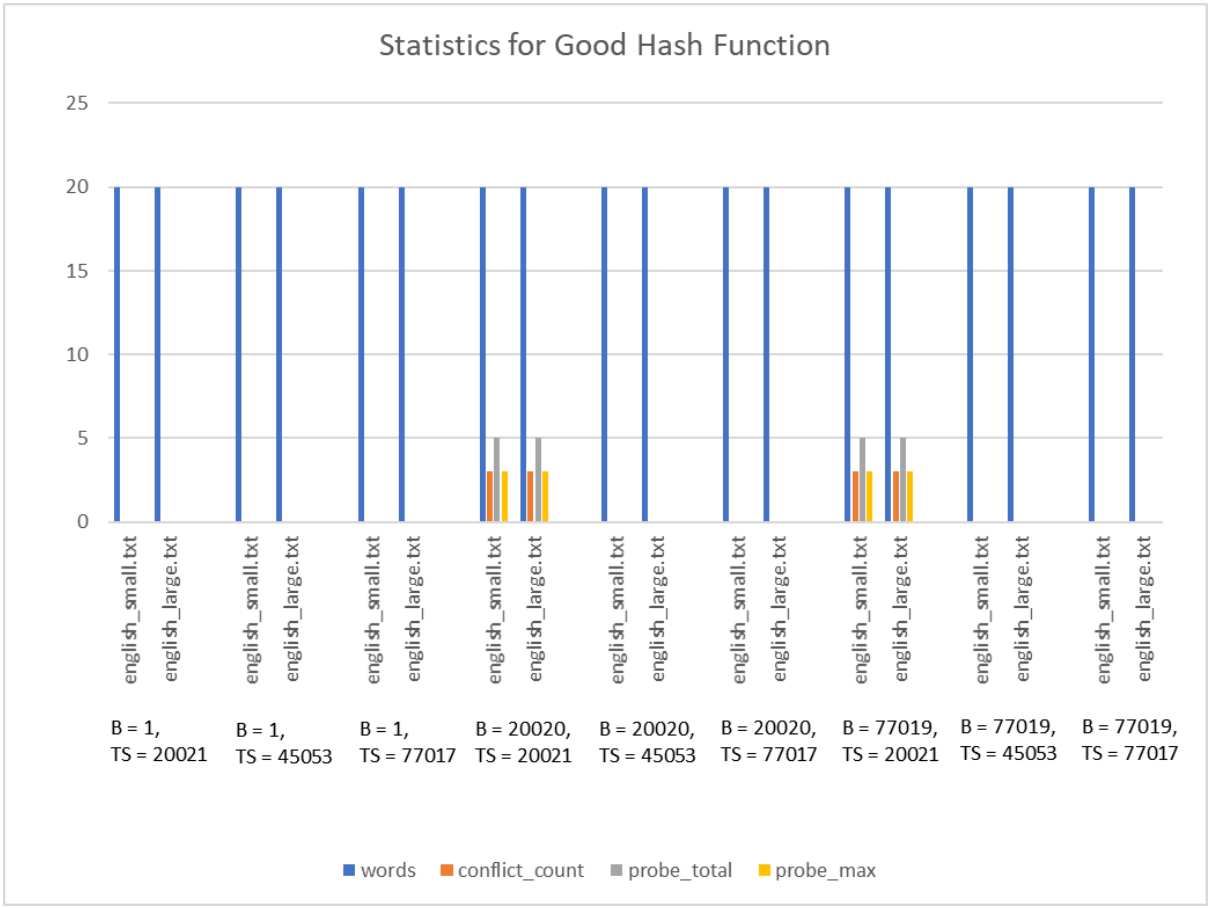
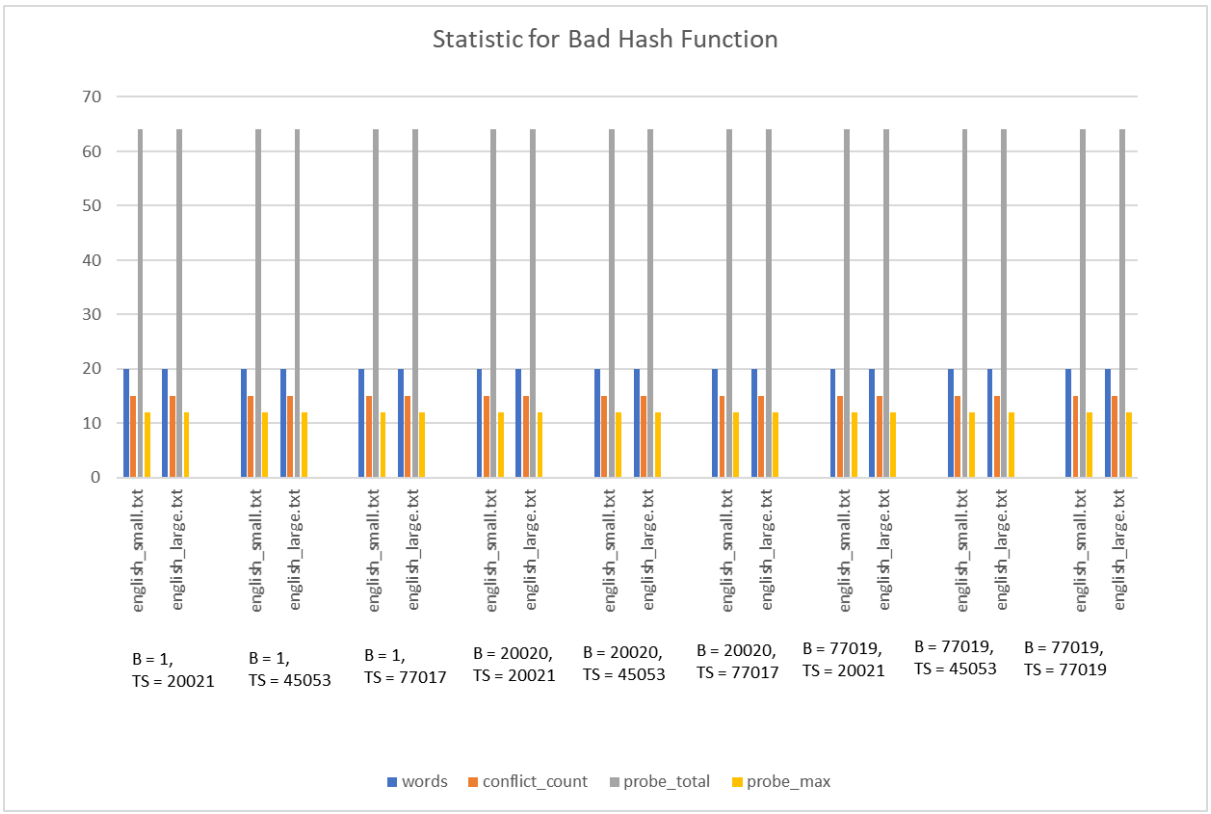
Analysis of statistics method on good hash and bad hash functions

The expected output of the statistics method for our good hash function is around (1,1,1) for a small table size, and (0,0,0) for a large table size (10000 < tablesize < 100000). It would not have many conflicts as the hashing of keys should be unique. The same goes for probing as there would not be as much probing if there are less conflicts. Small table sizes are more conflict-prone due to the mod(%) operation of table size. The good hash function does not perform that well in some cases when there are similar characters/words in the given keys (case sensitive).

The expected output of the statistics method for our bad hash function is around (2,3,2) for a small table size, and (20,30,20) for a large table size (10000 < tablesize < 100000). It would have more conflicts due to a simple hashing algorithm which returns the same position for a certain set of data (in this case all keys that start with the same character would get hashed to the same position). Since more conflicts occur, linear probing would be more often and thus leads to a bigger probe chain length. The bad hash function performs well in some special cases like when the keys given start with different characters (case sensitive).


Statistics for Good Hash Function
25
20
15
10
5
0
english_small.txt
english_large.txt
B = 1, TS = 20021
B = 1, TS = 45053
B = 1, TS = 77017
B = 20020, TS = 20021
B = 20020, TS = 45053
B = 20020, TS = 77017
B = 77019, TS = 20021
B = 77019, TS = 45053
B = 77019, TS = 77017
words
conflict_count
probe_total
probe_max


Statistic for Bad Hash Function
70
60
50
40
30
20
10
0
english_small.txt
english_large.txt
B = 1, TS = 20021
B = 1, TS = 45053
B = 1, TS = 77017
B = 20020, TS = 20021
B = 20020, TS = 45053
B = 20020, TS = 77017
B = 77019, TS = 20021
B = 77019, TS = 45053
B = 77019, TS = 77019
words
conflict_count
probe_total
probe_max

Fake data sets and their output:

Data set 1

```
l = LinearProbePotionTable(10, True, 10)
l["Potion of Health Regeneration"] = "p1"
l["Potion of Extreme Speed"] = "p2"
l["Potion of Deadly Poison"] = "p3"
l["Potion of Instant Health"] = "p4"
print(l.statistics())
```

Returns : (1,1,1)

```
l = LinearProbePotionTable(10, False, 10)
l["Potion of Health Regeneration"] = "p1"
l["Potion of Extreme Speed"] = "p2"
l["Potion of Deadly Poison"] = "p3"
l["Potion of Instant Health"] = "p4"
print(l.statistics())
```

Returns : (3,6,3)

Data set 2

```
l = LinearProbePotionTable(10, True, 10)
l["I need help"] = "p1"
l["you also need help"] = "p2"
l["We need help , so are you"] = "p3"
l["They need our help , we won't"] = "p4"
print(l.statistics())
```

Return(1,2,2)

```
l = LinearProbePotionTable(10, False, 10)
l["I need help"] = "p1"
l["you also need help"] = "p2"
l["We need help , so are you"] = "p3"
l["They need our help , we won't"] = "p4"
print(l.statistics())
```

Return (0,0,0)

Data set 3

```
l = LinearProbePotionTable(10, True, 10)
l["Nu gundam is the best"] = "p1"
l["Hi Nu is above nu gundam "] = "p2"
l["Sazabi is char's mobile suit"] = "p3"
l["Nitingale is the next levelk of Sazabi"] = "p4"
print(l.statistics())
```

Return (0,0,0)

```
l = LinearProbePotionTable(10, False, 10)
l["Nu gundam is the best"] = "p1"
l["Hi Nu is above nu gundam "] = "p2"
l["Sazabi is char's mobile suit"] = "p3"
l["Nitingale is the next levelk of Sazabi"] = "p4"
print(l.statistics())
```

Return (1,1,1)

Data set 4

```
l = LinearProbePotionTable(10, True, 10)
l["Teemo"] = "p1"
l["Elise"] = "p2"
l["Ezreal"] = "p3"
l["Talon"] = "p4"
print(l.statistics())
```

Return (1,1,1)

```
l = LinearProbePotionTable(10, False, 10)
l["Teemo"] = "p1"
l["Elise"] = "p2"
l["Ezreal"] = "p3"
l["Talon"] = "p4"
print(l.statistics())
```

Return(2,2,1)