7.10 All-pairs suffix-prefix matching

Kim, Taehee

All pairs suffix-prefix matching

Can be used in implementing fast approximation algorithms for the **shortest superstring problem** (will not be discussed in this chapter)

In this chapter, we will use this algorithm to extrapolate from the number of ACRs observed in the set of ESTs.

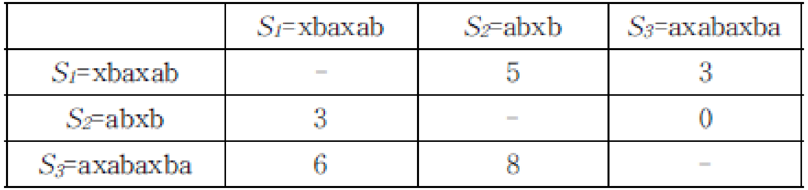
Terms

ACR ; Ancient Conserved Regions <= region, occurs in common between close but different two species
EST ; Expressed Sequence tag <= STS, that came from genes rather than parts of intergene DNA
STS ; Sequence Tagged Site <= a DNA string of length 200-300 nucleotides whose right and left ends, of length 20-30 nucleotides each, occur only once in the entire genome

What is all-pairs suffix-prefix problem?

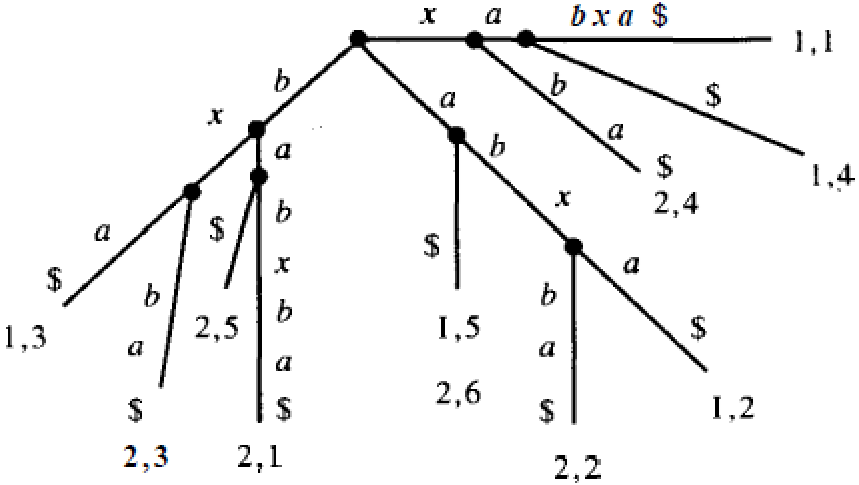
Def) Given two strings S1 and S2, any suffix of S1 that matches a prefix of S2, is called suffix-prefix match of S1 and S2.
 ex) ACGT - CGTC

Def) Given a collection of strings S = S1, S2, … , Sn of total length m, the all-pairs suffix prefix problem is the problem of finding, for each ordered pair Si, Sj in S, the longest suffix-prefix matching of them.
 ex)



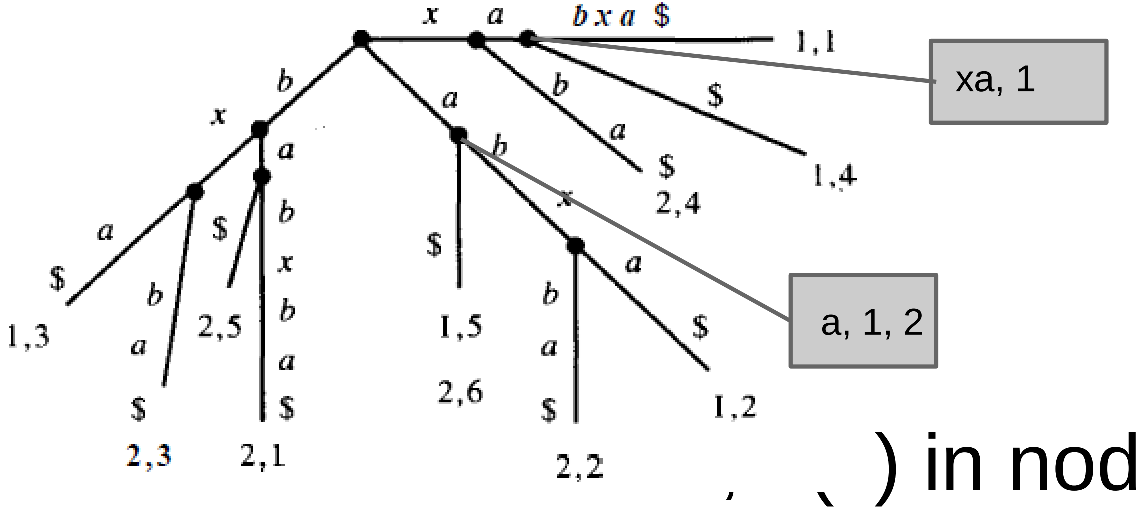
	S_1=xbaxab	S_2=abxb	S_3=axabaxba
S_1=xbaxab	–	5	3
S_2=abxb	3	–	0
S_3=axabaxba	6	8	–

Terminate Edge



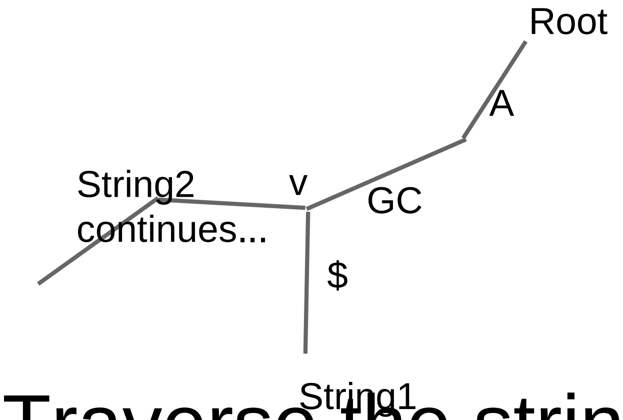
E string termination symbol.

Terminate Edge



V) in node v which is an internal node of terminate edge

all-pairs SP problem, in linear time



Traverse the string

We also will maintain a stack for each string...

Proof of linear time consumption

Total number of indices in all the lists=>O(m)
Total number of edges in suffix tree=>O(m)

push/pop is associated at most once on each leaf, total count is equal to the number of indices, push/pop consume constant time
we need to travel every edge to acquire information, traversing one edge consume constant time
O(1*m)+O(1*m) = O(m)

Extension

Maintain a linked list on stacks, so that you don’t have to find ‘v’s then access each stack.