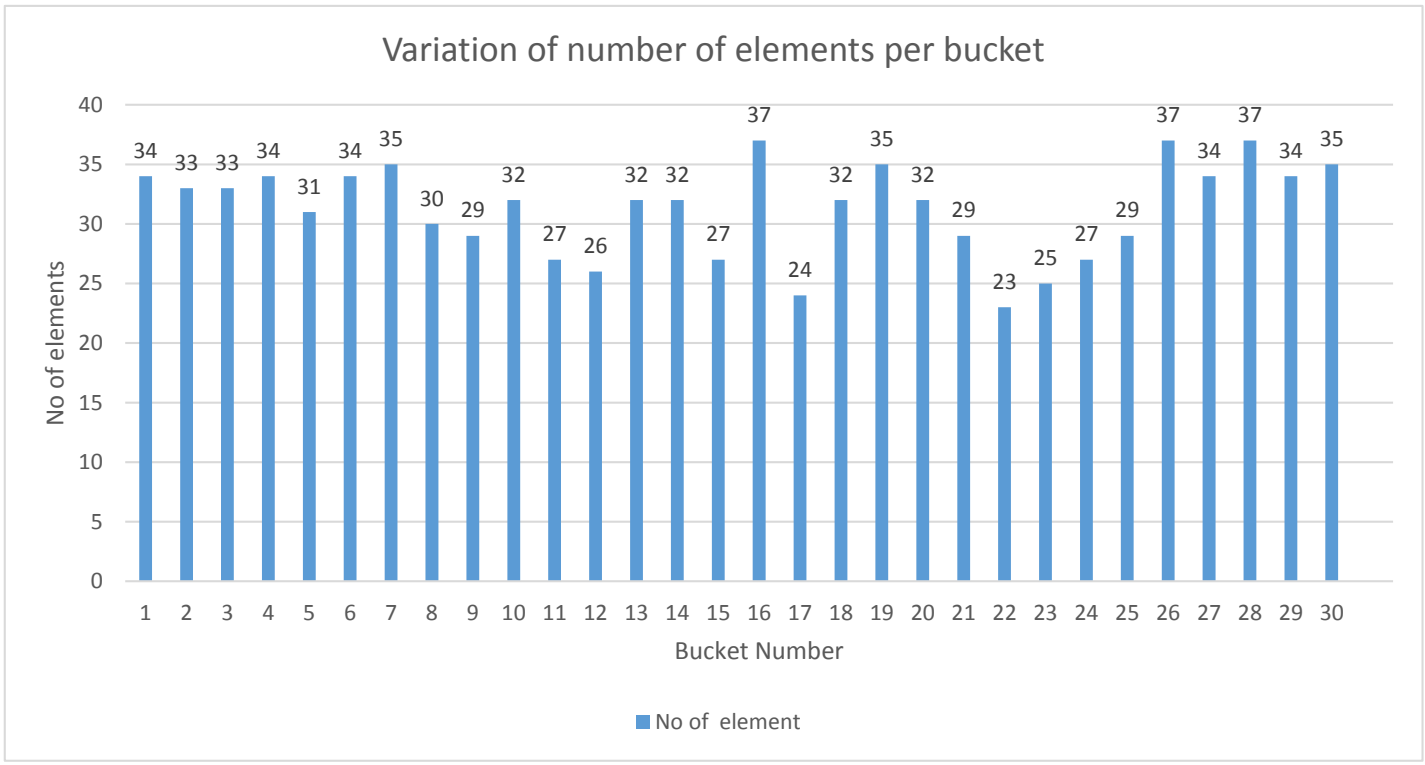
Lab 03

1) For sample-text1.txt

a) Hash function 1,

```
/**hash function1**/
public int hashfunc(String key){
        int val=0;
        int mul=43;
        //sum up the ascci value of the word
        for(int i=0;i<key.length();i++){
                val=(val*mul+key.charAt(i));
         }
        return val%hashTable.length;//return modulus of sum of ascii values
}
```

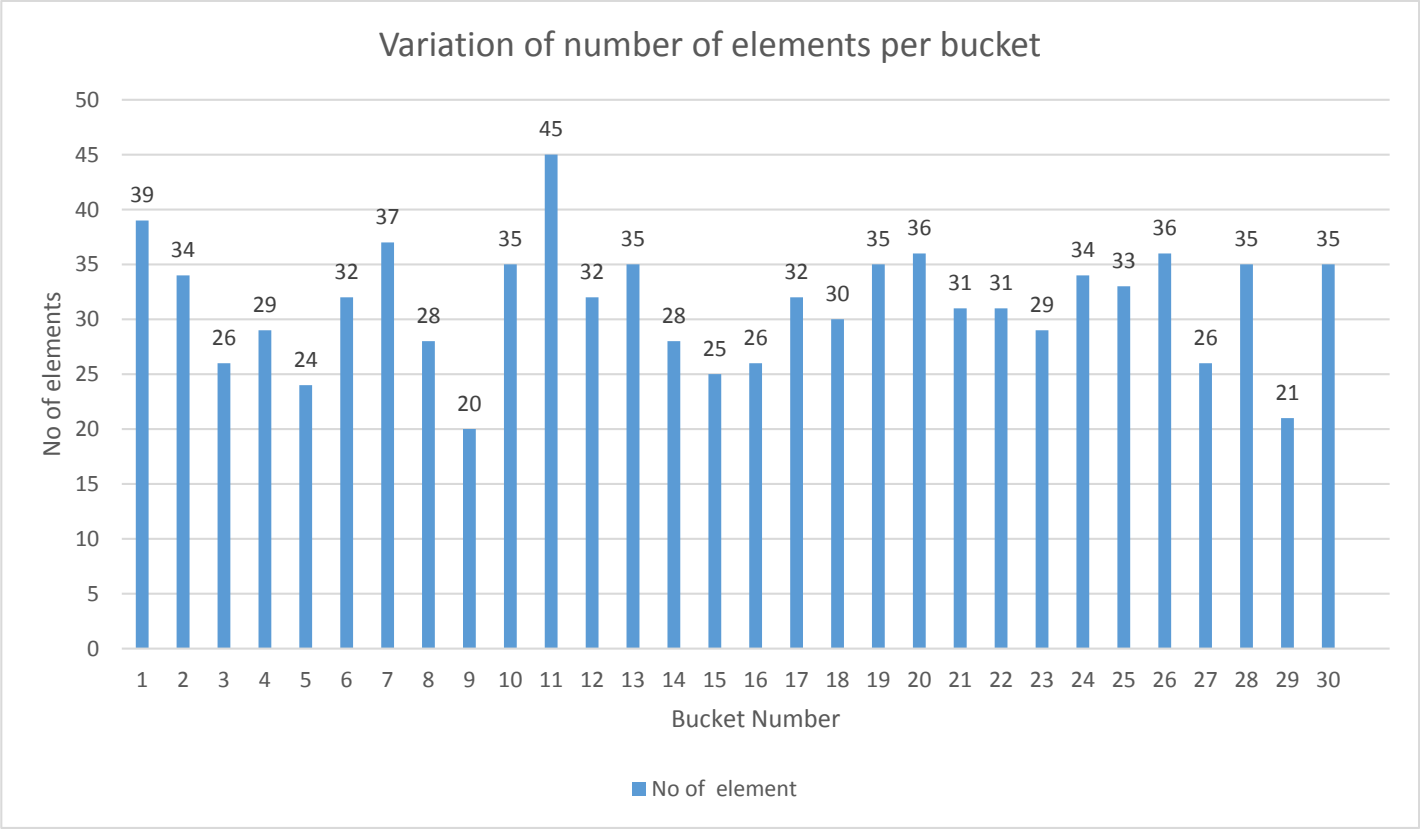


The minimum no of entries = 23
The maximum no of entries = 37
Average no of elements per bucket = 31.3
Standard deviation = 3.8570283867738637

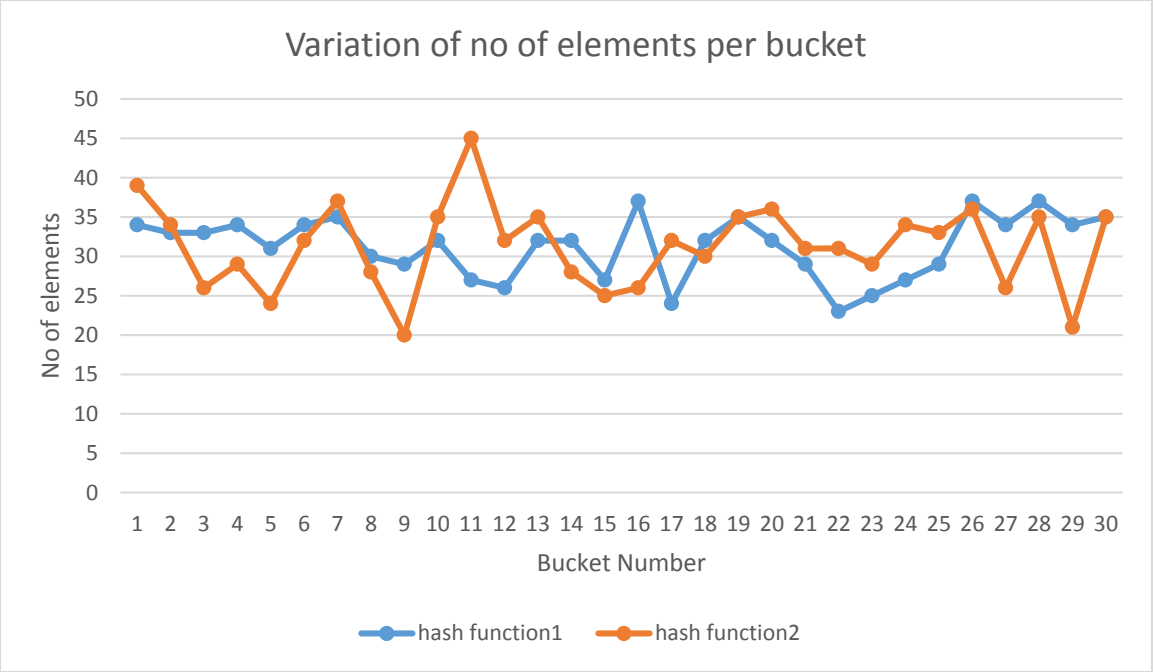
b) hash function 2,

By changing the multiplier we can get another hash function simply.

```
/**hash function2**/
public int hashfunc(String key){
        int val=0;
        int mul=71;
        //sum up the ascci value of the word
        for(int i=0;i<key.length();i++){
            val=(val*mul+key.charAt(i));
         }
        return val%hashTable.length;//return modulus of sum of ascii values
}
```



The minimum no of entries = 20
The maximum no of entries = 45
Average no of elements per bucket = 31.3
Standard deviation = 5.34259609680433

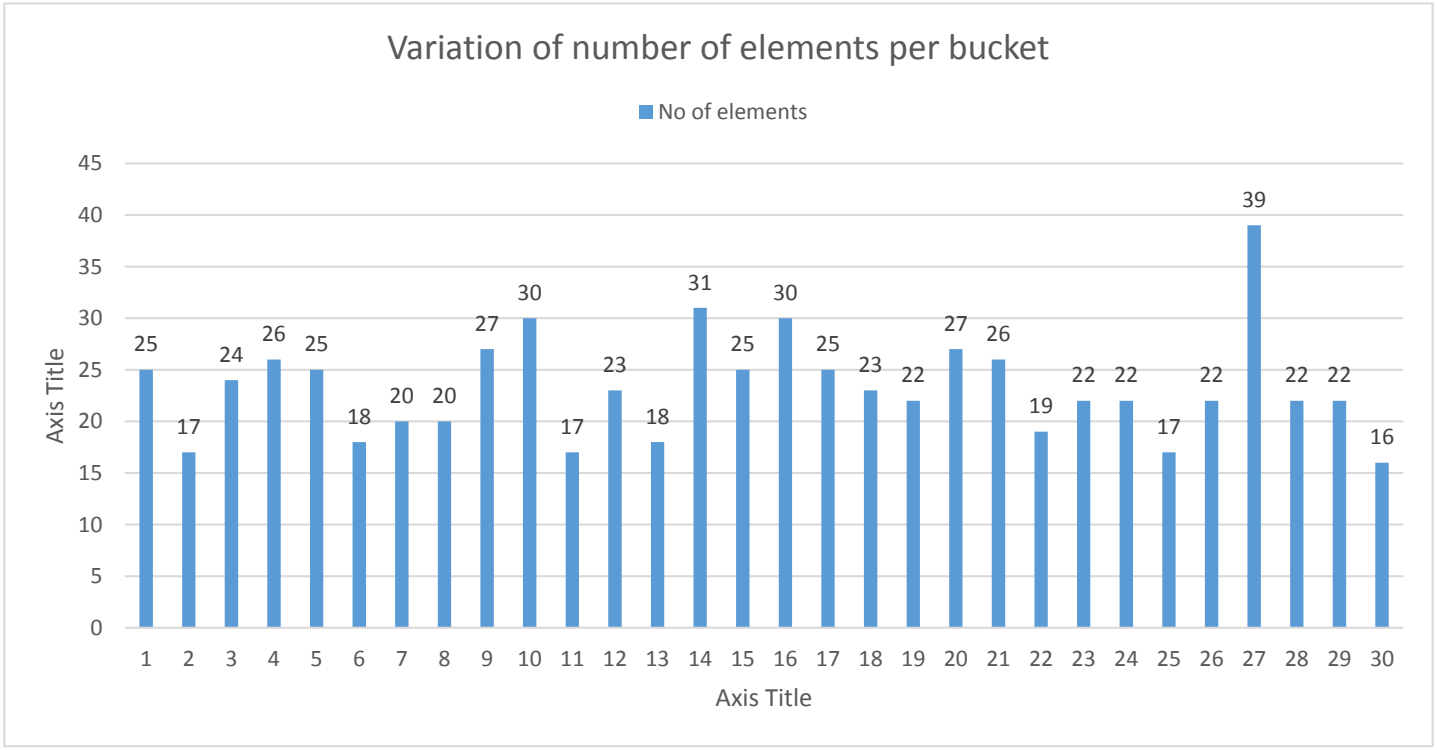


When comparing these two hash functions they have same average hits. But the maximum, minimum and standard deviation of hash function2 is high than hash function1. And the range that the values are spread is high in hash function2 according to above graph. In the ideal hash function we have same number of elements in every bucket. If we can have approximately same number of elements in buckets in the practical case it is better.

In hash function1 the range of values that are spread is narrow and standard deviation is low, therefore hash function1 is better than hash function2.

2) For sample-text2.txt
   a) For hash function 1,

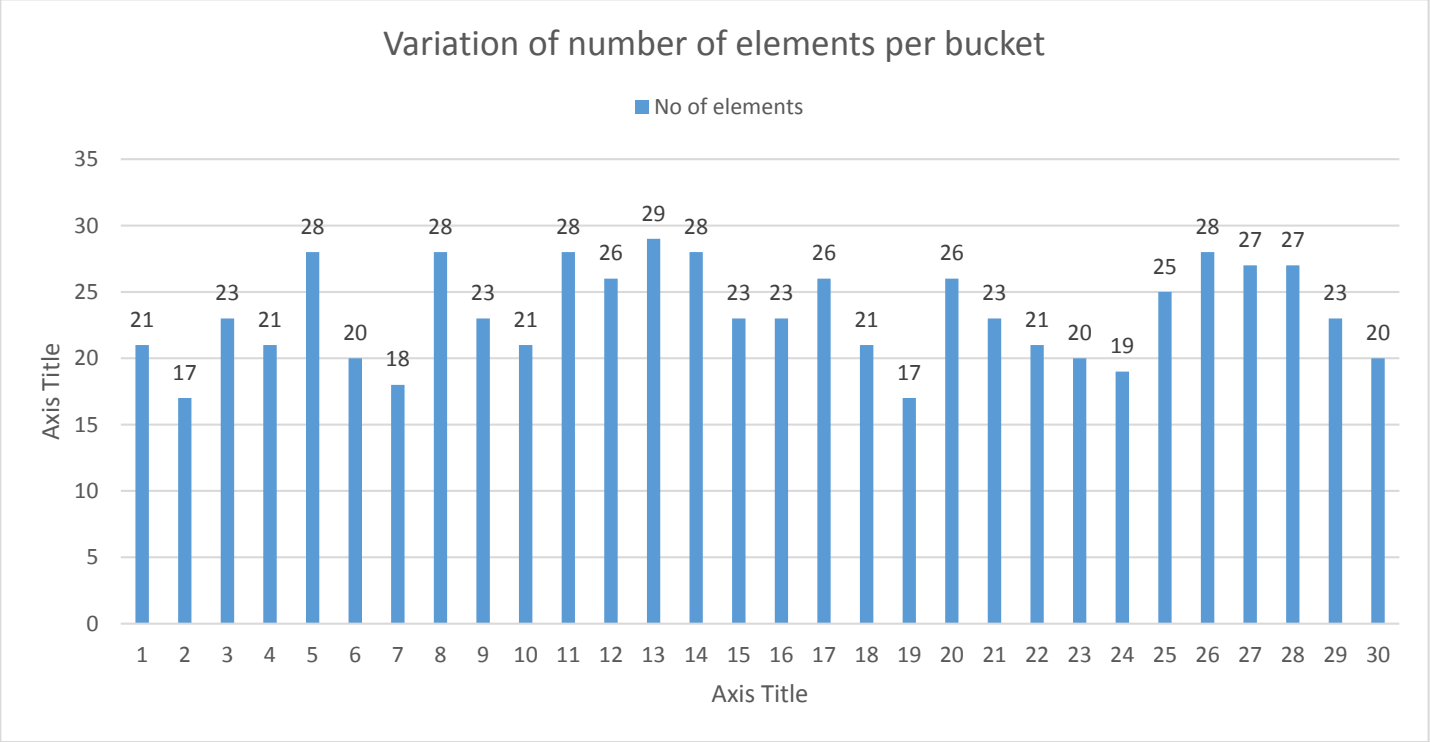
```
/**hash function**/
public int hashfunc(String key){
        int val=0;
        int mul=43;
        //sum up the ascci value of the word
        for(int i=0;i<key.length();i++){
                val=(val*mul+key.charAt(i));
         }
        return val%hashTable.length;//return modulus of sum of ascii values
}
```



The minimum no of entries = 16
The maximum no of entries = 39
Average no of elements per bucket = 23.333334
Standard deviation = 4.928375504826834

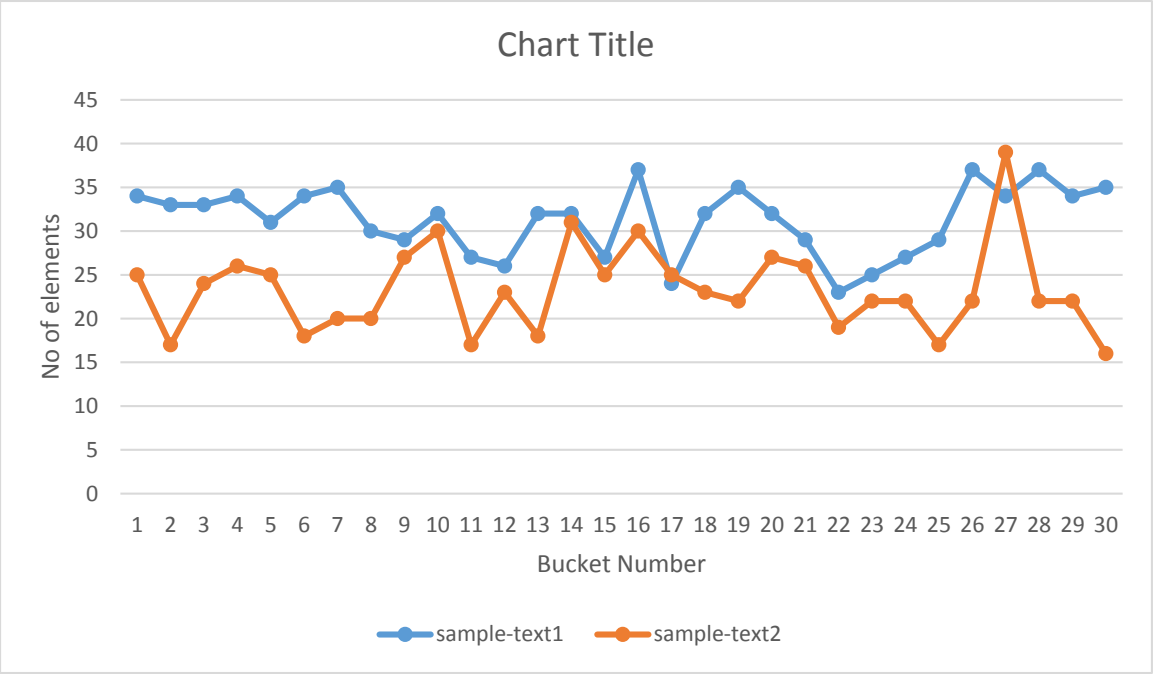
b) For hash function 2,

```
/**hash function**/
public int hashfunc(String key){
        int val=0;
        int mul=71;
        //sum up the ascci value of the word
        for(int i=0;i<key.length();i++){
                val=(val*mul+key.charAt(i));
         }
        return val%hashTable.length;//return modulus of sum of ascii values
}
```



The minimum no of entries = 17
The maximum no of entries = 29
Average no of elements per bucket = 23.333334
Standard deviation = 3.562146621042526

Hash function1, for both text files,



The same hash function for different files, it will give the different distributions according to above graph. Because the generated hash tables are totally different and the frequency of some words are very high compared to others in use(like 'the'). Then we cannot have same distribution.

If we use odd multiplier in hash function to generate the bucket number we can have better distribution close to the ideal case than using even multiplier.