Frequency Distribution of Digit in the Prime Number

Graduation Thesis

Chonnam National University

Department of Mathematics

Young-Hwa Park

Supervised by Professor Hyeong-Kwan Ju

December 5, 2019

Contents

Abstract

We obtained certain results about the frequency distribution of digit in the prime number with some constraint condition.

1 Introduction

This thesis is about the frequency distribution of each digit in the prime number. So i counted digits in the prime number with the java program which i made. For example there are three 1 in 171571.And there are two 7 in 171571. There are some result i got from my program.

2 Preliminaries

2.1 Benford's law

Benford's law, also called the Newcomb–Benford law, the law of anomalous numbers, or the first-digit law, is an observation about the frequency distribution of leading digits in many real-life sets of numerical data. The law states that in many naturally occurring collections of numbers, the leading significant digit is likely to be small. For example, in sets that obey the law, the number 1 appears as the leading significant digit about 30

2.2 Prime Number Theorem

In number theory, The prime number theorem (PNT) describes the asymptotic distribution of the prime numbers among the positive integers. It formalizes the intuitive idea that primes become less common as they become larger by precisely quantifying the rate at which this occurs[Wikipedia]

3 Result

There are some result about frequency distribution of digit in the prime number with some constraint condition.

3.1 First Case

This chart shows the order which digit appears more often in the prime number at the first digit.
So the data on the digit column means order and the data on the n column means interval from 0 to 10^x.

n	1	2	3	4	5	6	7	8	9
10^6	1	2	3	4	5	6	7	8	9
10^7	1	2	3	4	5	6	7	8	9

This chart shows which digit appears more often in the prime number at the first digit. So the figure on the digit column means ratio and the figure on the n column means interval from 0 to 10^x.

n	1	2	3	4	5	6	7	8	9
10^6	12.210	11.646	11.414	11.142	10.974	10.774	10.745	10.606	10.484
10^7	12.124	11.670	11.407	11.229	11.053	10.948	10.843	10.763	9.960

Analysis We can see the more digit increase, the more ratio decrease. This is the similar with Benford's Law.

3.2 Second Case

This chart shows the order which digit appears more often in the prime number except last digit.
So the data on the digit column means order and the data on the n column means interval from 0 to 10^x.

n	0	1	2	3	4	5	6	7	8	9
10^6	10	1	2	3	4	5	6	7	9	8
10^7	10	1	2	3	4	5	6	8	7	9
10^8	10	1	2	3	4	5	6	7	8	9

This chart shows which digit appears more often in the prime number except the last digit.
The reason why i didn't count last digit is the even number and 5 can't appear at the last digit in the prime number.
The data on the digit column means ratio and the data on the n column means interval from 0 to 10^x.

n	0	1	2	3	4	5	6	7	8	9
10^6	7.955	10.490	10.373	10.251	10.225	10.200	10.148	10.146	10.103	10.106
10^7	8.315	10.365	10.277	10.221	10.196	10.159	10.149	10.115	10.118	10.080
10^8	8.558	10.284	10.227	10.187	10.163	10.146	10.126	10.113	10.103	10.086

Analysis We can see the more digit increase, the more ratio decrease mostly except 0. And the 0 is the least digit since 0 can't appear at the first digit in the number.

3.3 The Main Case

This chart shows the order which digit appears more often in the prime number except the first and last digit.
So the data on the digit column means order and the data on the n column means interval from 0 to 10^x.

n	0	1	2	3	4	5	6	7	8	9
10^6	2	3	1	10	6	8	5	7	9	4
10^7	1	2	3	7	5	9	4	6	8	10
10^8	1	2	3	6	4	5	9	8	7	10
10^9	1	2	3	4	5	6	7	8	9	10

This chart shows which digit appears more often in the prime number except the first and last digit.
So the figure on the digit column means ratio and the figure on the n column means interval from 0 to 10^x.

n	0	1	2	3	4
	5	6	7	8	9
10^6	10.017097	10.044161	10.043831	9.950096	9.987062
	9.999604	9.986072	9.991682	9.97254	10.007855
10^7	10.024566	10.021473	10.008142	9.9936075	10.000659
	9.991566	10.000782	9.981267	10.000659	9.977278
10^8	10.016329	10.008383	10.006177	9.998375	9.998351
	10.000996	9.994188	9.99478	9.995081	9.987339
10^9	10.012193	10.006777	10.004244	10.001982	10.000601
	9.999767	9.996005	9.995016	9.99325	9.990169

Analysis I expected each digit will be equal. but we can see 0,1,2 appear more often than rest of digits.

4 Analysis

The first and second case are natural. Because the more number increase, the more prime frequency distribution decrease by the prime number theorem. So that is why the main case is fair for every digit. Therefore the digit are supposed to occur uniformly in the main case. But we can see 0,1,2 appear more often than rest of digit from 10^6 to 10^9. And it is difficult to understand the reason.

4.1 Guess

I guess 0,1,2 appear more often than rest of digit at the sufficiently big number in the main case.

References

[Wikipedia] https://en.wikipedia.org/wiki/Prime$_n umber_t heorem$

[Wikipedia] https://en.wikipedia.org/wiki/Benford

5 Java Program Code

public class engine1

static int y[] = 0,0,0,0,0,0,0,0,0,0;

public int prime$_producer(intn)//producennumberthprimenumber//timecomplexityisO(n^2)$

int count=0; int prime=0;

for (int i=2;;i++)

int sum=0; for (int j=2;j¡=Math.sqrt(i);j++) if (i sum=sum+1; break;

if (sum==0) count=count+1; prime=i;

if (count==n) break;

return prime;

public void count(int k) //count how many each digit(0 9) in the prime number except last digit. int x=0;

if (k¡10) k=k*10;

k=k/10;

while(true)

//int x=0;

x=k

if(k==0) break;

k=k/10;

switch(x) case 0: y[0]=y[0]+1; break; case 1: y[1]=y[1]+1; break; case 2: y[2]=y[2]+1; break; case 3: y[3]=y[3]+1; break; case 4: y[4]=y[4]+1; break; case 5: y[5]=y[5]+1; break; case 6: y[6]=y[6]+1; break; case 7: y[7]=y[7]+1; break; case 8: y[8]=y[8]+1; break; case 9: y[9]=y[9]+1; break;

public void count2(int k) //count how many each digit(0 9) in the prime number in the first digit. //benford's law

if (k¡10) k=k*10;

while(true)

k=k/10;

if(k==0) break;

if(k¿=10) continue;

switch(k) case 0: y[0]=y[0]+1; break; case 1: y[1]=y[1]+1; break; case 2: y[2]=y[2]+1; break; case 3: y[3]=y[3]+1; break; case 4: y[4]=y[4]+1; break; case 5: y[5]=y[5]+1; break; case 6: y[6]=y[6]+1; break; case 7: y[7]=y[7]+1; break; case 8: y[8]=y[8]+1; break; case 9: y[9]=y[9]+1; break;

public void printfunc2(int[] y) //this method print ratio of each digit in the prime number

int m;

m=y[0]+y[1]+y[2]+y[3]+y[4]+y[5]+y[6]+y[7]+y[8]+y[9];

System.out.print((float)y[0]/m*100 + " "); System.out.print((float)y[1]/m*100 + " "); System.out.print((float)y[2]/m*100 + " "); System.out.print((float)y[3]/m*100 + " "); System.out.print((float)y[4]/m*100 + " "); System.out.print((float)y[5]/m*100 + " "); System.out.print((float)y[6]/m*100 + " "); System.out.print((float)y[7]/m*100 + " "); System.out.print((float)y[8]/m*100 + " "); System.out.print((float)y[9]/m*100 + " "); System.out.println();

public void printfunc3(int[] y) // this method print digit according to size.

int z[] = 0,0,0,0,0,0,0,0,0,0;

for (int i1=0;i1¡10;i1++) z[i1]=y[i1];

int a,b;

for (int g=0;g¡10;g++) a=g; b=z[g]; for(int h=0;h¡10;h++) if (b¡z[h]) a=h; b=z[h];

z[a]=-1; System.out.print(a + " "); System.out.println();

public static void main(String[] args)

int n=20000; int a;

long b;

int c=0;

engine1 um = new engine1();

System.out.print(0 + " "); System.out.print(1 + " "); System.out.print(2+ " "); System.out.print(3 + " "); System.out.print(4 + " "); System.out.print(5 + " "); System.out.print(6 + " "); System.out.print(7 + " "); System.out.print(8 + " "); System.out.print(9 + " "); System.out.println(); System.out.println();

for(int i=1;i¡=n;i++) //time complexity is O(n^3)$a = um.prime_producer(i)$;

um.count2(a);

c=c+1; if (c==n/10) um.printfunc2(y); c=0;