Εισαγωγή στη Βιοπληροφορική

Μάθημα Ελεύθερης Επιλογής Τομέα Λογικού των Υπολογιστών

Εαρινό Εξάμηνο

Ορισμένα σημεία διαφανειών είναι από το βιβλίο του

Dan Gusfield, **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology**

Εφαρμογές Aho-Corasick(1)

Exact pattern matching with do not cares (‘*’) character

Έστω T μια συμβολοσειρά με n χαρακτήρες και P μια συμβολοσειρά με k-1 do not care ('*') χαρακτήρες συνολικού μήκους m.

Αλγόριθμος

0. Έστω C ένας πίνακας ακεραίων μήκους n αρχικοποιημένος σε μηδενικά.

1. Έστω *P* = ***{p1, p2,...,pk)*** το (πολυ-)σύνολο των υποσυμβολοσειρών του ***P*** που δεν περιέχουν χαρακτήρες μπαλαντέρ. Έστω l1, 12, . . . , lk είναι οι αρχικές θέσεις στο P καθεμιάς από αυτές τις υποσυμβολοσειρές (l1=1).

2. Χρησιμοποιώντας τον αλγόριθμο Aho-Corasick βρείτε για κάθε συμβολοσειρά ***Pi*** στο *P*, όλες τις αρχικές θέσεις του ***Pi*** στο T. Για κάθε θέση j του ***Pi*** στο T, αυξήστε τον αριθμό στο κελί j - ***li*** + ***1*** του ***C*** κατά ένα.

3. Σάρωση του πίνακα C για εύρεση κελιών με τιμή ***k.*** Υπάρχει εμφάνιση του P στο T ξεκινώντας από τη θέση ***p*** εάν και μόνο εάν ***C (p) = k.***

Εφαρμογές Aho-Corasick(2)

Pattern matching δύο διαστάσεων

Έστω T ένα κείμενο δύο διαστάσεων με n=n1 x n2 κελιά και P ένα πρότυπο δύο διαστάσεων m=m1 x m2 κελιών. Θέλουμε να εντοπίσουμε όλες τις εμφανίσεις του P στο T.

Η μέθοδος χωρίζεται σε δύο φάσεις.

Στην πρώτη φάση, αναζητήστε όλες τις εμφανίσεις καθεμιάς από τις σειρές του P μεταξύ των σειρών του T. Για να το κάνετε αυτό, προσθέστε ένα δείκτη τέλους γραμμής (κάποιος χαρακτήρας που δεν υπάρχει στο αλφάβητο) σε κάθε γραμμή του *T* και συνενώστε αυτές τις γραμμές σε μία συμβολοσειρά κειμένου T' μήκους O(n).

Στη συνέχεια, αντιμετωπίζοντας κάθε γραμμή του P ως ξεχωριστό μοτίβο, χρησιμοποιήστε τον αλγόριθμο Aho-Corasick για να αναζητήσετε όλες τις εμφανίσεις στο T' οποιασδήποτε σειράς του P.

Ως εκ τούτου, η πρώτη φάση προσδιορίζει όλες τις εμφανίσεις πλήρων σειρών P και παίρνει χρόνο O(n + m).

Εφαρμογές Aho-Corasick(2)

Κάθε φορά που εντοπίζεται εμφάνιση της γραμμής i του P ξεκινώντας από τη θέση (p, q) του T, γράψτε τον αριθμό i στη θέση (p, q) ενός άλλου πίνακα M με τις ίδιες διαστάσεις με το T. Επειδή κάθε γραμμή του P θεωρείται ξεχωριστή και επειδή το P είναι ορθογώνιο, το πολύ ένας αριθμός θα γραφτεί σε οποιοδήποτε κελί του M.

Στη δεύτερη φάση, σαρώστε κάθε στήλη του M, αναζητώντας μια εμφάνιση της συμβολοσειράς 1,2, . . . , m1 σε διαδοχικά κελιά μίας στήλης.

Αυτό δίνει μια λύση O(n+m) αν χρησιμοποιηθεί αλγόριθμος linear time pattern matching σε κάθε στήλη, ανεξάρτητα από το αλφάβητο (Z-algorithm, Knuth Morris Pratt)

Τώρα ας υποθέσουμε ότι οι σειρές του P δεν είναι όλες διακριτές. Αρκεί να εντοπίσουμε όλες τις πανομοιότυπες σειρές του P και να τους δώσουμε μια κοινή ετικέτα.

Τεχνικές Ανάλυσης και Σύγκρισης Ακολουθιών Βιολογικών Δεδομένων

- Δέντρο Επιθεμάτων- Suffix Tree
- Γενικευμένο Δέντρο Επιθεμάτων -Generalized Suffix Tree
- Εφαρμογές σε Προβλήματα Μοριακής Βιολογίας

Βασικοί Ορισμοί

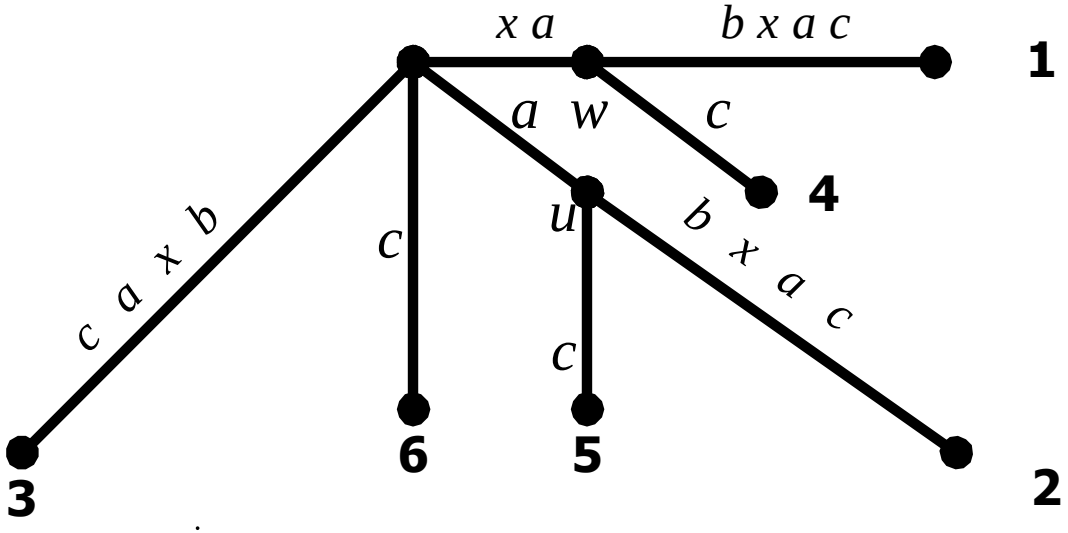
- Συμβολοσειρά-string: x=x[1]x[2].....x[n], x[i]$\in\Sigma$ & |x|=n
  x= acgttaaaca, |x|=10 & Σ={a,c,g,t}

- Κενή συμβολοσειρά: ε
- Υπο-συμβολοσειρά-substring w: x=uwv
- Πρόθεμα –Prefix w: x=wu
- Επίθεμα-Suffix w: x=uw
- Κάθε συμβολοσειρά S, μήκους |S|=m, έχει m δυνατά μη κενά επιθέματα που είναι τα ακόλουθα: S[1…m], S[2…m], …. S[m-1…m] και S[m].
- Παράδειγμα "sequence" : *sequence, equence, quence, uence, ence, nce, ce, e.*

Το Δέντρο Επιθεμάτων
Suffix Tree

Ορισμός: *«αποθηκεύει όλα τα δυνατά επιθέματα μιας συμβολοσειράς S».*

p= xabxac

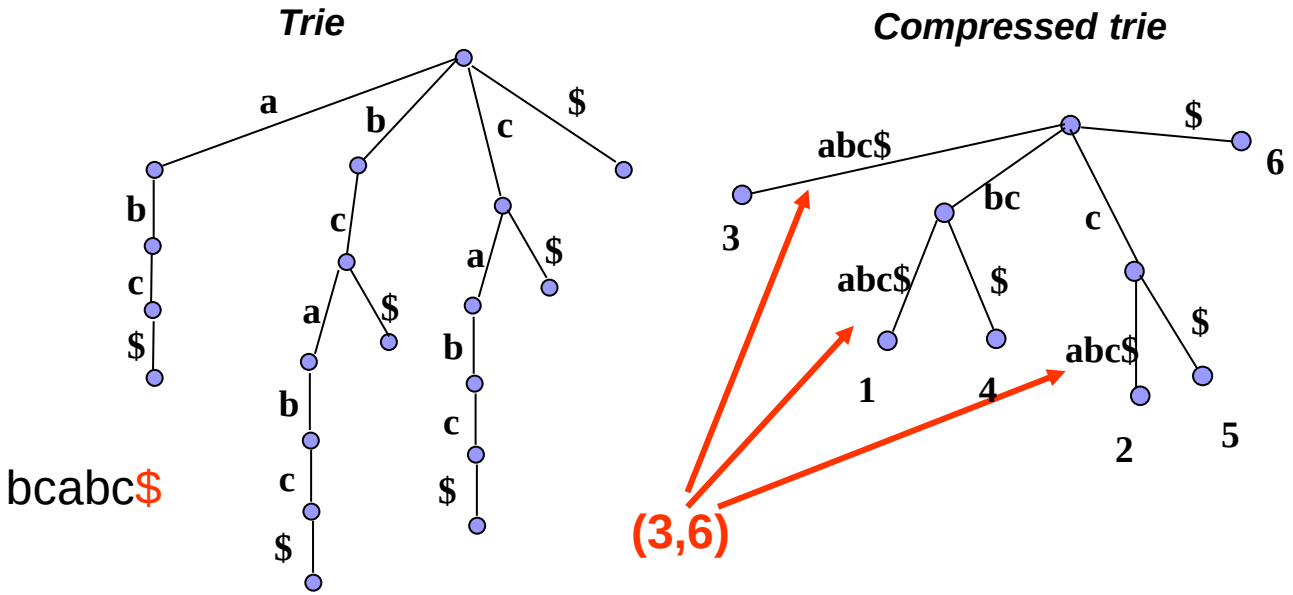


Ορισμός Ένα δέντρο επιθεμάτων **T** για μια συμβολοσειρά χαρακτήρων n χαρακτήρων είναι ένα rooted κατευθυνόμενο δέντρο με ακριβώς n φύλλα αριθμημένα από το 1 έως το n**.** Κάθε εσωτερικός κόμβος, εκτός από τη ρίζα, έχει τουλάχιστον δύο παιδιά και κάθε ακμή επισημαίνεται με μια μη κενή υποσυμβολοσειρά. Δεν είναι εφικτό από ένα κόμβο να υπάρχουν δύο ακμές που ξεκινούν από τον ίδιο χαρακτήρα. Το βασικό χαρακτηριστικό του δέντρου επιθημάτων είναι ότι για οποιοδήποτε φύλλο i**,** η συνένωση των ακμών-ετικετών στη διαδρομή από τη ρίζα στο φύλλο i είναι ίση με το επίθεμα της συμβολοσειράς που ξεκινά από την θέση i.

από το βιβλίο: Dan Gusfield Algorithms on Strings, Trees and Sequences, Cambridge University Press,

Suffix Tree

Ορισμός: Το suffix tree μιας συμβολοσειράς S[1…n] είναι ένα *συμπαγές* **trie** που περιέχει ως κλειδιά όλα τα επιθέματα S[i…n], 1≤i≤n.



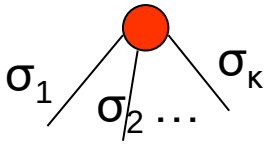
Trie

Ορισμός:

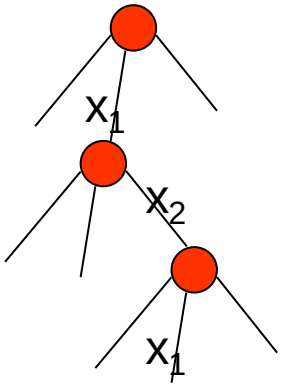
Έστω σύμπαν U= $\Sigma^0 \cup \ldots \cup \Sigma^l$ για αλφάβητο Σ και l>0.

$$(x \in U : x = d_1 d_2 \cdots d_l)\ S \subseteq U :$$

Trie καλείται το κ-δικό δένδρο (κ=|Σ|) το οποίο περιέχει όλα τα προθέματα των στοιχείων του S. Κάθε επίπεδο του δένδρου αντιστοιχίζεται και σε ένα d_i (d_1→ρίζα). Κάθε στοιχείο $x=x_1x_2\ldots x_l$ τοποθετείται στο υποδένδρο $x_1 \rightarrow x_2 \rightarrow \ldots$.

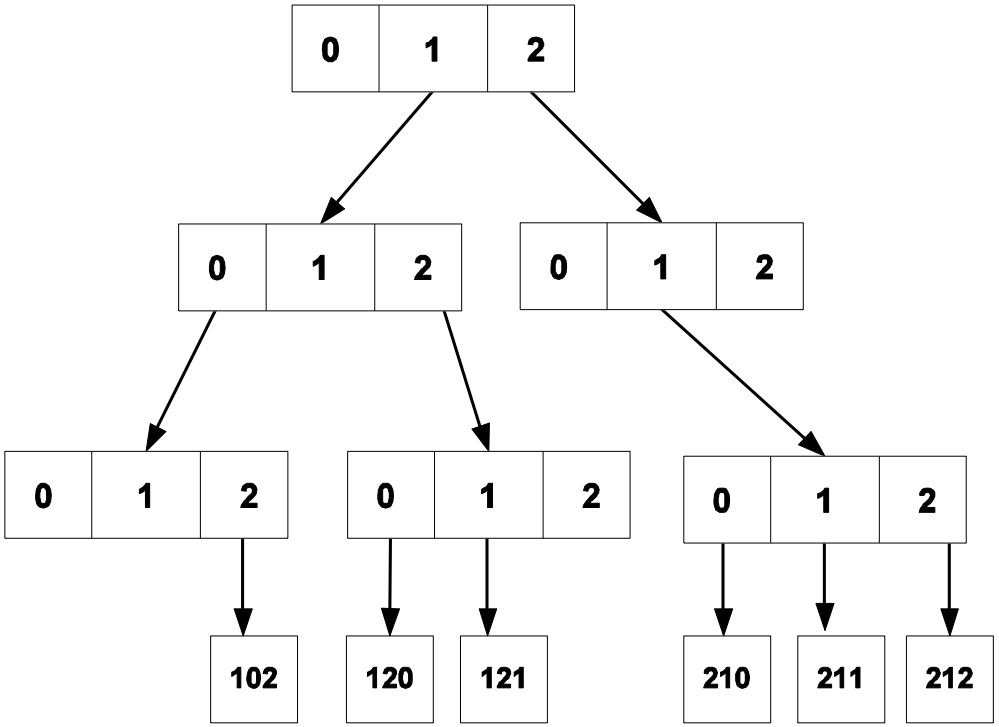


$(\sigma_i \in \Sigma)$



Trie - παράδειγμα

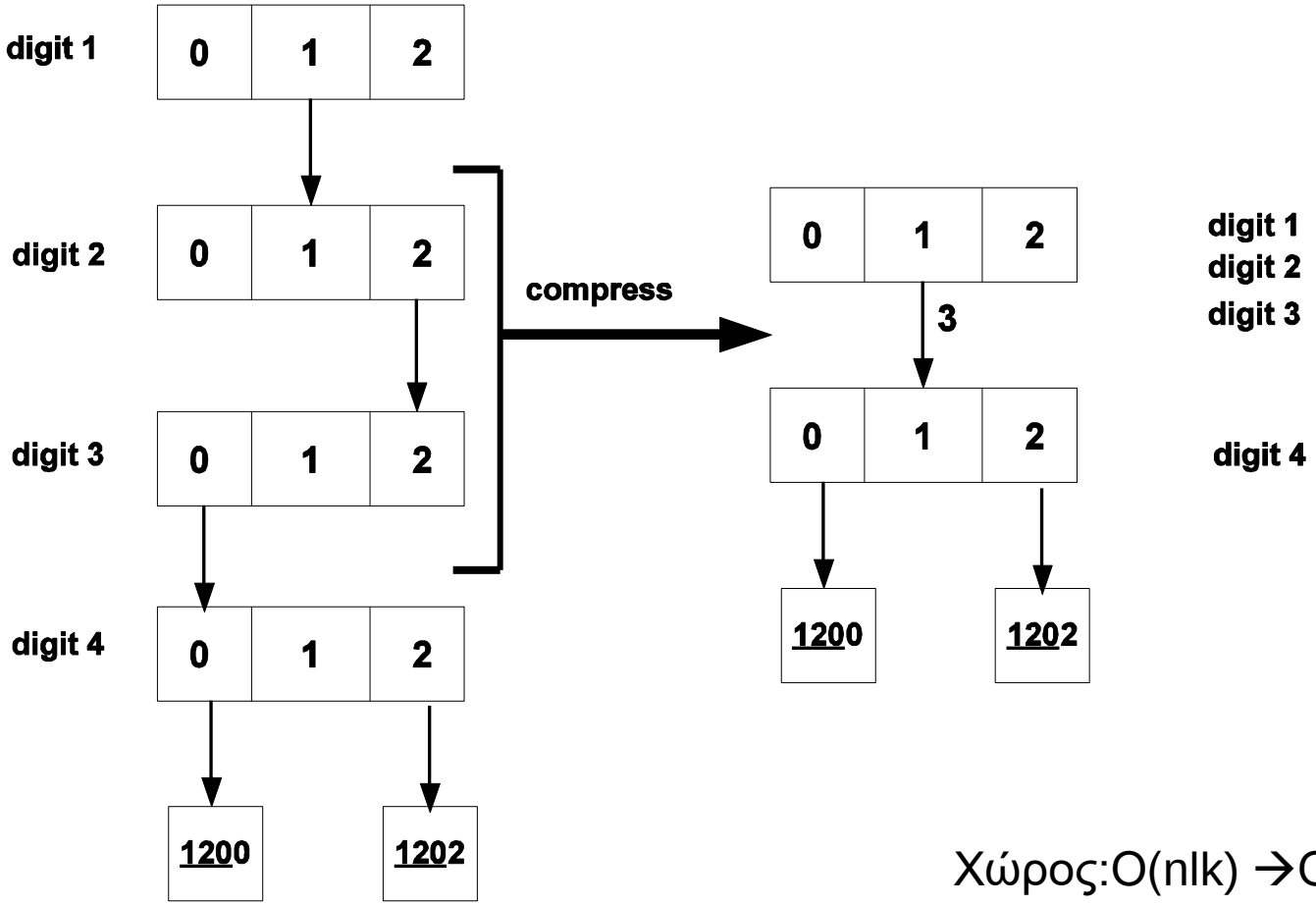
S={ 102, 120, 121, 212, 211, 120}, Σ={0,1,2}



Χρόνος ins/del/search : O(l)

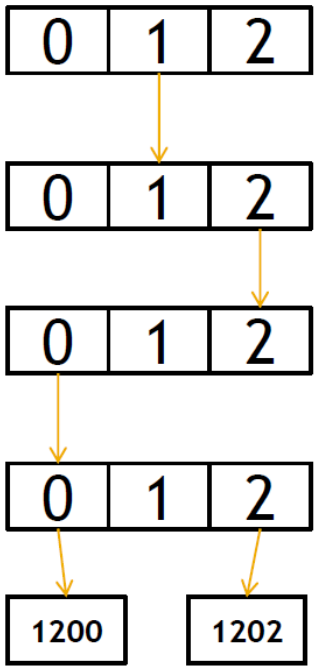
Χώρος: O(nlk)

Compressed Trie

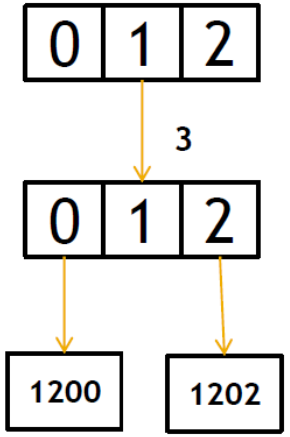


Χώρος:O(nlk) →O(nk)=O(n)

Compressed Trie - example



TRIE



ΣΥΜΠΑΓΕΣ
TRIE

Δημιουργία του δέντρου

- Weiner’s algorithm [FOCS, 1973]

  Knuth ”The algorithm of 1973”
- McCreight’s algorithm [JACM, 1976]

  Linear time and space
- Ukkonnen's algorithm [Algorithmica, 1995]

  Linear time and less space
- Farach’s algorithm [FOCS 1997],

  έδωσε τον πρώτο γραμμικό χρόνο για κάθε αλφάβητο

Implementation (https://en.wikipedia.org/wiki/Suffix_tree)

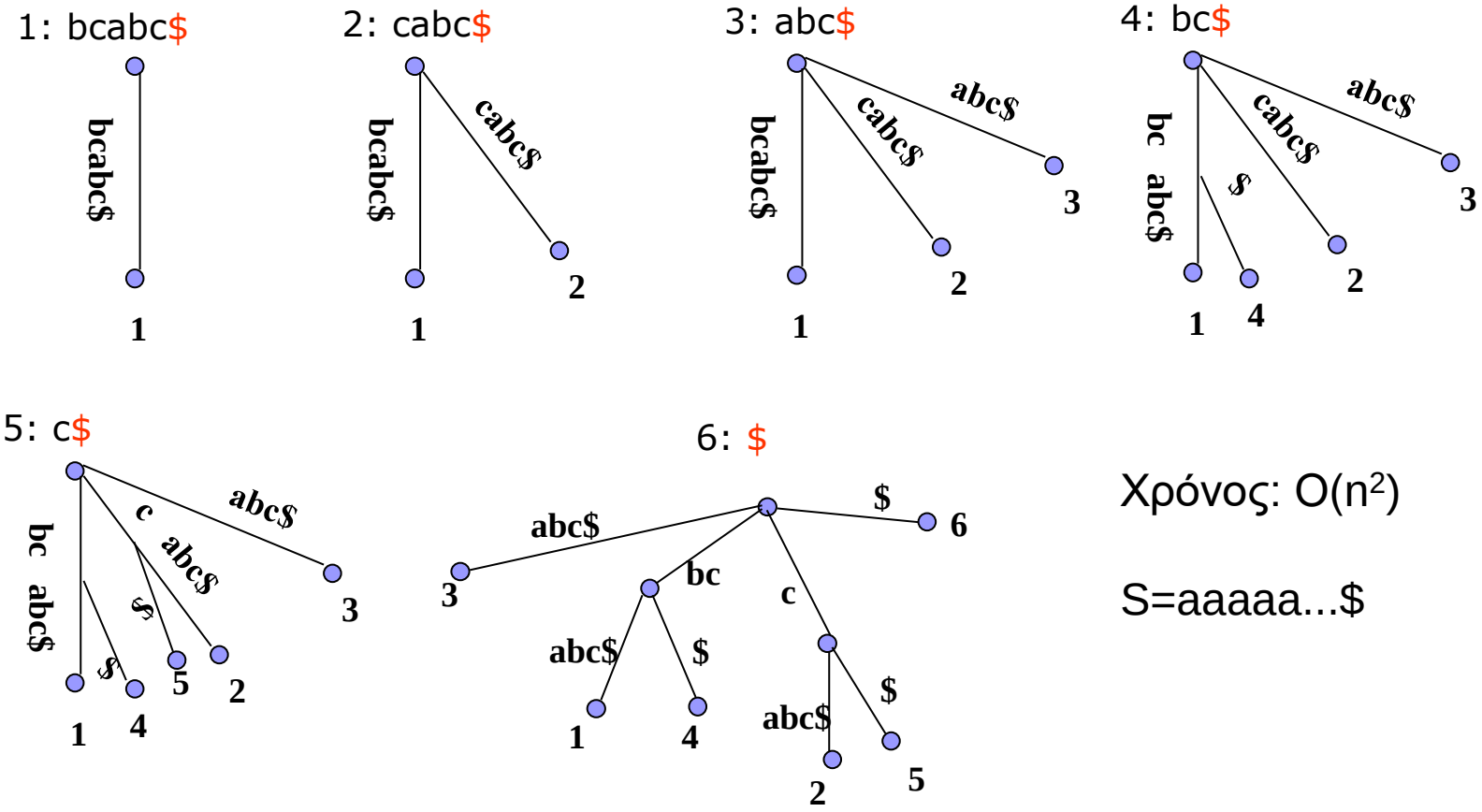
	Lookup	Insertion	Traversal
Sibling lists / unsorted arrays	$O(\sigma)$	$\Theta(1)$	$\Theta(1)$
Bitwise sibling trees	$O(\log \sigma)$	$\Theta(1)$	$\Theta(1)$
Hash maps	$\Theta(1)$	$\Theta(1)$	$O(\sigma)$
Balanced search tree	$O(\log \sigma)$	$O(\log \sigma)$	$O(1)$
Sorted arrays	$O(\log \sigma)$	$O(\sigma)$	$O(1)$
Hash maps + sibling lists	$O(1)$	$O(1)$	$O(1)$

Κατασκευή του Δέντρου Επιθεμάτων

- Μια απλοϊκή θεώρηση:
  - Ένθεση μιας πλευράς στο δέντρο για το επίθεμα S[1…m]$,
  - Διαδοχική ένθεση των επιθεμάτων S[i…m]$, για i=2→m.

Naïve Κατασκευή

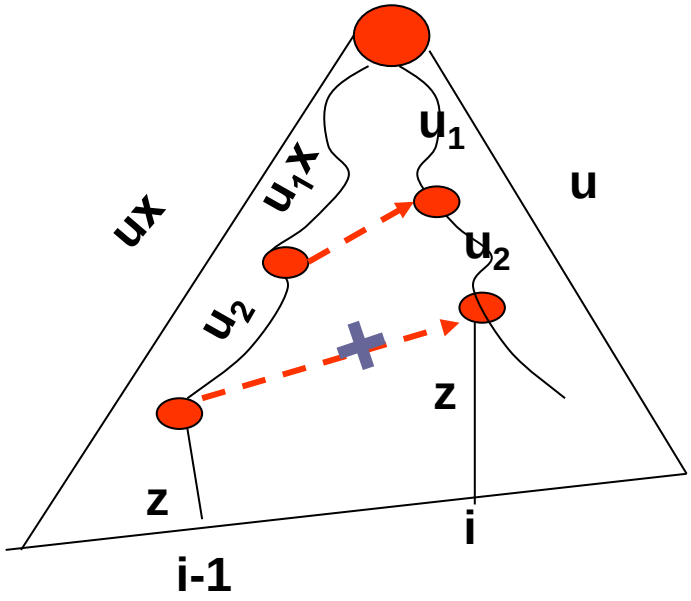
S=bcabc$



Χρόνος: $O(n^2)$

S=aaaaa...$

Suffix Links – Speed Up



i-1	i		
x	u		z
x	u_1	u_2	z

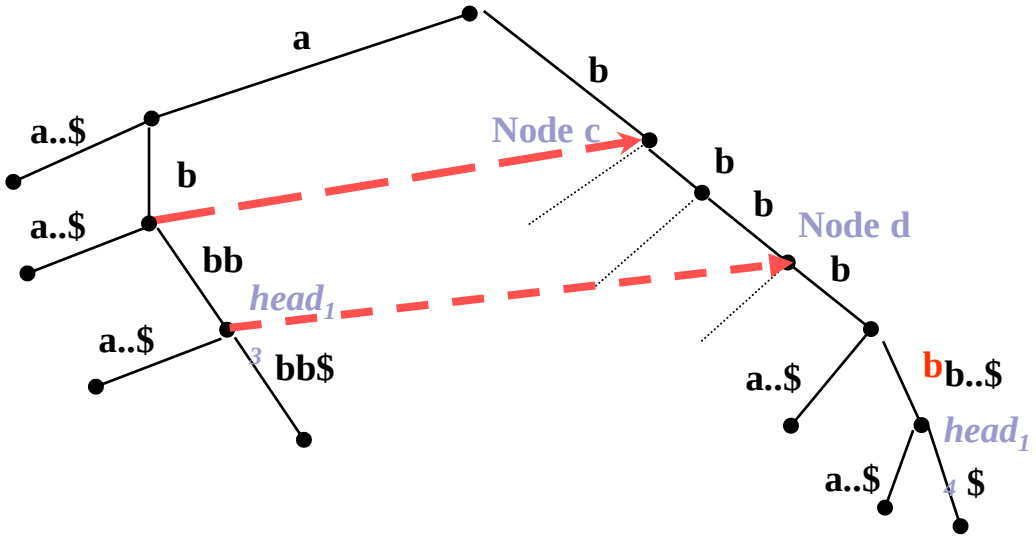
res(i)=|u2|+|z|

res(i+1)|<=res(i)-int_i

Scan cost=length($head_i$)-length($head_{i-1}$)

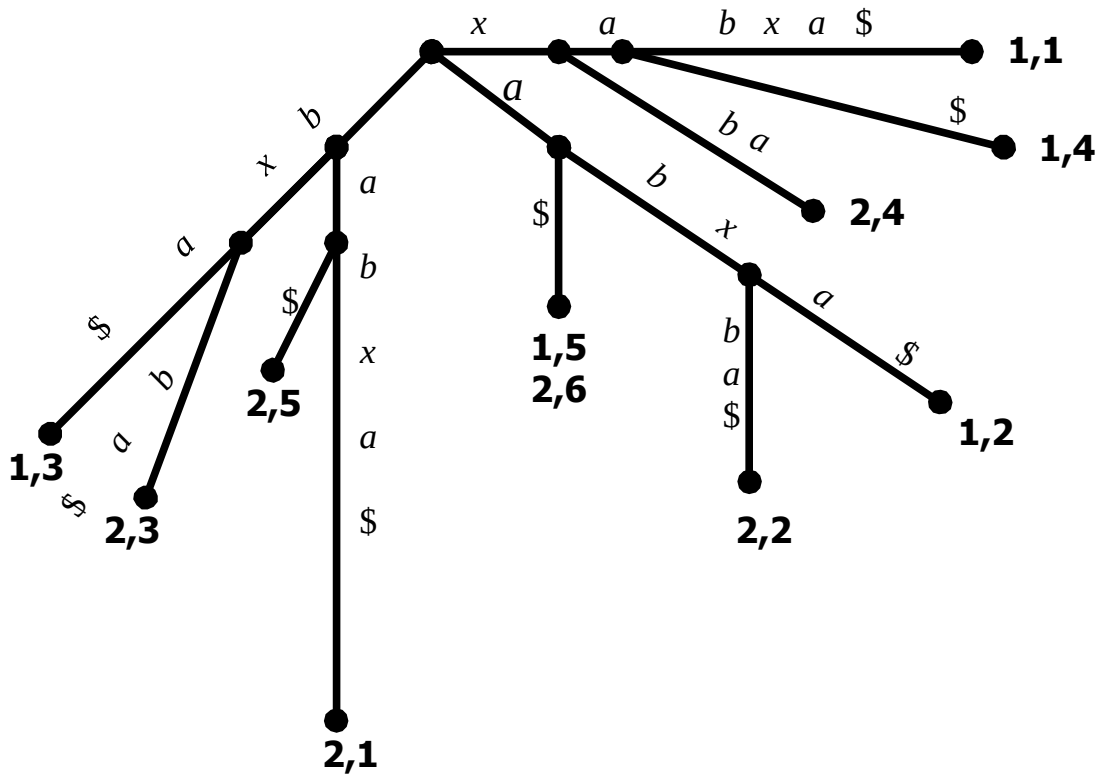
Suffix Links – Speed Up

S=bbbbbababbbaabbbbb$
i-1:**Inserted** S[13..19]=abbbbb$
x=a, u=bbb, z=bb$
u_1=b, u_2=bb
i: **Insert** S[14..19]=bbbbb$



Το Γενικευμένο Δέντρο Επιθεμάτων Generalized Suffix Tree

Ορισμός: *«Το **Γενικευμένο Δέντρο Επιθεμάτων**, αποτελεί ένα Δέντρο Επιθεμάτων το οποίο αποθηκεύει όλα τα δυνατά επιθέματα ενός συνόλου συμβολοσειρών* $S=\{S_1,S_2,...S_n\}$*».*



x_1 = xabxa$

x_2 = babxba$

Κατασκευή του Γενικευμένου Δέντρου Επιθεμάτων

- Μια απλοϊκή θεώρηση:
  - Διαδοχική ένθεση όλων των επιθεμάτων των συμβολοσειρών εισόδου.

Εφαρμογές στην ανάλυση ακολουθιών βιολογικών δεδομένων

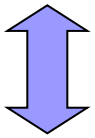
- Ακριβής Εύρεση Προτύπου - Exact pattern matching problem
- Ακριβής Εύρεση Πολλαπλών Προτύπων- Σύγκριση με το Αυτόματο Aho- Corasick
- Μέγιστη κοινή υποσυμβολοσειρά 2 ακολουθιών- Longest Common Substring Problem
- DNA Contamination Problem
- Εύρεση Κοινών Μοτίβων σε 2 ή περισσότερες Βιολογικές Ακολουθίες
- Εύρεση Επαναλήψεων σε Βιολογικές Ακολουθίες

Ακριβής Εύρεση Προτύπου - Exact pattern matching problem

- Κατασκευή του Δ.Ε. για την ακολουθία εισόδου T σε O(|T|) χρόνο,
- Ξεκινώντας από τη ρίζα, σύγκρινε έναν προς έναν τους χαρακτήρες του P, ακολουθώντας το κατάλληλο μονοπάτι.
  - Εάν εμφανιστεί κάποιο μη-ταίριασμα, τότε το πρότυπο δεν εμφανίζεται στην ακολουθία,
  - διαφορετικά ανέφερε ως απάντηση όλα τα φύλλα που βρίσκονται κάτω από τον κόμβο του τελευταίου χαρακτήρα του P.
- Η αναζήτηση στοιχίζει O(m+k) χρόνο, |P|=m & k: το πλήθος εμφανίσεων του P στο T.

Ακριβής Εύρεση Πολλαπλών Προτύπων

- Κατασκευή του Δ.Ε. για την ακολουθία εισόδου Τ σε Ο(|Τ|) χρόνο,
- Ξεκινώντας από τη ρίζα, αναζητούμε διαδοχικά όπως και στην προηγούμενη εφαρμογή το σύνολο των προτύπων $P=\{P_1, P_2, ..\}$
- Η αναζήτηση στοιχίζει $O(n+|P|+ k_p)$ χρόνο, $|P|= |P_1| +|P_2|+.....$ & k_p: το πλήθος εμφανίσεων των προτύπων στο Τ.



- Κατασκευή του Αυτομάτου σε χρόνο Ο(|P|)
- Αναζήτηση: Ο(m+k)