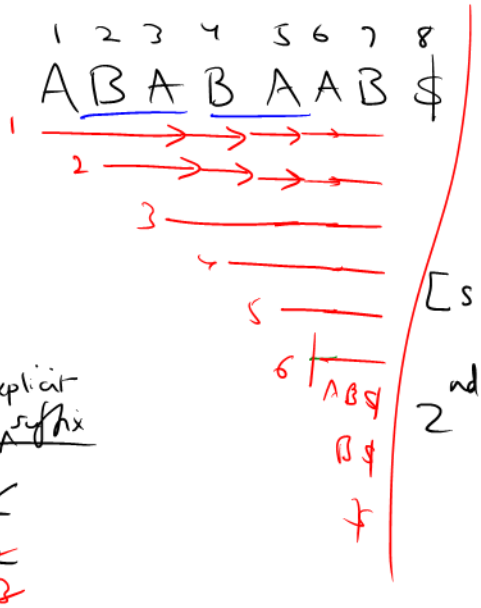
Suffix Tree

$O(n^2)$ time & space ← naive

$O(n)$ space & time ← Ukkonen's Algo

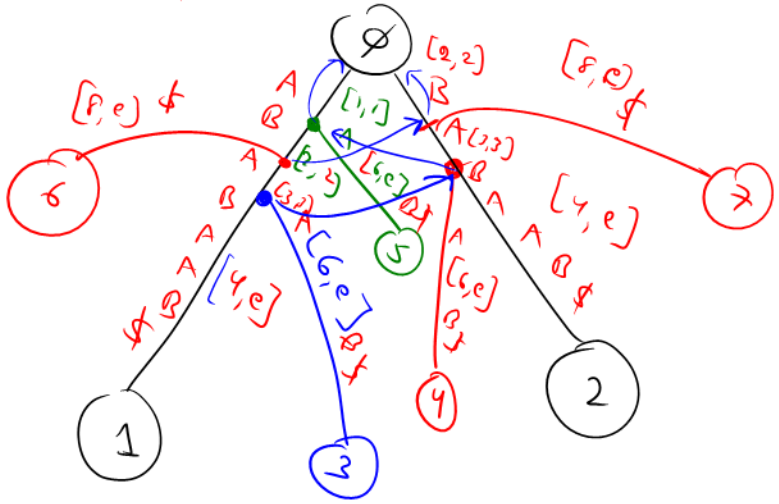


$\Sigma = \{A, B\}$

[s, e] ← [start, end]

2nd suffix : [2, e]

e	explicit last suffix
~~1~~	~~1~~
~~2~~	~~2~~
~~3~~	~~3~~
~~4~~	~~4~~
~~5~~	~~5~~
~~6~~	~~6~~
~~7~~	~~7~~
8	



suffix links

$s(ABA) = BA$

$s(AB) = A$

$s(A) = \emptyset$

$s(BA) = A$

$s(B) = \emptyset$

Mississippi $ ← try it!

1) querying

$q = BA$: answer : $(2, 4)$

$O(|q|)$ time + # of matches

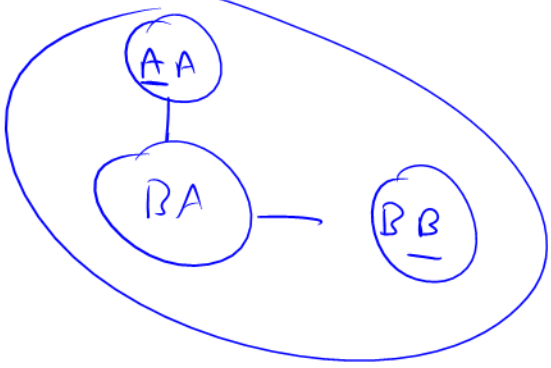
2) Inexact matches : $|q| = l$ ← length of the query

d ← # of mismatches

$d < l$

Naive approach:

$q = BA$ $d = 1$



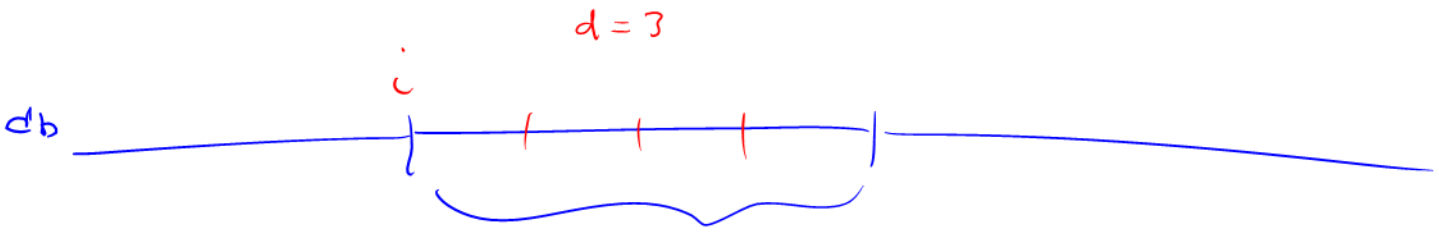
d - Neighborhood

Use the suffix tree to spell out the neighbors "with reserve counts"

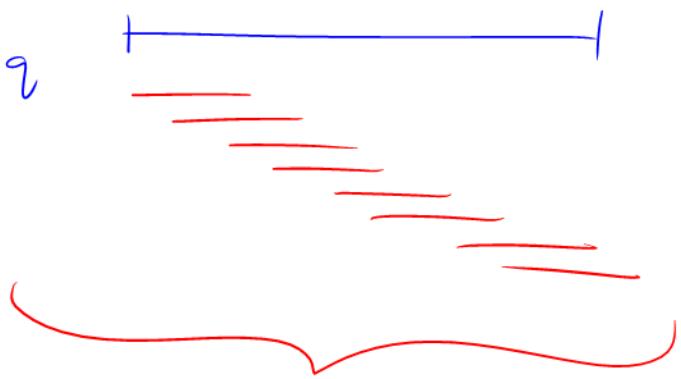
Suffix Tree : Human genome

$O\left(|\Sigma|^d\right)$ ← size of the neighborhood

d has to be small



q must have an exact match of length at least $\left\lfloor \frac{l}{d+1} \right\rfloor = w$

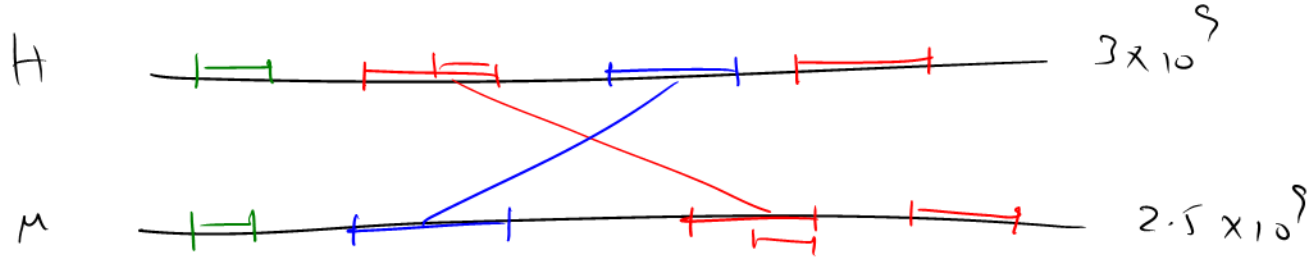


generate all queries of len w

→ look for an exact match

→ extend to longer matches

2) Genome Alignment



MUMer

↳ Maximal Unique Matches

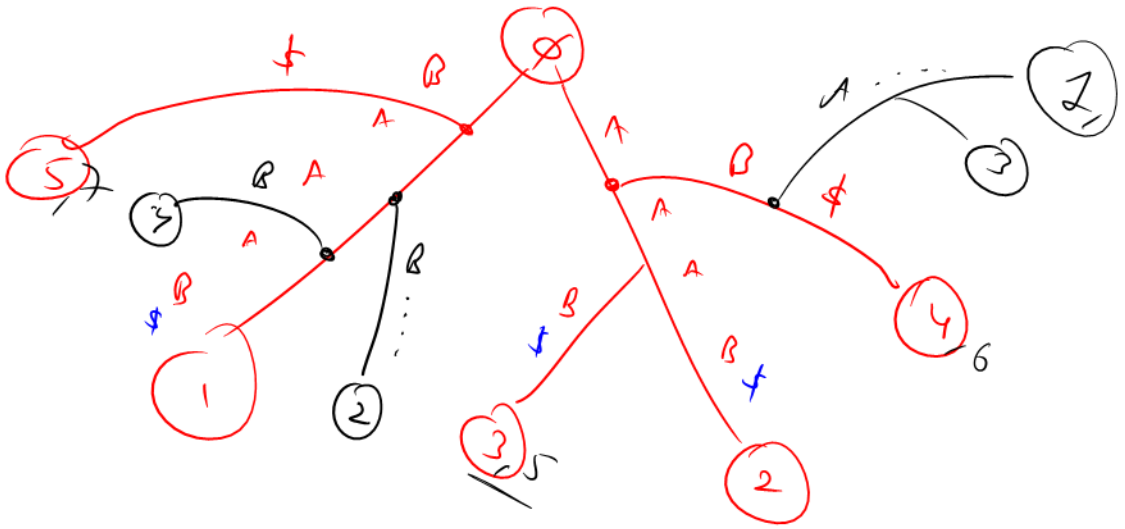
$O(n)$ time !

maximal ⟹ it cannot be extended ↦ left or right in either string

unique ⟹ one occurrence in each string

1) build ST for one of the strings
2) add 2nd string to the suffix tree

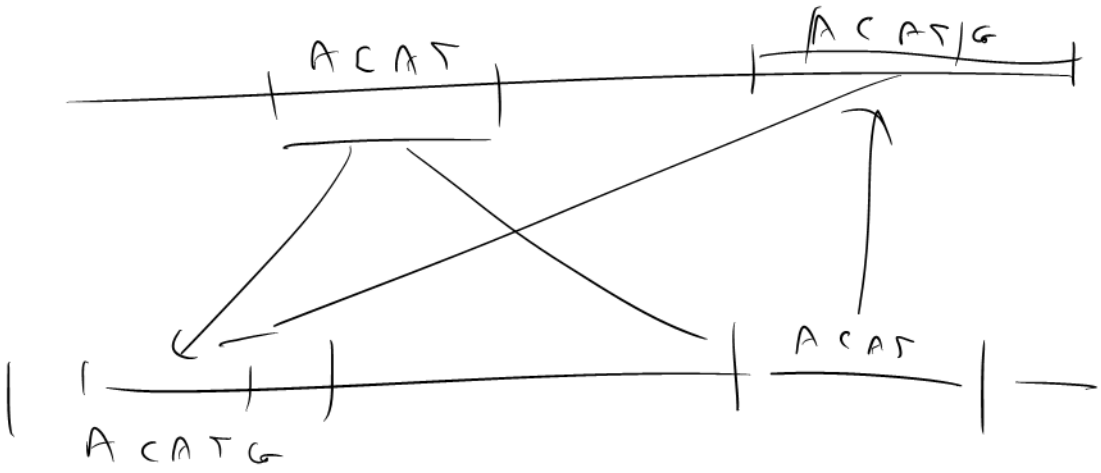
generalized suffix tree



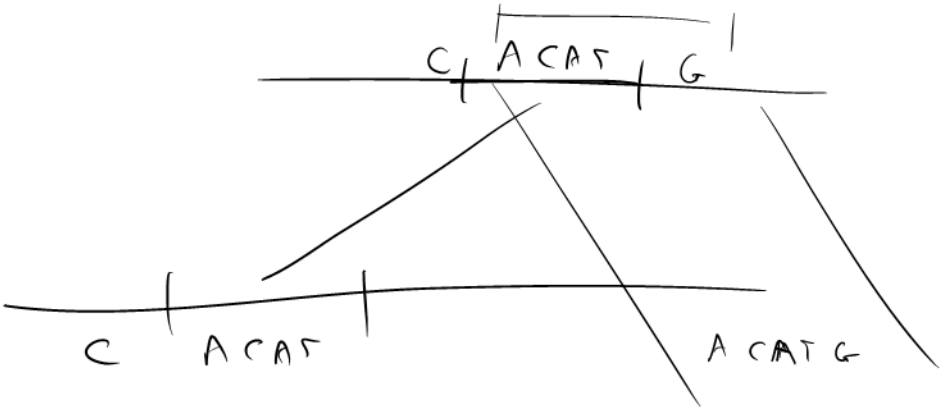
maximal

unique matches: AAB (M 3, H 5) ✓

BAA (M 1, H 4) ✓

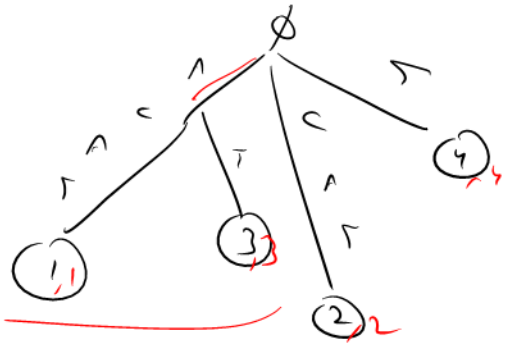


ACAT ← not unique



H : ACAT

M : ACAT



ACAT

CAT

AT

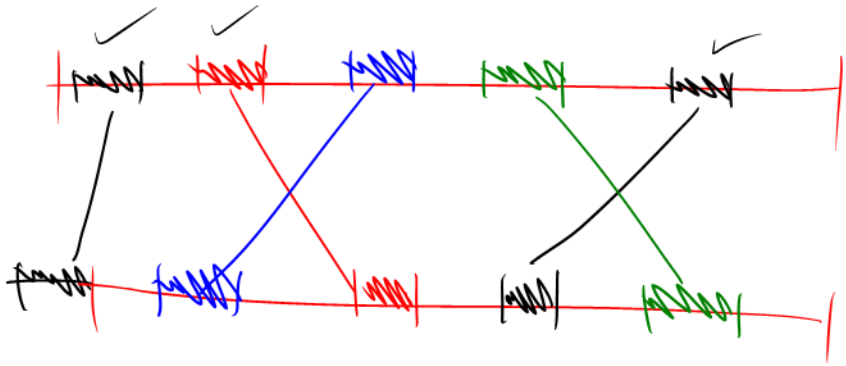
T

1) find the MUMs : $O(n) + O(m)$

suffix Tree — maximality check over the # of Unique matches (m)

$O(n|\Sigma|)$ ← real cost

2)



find the maximum weighted subsequence of segments between the 2 genomes : DP

---

suffix arrays

sorted list of suffixes of a DB string

1	2	3	4	5	6	7	8
A	B	A	B	A	A	B	$

$\$ < A < B$

total order on $\Sigma \cup \$$

suffix array L_1 → lex order

8	:	$
5	:	A A B $
6	:	A B $
L_2 → 3	:	A B A A B $
1	:	A B A B A B $
L_3 → 7	:	B $
→ 4	:	B A A B $
R → 2	:	B A B A A B $

conceptual

q: BA ?

Answer: 2, 4

$O(|q| \log \frac{n}{\uparrow})$

DB size

$O(|q| \log n) + O(\#\text{matches})$

$O(\log n) + O(|q|)$

how to do this ?