Bioinformatics

Finding Repetitions in Biological Sequences

- Repetitions in biological sequences are categorized into the following three main categories:
  - repetitions of limited length occurring locally, and whose function is known,
  - repetitions of limited length occurring throughout the sequence, the operation of which is not fully known,
  - Long length structured repetitions whose function has not been determined.

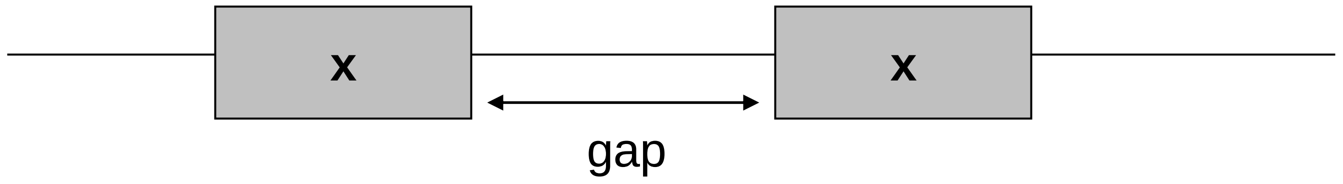
Examples of repetitions

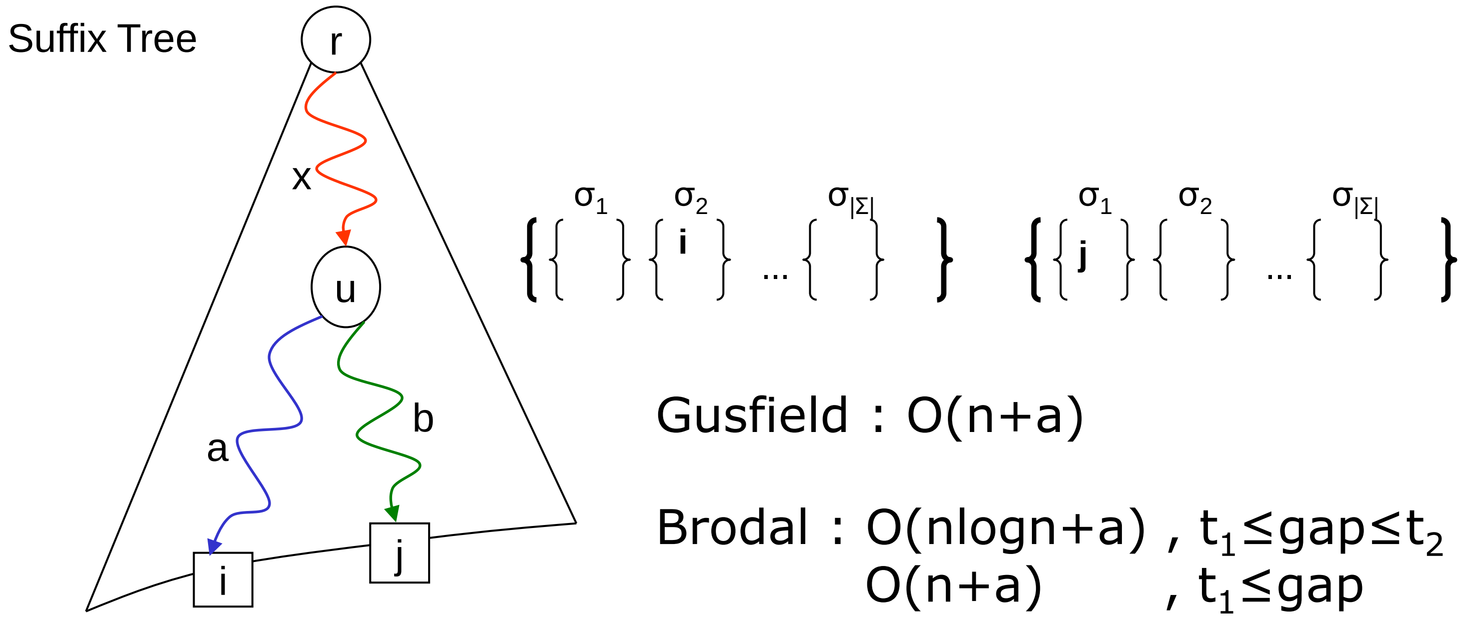
- 1st class:
  - complementary regressions in DNA & RNA sequences, which regulate DNA transcription,
  - nested complementary palindromes in tRNA sequences,
- 2nd class:
  - **tandem repeats**,
  - **satellite DNA**, (**micro** & **mini satellite DNA**)
- 3rd class:
  - **SINE-Short Interspersed Nuclear Sequences (π.χ.:** *Alu family)*
  - **LINE-Long Interspersed Nuclear Sequences**.

Palindromes

- ***Definition:*** *A palindrome is the occurrence of the sub-string read as the same in both directions (left to right and right to left):xyaayx*
- ***Ορισμός:*** *A palindrome in a DNA or RNA sequence is called a complemented palindrome if it results from replacing all characters from beginning to middle with corresponding complementary bases: agctcgcgagct*

Maximal Pairs

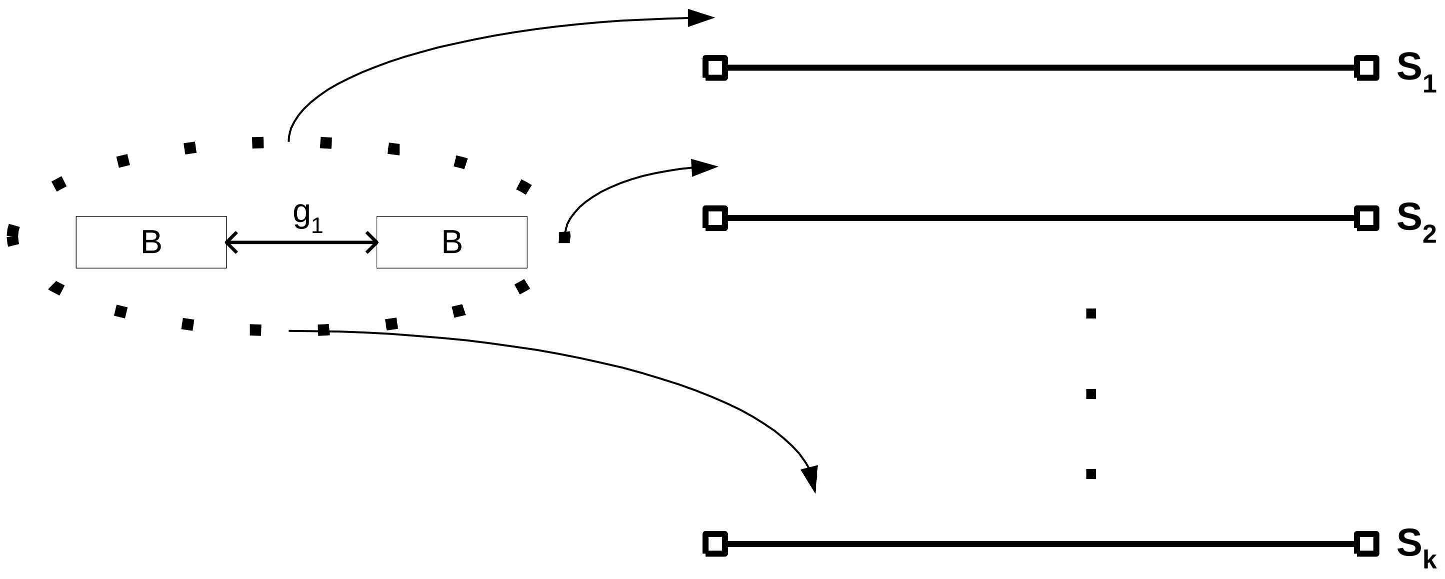


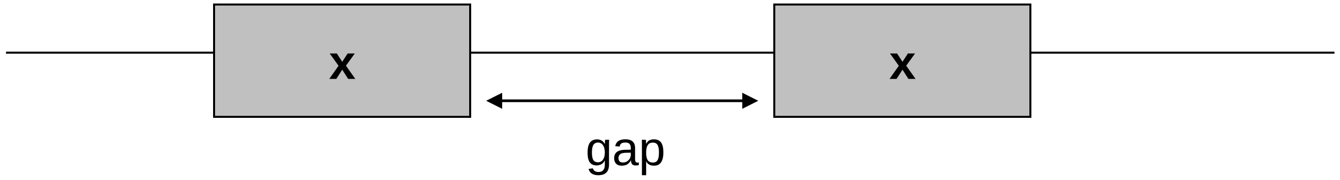


Gusfield : $O(n+a)$

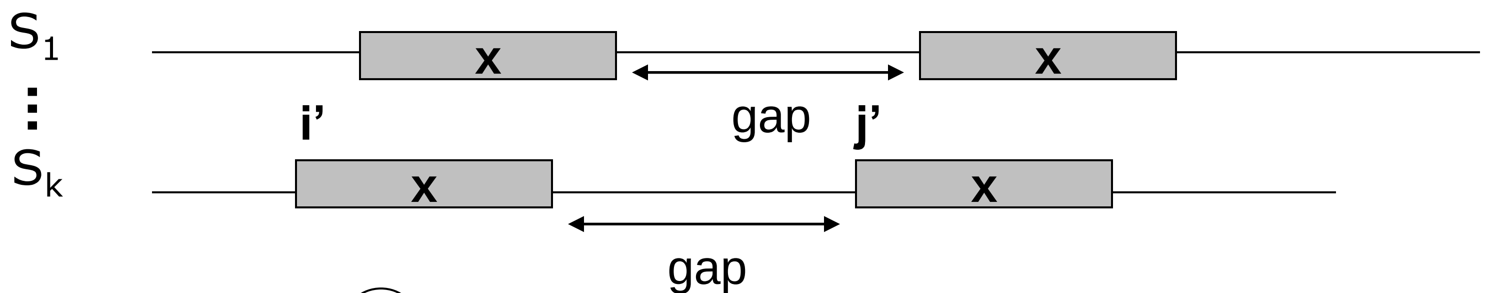
Brodal : $O(n\log n+a)$, $t_1 \leq gap \leq t_2$
$O(n+a)$, $t_1 \leq gap$

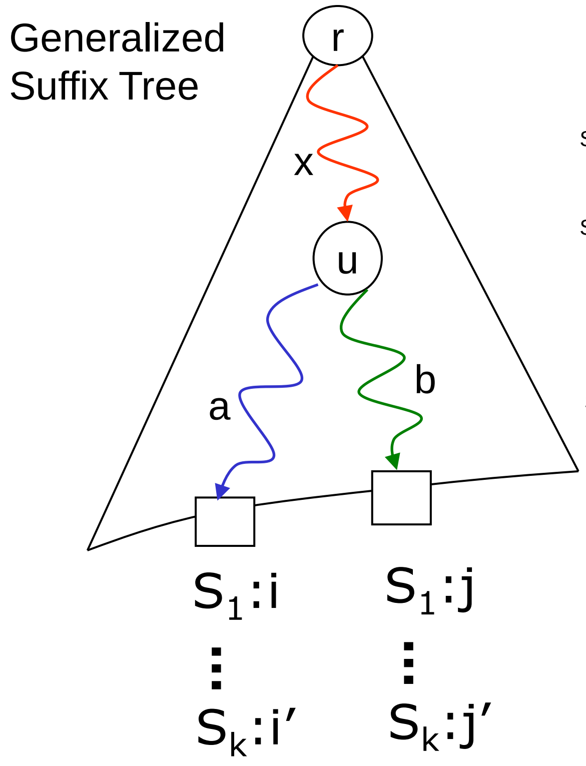
Maximal Pairs in Multiple Strings

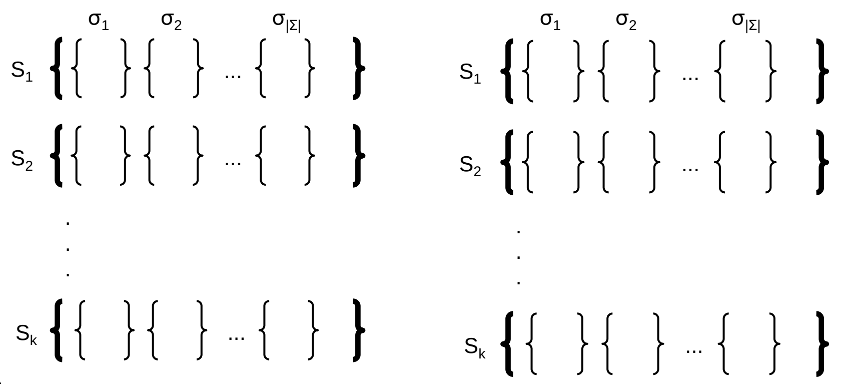




Maximal Pairs in Multiple Strings

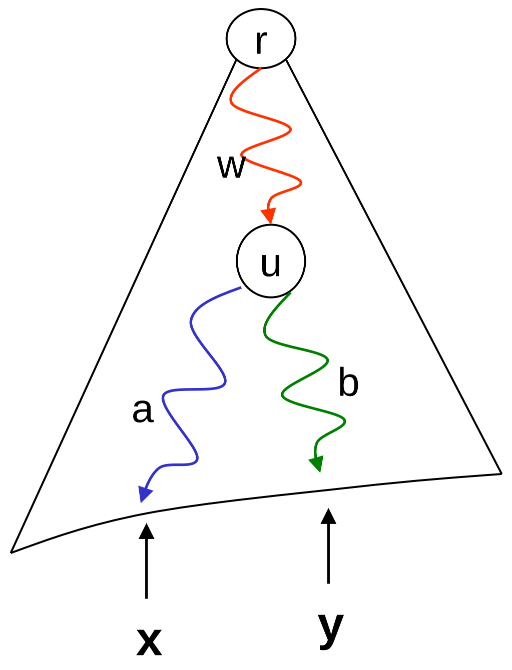




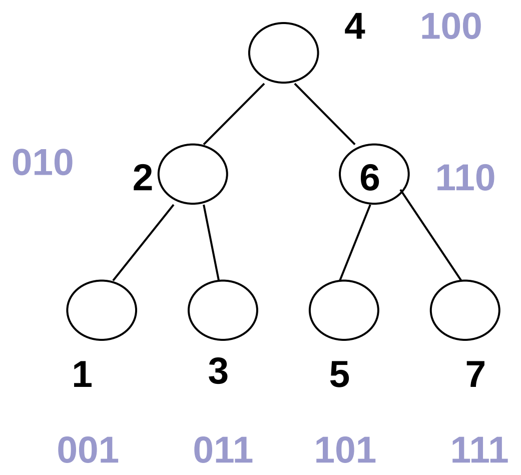


O(nlogn$^+$ak) , gap ≤t_1
O(n+a) , gap unbounded

Nearest Common Ancestor & Suffix Tree



nca(x,y)=u in time O(1)



nca(001, 101)=leftmost$_1$(XOR(001, 101)) = 100 =100

nca(001, 111)= leftmost$_1$(XOR(001, 111)) = 110 =100

nca(011, 010)=leftmost$_1$(XOR(011, 010) = 010

Longest Common Extension - LCE) :

Two strings S1 and S2 of total length n are given (in an initial preprocessing phase) and, a long sequence of queries for position pairs (i,j)

For each specified pair of pointers (i, j), we need to find the length of the longest substring of S1 starting at position i that matches a substring of S2 starting at position j. That is, we need to find the length of the longest prefix of the suffix i of S1 that matches a prefix of the suffix j of S2.

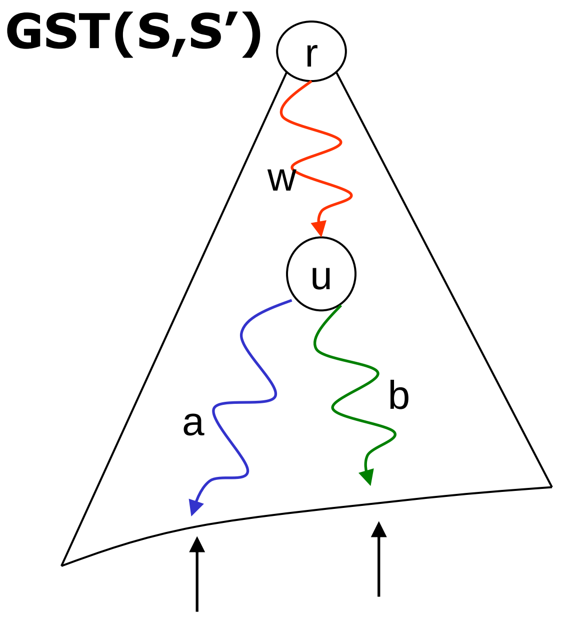
Suggestion: Simple use of generalized suffix tree and NCA processing

From Dan Gusfield, **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology Cambridge University Press**

Maximal Palindromes

αvvα **aabactgaaccaat**

taaccaagtcabaa



From Dan Gusfield, **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology Cambridge University Press**

Maximal Palindromes

There is a simple linear time method for finding all maximum even length palindromes in S (for a redundant corresponding algorithm):

1. In linear time, create the reverse string Sr from S and preprocess the two strings so that any Longest Common Extension query can be resolved in fixed time.
2. For each q from 1 to n - 1, solve the Longest Common Extension query for the pair (q+1, n-q+1) in S and Sr, respectively. If the extension has a nonzero length k, then there is a maximum palindrome of radius k centered on q.

The method takes O(n) time since the suffix tree can be created and preprocessed for Longest Common Extenion queries at this time interval, and each of the extension queries is resolved in fixed time.

From Dan Gusfield, **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology Cambridge University Press**

Exact Matching with wild cards

text

pattern * * * * * *

acgtttaacctttgagtttgggcv

at**