COMP9313 Project 1
Yu Zhang
(z5238743)

Q1. Implementation details of your c2lsh(). Explain how your major transform function works.

I write two function in the code---submission.py.

The first function named match_function has 4 parameters, data_hashes, query_hashes, alpha_m and offset. Every Data_hash compares with query_hashe based on offset number. The difference between every data_hashes and query_hashes use two offset numbers, negative offset and positive function, to compare the number. If the difference satisfy the condition, it will accumulate by count. If count bigger than alpha_m, the result will return True. On the contrary, it will return False.

The Second function is c2lsh(), I give two local variable offset and length, both of them applied -1. I have a while loop to use length compare with beta_n, I use flatMap to return a new RDD named candidate, which by first applying a lambda function to use the first match_function to all elements of the data_hashes, and then flattening the result. Besides, I use count function to return the number of rows in this candiates.

Q2.Show the evaluation result of your implementation using your own test cases.

1.The Toy datasets:
In this dataset, there are 100 data_hashes.

i. a =10, b =10
running time: 1.6707708835601807
Number of candidate: 10
set of candidate: {0, 70, 40, 10, 80, 50, 20, 90, 60, 30}

ii. a = 20, b = 20
running time: 2.3055028915405273
Number of candidate: 100

iii. a =30, b =30
running time: 2.2662711143493652
Number of candidate: 100

2.Test Case 1 Experiment
In this dataset, there are 5000 data_hashes.

i. a =10, b =10
running time: 0.7595720291137695
Number of candidate: 1000

ii. a =20, b =20
running time: 0.5906140804290771
Number of candidate: 1000
iii. a =20, b =4000
running time: 3.0655229091644287
Number of candidate: 4044

iv. a =10, b =5000
running time: 3.0464818477630615
Number of candidate: 5000

2.Test Case 3 Experiment
In this dataset, there are 500000 data_hashes.

i. a=10, b=10
running time: 2.056648015975952
Number of candidate: 100000

ii. a=20, b =20
running time: 2.265681028366089
Number of candidate: 100000

iii. a = 20, b =4000
running time: 1.963404893875122
Number of candidate: 100000

iv. a = 20, b = 40000
running time: 2.3028311729431152
Number of candidate: 100000

Q3. What did you do to improve the efficiency of your implementation?

Different machine run different result.

1.FlatMap() function may not the best solution to speed up
It may have other efficient function to implement.

2. For evey while loop, it will continue large repetitive work to count difference, it can calculate difference before enter loop