Εισαγωγή στη Βιοπληροφορική

Μερικές διαφάνειες από διαφάνειες βιβλίου Neil C. Jones, Pavel A. Pevzner, Εισαγωγή στους Αλγορίθμους Βιοπληροφορικής

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

(http://bix.ucsd.edu/bioalgorithms/slides.php)

RABIN-KARP ALGORITHM

Karp, Richard M.; Rabin, Michael O. (March 1987). "Efficient randomized pattern-matching algorithms". IBM Journal of Research and Development. 31 (2): 249–260. CiteSeerX 10.1.1.86.9502. doi:10.1147/rd.312.0249.

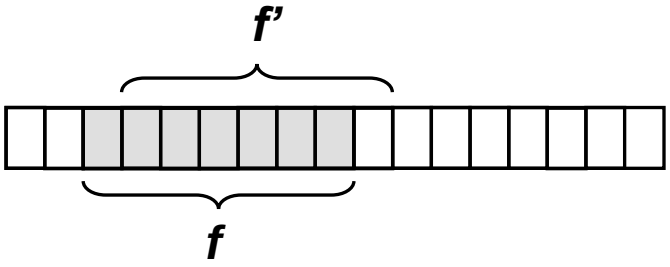
Slides from Can Alkan

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/

using a presentation from Saltenis.

Fingerprint idea

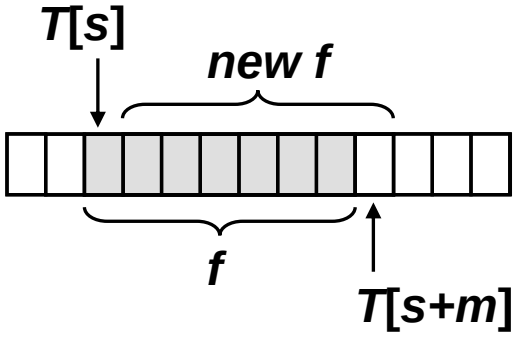
- Assume:
  - We can compute a fingerprint f(P) of P in O(m) time.
  - If f(P) ≠ f(T[s .. s+m–1]), then P ≠ T[s .. s+m–1]
  - We can compare fingerprints in O(1)
  - We can compute f' = f(T[s+1.. s+m]) from f(T[s .. s+m–1]), in O(1)



Algorithm with Fingerprints

- Let the alphabet Σ ={0,1,2,3,4,5,6,7,8,9}
- Let fingerprint to be just a decimal number, i.e., f("1045") = $1*10^3 + 0*10^2 + 4*10^1 + 5 = 1045$
- **Fingerprint-Search(T,P)**

```
01 fp ← compute f(P)
02 f ← compute f(T[0..m−1])
03 for s ← 0 to n − m do
04     if fp = f return s
05     f ← (f − T[s]*10^(m−1))*10 + T[s+m]
06 return −1
```



  - **Running time $2O(m) + O(n–m) = O(n)$**

Using a Hash Function

- Problem:
  - we can not assume we can do arithmetics with m-digits-long numbers in O(1) time
- Solution: Use a hash function h = f mod q
  - For example, if q = 7, h(“52”) = 52 mod 7 = 3
  - h(S1) ≠ h(S2) ⇒ S1 ≠ S2
  - But h(S1) = h(S2) does not imply S1=S2
    - For example, if q = 7, h(“73”) = 3, but “73” ≠ “52”
- Basic “mod q” arithmetics:
  - (a+b) mod q = (a mod q + b mod q) mod q
  - (a*b) mod q = (a mod q)*(b mod q) mod q

Rabin-Karp Algorithm

```
Rabin-Karp-Search(T,P)
01 q ← a prime larger than m
02 c ← 10^(m-1) mod q  // run a loop multiplying by 10 mod q
03 fp ← 0; ft ← 0
04 for i ← 0 to m-1  // preprocessing
05    fp ← (10*fp + P[i]) mod q
06    ft ← (10*ft + T[i]) mod q
07 for s ← 0 to n - m  // matching
08    if fp = ft then   // run a loop to compare strings
09       if P[0..m-1] = T[s..s+m-1] return s
10    ft ← ((ft - T[s]*c)*10 + T[s+m]) mod q
11 return -1
```

Semi-Numerical Algorithms

Wu, Sun; Manber, Udi (20–24 January 1992). Agrep -- a fast approximate pattern-matching tool. 1992 Winter USENIX Conference. San Francisco, California. CiteSeerX 10.1.1.89.5424

Udi Manber, Sun Wu. "Fast text search allowing errors." Communications of the ACM, 35(10): pp. 83–91, October 1992, doi:10.1145/135239.135244.

Ο αλγόριθμος Shift-Or

- Ο αλγόριθμος χρησιμοποιεί αριθμητικές τεχνικές:

1. Έστω για κάθε char c $\in$ Σ, το διάνυσμα S_c μεγέθους m=|P|, που αποθηκεύει τις εμφανίσεις του c μέσα στο πρότυπο (με 0 p προσδιορίζεται η εμφάνιση),

2. Ο πίνακας R[mxn]: bit-array όπου R[i,j] είναι 0 αν και μόνο αν οι πρώτοι i χαρακτήρες του P ταιριάζουν με τους i χαρακτήρες που τελειώνουν στο j-οστό χαρακτήρα του T.

3. $R_{j+1} = \text{Shift}\ (R_j\ )\ \text{OR}\ S_{T[j+1]}$

Ας δούμε τον αλγόριθμο στην πράξη

1ο βήμα: Υπολογίζω τα διανύσματα S_C για το πρότυπο x=gcagagag

	Sa	S_c	S_g	S_t
g	1	1	0	1
c	1	0	1	1
a	0	1	1	1
g	1	1	0	1
a	0	1	1	1
g	1	1	0	1
a	0	1	1	1
g	1	1	0	1

...Ας δούμε τον αλγόριθμο στην πράξη

2ο βήμα: Υπολογίζω τις τιμές του πίνακα R, σύμφωνα με τον τύπο:
R_{j+1} = Shift (R_j) Or $S_{T[j+1]}$

		1	2	3	4	5	6	7	8	9	10	11	12	13
		g	c	a	t	c	g	c	a	g	a	g	a	g
1	g	0	1	1	1	1	0	1	1	0	1	0	1	0
2	c	1	0	1	1	1	1	0	1	1	1	1	1	1
3	a	1	1	0	1	1	1	1	0	1	1	1	1	1
4	g	1	1	1	1	1	1	1	1	0	1	1	1	1
5	a	1	1	1	1	1	1	1	1	1	0	1	1	1
6	g	1	1	1	1	1	1	1	1	1	1	0	1	1
7	a	1	1	1	1	1	1	1	1	1	1	1	0	1
8	g	1	1	1	1	1	1	1	1	1	1	1	1	0

Ανάλυση του αλγορίθμου Shift-Or σε χρόνο

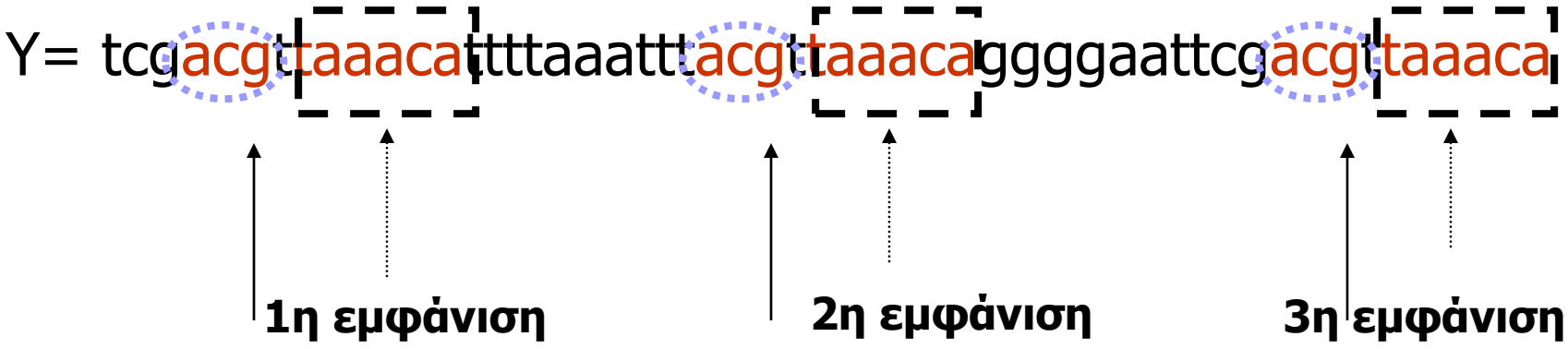
- Πολυπλοκότητα μεθόδου: O(n+m), όπου |T|=n & |P|=m

1. Σε χρόνο O(m*σ) υπολογίζω, τα διανύσματα S_C

Ακριβής Εύρεση για ένα σύνολο προτύπων

<u>Ορισμός:</u> *«έστω μια ακολουθία χαρακτήρων Υ. Αναζητούμε τις θέσεις εμφάνισης* ενός συνόλου προτύπων P μέσα *στην ακολουθία».*

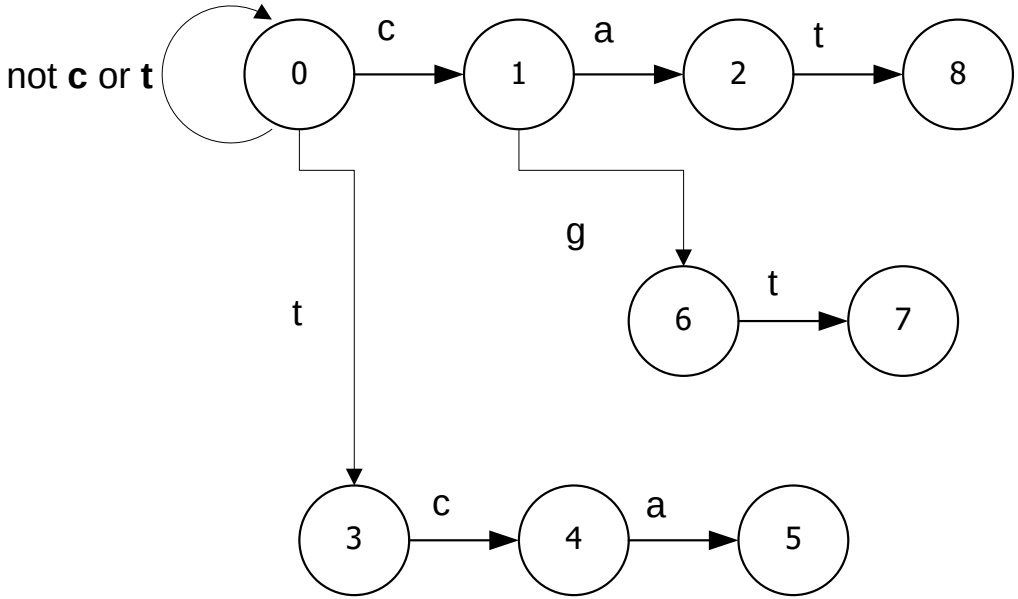
P= {acg, taaaca}



*Aho, Alfred V.; Corasick, Margaret J. (June 1975). "Efficient string matching: An aid to bibliographic search". Communications of the ACM. **18** (6): 333–340. doi:10.1145/360825.360855. MR 0371172. S2CID 207735784.*

Το Αυτόματο Aho-Corasick

Έστω P={ca, tca, cgt, cat}



goto function

Η Συνάρτηση “goto”

- g(s, α) = s’: το αυτόματο μεταπηδά στην κατάσταση s’και ο επόμενος χαρακτήρας της ακολουθίας διαβάζεται στην είσοδο,
- g(s, α) = fail, το αυτόματο μεταβαίνει στην κατάσταση s’=f(s) σύμφωνα με τη failure function. Η αναζήτηση συνεχίζεται με τρέχουσα κατάσταση την s’ και σύμβολο εισόδου το χαρακτήρα που ήδη έχει διαβαστεί στην είσοδο $\rightarrow$ α.

Η Συνάρτηση “failure-function”

- f(s) = 0, για κάθε κατάσταση s, βάθους 1.
- Για να υπολογίσεις το f(s), για κάθε κατάσταση s, βάθους d, θεώρησε όλες τις καταστάσεις r, βάθους d-1 :
  - Αν g(r,α) = fail, κάθε α, μην κάνεις τίποτα,
  - Διαφορετικά, για κάθε σύμβολο α, έτσι ώστε g(r,α)= s, τότε:
    - Θέσε state = f(r)
    - Εκτέλεσε την εντολή state $\leftarrow$ f(state), έως ότου g(state, α) $\neq$fail
- Θέσε f(s)= g(state, α)

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

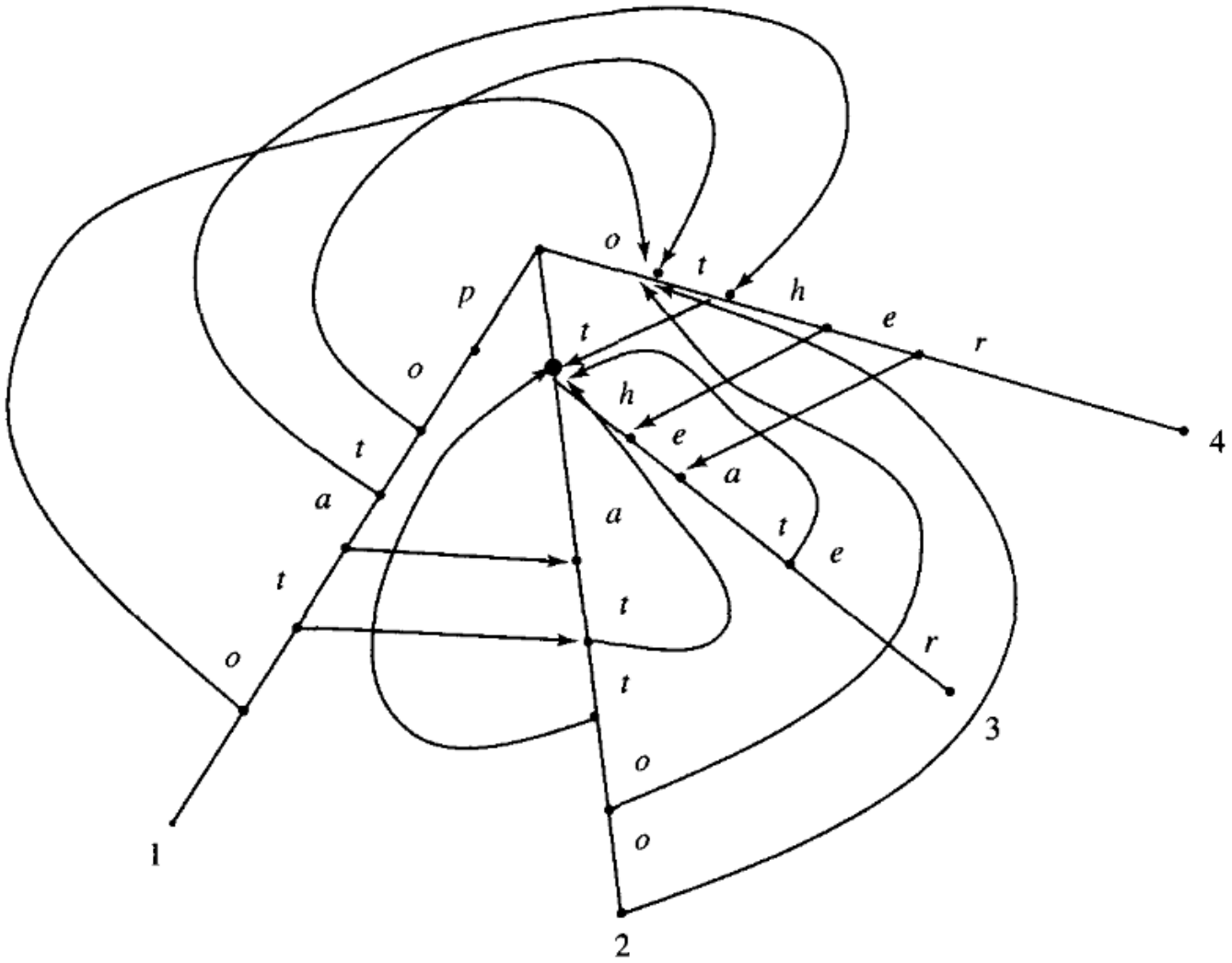


Figure 3.16: Keyword tree showing the failure links.

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

Ο αλγόριθμος Aho-Corasick (1)

- l:=1; c:=1; w:=root;
- ΕΠΑΝΑΛΑΒΕ
- ΕΝΟΣΩ υπάρχει ακμή (w,w') με ετικέτα T(c)
- ΕΑΝ ο κόμβος w' έχει ετικέτα i ανέφερε την εμφάνιση του pi
- w:=w'; και c:=c+1;
  - Διαφορετικά w:=failure_link(w); l:=c-βαθος(w).
- ΜΕΧΡΙ c>n

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

Ο αλγόριθμος Aho-Corasick (2)

P = {acatt, ca) and T = acatg
Αν πρότυπα είναι υποσυμβολοσειρά άλλων αλλαγή αλγορίθμου:

- l:=1; c:=1; w:=root;
- ΕΠΑΝΑΛΑΒΕ
- ΕΝΟΣΩ υπάρχει ακμή (w,w') με ετικέτα T(c)
- ΕΑΝ ο κόμβος w' έχει ετικέτα i ή υπάρχει αλυσίδα από failure links σε
- κόμβο με ετικέτα i, ανέφερε την εμφάνιση του pi
- w:=w'; και c:=c+1;
  - Διαφορετικά w:=failure_link(w); l:=c-βαθος(w).
- ΜΕΧΡΙ c>n

Αντικατάσταση αλυσίδας failure links με ένα output link.
Συνολικός χρόνος O(m)

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

Εφαρμογές εύρεσης προτύπων σε προβλήματα Βιοπληροφορικής

- Αναζήτηση ***Sequence-tagged-site (STS) & Expressed Sequence Tags (ESTs)*** σε ακολουθίες γονιδιωμάτων.
  - ***STS:*** τμήματα του DNA μήκους 200-300 νουκλεοτιδίων
  - ***ESTs:*** τμήματα mRNA & cDNA ακολουθίες που αντιπροσωπεύουν τα τμήματα κωδικοποίησης μιας πρωτεΐνης σε μια ακολουθία γονιδίων.

- Αναζήτηση transcription factors

- Αναζήτηση ***"κανονικών εκφράσεων"*** (regular expressions)
  - [ED]-[EN]-L-[SAN]-x-x-[DE]-x-E-L $\Rightarrow$ **ENLSSEDEEL**

  PROSITE