

# Ανεύρεση Κωδικού με Χρήση Μαρκοβιανών Αλυσίδων

Ασφάλεια Συστημάτων Υπολογιστών  
Γεώργιος Δροσάτος

---

Αναστασιάδου Μαγδαλινή   Γκαντίδης Χρήστος  
Ιανουάριος 2018

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Δημοκρίτειο Πανεπιστήμιο Θράκης

1. Περίληψη Μαρκοβιανών Αλυσίδων
2. Κατακερματισμός Κωδικών
3. Μέθοδοι Ανεύρεσης
4. Υλοποίηση Μεθόδου Μαρκοβιανών Αλυσίδων
5. Προτάσεις για Επιλογή Καλύτερων Κωδικών

# Περίληψη Μαρκοβιανών Αλυσίδων

---

## Ορισμός

Οι Μαρκοβιανές αλυσίδες, είναι μαθηματικές κατασκευές οι οποίες περιγράφουν την μετάβαση ενός συστήματος από μια κατάσταση σε όλες τις υπόλοιπες δυνατές καταστάσεις του συστήματος.

## Ορισμός

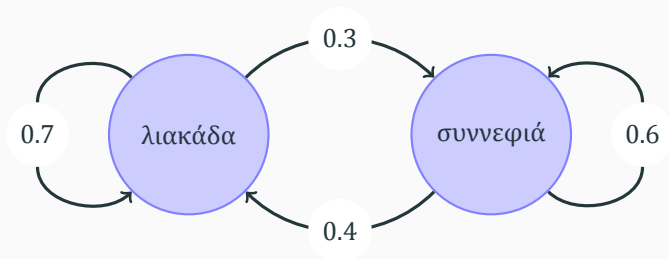
Οι Μαρκοβιανές αλυσίδες, είναι μαθηματικές κατασκευές οι οποίες περιγράφουν την μετάβαση ενός συστήματος από μια κατάσταση σε όλες τις υπόλοιπες δυνατές καταστάσεις του συστήματος.

## Παράδειγμα

Πώς θα μπορούσαμε να αναπαραστήσουμε τις μεταβάσεις ενός συστήματος το οποίο περιγράφει τον καιρό μιας περιοχής, θεωρώντας 2 δυνατές καταστάσεις {λιακάδα, συννεφιά};

# Μαρκοβιανές αλυσίδες

Για 2 δυνατές καταστάσεις, υπάρχουν 4 δυνατές μεταβάσεις, καθώς από κάθε κατάσταση υπάρχει η πιθανότητα να μεταβούμε στην άλλη ή να παραμείνουμε στην ίδια. Γενικά για κάθε  $n$  καταστάσεις υπάρχουν  $n^2$  δυνατές μεταβάσεις.



Επειδή για μεγαλύτερα  $n$  οι αναπαραστάσεις με μορφή γράφου αρχίζουν να γίνονται χαοτικές, προτιμούμε την αναπαράσταση των Μαρκοβιανών αλυσίδων με μορφή πίνακα.

	λιακάδα	συννεφιά
λιακάδα	0.7	0.3
συννεφιά	0.4	0.6

# Κατακερματισμός Κωδικών

---



## Ορισμός

Οι *Συναρτήσεις Κατακερματισμού (hash functions)*, είναι συναρτήσεις οι οποίες δέχονται ως είσοδο δεδομένα οποιουδήποτε μεγέθους, και τα αντιστοιχίζουν ντετερμινιστικά σε κάποια άλλα δεδομένα σταθερού μεγέθους.

## Ορισμός

Οι *Συναρτήσεις Κατακερματισμού (hash functions)*, είναι συναρτήσεις οι οποίες δέχονται ως είσοδο δεδομένα οποιουδήποτε μεγέθους, και τα αντιστοιχίζουν ντετερμινιστικά σε κάποια άλλα δεδομένα σταθερού μεγέθους.

## Χρήση

Αποθήκευση κωδικών των χρηστών μιας σελίδας, στην βάση δεδομένων, με στόχο την προστασία των προσωπικών δεδομένων τους.

## Κωδικοί με το MD5 hash τους

κωδικός	MD5 hash
apple	1f3870be274f6c49b3e31a0c6728957f
Apple	9f6290f4436e5a2351f12e03b6433c3c
apple_	e307c4bc0265c934316dbd59f86336fd

Για να βρούμε ποιος κωδικός απλού κειμένου αντιστοιχεί στον κωδικό κατακερματισμού πρέπει:

Για να βρούμε ποιος κωδικός απλού κειμένου αντιστοιχεί στον κωδικό κατακερματισμού πρέπει:

ή Να ξέρουμε εξαρχής τον κωδικό απλού κειμένου.

Για να βρούμε ποιος κωδικός απλού κειμένου αντιστοιχεί στον κωδικό κατακερματισμού πρέπει:

- ή Να ξέρουμε εξαρχής τον κωδικό απλού κειμένου.
- ή Να δοκιμάσουμε διάφορους κωδικούς απλού κειμένου ως εισόδους της συνάρτησης μέχρι να βρούμε την έξοδο που είναι ίδια με τον κατακερματισμένο κωδικό.

# Μέθοδοι Ανεύρεσης

---

## Ορισμός

Δοκιμάζουμε όλους τους πιθανούς συνδυασμούς των χαρακτήρων που ανήκουν στο πεδίο χαρακτήρων (*character space*), συνήθως κατά αύξουσα σειρά.



# Brute force

## Ορισμός

Δοκιμάζουμε όλους τους πιθανούς συνδυασμούς των χαρακτήρων που ανήκουν στο πεδίο χαρακτήρων (*character space*), συνήθως κατά αύξουσα σειρά.

## Παράδειγμα

Εστω *character space* = (a-z) και ότι θέλουμε να δοκιμάσουμε όλους τους κωδικούς με μήκος τρεις χαρακτήρες, τότε οι κωδικοί θα δοκιμαστούν με την ακόλουθη σειρά.

aaa	aba	...	baa	bba	...	zza
aab	abb	...	bab	bbb	...	zzb
aac	abc	...	bac	bbc	...	zzc
⋮	⋮	⋮	⋮	⋮	⋮	⋮
aaz	abz	...	baz	bbz	...	zzz

## Ορισμός

Η μέθοδος λεξικού (dictionary attack) βασίζεται στην δοκιμή λέξεων από μια υπάρχουσα λίστα. Περιορίζεται μόνο σε συνηθισμένες ακολουθίες χαρακτήρων που είναι εύκολο να απομνημονευθούν.

## Ορισμός

Η μέθοδος λεξικού (dictionary attack) βασίζεται στην δοκιμή λέξεων από μια υπάρχουσα λίστα. Περιορίζεται μόνο σε συνηθισμένες ακολουθίες χαρακτήρων που είναι εύκολο να απομνημονευθούν.

## Παράδειγμα

Το πλήθος των λέξεων στο Oxford English Dictionary είναι περίπου 600 000 , θεωρώντας ότι ο μέσος άνθρωπος χρησιμοποιεί, από 10 000 έως 40 000 από αυτές, με έναν ρυθμό δοκιμής 5 000 000 κωδικών το δευτερόλεπτο, θα απαιτούνταν γύρω στα 8 ms για να δοκιμαστούν όλες οι λέξεις!

## Ορισμός

Με την χρήση των Μαρκοβιανών αλυσίδων, μπορούμε να βελτιώσουμε την μέθοδο ωμής βίας. Στις λέξεις της φυσικής γλώσσας, η συχνότητα και η ακολουθία των γραμμάτων δεν είναι ισοπίθανη.

## Ορισμός

Με την χρήση των Μαρκοβιανών αλυσίδων, μπορούμε να βελτιώσουμε την μέθοδο ωμής βίας. Στις λέξεις της φυσικής γλώσσας, η συχνότητα και η ακολουθία των γραμμάτων δεν είναι ισοπίθανη.

## Παράδειγμα

Στην Αγγλική γλώσσα, περισσότερες λέξεις ξεκινάνε με τον χαρακτήρα 'c' παρά με τον χαρακτήρα 'b'. Επίσης, υπάρχουν περισσότερες λέξεις, στις οποίες ο χαρακτήρας 'a' ακολουθείται από τον χαρακτήρα 'b', παρά από τον χαρακτήρα 'a'.

# Σύγκριση μεθόδων

	Brute Force	Dictionary	Markov Chains
Πολυπλοκότητα	$\mathcal{O}(s^n)$	$\mathcal{O}(l)$	$\mathcal{O}(t^n)$

$n$  μήκος του κωδικού απλού κειμένου

$s$  πληθικότητα του πεδίου χαρακτήρων

$l$  πλήθος λέξεων στο λεξικό

$t$  πληθικότητα του πεδίου χαρακτήρων  $t \leq s$

# Υλοποίηση Μεθόδου Μαρκοβιανών Αλυσίδων

---

## Στόχος

- Να δημιουργήσουμε ένα Μαρκοβιανό πίνακα για κάθε θέση του κωδικού (per position Markov table).
- Να χρησιμοποιήσουμε τους πίνακες για να παράγουμε πραγματικούς κωδικούς!



## Στόχος

- Να δημιουργήσουμε ένα Μαρκοβιανό πίνακα για κάθε θέση του κωδικού (per position Markov table).
- Να χρησιμοποιήσουμε τους πίνακες για να παράγουμε πραγματικούς κωδικούς!

## Χαρακτηριστικά υλοποίησης

- Οι πίνακες μπορούν να παραχθούν από οποιαδήποτε λίστα κωδικών απλού κειμένου (*wordlist*).
- Οι κωδικοί έχουν μήκος μέχρι 10 χαρακτήρες.
- Το πεδίο χαρακτήρων είναι το πλήρες διάστημα εκτυπώσιμων χαρακτήρων του πίνακα ASCII (κωδικοί 32–126), με πληθικότητα 95.

## **Πόσους πίνακες θα χρειαστούμε;**

Αφού από κάθε μια κατάσταση, μπορούμε να πάμε σε οποιαδήποτε κατάσταση, δημιουργούμε έναν πίνακα με διαστάσεις  $95 \times 95$  για κάθε θέση, δηλαδή 10 στο πλήθος.

## **Πόσους πίνακες θα χρειαστούμε;**

Αφού από κάθε μια κατάσταση, μπορούμε να πάμε σε οποιαδήποτε κατάσταση, δημιουργούμε έναν πίνακα με διαστάσεις  $95 \times 95$  για κάθε θέση, δηλαδή 10 στο πλήθος.

## **Πώς θα αναπαραστήσουμε τους πίνακες;**

Υπό την μορφή ενός τρισδιάστατου πίνακα, με μήκος 1<sup>ης</sup> διάστασης το μέγιστο μήκος κωδικού και μήκος 2<sup>ης</sup> και 3<sup>ης</sup> διάστασης την πληθικότητα του πεδίου χαρακτήρων.

# Αλγόριθμος παραγωγής Μαρκοβιανών Πινάκων

---

## Algorithm 1 Generate Markov Tables

---

```
1: // Initialize all frequencies to 0
2: for  $m = 0$  to 9 do
3:     for  $i = 0$  to 94 do
4:         for  $j = 0$  to 94 do
5:              $markov[m][i][j] = 0$ 
6:         end for
7:     end for
8: end for
9: // Compute Frequencies
10: for all words in dictionary do
11:      $markov[0][0][word[0]] += 1$ 
12:     for  $c = 1$  to  $length(word)$  do
13:          $markov[c][word[c - 1]][word[c]] += 1$ 
14:     end for
15: end for
```

---

## Υποθέσεις

- Το πεδίο χαρακτήρων είναι το {a, b, c, d, e}
- Το μέγιστο μήκος κωδικού είναι 4
- Η λίστα με τους κωδικούς είναι η ακόλουθη

ace	bee
add	cab
bad	dad
bed	dead



## Βήμα 1: ace

1	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[0]

0	0	1	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[1]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0

markov[2]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[3]

## Βήμα 2: add

2	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[0]

0	0	1	1	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[1]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	0	0

markov[2]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[3]



## Βήμα 3: bad

2	1	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[0]				
0	0	1	1	0
1	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[1]				
0	0	0	1	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	0	0
markov[2]				
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[3]				

## Βήμα 4: bed

2	2	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[0]

0	0	1	1	0
1	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[1]

0	0	0	1	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	0

markov[2]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[3]

## Βήμα 5: bee

2	3	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[0]

0	0	1	1	0
1	0	0	0	2
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[1]

0	0	0	1	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1

markov[2]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[3]

## Βήμα 6: cab

2	3	1	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[0]				
0	0	1	1	0
1	0	0	0	2
1	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[1]				
0	1	0	1	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
markov[2]				
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[3]				

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

markov[3]

## Βήμα 8: dead

2	3	1	2	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[0]				
0	0	1	1	0
1	0	0	0	2
1	0	0	0	0
1	0	0	0	1
0	0	0	0	0
markov[1]				
0	1	0	2	0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
1	0	0	1	1
markov[2]				
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
markov[3]				

# Παραγωγή Κωδικών με Βάση τους Μαρκοβιανούς Πίνακες

- Δημιουργούμε έναν πίνακα με διαστάσεις  $95 \times 95$  για κάθε θέση, δηλαδή 10 στο πλήθος
- Αυτή τη φορά, αντί να κρατάμε μόνο την συχνότητα εμφάνισης του χαρακτήρα σε κάθε κελί κρατάμε ένα ζευγάρι που έχει
  - ως 1<sup>ο</sup> στοιχείο τον χαρακτήρα
  - ως 2<sup>ο</sup> στοιχείο την συχνότητα εμφάνισης αυτού του χαρακτήρα
- Ταξινομούμε τα ζεύγη κάθε γραμμής του πίνακα κατά φθίνουσα σειρά με βάση την συχνότητα, σπάζοντας τις ισοβαθμίες κατά αύξουσα σειρά με βάση τον χαρακτήρα.

# Παράδειγμα

('a',2)	('b',3)	('c',1)	('d',2)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

pairs[0]

('b',3)	('a',2)	('d',2)	('c',1)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

sorted\_pairs[0]



# Παράδειγμα

('a',0)	('b',0)	('c',1)	('d',1)	('e',0)
('a',1)	('b',0)	('c',0)	('d',0)	('e',2)
('a',1)	('b',0)	('c',0)	('d',0)	('e',0)
('a',1)	('b',0)	('c',0)	('d',0)	('e',1)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

pairs[1]

('c',1)	('d',1)	('a',0)	('b',0)	('e',0)
('e',2)	('a',1)	('b',0)	('c',0)	('d',0)
('a',1)	('b',0)	('c',0)	('d',0)	('e',0)
('a',1)	('e',1)	('b',0)	('c',0)	('d',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

sorted\_pairs[1]

# Παράδειγμα

('a',0)	('b',1)	('c',0)	('d',2)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',1)
('a',0)	('b',0)	('c',0)	('d',1)	('e',0)
('a',1)	('b',0)	('c',0)	('d',1)	('e',1)

pairs[2]

('d',2)	('b',1)	('a',0)	('c',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('e',1)	('a',0)	('b',0)	('c',0)	('d',0)
('d',1)	('a',0)	('b',0)	('c',0)	('e',0)
('a',1)	('d',1)	('e',1)	('b',0)	('c',0)

sorted\_pairs[2]

# Παράδειγμα

('a',0)	('b',0)	('c',0)	('d',1)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

pairs[3]

('d',1)	('a',0)	('b',0)	('c',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)
('a',0)	('b',0)	('c',0)	('d',0)	('e',0)

sorted\_pairs[3]

## Υποθέσεις

- το threshold  $t = 3$ , δηλαδή για κάθε θέση θα δοκιμάζουμε κάθε φορά μόνο τους 3 πιο πιθανούς χαρακτήρες
- το μήκος των κωδικών που θέλουμε να παράγουμε  $n = 3$

# Παράδειγμα Παραγωγής Κωδικών

Παράγουμε  $t^n = 3^3 = 27$  διαφορετικούς κωδικούς.

bea	ace	dad
bed	aca	dab
bee	acb	daa

bad	add	dea
bab	ada	ded
baa	adb	dee

bba	aad	dba
bbb	aab	dbb
bbc	aaa	dbc

# Προτάσεις για Επιλογή Καλύτερων Κωδικών

---

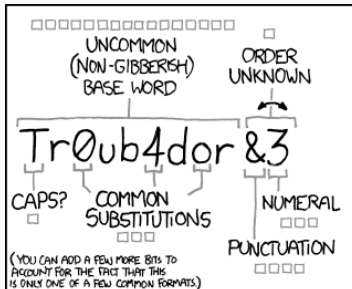
# Τι να αποφύγετε

- Αποφύγετε την επαναχρησιμοποίηση κωδικών.
- Αποφύγετε εύκολους κωδικούς που προκύπτουν από μονοπάτια στο πληκτρολόγιο (π.χ. `asdf`).
- Αποφύγετε λέξεις που βρίσκονται αυτούσιες σε λεξικά (π.χ. `football`), αριθμητικές ακολουθίες (π.χ. `123789654`) και προσωπικά στοιχεία (π.χ. `17051994`).

## Τι να προτιμήσετε;

- Χρησιμοποιήστε κωδικούς με μήκος τουλάχιστον 12 με 14 χαρακτήρες.
- Χρησιμοποιήστε κεφαλαίους και μικρούς χαρακτήρες, νούμερα, και ειδικούς χαρακτήρες αν επιτρέπεται.
- Παράγετε τους κωδικούς σας τυχαία όπου μπορείτε.





~28 BITS OF ENTROPY


$2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}$

(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

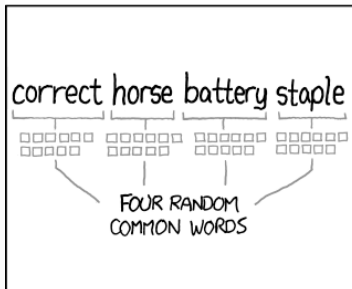
DIFFICULTY TO GUESS: **EASY**

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?

AND THERE WAS SOME SYMBOL...



DIFFICULTY TO REMEMBER: **HARD**



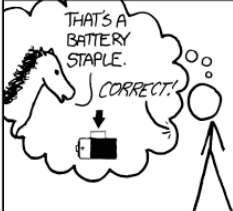
~44 BITS OF ENTROPY

$2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}$

DIFFICULTY TO GUESS: **HARD**

THAT'S A BATTERY STAPLE.

CORRECT!



DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.