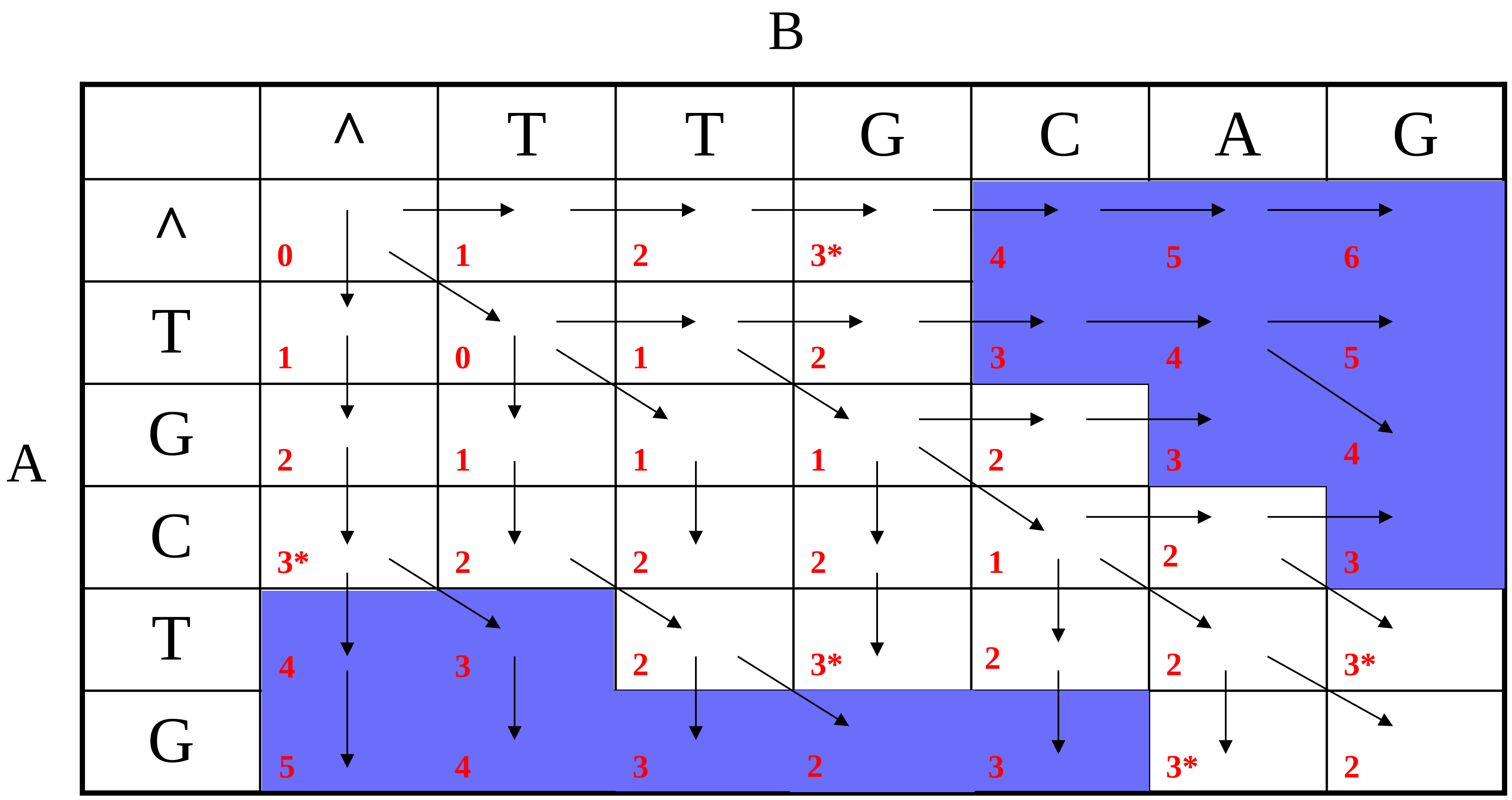
EXTENDING

break point = B



break length = A

BANDED ALIGNMENT



B

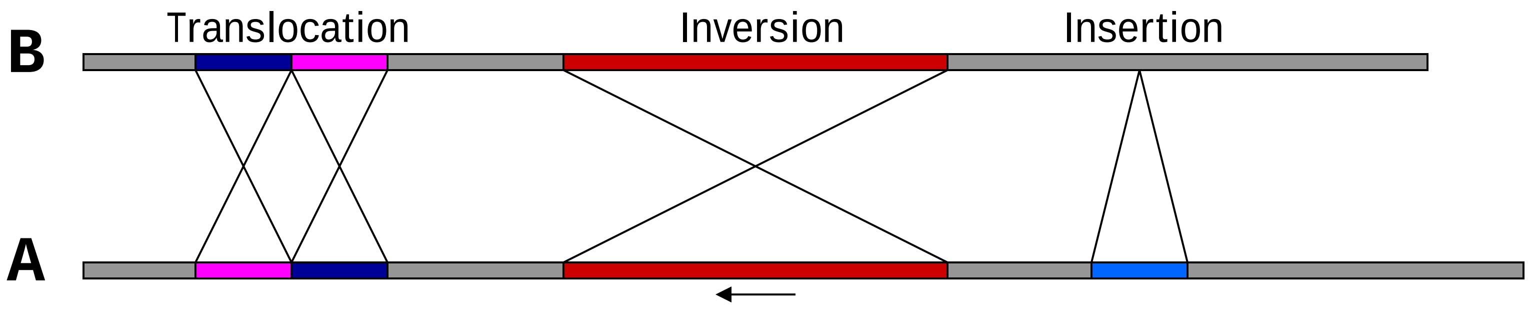
A

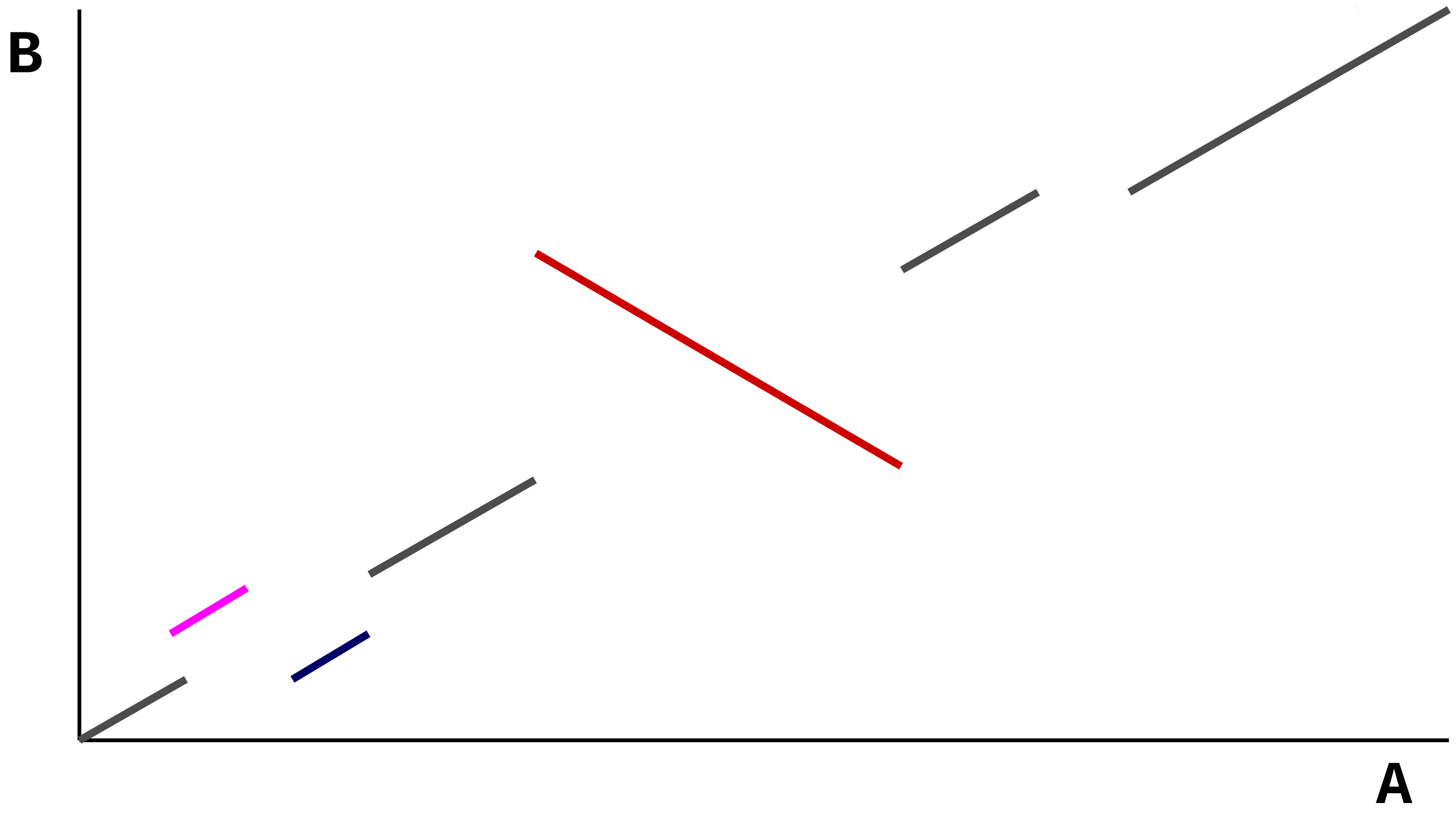
	^	T	T	G	C	A	G
^	0	1	2	3*	4	5	6
T	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3*	2	2	2	1	2	3
T	4	3	2	3*	2	2	3*
G	5	4	3	2	3	3*	2

SIDETRACK: PLOTS

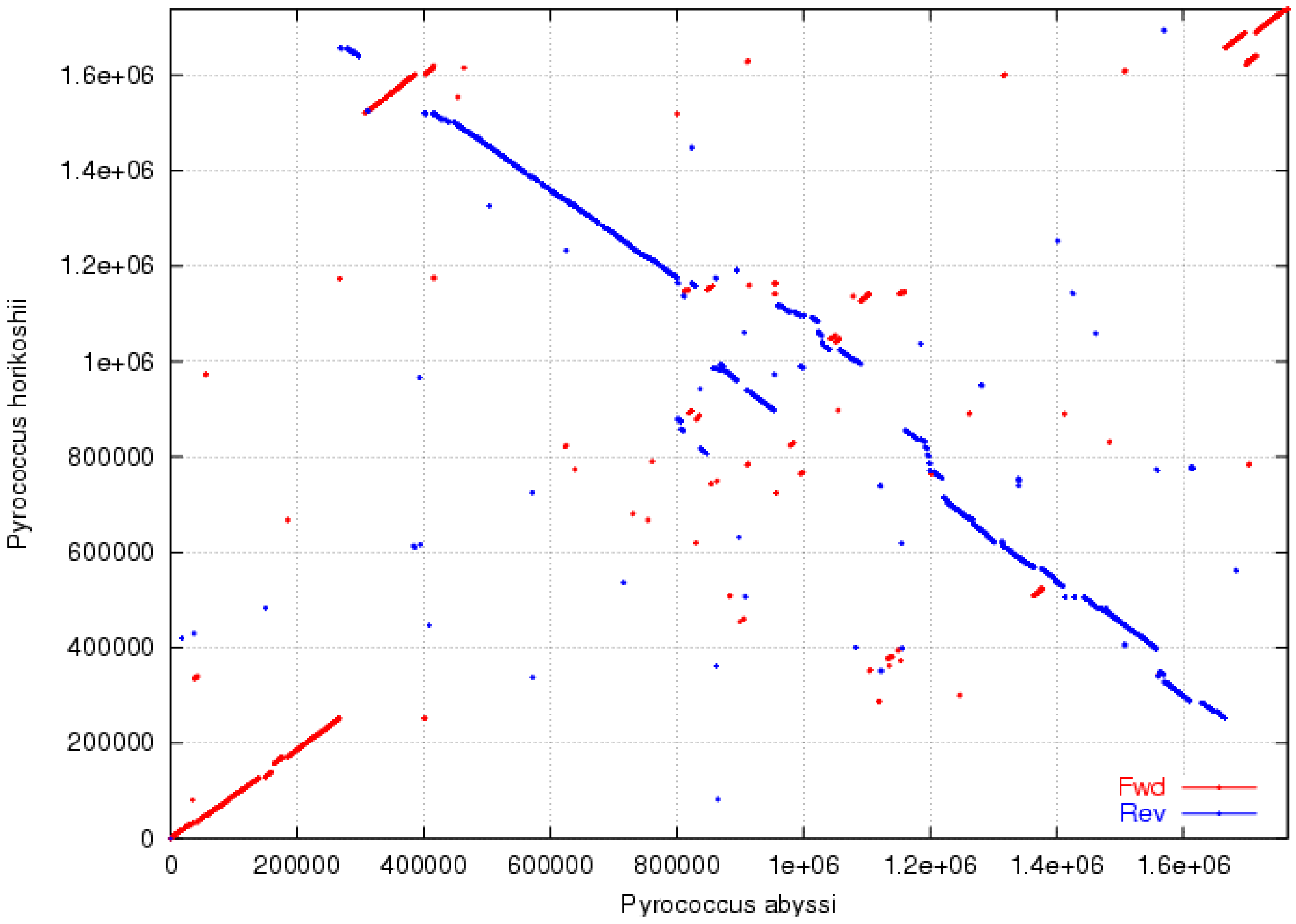
- How can we visualize *whole* genome alignments?

- With an alignment dot plot
  - N x M matrix
    - Let i = position in genome A
    - Let j = position in genome B
    - Fill cell (i,j) if A_i shows similarity to B_j
  - A perfect alignment between A and B would completely fill the positive diagonal





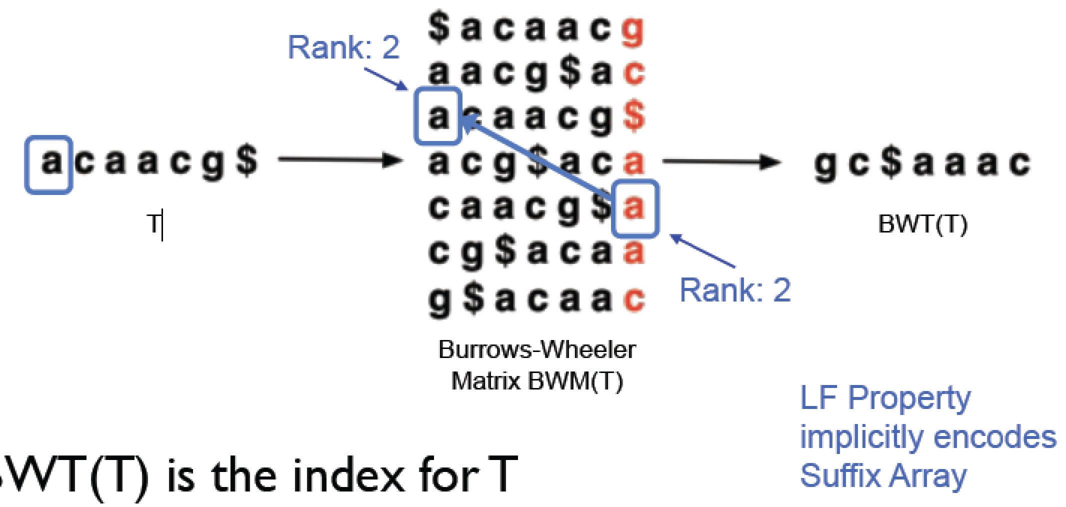
http://mummer.sourceforge.net/manual/AlignmentTypes.pdf dotplot from promer-based mummerplot



BOWTIE: ULTRAFAST AND MEMORY EFFICIENT ALIGNMENT OF SHORT DNA SEQUENCES TO THE HUMAN GENOME

- Uses Burrows-Wheeler Transform in addition to suffix trees

- Reversible permutation of the characters in a text



"BWT rearranges a character string into runs of similar characters. This is useful for compression, since it tends to be easy to compress a string that has runs of repeated characters by techniques such as move-to-front transform and run-length encoding" (wikipedia)

- BWT(T) is the index for T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Slides Courtesy of Ben Langmead (langmead@umiacs.umd.edu)

BWT SHORT READ MAPPING

- Trim off very low quality bases & adapters from ends of sequences
- Execute depth-first-search of the implicit suffix tree represented by the BWT
  - If we fail to reach the end, back-track and resume search
  - BWT enables searching for good end-to-end matches entirely in RAM
    - 100s of times faster than competing approaches
- 3. Report the "best" n alignments
  - Best = fewest mismatches/edit distance, possibly weighted by QV
  - Some reads will have millions of equally good mapping positions
  - If reads are paired, try to find mapping that satisfies both