Amasith KTD

E/16/025

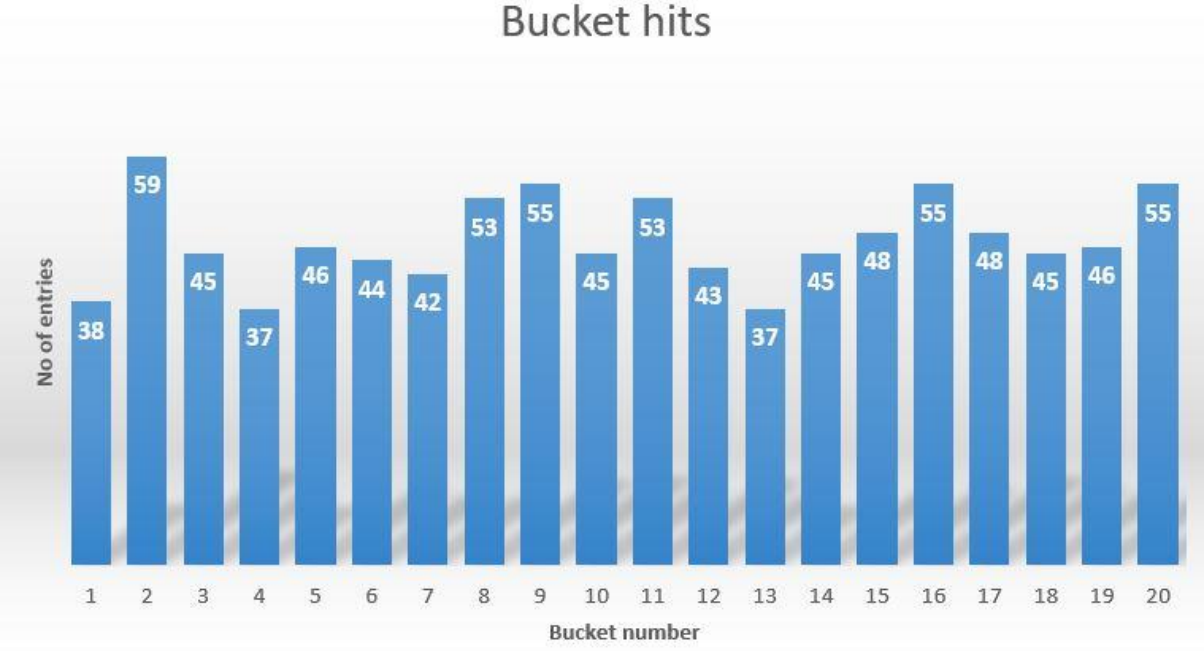
Co322-Lab3 -Report

1) For sample-text1.txt file

a) Hash Code 1

```
private int hash(String key)
 {
       int h = 0;
       for (int i = 0; i < key.length(); i++)
       {
             h = (29* h + key.charAt(i))%table.length;
       }
       return h%table.length;  //mapping to table length
 }
```

1) For Hash Code 1 when bucket size 20



Min keys in bucket : 37

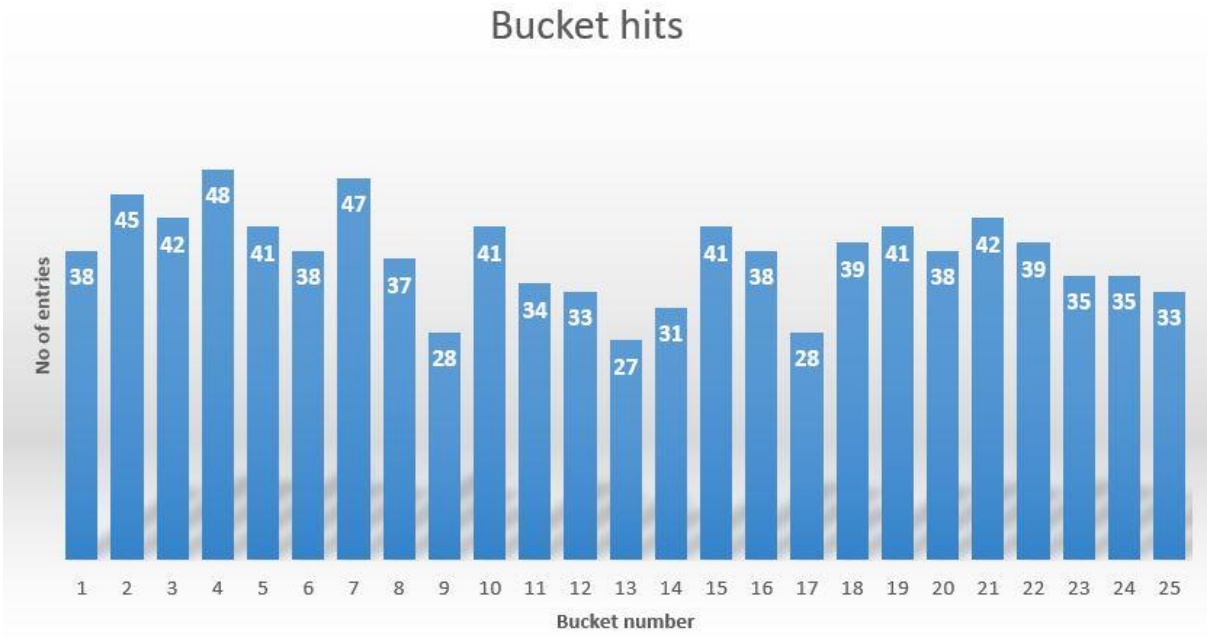
Max keys in bucket : 59

Total keys : 939

Avg keys in one bucket:46.95

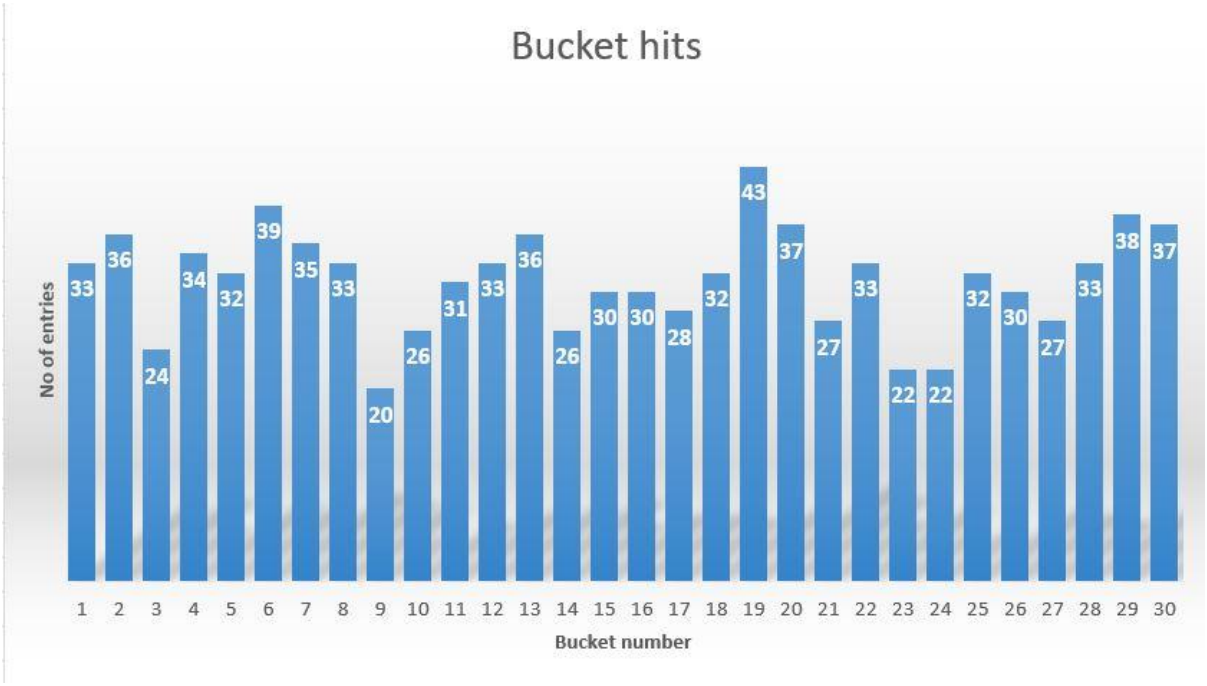
Deviation of the keys in buckets:6.139829783114919

2) For Hash Code 1 when bucket size 25



Min keys in bucket : 27
Max keys in bucket : 48
Total keys : 939
Avg keys in one bucket:37.56
Deviation of the keys in buckets:5.407988340466813

3) For Hash Code 1 when bucket size 30



Min keys in bucket : 20

Max keys in bucket : 43

Total keys : 939

Avg keys in one bucket:31.3

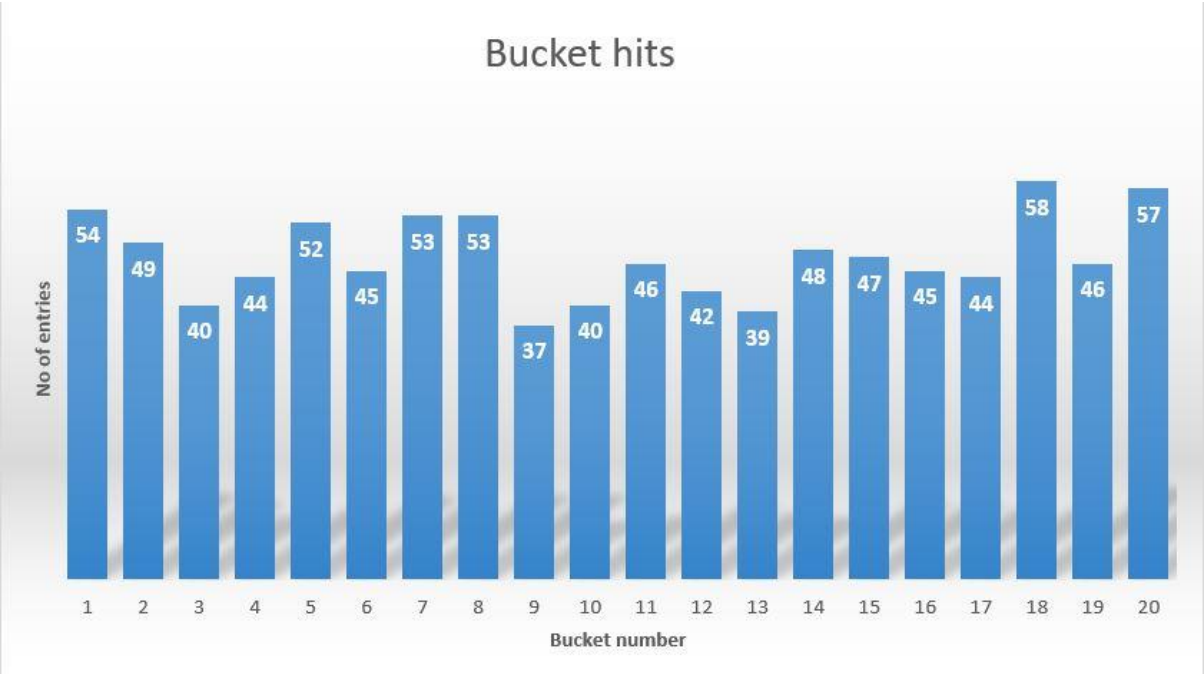
Deviation of the keys in buckets:5.320719743958518

b) Hash Code 2

```
private int hash(String key)
 {
      int h = 0;
```

```
        for (int i = 0; i < key.length(); i++)
        {
              h = (67* h + key.charAt(i))%table.length;
        }
        return h%table.length;  //mapping to table length
  }
```

1) For Hash Code 2 when bucket size 20



Min keys in bucket : 37

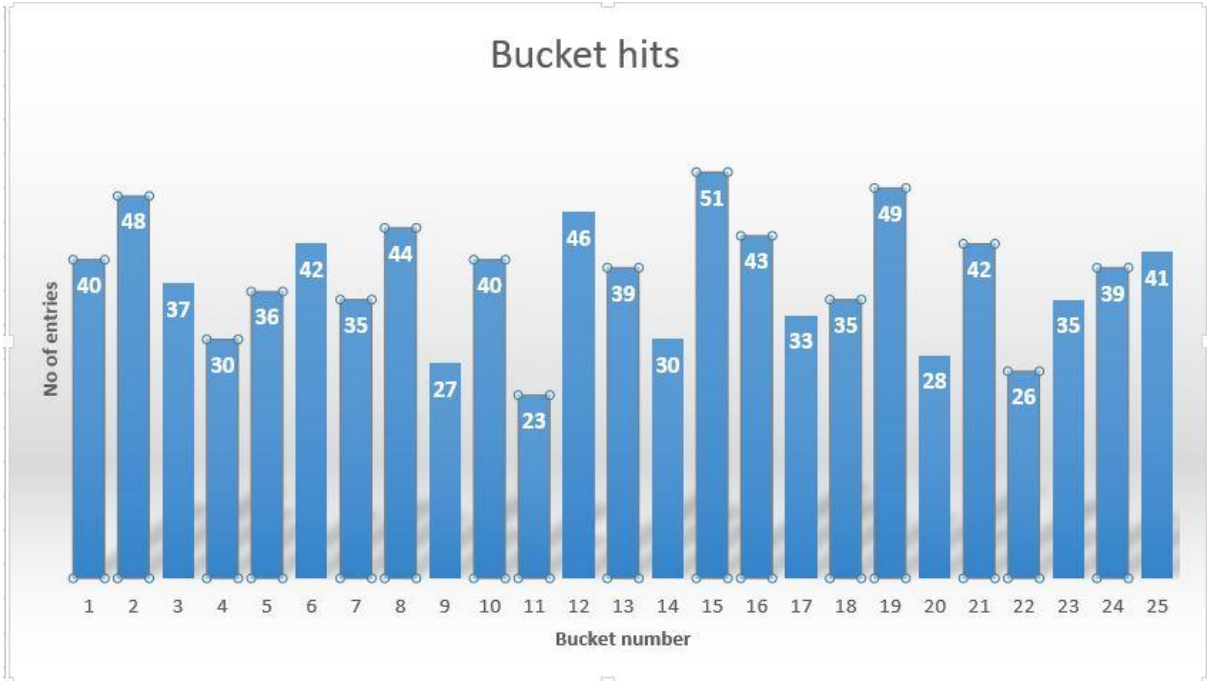
Max keys in bucket : 58

Total keys : 939

Avg keys in one bucket:46.95

Deviation of the keys in buckets:5.804955621331226

2) For Hash Code 2 when bucket size 25



Min keys in bucket : 23

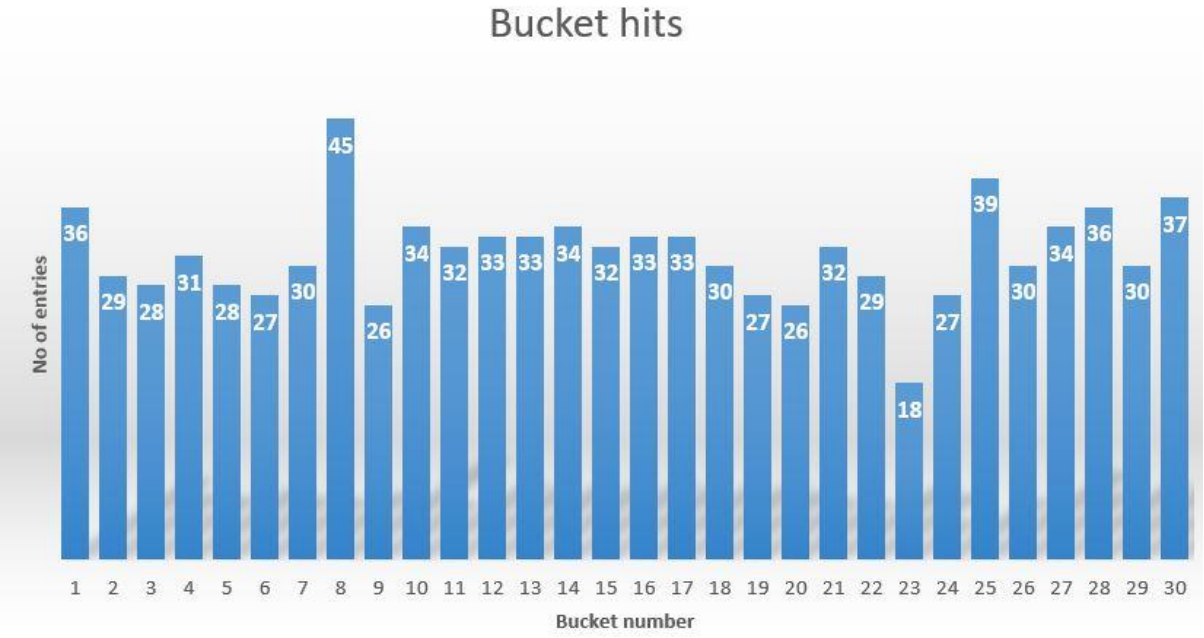
Max keys in bucket : 51

Total keys : 939

Avg keys in one bucket:37.56

Deviation of the keys in buckets:7.297008831749143

3) For Hash Code 2 when bucket size 30

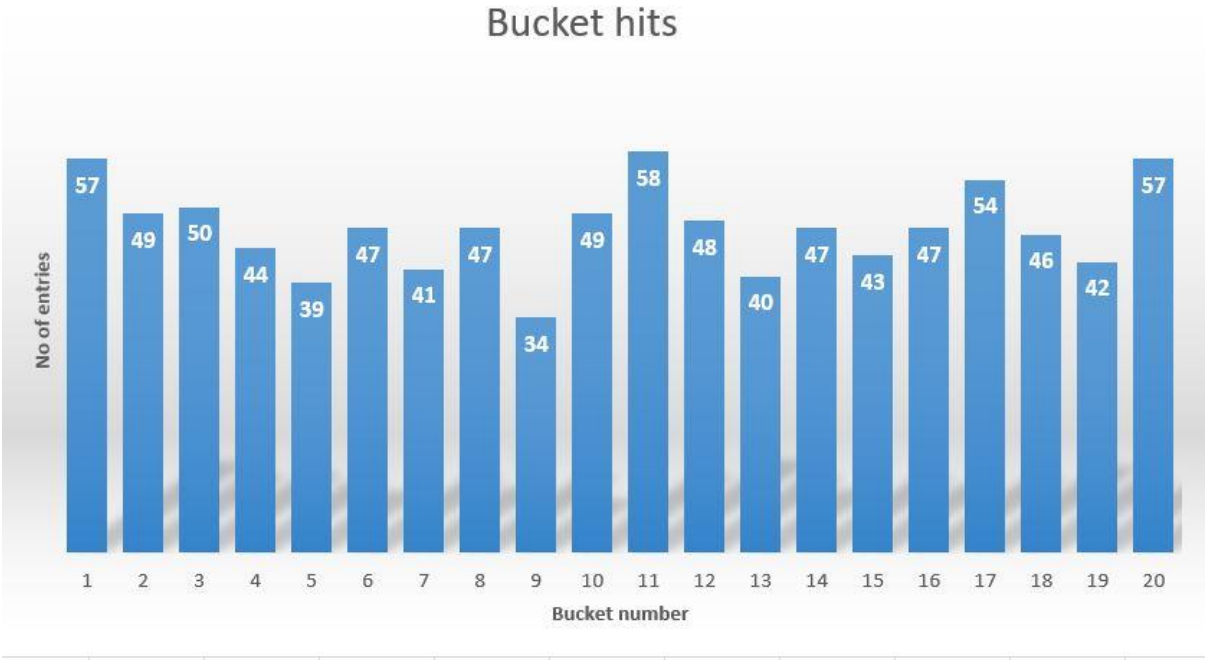



```
Min keys in bucket : 18
Max keys in bucket : 45
Total keys : 939
Avg keys in one bucket:31.3
Deviation of the keys in buckets:4.723352474011441
```

c) Hash Code 3

```
private int hash(String key)
   {
         int h = 0;
         for (int i = 0; i < key.length(); i++)
         {
               h = (541* h + key.charAt(i))%table.length;
         }
         return h%table.length;  //mapping to table length
   }
```

1) For Hash Code 3 when bucket size 20



Min keys in bucket : 34

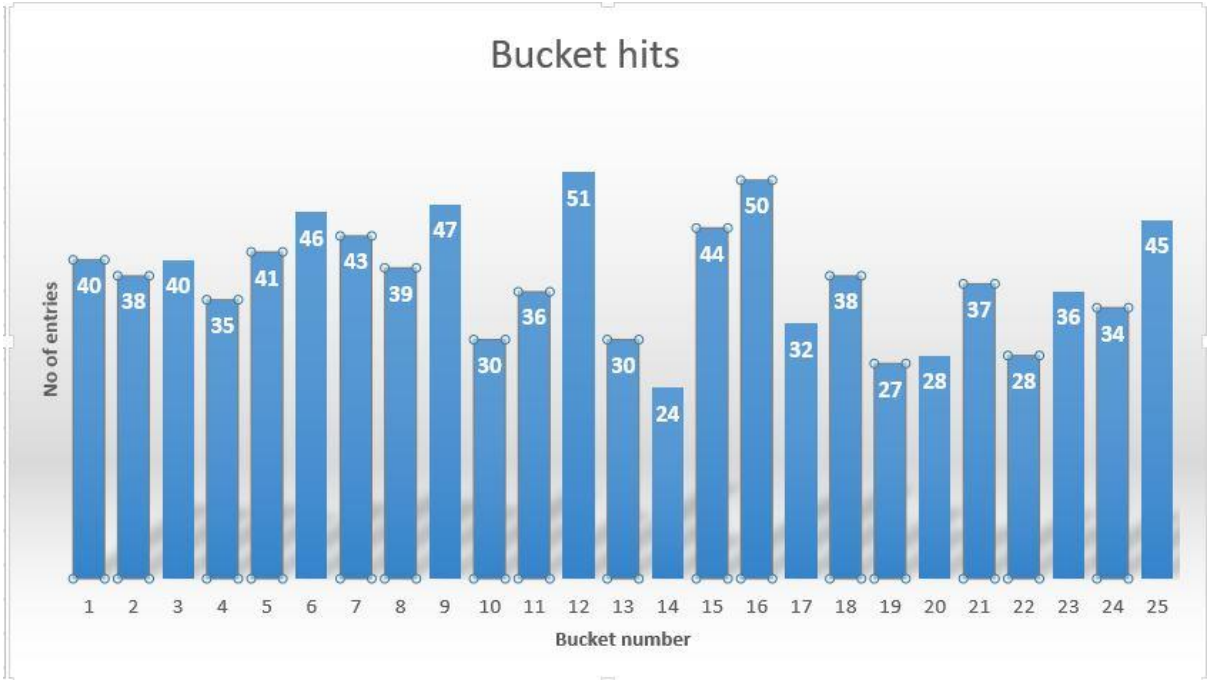
Max keys in bucket : 58

Total keys : 939

Avg keys in one bucket:46.95

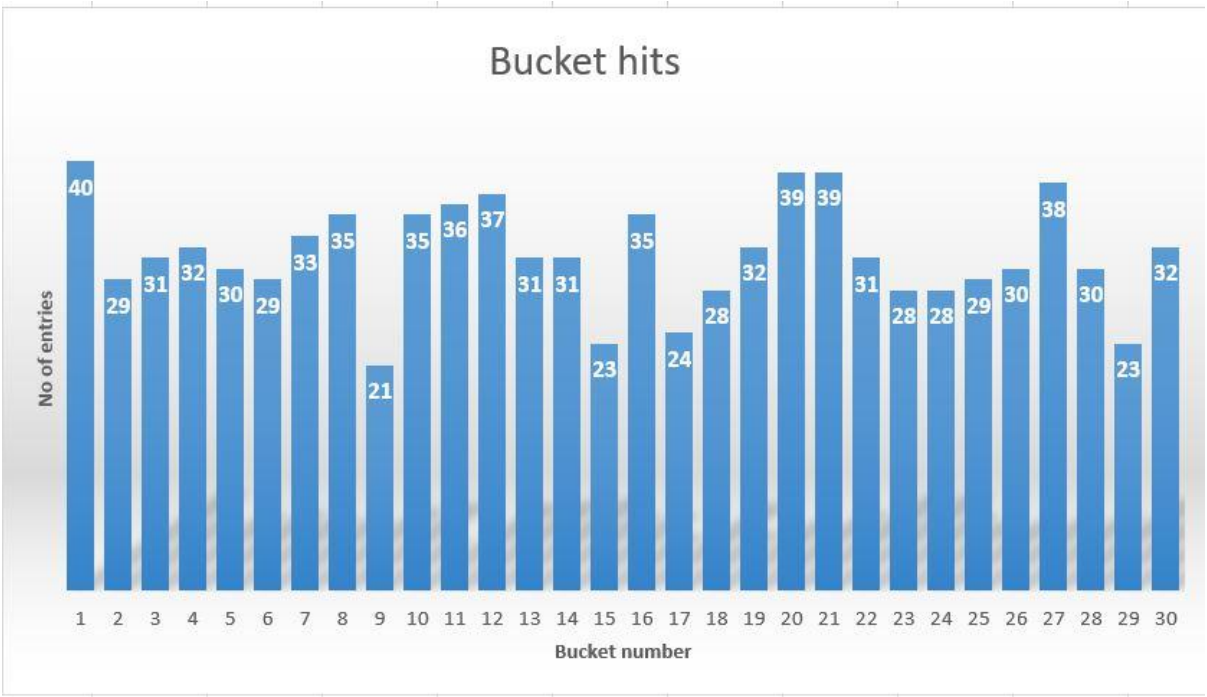
Deviation of the keys in buckets:6.139829783114919

2) For Hash Code 3 when bucket size 25



Min keys in bucket : 24
Max keys in bucket : 51
Total keys : 939
Avg keys in one bucket:37.56
Deviation of the keys in buckets:7.158654754255509

3) For Hash Code 3 when bucket size 30



Min keys in bucket : 21

Max keys in bucket : 40

Total keys : 939

Avg keys in one bucket:31.3

Deviation of the keys in buckets:4.723352474011441

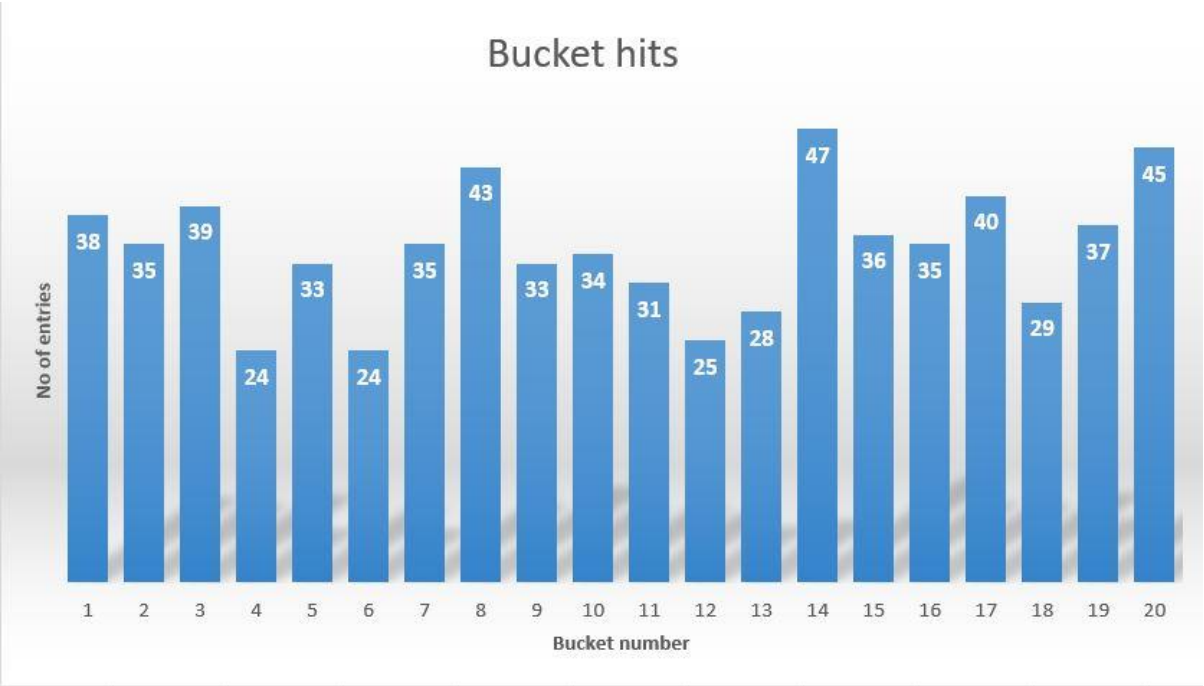
When we compare hash codes 1,2 & 3 we can see that average hits per bucket is same for all hash codes when number of buckets are fixed. Also three hash codes were able to distribute keys through buckets in an almost uniform manner. when consider minimum, maximum hits and Standard Deviation of hits/bucket, hash code 3 has better performances Thus we can say that hash code 3 will be more suitable to expect a better performance of the hash table.

2) For sample-text2.txt file

a) Hash Code 1

```
private int hash(String key)
   {
         int h = 0;
         for (int i = 0; i < key.length(); i++)
         {
               h = (29* h + key.charAt(i))%table.length;
         }
         return h%table.length;  //mapping to table length
   }
```

1) For Hash Code 1 when bucket size 20



Min keys in bucket : 24

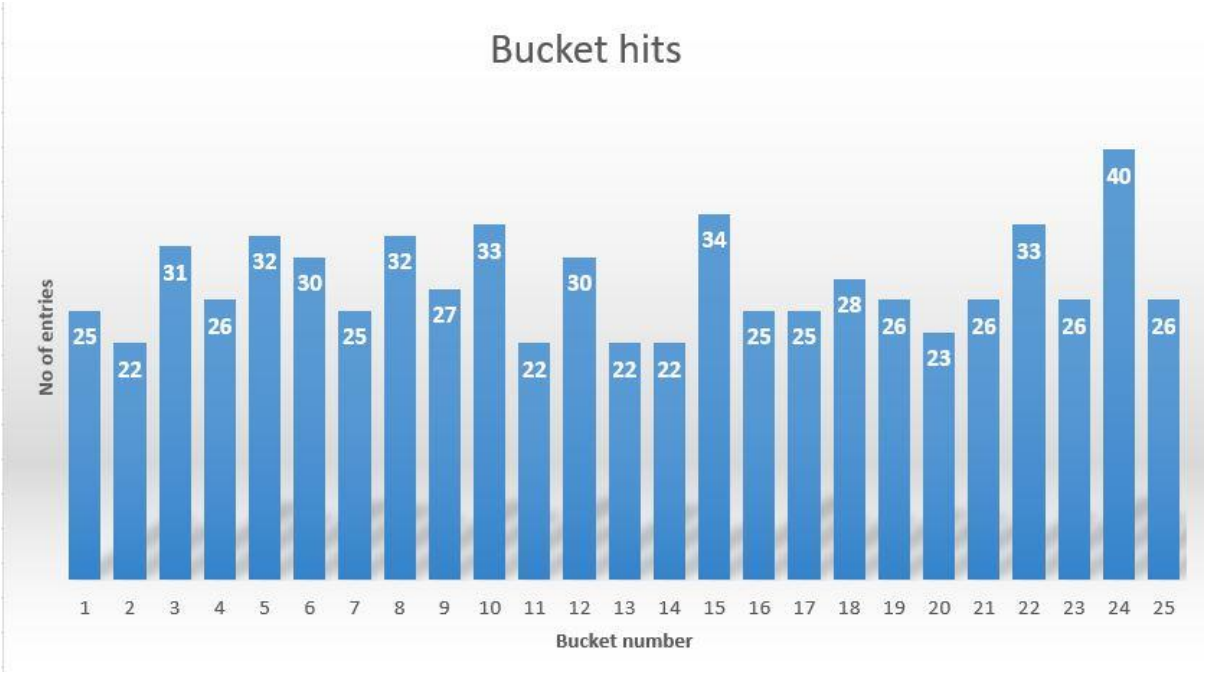
Max keys in bucket : 47

Total keys : 691

Avg keys in one bucket:34.55

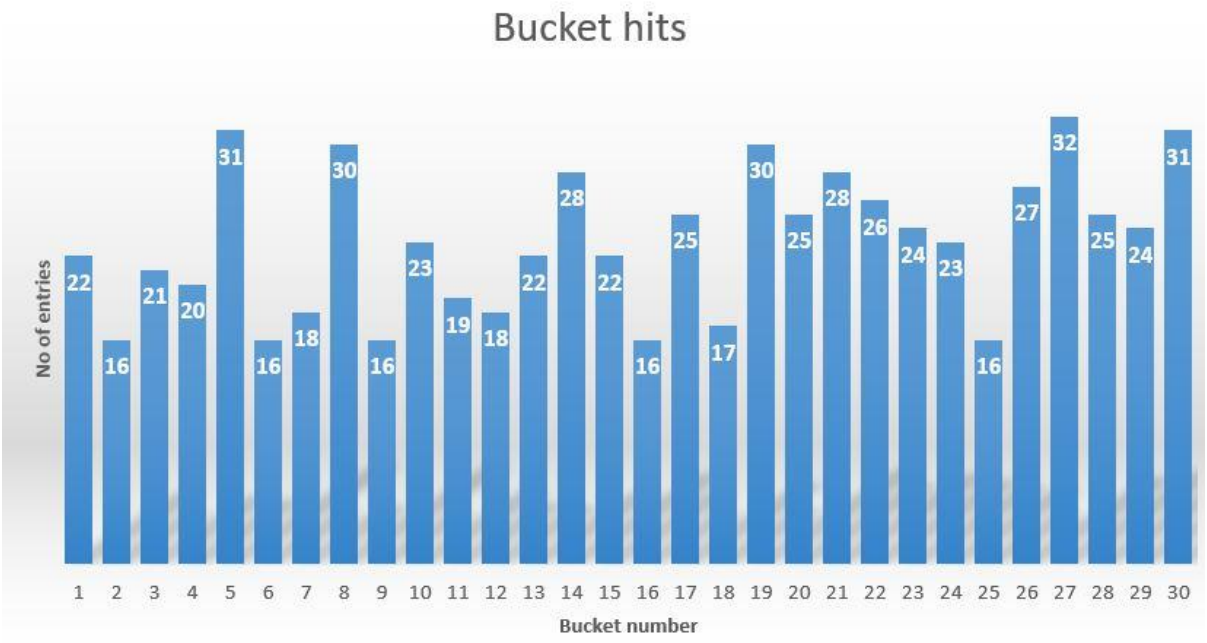
Deviation of the keys in buckets:6.348039651882697

2) For Hash Code 1 when bucket size 25



Min keys in bucket : 22
Max keys in bucket : 40
Total keys : 691
Avg keys in one bucket:27.64
Deviation of the keys in buckets:4.475539804645776

3) For Hash Code 1 when bucket size 30



Min keys in bucket : 16

Max keys in bucket : 32

Total keys : 691

Avg keys in one bucket:23.033333

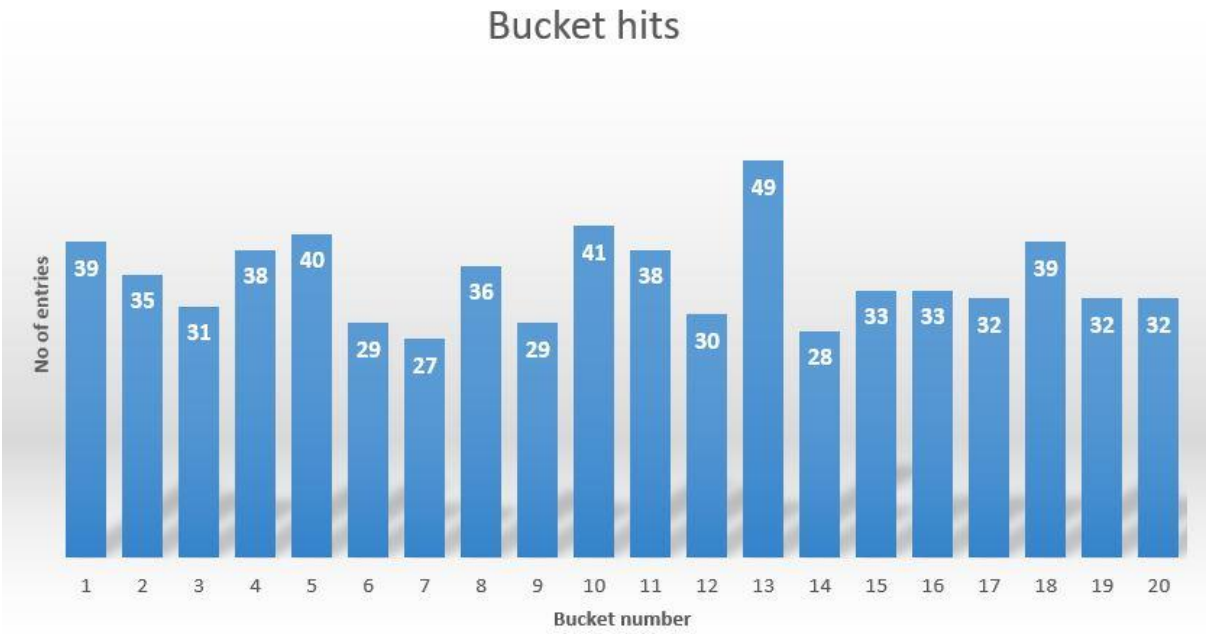
Deviation of the keys in buckets:4.9462689142296945

a) Hash Code 2

```
private int hash(String key)
   {
         int h = 0;
         for (int i = 0; i < key.length(); i++)
         {
               h = (67* h + key.charAt(i))%table.length;
         }
         return h%table.length;  //mapping to table length
```

```
}
```

1) For Hash Code 2 when bucket size 20



Min keys in bucket : 27

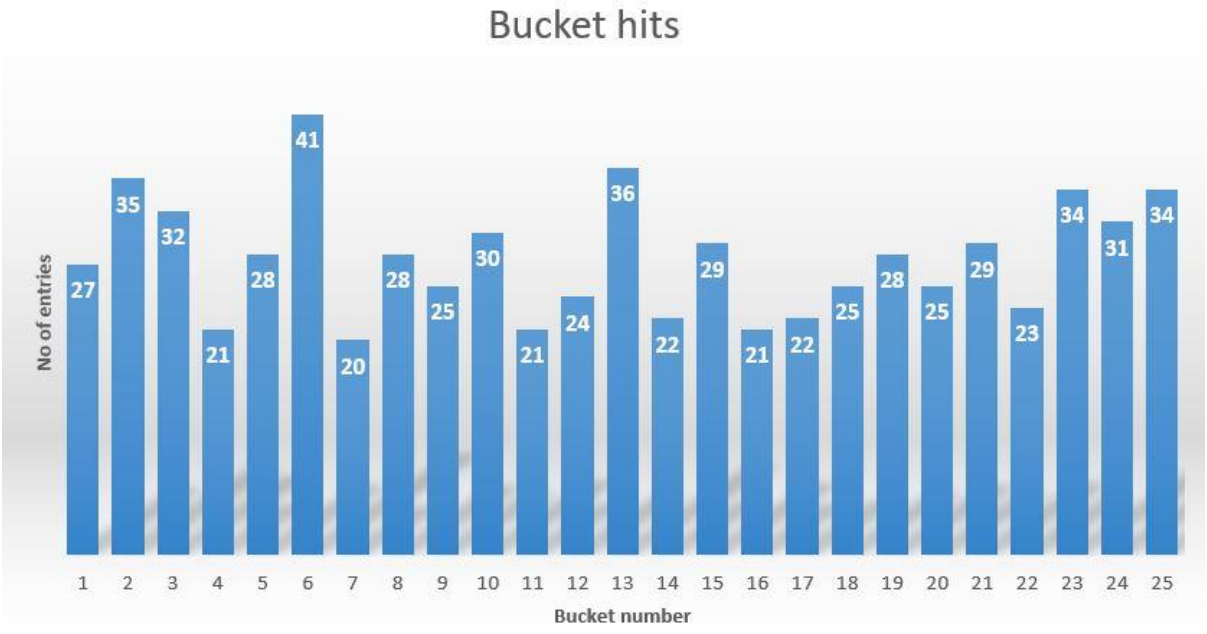
Max keys in bucket : 49

Total keys : 691

Avg keys in one bucket:34.55

Deviation of the keys in buckets:5.224711228563259

2)For Hash Code 2 when bucket size 25



Min keys in bucket : 20

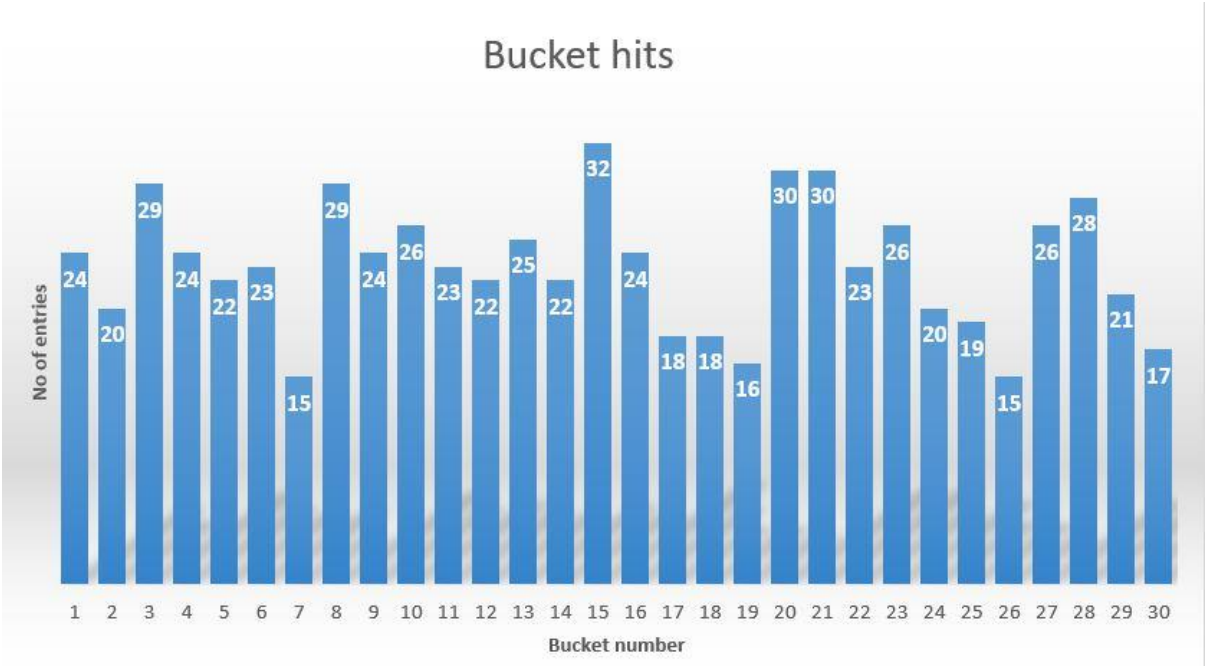
Max keys in bucket : 41

Total keys : 691

Avg keys in one bucket:27.64

Deviation of the keys in buckets:5.387991884085271

3)For Hash Code 2 when bucket size 30



Min keys in bucket : 15

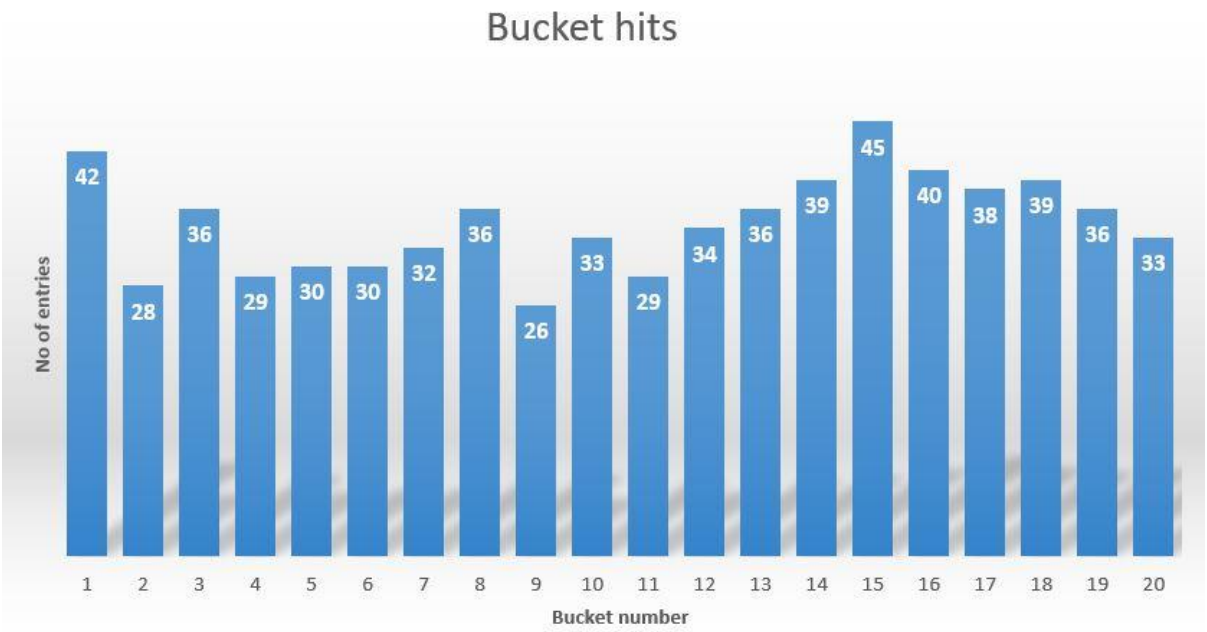
Max keys in bucket : 32

Total keys : 691
Avg keys in one bucket:23.033333
Deviation of the keys in buckets:4.523889495984069

a) Hash Code 3

```
private int hash(String key)
   {
         int h = 0;
         for (int i = 0; i < key.length(); i++)
         {
               h = (541* h + key.charAt(i))%table.length;
         }
         return h%table.length;  //mapping to table length
   }
```

1)For Hash Code 3 when bucket size 20



Min keys in bucket : 26

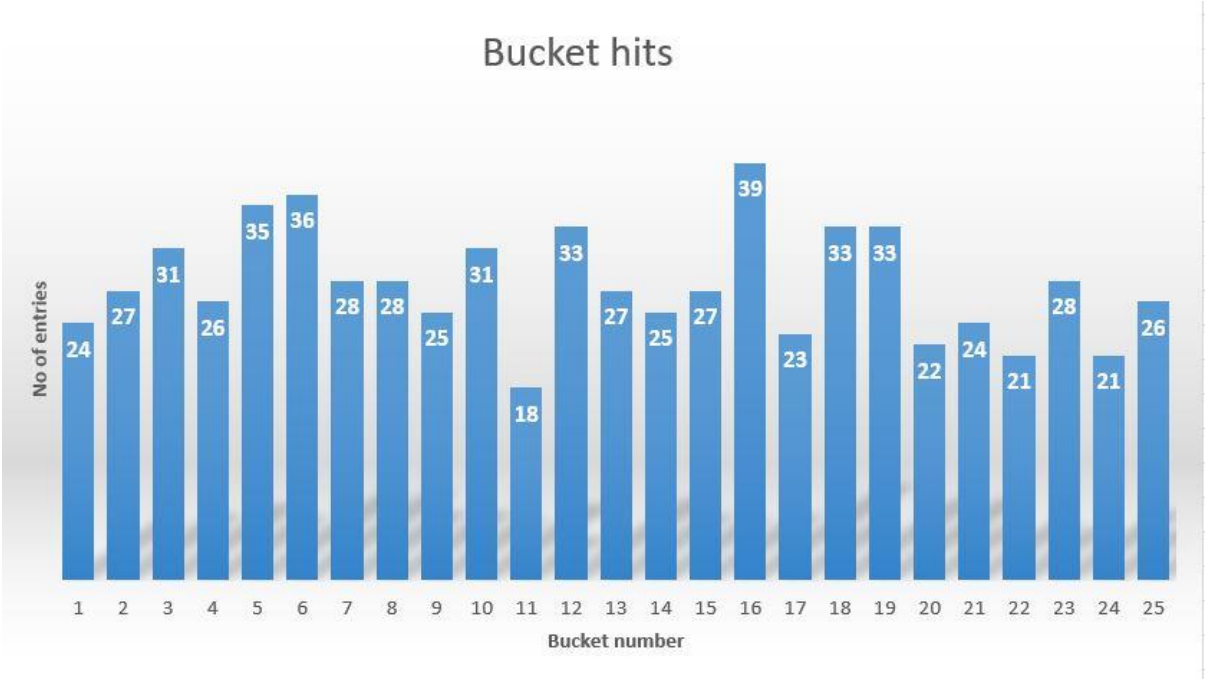
Max keys in bucket : 45

Total keys : 691

Avg keys in one bucket:34.55

Deviation of the keys in buckets:4.826759515645564

2)For Hash Code 3 when bucket size 25



Min keys in bucket : 18

Max keys in bucket : 39

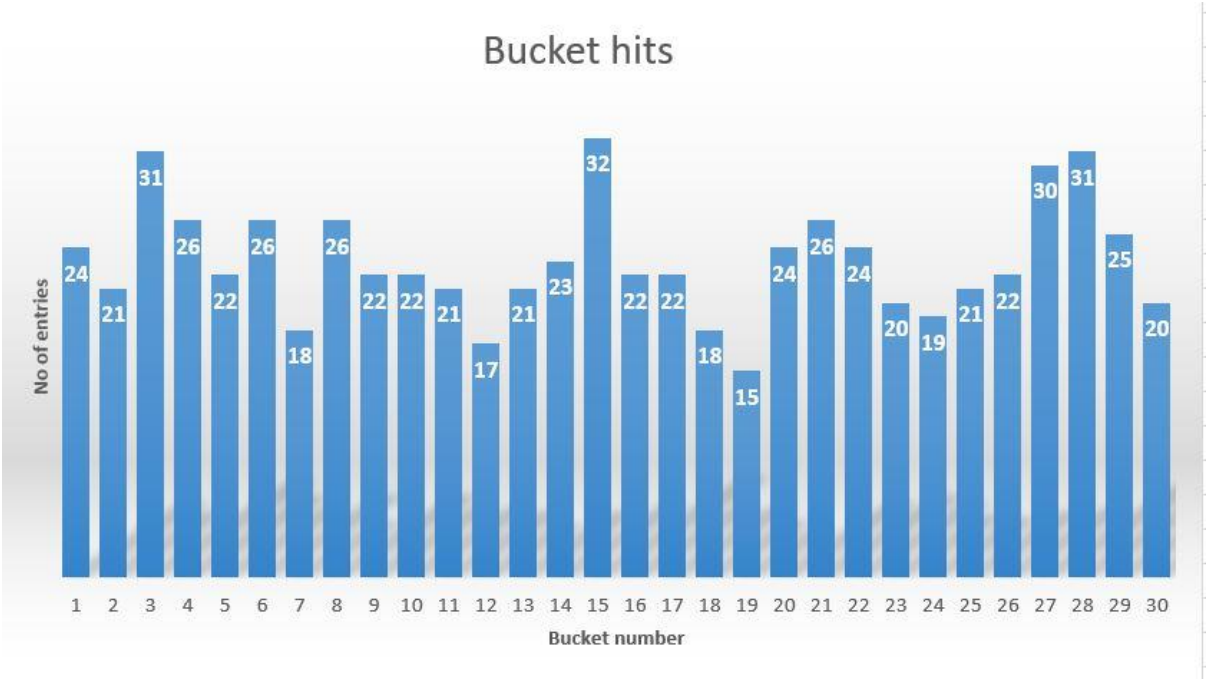
Total keys : 691

Avg keys in one bucket:27.64

Deviation of the keys in buckets:5.003044727260466

30

2) For Hash Code 3 when bucket size 30



Min keys in bucket : 15

Max keys in bucket : 32

Total keys : 691

Avg keys in one bucket:23.033333

Deviation of the keys in buckets:4.057779709628777

All above values changes for different bucket sizes. Here bucket sizes 20,25 and 30 are considered for all the situations for comparing.

When we compare hash codes 1,2 & 3 for text file 2 we can see that average hits per bucket is same for all hash codes when number of buckets are fixed. Also three hash codes were able to distribute keys through buckets in an almost uniform manner. when consider minimum, maximum hits and Standard Deviation of hits/bucket, hash code 3 has better performances Thus we can say in this case also hash code 3 will be more suitable better performance of the hash table.

A good hash function should have the following properties:

1. Efficiently computable.
2. Should uniformly distribute the keys (Each table position equally likely for each key)

When the same hash function is used for different files it has given different results since the complexity of vocabulary used in 2 files is almost completely different therefore the domain of generated hash codes are also completely different. However despite the complexity of 2 files when an odd number is used as the multiplier in the hash function better uniformity has been occurred than when an even number is used. Also several odd numbers has given the best uniformity above all other odd multipliers. Hash code 3 is an example for such situation.