CO 322 Data Structures and Algorithms
Lab 03 - Hash Tables

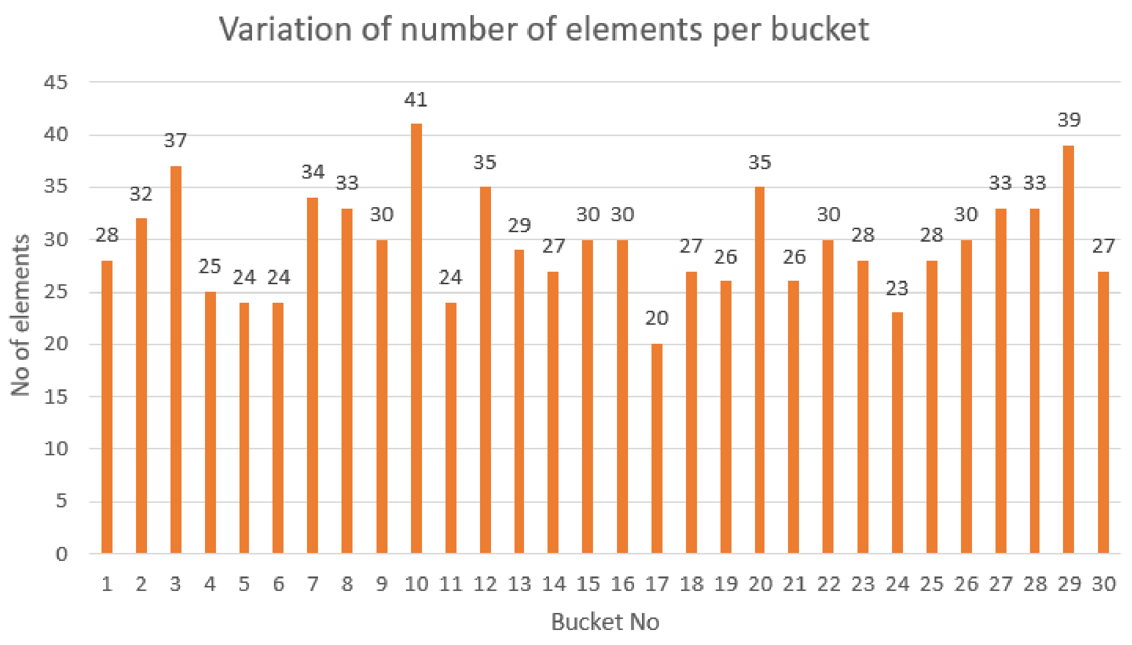
Paracrama G.T.W

E/16/267

1)For sample-text1.txt

a) Hash function 1,

```
public int hashfunction(String key){
            //set the multiplier
            int mul=43;
            int h=0;

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

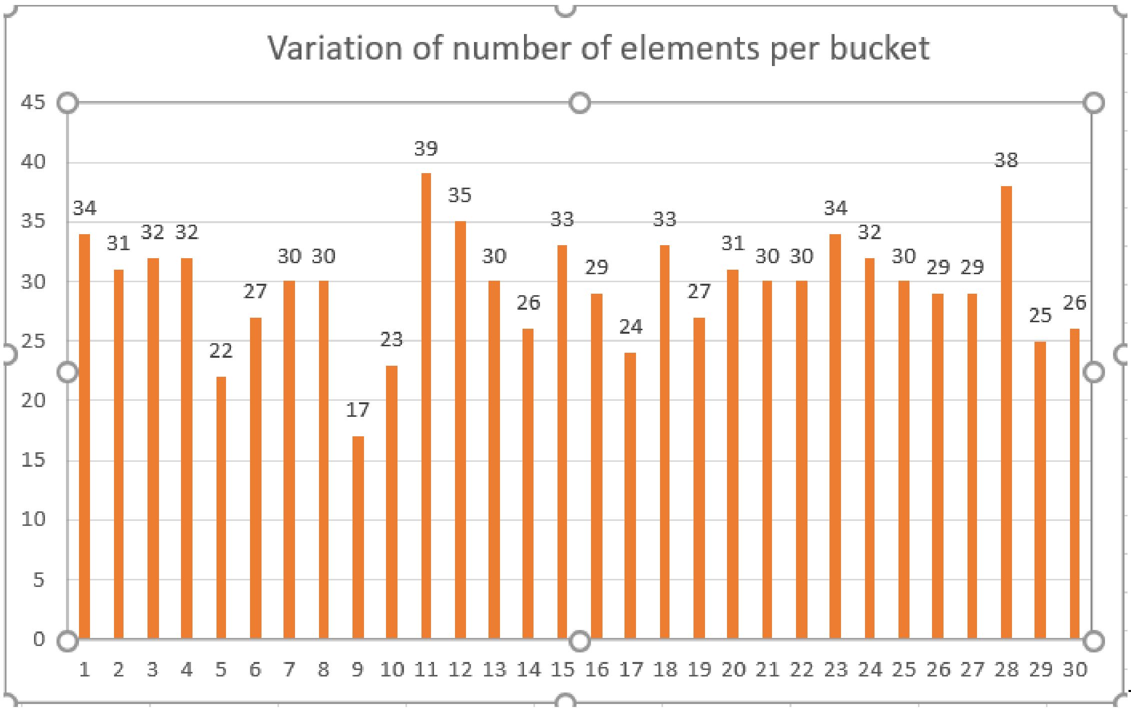


```
The minimum no of entries : 20

The maximum no of entries : 41
Total elements : 888
Average no of elements per bucket : 29.6
Standard deviation : 4.820788095713141
```

a) Hash function 2,

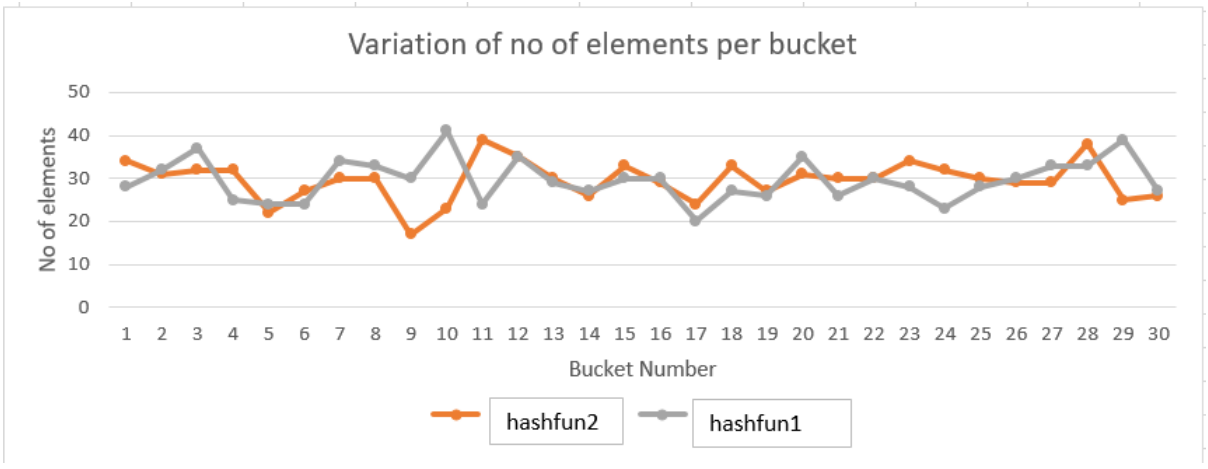
By changing the multiplier we can get another hash function simply

```
public int hashfunction(String key){

            //set the multiplier
            int mul=71;
            int h=0;

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



The

```
The minimum no of entries : 17
The maximum no of entries : 39
Total elements : 888
Average no of elements per bucket : 29.6
Standard deviation : 4.565084692011667
```



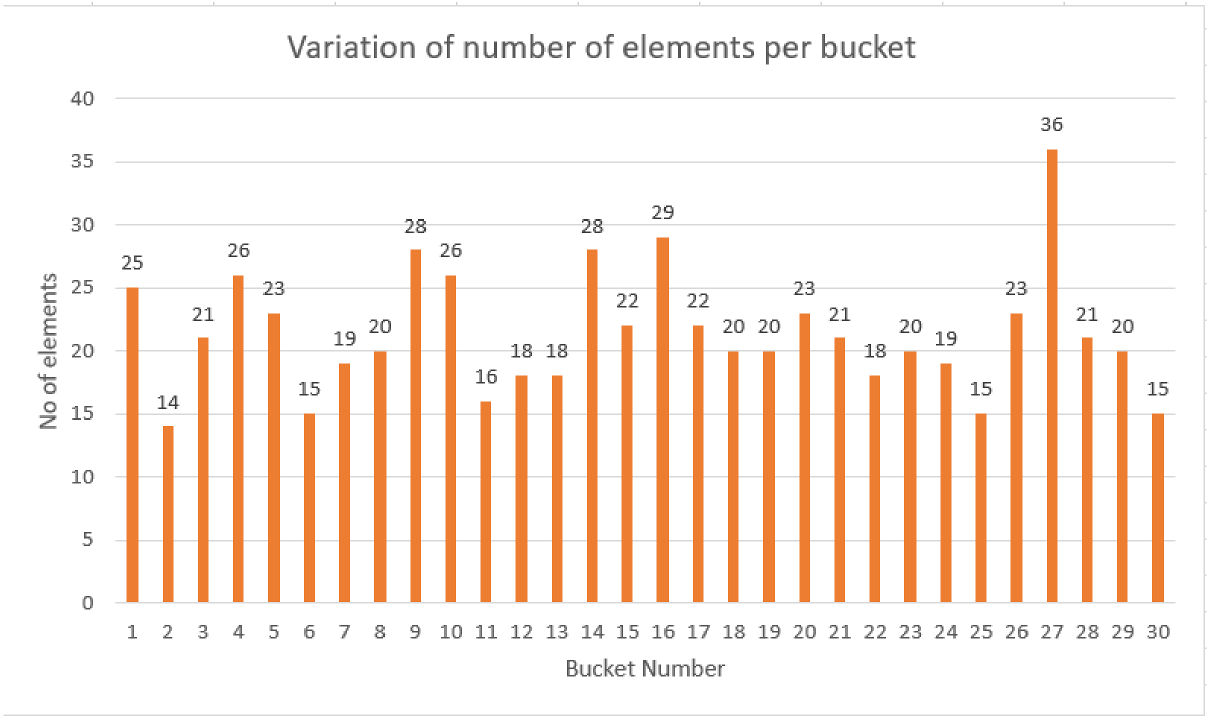
When comparing these two hash functions they have same average . But the maximum, minimum and standard deviation of hash function2 is high than hash function1. And the range that the values are spread is high in hash function2 according to above graph. In the ideal hash function we have same number of 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,

```
public int hashfunction(String key){
            //set the multiplier
            int mul=43;
            int h=0;

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

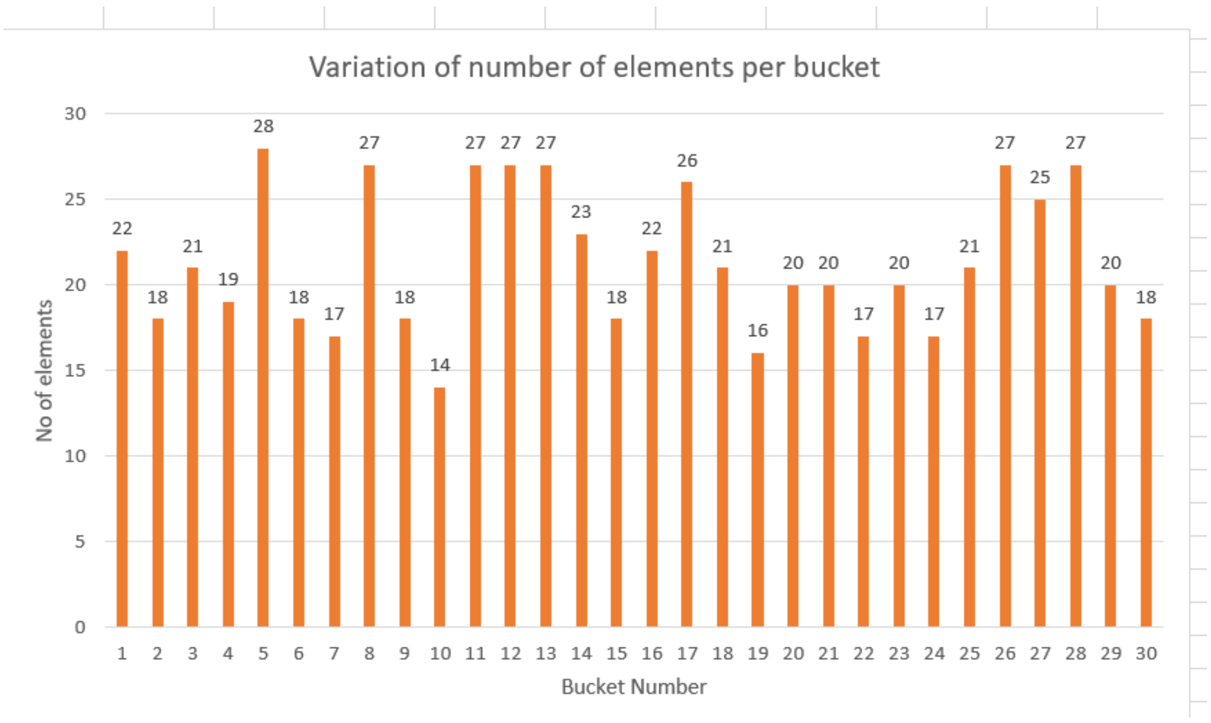


```
}
```

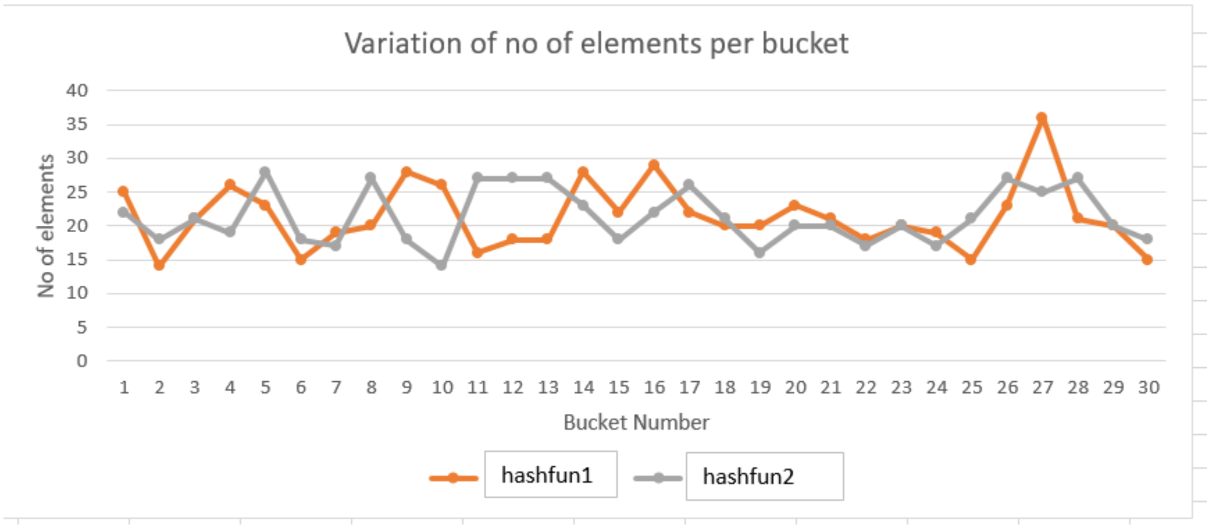
```
The minimum no of entries : 14
The maximum no of entries : 36
Total elements : 641
Average no of elements per bucket : 21.366667
Standard deviation : 4.778307439341182
```

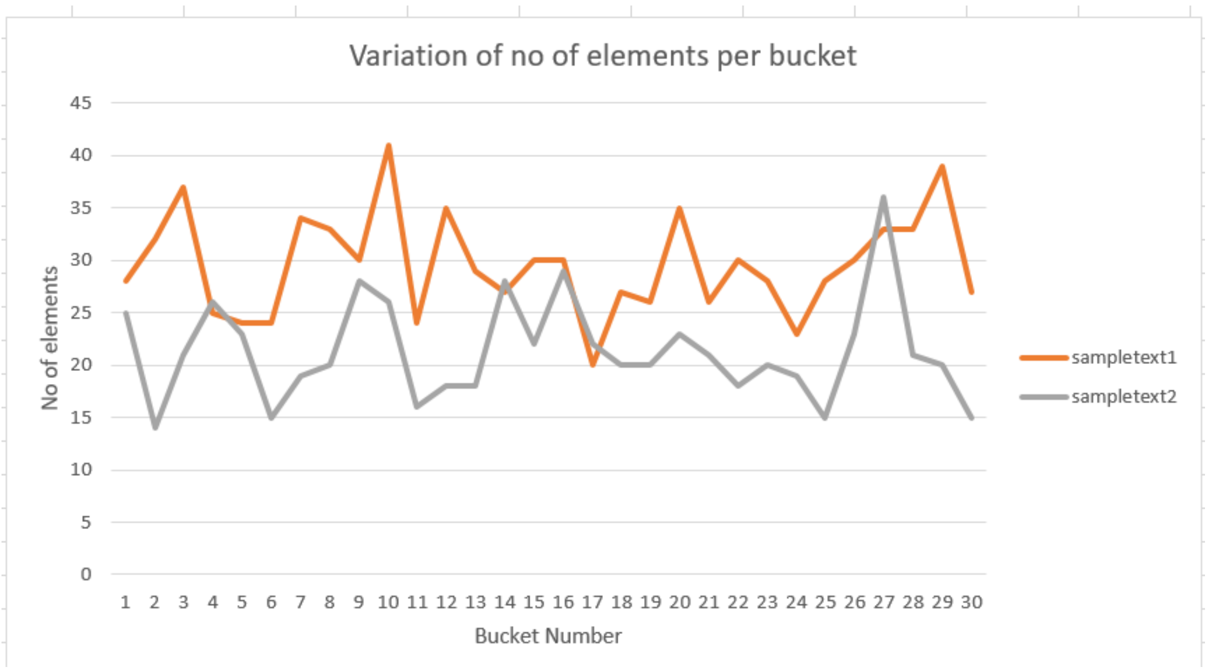
b) For hash function 2,

```
public int hashfunction(String key){
            //set the multiplier
            int mul=71;
            int h=0;

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



```
The minimum no of entries : 14
The maximum no of entries : 28
Total elements : 641
Average no of elements per bucket : 21.366667
Standard deviation : 4.01234177131861
```





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

About division method

```
public int hashfunctionDivision(String key){



                    int h=0;
                    //sum up the ascci value of the word
                    for(int i=0;i<key.length();i++){
                     h=(key.charAt(i))%hashTable.length;
                    }
                    //return modulus of sum of ascii values
                    return h%hashTable.length;
          }
```

For sample text2

```
The minimum no of entries : 14
The maximum no of entries : 36
Total elements : 641
Average no of elements per bucket : 21.366667
Standard deviation : 4.778307439341182
```

For sample text1

```
The minimum no of entries : 20
The maximum no of entries : 41
Total elements : 888
Average no of elements per bucket : 29.6
Standard deviation : 4.820788095713141
```

About multiplication method 2

```
public int hashfunctionMultiplication(String key){
                    //can chage the A value;
                    double A=0.61;
                    int h;
                    long sum=0;
                    //sum up the ascci value of the word
                    for(int i=0;i<key.length();i++){
                          sum=sum+(int)key.charAt(i);


                    }
                    //return modulus of sum of ascii values
                    h=(int)Math.floor(((hashTable.length)*((A*sum)%1)));
                    return h;
              }
```

sample test1.txt

```
The minimum no of entries : 20
The maximum no of entries : 41
Total elements : 888
Average no of elements per bucket : 29.6
Standard deviation : 4.820788095713141
```

sample test2.txt

```
The minimum no of entries : 14

The maximum no of entries : 36

Total elements : 641

Average no of elements per bucket : 21.366667

Standard deviation : 4.778307439341182
```

Comparing those Standard deviations we can say we have to choose what is the good hash function according to our purpose