E/15/211
CO322-Lab 03

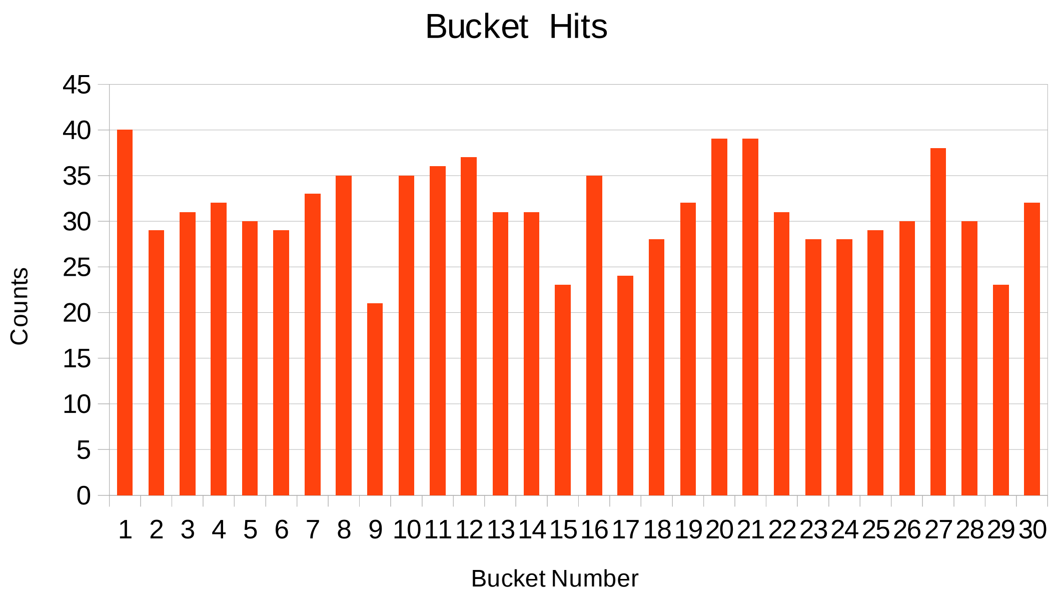
Hash Tables

(1) Considering sample-text1.text file

when bucket size is 30

case (a)

```
private int hash(String key)
  {
          int h = 0;
          for (int i = 0; i < key.length(); i++)
          {
                  h = (31*h + key.charAt(i))%table.length;
          }
          return (h%table.length);
  }
```



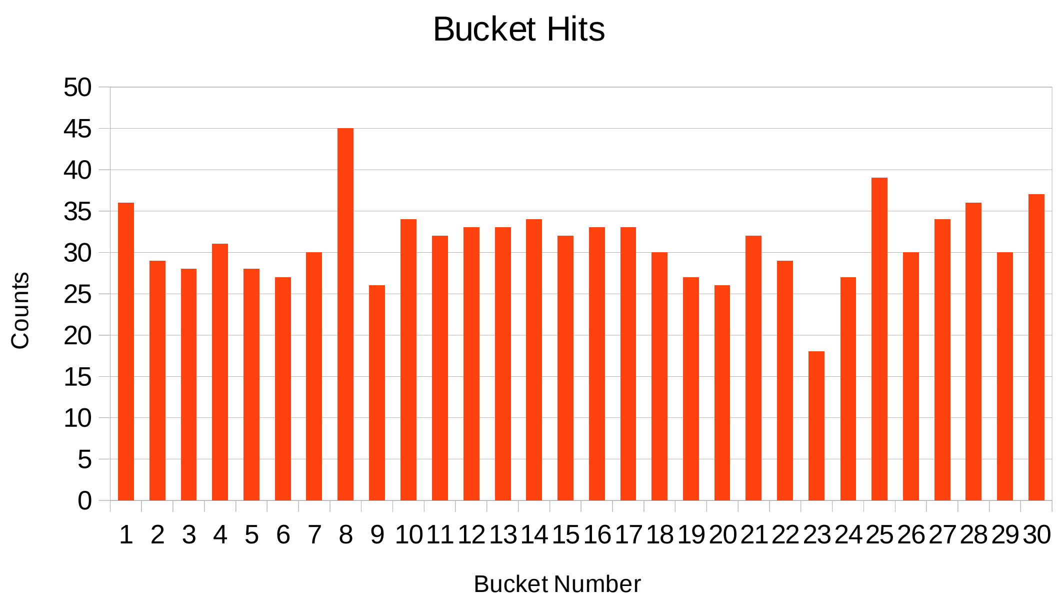
Bucket No	Counts
1	40
2	29
3	31
4	32
5	30
6	29
7	33
8	35
9	21
10	35
11	36
12	37
13	31
14	31
15	23
16	35
17	24
18	28
19	32
20	39
21	39
22	31
23	28
24	28
25	29
26	30
27	38
28	30
29	23
30	32

Maximum Hits:40
Minimum Hits:21
Average Hits:31.3
Standard Deviation:4.7234

case (b)

```
private int hash(String key)
  {
          int h = 0;
          for (int i = 0; i < key.length(); i++)
          {
                  h = (97*h + key.charAt(i))%table.length;
          }

          return (h%table.length);     }
```

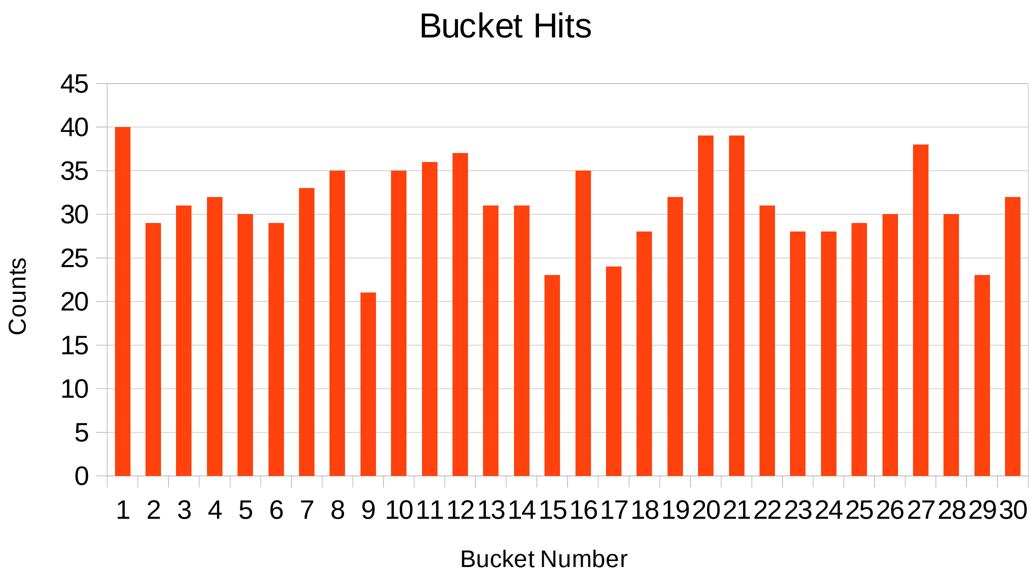


Bucket No	Counts
1	36
2	29
3	28
4	31
5	28
6	27
7	30
8	45
9	26
10	34
11	32
12	33
13	33
14	34
15	32
16	33
17	33
18	30
19	27
20	26
21	32
22	29
23	18
24	27
25	39
26	30
27	34
28	36
29	30
30	37

Maximum Hits:45
Minimum Hits:18
Average Hits:31.3
Standard Deviation:4.7234

case (c)

```
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);
  }
```



Bucket No	Counts
1	40
2	29
3	31
4	32
5	30
6	29
7	33
8	35
9	21
10	35
11	36
12	37
13	31
14	31
15	23
16	35
17	24
18	28
19	32
20	39
21	39
22	31
23	28
24	28
25	29
26	30
27	38
28	30
29	23
30	32

Maximum Hits:45
Minimum Hits:18
Average Hits:31.3
Standard Deviation:4.7234

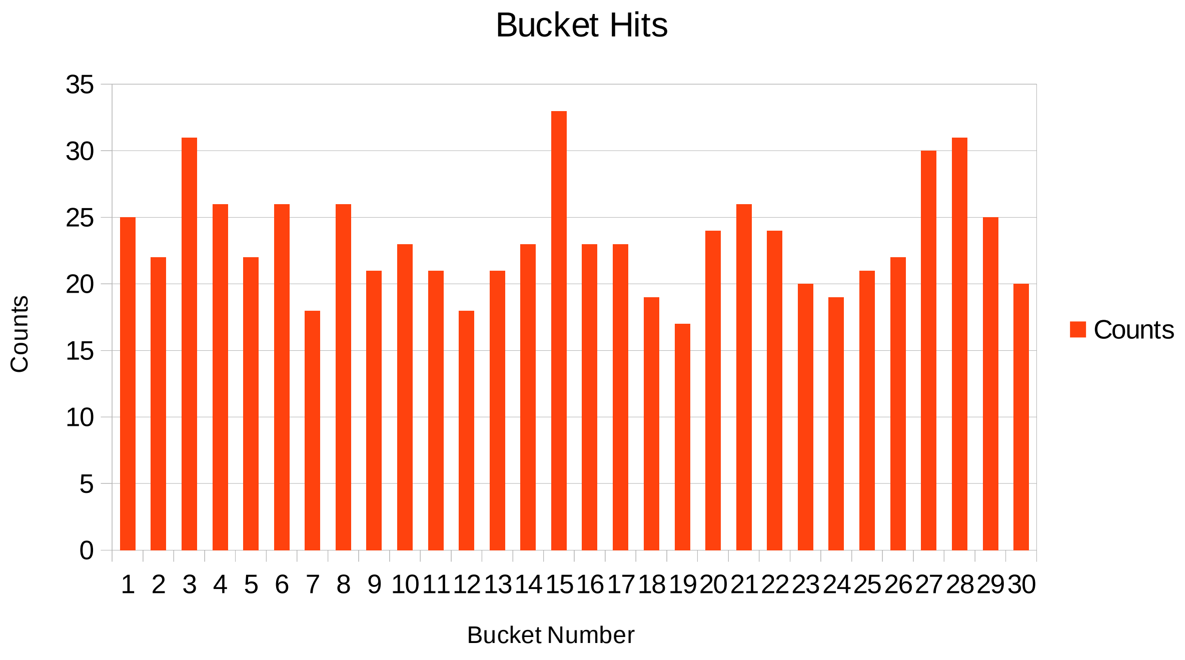
(2) Considering sample-text2.text file

when bucket size is 30

Case (a)

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

            for (int i = 0; i < key.length(); i++)          {
                    h = (31* h + key.charAt(i))%table.length;
            }
            return (h%table.length);
  }
```



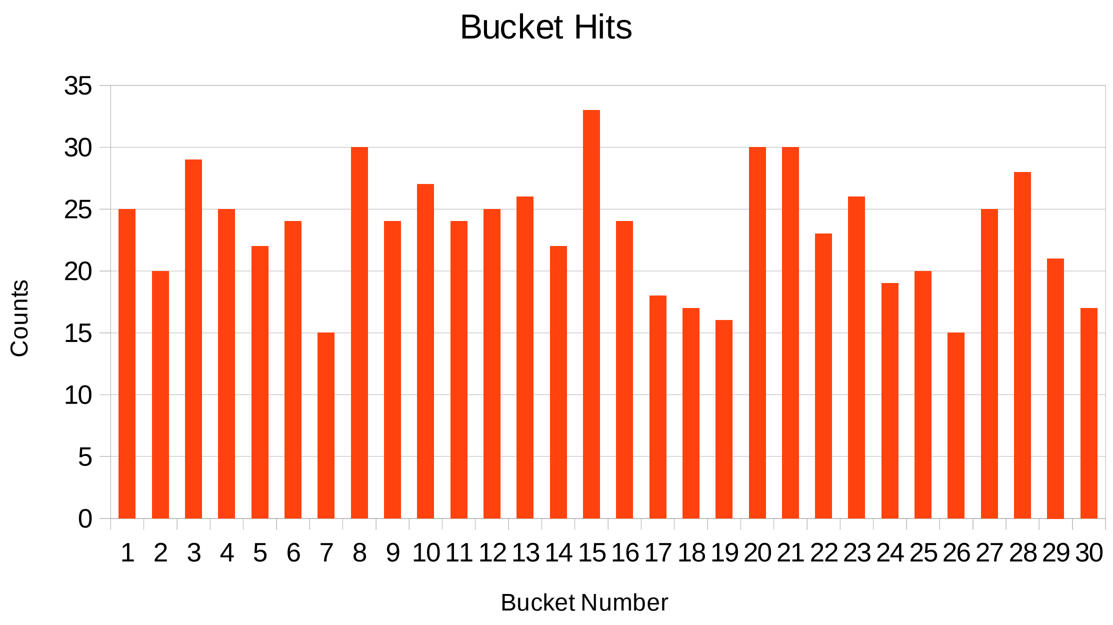
Bucket No	Counts
1	25
2	22
3	31
4	26
5	22
6	26
7	18
8	26
9	21
10	23
11	21
12	18
13	21
14	23
15	33
16	23
17	23
18	19
19	17
20	24
21	26
22	24
23	20
24	19
25	21
26	22
27	30
28	31
29	25
30	20

Maximum Hits:33
Minimum Hits:17
Average Hits:23.3333
Standard Deviation:3.9441

Case (b)

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

        for (int i = 0; i < key.length(); i++)
        {
                h = (97* h + key.charAt(i))%table.length;
        }
        return (h%table.length);
 }
```



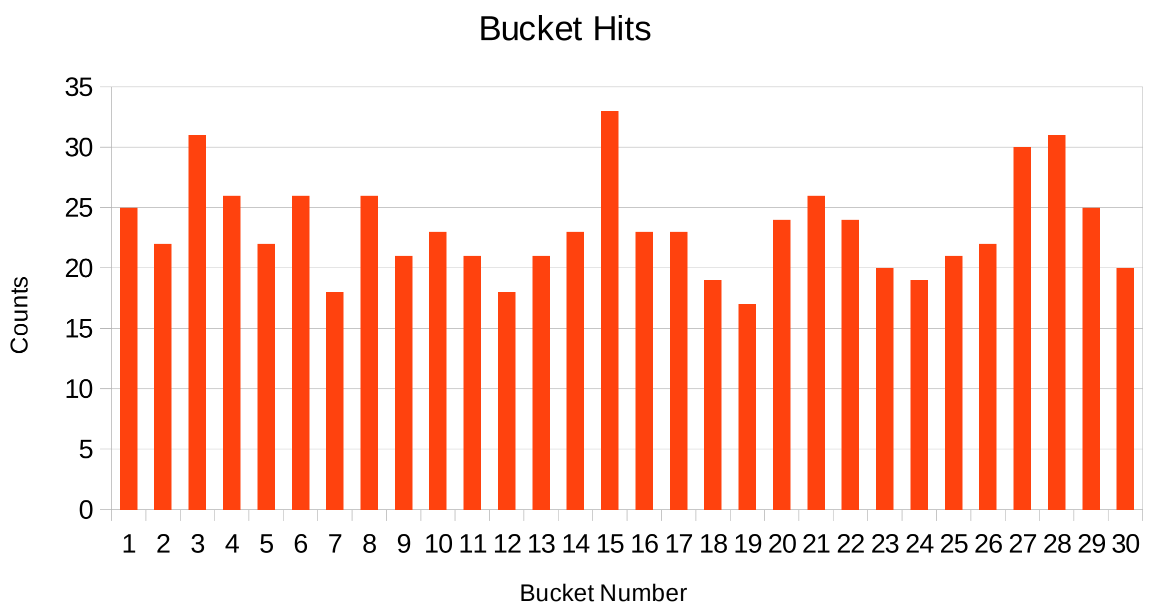
Bucket No	Counts
1	25
2	20
3	29
4	25
5	22
6	24
7	15
8	30
9	24
10	27
11	24
12	25
13	26
14	22
15	33
16	24
17	18
18	17
19	16
20	30
21	30
22	23
23	26
24	19
25	20
26	15
27	25
28	28
29	21
30	17

Maximum Hits:33
Minimum Hits:15
Average Hits:23.3333
Standard Deviation:4.6427

<u>case(c)</u>

```
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);
  }
```



Bucket No	Counts
1	25
2	22
3	31
4	26
5	22
6	26
7	18
8	26
9	21
10	23
11	21
12	18
13	21
14	23
15	33
16	23
17	23
18	19
19	17
20	24
21	26
22	24
23	20
24	19
25	21
26	22
27	30
28	31
29	25
30	20

Maximum Hits:33
Minimum Hits:17
Average Hits:23.3333
Standard Deviation:3.9441

When we compare hash codes for sample-text2 in case (a), (b) and (c) we can see that average hits per bucket is same in these three situations. Although codes were able to distribute keys through buckets in an almost uniform manner, when consider minimum, maximum hits and Standard Deviation of hits and buckets case (b) is only have little bit different variation than case(a) and case(c).

Therefore we can say that in a limited memory system where number of collisions in a bucket is not a critical issue because, here case(b) will be most suitable to expect a better performance of the hash table.

Considering Sample-text1 and Sample-text2, 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. The complexity of two files when an odd number is used as the multiplier in the hash function better uniformly has been occurred than when an even number is used. Also several odd numbers have given the best uniformly above all aother odd multipliers.(like case(c))