CO322: Data structures and algorithms

LAB 03

Hash Function 1

```java
public static int hashone(String word,int buckets){



        char ch[];
        ch = word.toCharArray();
        int wordlength = word.length();

        int i, sum;
        for (sum = 0, i = 0; i < word.length(); i++)
            sum += ch[i];

                return sum % buckets;


}
```

This function takes the sum of ASCII values of the letters of the word. Order of the characters does not affect the result. Finally the modulus of the sum has taken to generate a value in the range of buckets.

Hash Function 2

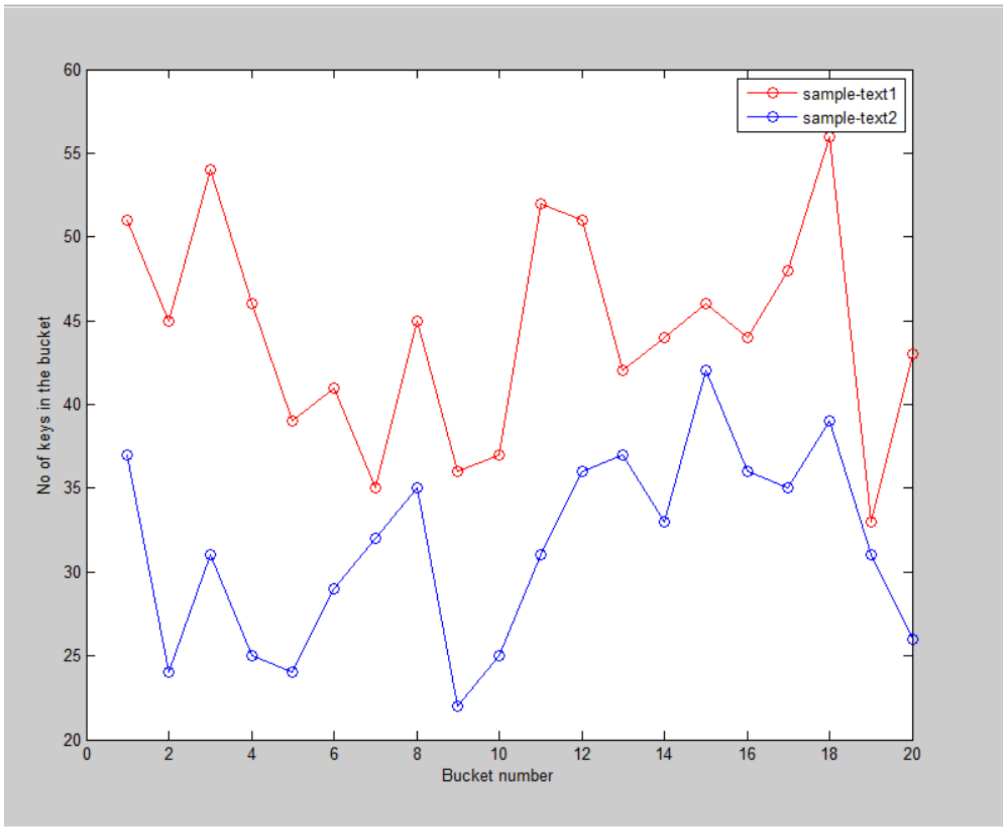
```java
public int hashtwo(String word,int buckets) {
    int intLength = word.length() / 4;
    long sum = 0;
    for (int j = 0; j < intLength; j++) {
        char c[] = word.substring(j * 4, (j * 4) + 4).toCharArray();
        long mult = 1;
        for (int k = 0; k < c.length; k++) {
            sum += c[k] * mult;
            mult *= 256;
        }
    }

    char c[] = word.substring(intLength * 4).toCharArray();
    long mult = 1;
    for (int k = 0; k < c.length; k++) {
        sum += c[k] * mult;
        mult *= 256;
    }

    return (int) (Math.abs(sum) % buckets);


}
```

This function takes a word and processes the word 4 bytes at a time(4 characters at a time). And it then interprets each of the 4 byte chunks as a single long integer value. The integer values for the 4 byte chunks are added together. And at the end, the resulting sum is converted to the range of buckets using modulus operator.

1) .Analysis of hash function 1 and hash table size =20

For sample-text1.txt bucket no 1-20 no of keys =[51,45,54,46,39,41,35,45,36,37,52,51,42,44,46,44,48,56,33,43]

For sample-text2.txt bucket no 1-20 no of keys =[37,24,31,25,24,29,32,35,22,25,31,36,37,33,42,36,35,39,31,26]



Here hash table size(M buckets) and the hash function is constant. Therefore, according to the graph, it is clear that the no of keys in each bucket has changed depending on the input text file.

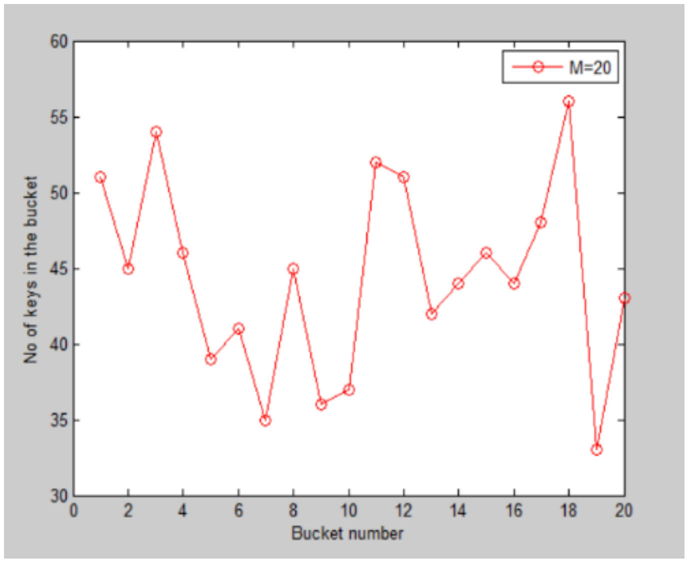
2) .Analysis of hash function 1 and sample-text1.txt file

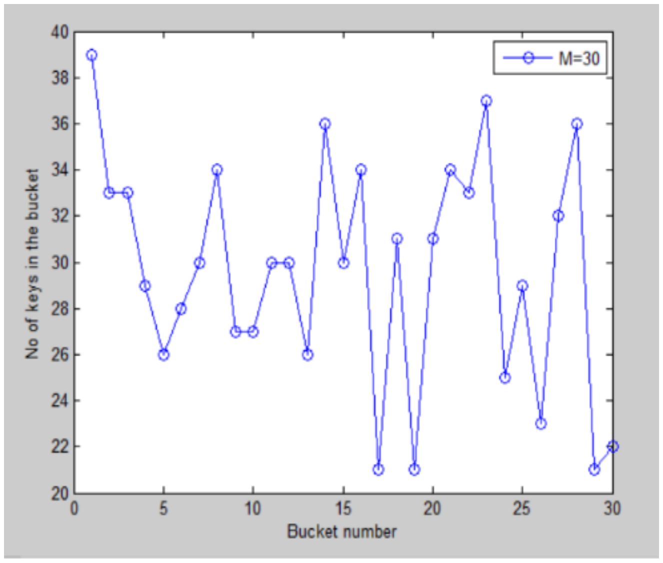
Here input text file and the hash function is constant. Only the hash table size changes.

For hash table size =20 - [51,45,54,46,39,41,35,45,36,37,52,51,42,44,46,44,48,56,33,43]

For hash table size =30 -
```
[39,33,33,29,26,28,30,34,27,27,30,30,26,36,30,34,21,31,21,31,34,33,37,25,29,23
,32,36,21,22]
```

For hash table size =40 -
```
[24,17,22,19,24,15,16,25,19,21,28,28,21,22,25,22,22,31,16,18,27,28,32,27,15,2
6,19,20,17,16,24,23,21,22,21,22,26,25,17,25]
```





Maximum no of keys in a bucket= 56

Minimum no of keys in a bucket= 33

Average no of keys in a bucket = 888/20

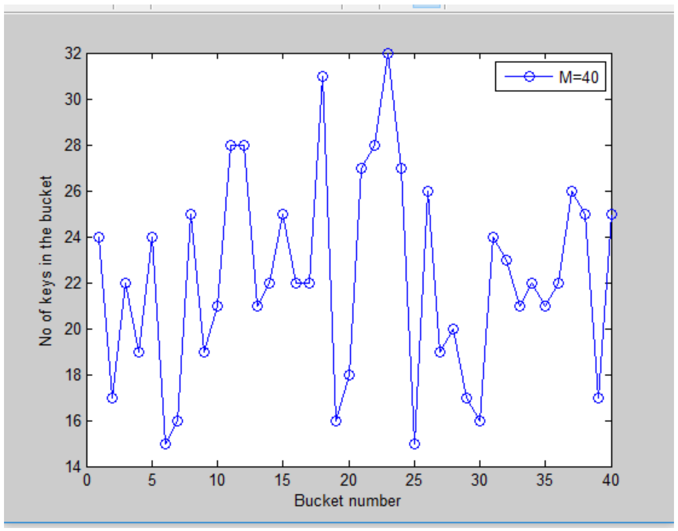
= 44.4

Maximum no of keys in a bucket= 39

Minimum no of keys in a bucket= 21

Average no of keys in a bucket =888/30

= 29.6



Maximum no of keys in a bucket= 32

Minimum no of keys in a bucket= 15

Average no of keys in a bucket =888/40

= 22.2

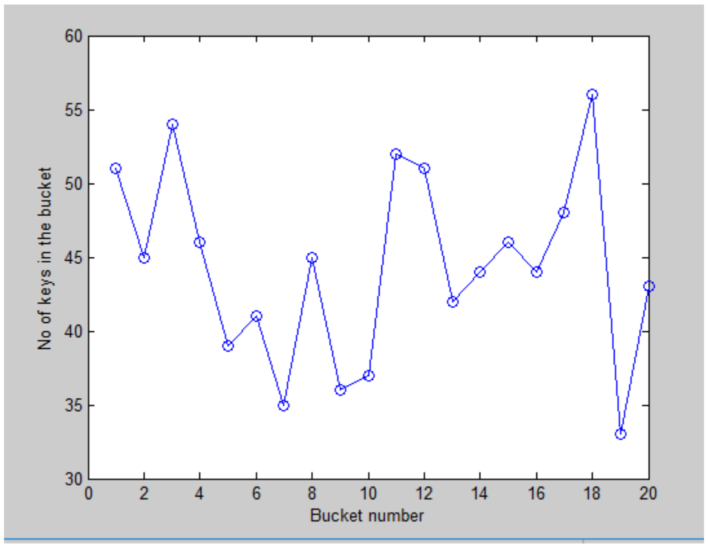
Here the input text file and the hash function is constant. Hash table size(M buckets) is changing. According to the graph, when M increases, both the maximum number of keys in a bucket and the minimum number of keys in a bucket decreases. Average number of keys in a bucket also decrease when the hash table size increases.

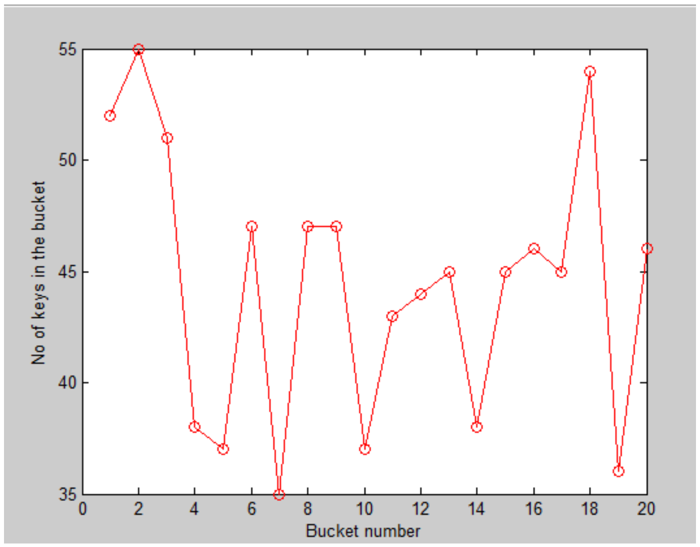
1) .Analysis of hash table size =20 and sample-text1.txt file

Here hash table size(M buckets) and the input text file is constant. Only the hash function changes.

Using hash function 1 - [51,45,54,46,39,41,35,45,36,37,52,51,42,44,46,44,48,56,33,43]

Using hash function 2 -
`[52,55,51,38,37,47,35,47,47,37,43,44,45,38,45,46,45,54,36,46]`





Hash function 1	hash function 2
Maximum no of keys in a bucket= 56	Maximum no of keys in a bucket= 55
Minimum no of keys in a bucket= 33	Minimum no of keys in a bucket= 35
Average no of keys in a bucket = 888/20	Average no of keys in a bucket = 888/20
= 44.4	=44.4

When comparing these graphs we can see that the 1st graph has spread between 23 units(from 33 to 56) and the 2nd graph has spread between 20 units (from 35 to 55). Therefore the 2nd graph has not spread as much as the 1st one. Therefore, the better hash function from these 2 functions is the 2nd one since it has distributed the keys to buckets more evenly than the 1st one.