Bertrand Chazot

UPM-64
Linear-Time String Matching Algorithms
The Karp-Rabin Algorithm

The goal of the Karp-Rabin Algorithm is to find in a text T, the positions -if any- of the pattern P.

This algorithm is based on a hashing technique : a fingerprint is calculated for each pattern, and also for each substring of T, substrings whose length is the size of P. The algorithm is efficient since, except for the first one, the hashes of T are being computed in constant time.

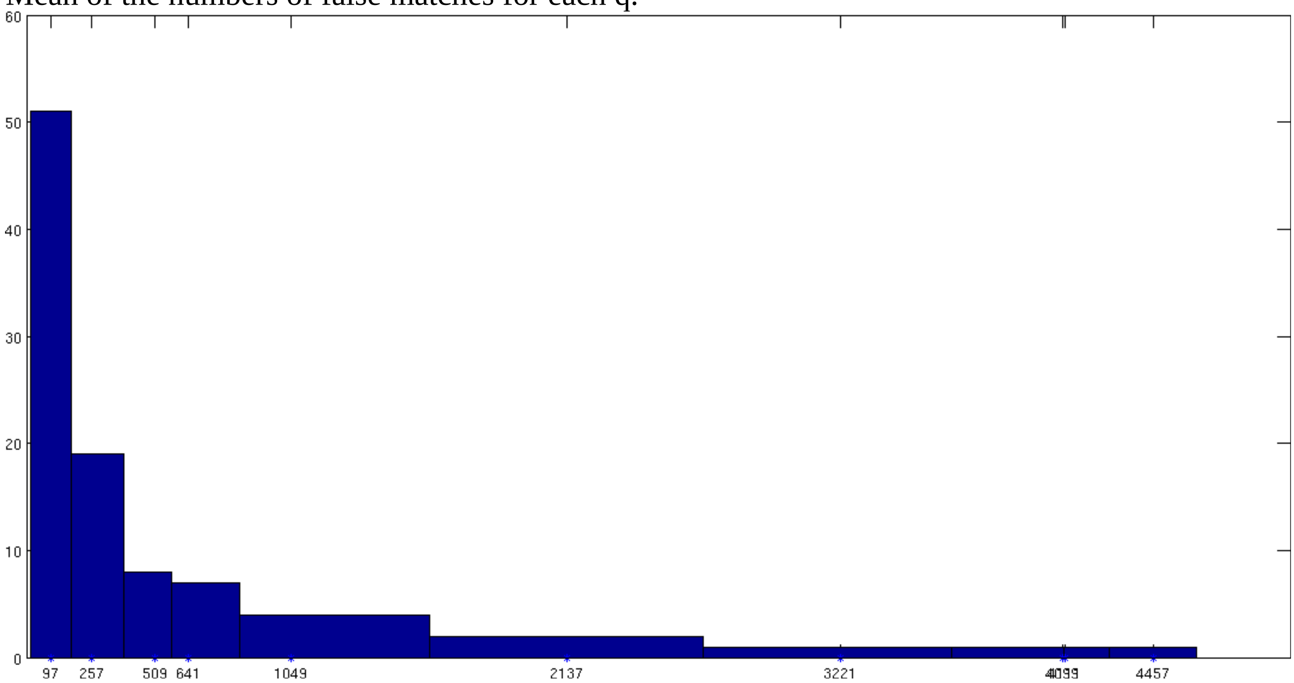
When a hit occur, *i.e.* one of the fingerprints of T is equal to the one of P, it only remains to check whether this segment of T is really P. If so, a correct position has been found. One of the purposes of the following study is to minimize the number of false matches because the are quite time-consuming.

I used C++ to implement this algorithm. A class Espr (exact string pattern recognition) owns a text T (randomly initialised or not). It allows the user to extract random substrings from T, and to use the Karp-Rabin algorith on T for a given p, d, q. In these exercises, d is fixed to 10, and the influence of q is analysed.

I repeated the experiment 1 with ten differents q, and did some statistical analysis on the false matches (I used the GNU Scientific Library).

$q \in \{97, 257, 509, 641, 1049, 2137, 3221, 4099, 4111, 4457\}$

Mean of the numbers of false matches for each q.



Bertrand Chazot

q= 97

Mean : 51.42
variance : 45.2966
Median : 51
Probability of false matches : 0.0103879

q= 257

Mean : 19.69
variance : 17.3272
Median : 19
Probability of false matches : 0.00397778

q= 509

Mean : 8.9
variance : 9.32323
Median : 9
Probability of false matches : 0.00179798

q= 641

Mean : 7.46
variance : 10.6145
Median : 7
Probability of false matches : 0.00150707

q= 1049

Mean : 4.9
variance : 5.88889
Median : 5
Probability of false matches : 0.000989899

q= 2137

Mean : 2.43
variance : 2.22737
Median : 2
Probability of false matches : 0.000490909

q= 3221

Mean : 1.64
variance : 1.70747
Median : 1
Probability of false matches : 0.000331313

q= 4099
Mean : 1.11
variance : 1.02818
Median : 1
Probability of false matches : 0.000224242

q= 4111

Mean : 1.3
variance : 1.42424
Median : 1
Probability of false matches : 0.000262626

q=4457

Mean : 1.11
variance : 0.947374
Median : 1
Probability of false matches : 0.000224242

Bertrand Chazot

It appears quite clearly that the biggest q is, the less false matches there are. Indeed, the probability of collision (two different strings with the same hash) decreases when q increases since the interval to which the hashes belong increases as well. However; it does not seem relevant to take a q big in comparison to the size of the text T. As soon as q is bigger than n/2, the number of mismatches seems acceptable.