Hash Table Analysis

T01G02

For the analysis of our hash table, we will use the random library to help us with generating the data and time library to measure the time needed for the hash function.

The random name will be generated from a combination of:

```
character_list = "abcdefghijklmnopqrstuvwxyz1234567890"
```

We will set the seed to 0 so the output is replicable:

```
random.seed(0)
```

We will create a function to generate random generated string from the previous character list:

```
def generate_random(string_len: int):
    random_gen = ''.join(random.choice(character_list) for i in
range(string_len))
    return random_gen
```

Next, we will create a function to generate the statistics itself:

```
def generate_statistics(iteration, max_item, random_string_len, good_hash=True):
    time_start = time.time()
    all = []

    for i in range(iteration):
        lppt = LinearProbePotionTable(max_item, good_hash)
        for i in range(max_item):
            random_gen = generate_random(random_string_len)
            lppt[random_gen] = random_gen
        all.append(lppt.statistics())

    average_conflict_count = 0
    average_probe_total = 0
    average_probe_max = 0

    for i in range(len(all)):
        average_conflict_count+= all[i][0]
        average_probe_total += all[i][1]
        average_probe_max += all[i][2]

    average_conflict_count /= len(all)
```

```
    average_probe_total /= len(all)
    average_probe_max /= len(all)

    time_end = time.time()
    time_needed = time_end - time_start

    return (average_conflict_count, average_probe_total, average_probe_max,
time_needed)
```

The function will generate average conflict count, probe total, probe max, and also time needed for the function to execute.

From testing, the factor that affects the statistics the most is how many items there are.

At a small enough item amount, the difference between good hash function and bad hash function is negligible. However, as the number of items get larger, the difference is very clear.

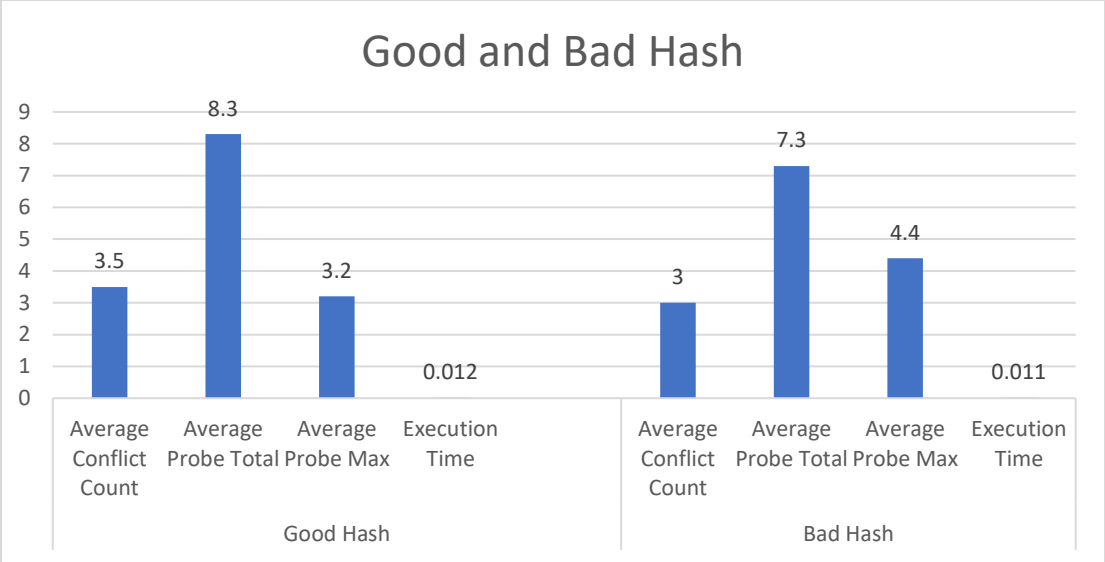
Every time we need to probe, it is counted as one conflict count.

Probe total is how many spots we need to probe when in total when inputting the items.

Probe max is the maximum distance of probe during all item insertion.

Statistics:

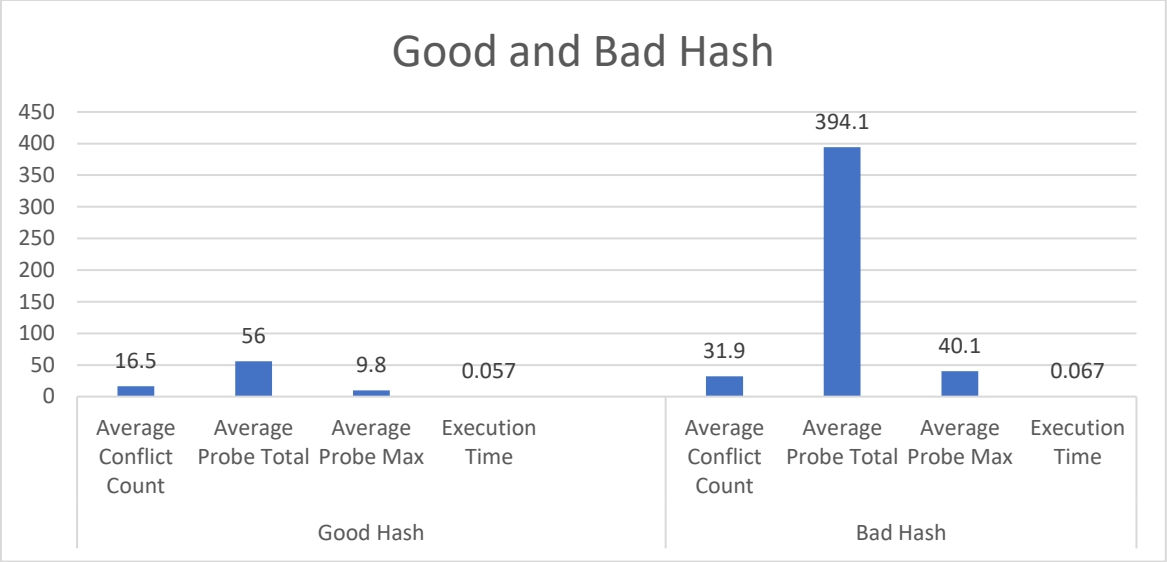
- Iteration = 10, Item = 10, String Length = 10:
  - Good Hash Function:
    - 3.5 Average conflict count
    - 8.3 Average probe total
    - 3.2 Average probe max
    - Executed in about 0.012 second
  - Bad Hash Function:
    - 3.0 Average conflict count
    - 7.3 Average probe total
    - 4.4 Average probe max
    - Executed in about 0.011 second



With a small item amount, the difference is not significant at all, and the bad hash is even better at conflict count and probe total.

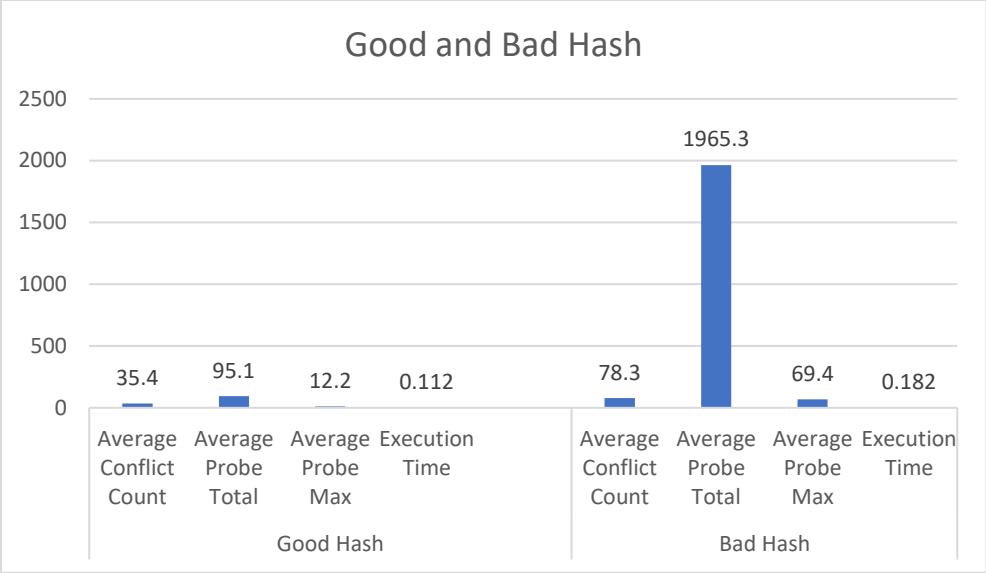
- Iteration = 10, Item = 50, String Length = 10:
  - Good Hash Function:
    - 16.5 Average conflict count
    - 56.0 Average probe total
    - 9.8 Average probe max
    - Executed in about 0.057 second
  - Bad Hash Function:
    - 31.9 Average conflict count
    - 394.1 Average probe total
    - 40.1 Average probe max
    - Executed in about 0.067 second

  By increasing the amount of item that we hash from 10 to 50, the difference can already be seen. Bad hash has nearly double the conflict count, a significantly larger probe total, and the maximum probe distance is also larger.



- Iteration = 10, Item = 100, String Length = 10:
  - Good Hash Function:
    - 35.4 Average conflict count
    - 95.1 Average probe total
    - 12.2 Average probe max
    - Executed in about 0.112 second
  - Bad Hash Function:
    - 78.3 Average conflict count
    - 1,965.3 Average probe total
    - 69.4 Average probe max
    - Executed in about 0.182 second

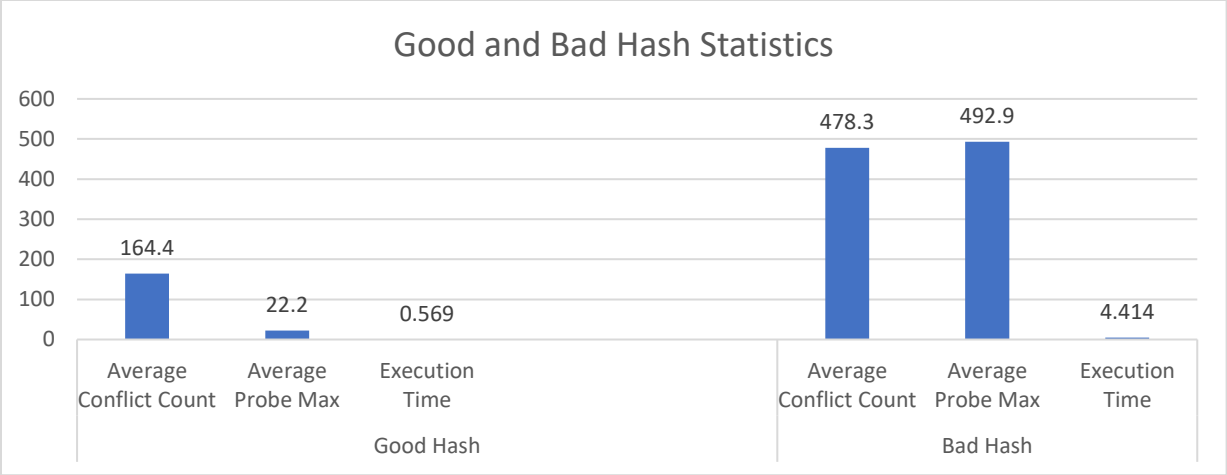
  The difference gets even larger, the most interesting part is probe total. We can see that in the good hash function, we only needed to probe 95 spots. While for the bad hash, we needed to probe nearly two thousand spots. A very large difference.



The Probe Total for the bad hash is now very big, making the others look very small.

- Iteration = 10, Item = 500, String Length = 10:
  - Good Hash Function:
    - 164.4 Average conflict count
    - 472.9 Average probe total
    - 22.2 Average probe max
    - Executed in about 0.569 second
  - Bad Hash Function:
    - 478.3 Average conflict count
    - 101,569.8 Average probe total
    - 492.9 Average probe max
    - Executed in about 4.414 second

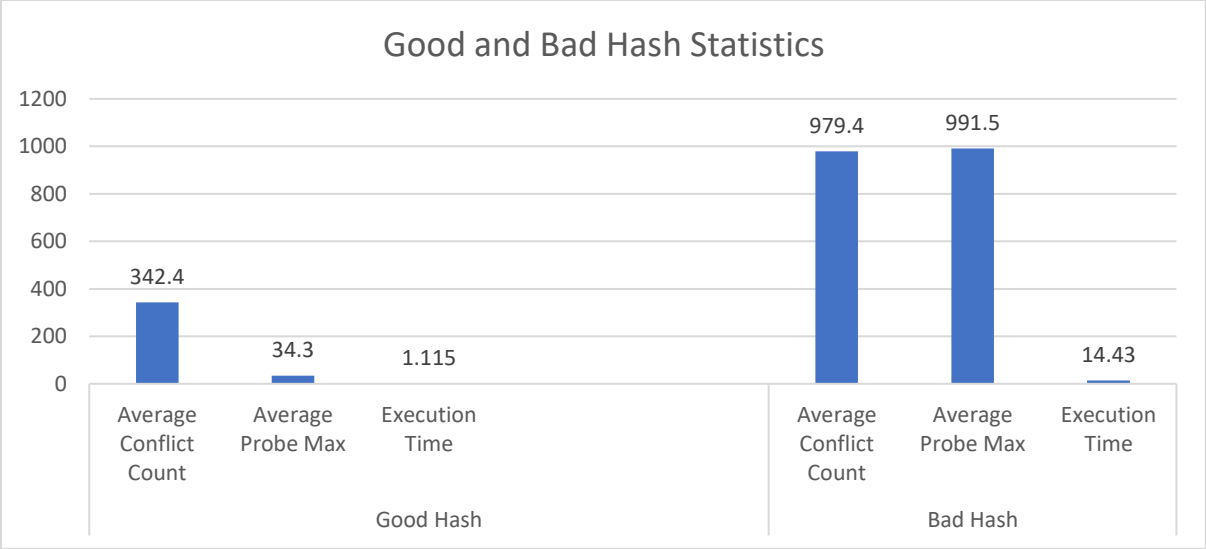
With 500 items, we can really feel the difference as the running time for bad hash is significantly larger compared to good hash function. The bad hash has nearly triple the conflict count of good hash. As for the total amount of probing, good hash only needed to probe around 473 spots, while for the bad hash, it probed more than 100 thousand spots. The maximum distance probed for good hash is also only around 22 spots, while the bad hash function maximum probe distance is nearly 493 spots.



From this bar chart we omit the Average Probe Total, because it is too large, making the other data unclear.

- Iteration = 10, Item = 1000, String Length = 10
  - Good Hash Function:
    - 342.4 Average conflict count
    - 1,074.4 Average probe total
    - 34.3 Average probe max
    - Executed in about 1.115 second
  - Bad Hash Function:
    - 979.4 Average conflict count
    - 453,322.2 Average probe total
    - 991.5 Average probe max
    - Executed in about 14.430 second

  With 1000 items, the running time of the bad hash function is around 14 times than the good hash function. In total, good hash only probed around 1074 spots, while the bad hash needed to probe around 453 thousand spots. The difference is obvious and now it is clear that the good hash function is better.



- Iteration = 10, Item = 10000, String Length = 10
  - Good Hash Function:
    - 3,323.0 Average conflict count
    - 9,975.7 Average probe total
    - 47.5 Average probe max
    - Executed in about 7.831 second
  - Bad Hash Function:
    - 9,977.2 Average conflict count
    - 49,538,215.1 Average probe total
    - 9,991.6 Average probe max
    - Executed in about 1287.213 second, or around 21 minutes.

With 10000 items, the average probe total of bad hash becomes over 49.5 million probes. The maximum probe distance is nearly 10000, and it needed more than 21 minutes to compute.