Lab 10 Hashing **Tanadon Santisan 6538068821**

Plotting Collision Frequency vs. Load Factor in Open Address Hashing

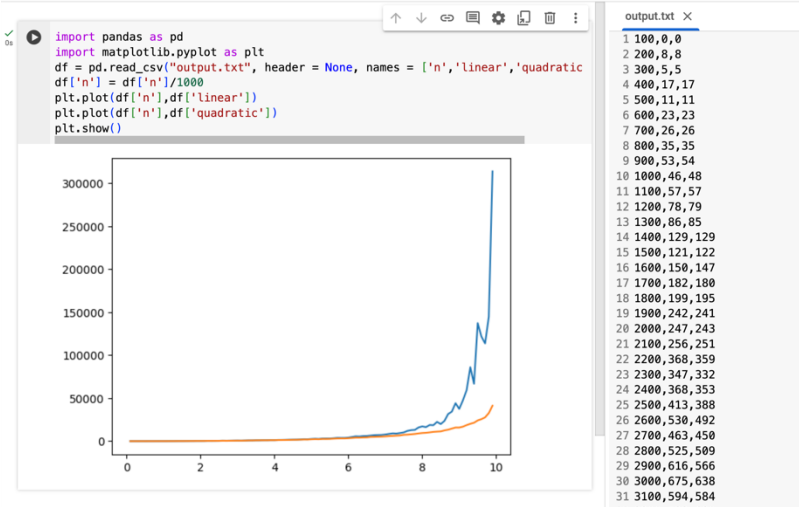
Objective: In this lab, you will explore the relationship between collision frequency and load factor in open address hashing using linear and quadratic probing methods.

Instructions:

1. **Understanding Open Address Hashing:**
    - Review the concept of open address hashing, including linear and quadratic probing methods.
    - Understand how collisions occur and how they are resolved in each probing method.
2. **Implementing Hash Table:**
    - Use the code provided in the lecture class for implementing a hash table data structure with open address hashing.
    - Ensure your implementation includes functions for inserting data into the hash table and calculating load factor.
3. **Generating Data:**
    - Vary the size of the dataset from 100 to 9900 random (increased by 100) integers to insert into a hash table.
    - The hash table is fixed by the size of buckets to 10000 and record the number of collisions that occur during insertion.
4. **Calculating Load Factor:**
    - Calculate the load factor for each number of data. Load factor is defined as the ratio of the number of elements stored in the hash table to the size of the hash table.
5. **Plotting Collision Frequency vs. Load Factor:**
    - Plot a line chart with load factor on the x-axis and collision frequency on the y-axis.
    - Plot separate lines for linear probing and quadratic probing methods.
6. **Analysis:**
    - Analyze the plotted data and observe the relationship between load factor and collision frequency for both linear and quadratic probing methods.
    - Discuss any differences or similarities observed between the two probing methods.

Submission: Submit your chart and analysis in a pdf file.

Tanadon Santisan 6538068821



```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("output.txt", header = None, names = ['n','linear','quadratic
df['n'] = df['n']/1000
plt.plot(df['n'],df['linear'])
plt.plot(df['n'],df['quadratic'])
plt.show()
```


output.txt

```
100,0,0
200,8,8
300,5,5
400,17,17
500,11,11
600,23,23
700,26,26
800,35,35
900,53,54
1000,46,48
1100,57,57
1200,78,79
1300,86,85
1400,129,129
1500,121,122
1600,150,147
1700,182,180
1800,199,195
1900,242,241
2000,247,243
2100,256,251
2200,368,359
2300,347,332
2400,368,353
2500,413,388
2600,530,492
2700,463,450
2800,525,509
2900,616,566
3000,675,638
3100,594,584
3200,723,680
3300,856,825
3400,869,837
3500,1014,957
3600,1023,1011
3700,1137,1044
3800,1140,1044
3900,1233,1147
4000,1197,1146
4100,1420,1289
```

Linear probing in hashing checks all slots one by one until it finds an empty one, which can lead to clusters of filled slots. Quadratic probing, on the other hand, spreads out the search more evenly, reducing the chance of such clusters. So, linear probing tends to have more collisions because it's more likely to create these clusters of filled slots.