2020

Project 1

COMP9313 | BIG DATA MANAGEMENT |20T2

ANANT KRISHNA MAHALE | Z5277610

This assignment requires student to implement the C2LSH. The psudocode of C2LSH w.r.t greedy algorithm is as follows:

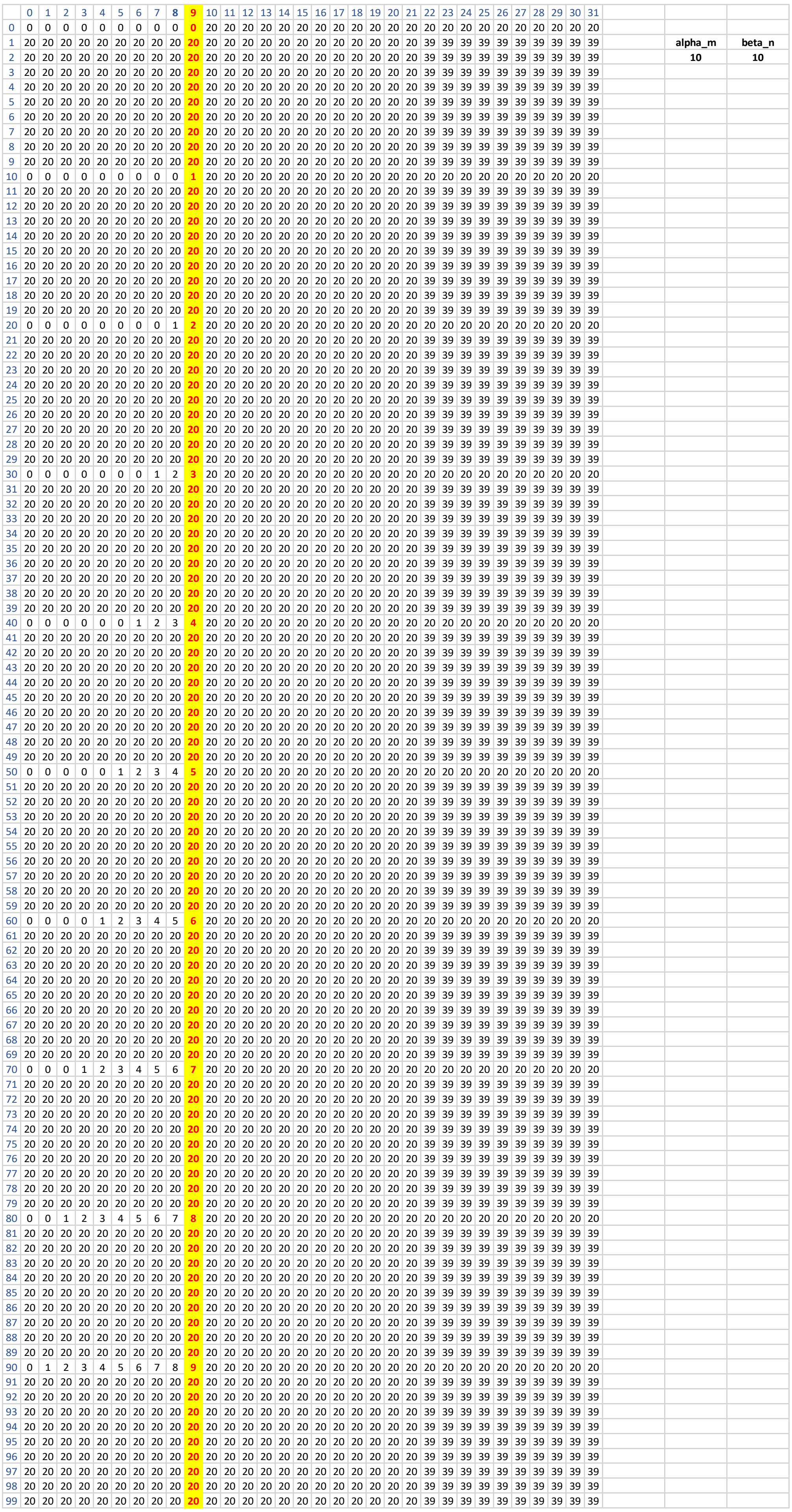
```
count(hashes_1, hashes_2, offset):
    counter ← 0
    for each $hash_1$, $hash_2$ in hashes_1, hashes_2:
        if $|hash_1 - hash_2| \leq$ offset:
            counter ← counter + 1
    return counter


candGen(data_hashes, query_hashes, $\alpha m$, $\beta n$):
    offset ← 0
    cand ← $\emptyset$
    while true:
        for each (id, hashes) in data_hashes:
            if count(hashes, query_hashes, offset) $\geq \alpha m$:
                cand ← cand ∪ {id}
        if $|cand| < \beta n$:
            offset ← offset + 1
        else:
            break
    return cand
```

After implementing above pseudocode, I realized that the count function will be called each time till we get the right offset and especially dataset that needs offset ~50000, is computationally expensive. It takes around ~15mins on 4 core 16gb machine. Hence, I wanted to find the relation between the offset, alpha and beta since same computations were being done again and again.

Initially, I calculated |data_hash – query_hash| and started computing on this RDD. Again, same computations were being carried out in loops. I realized if I can compute the right offset, algorithm takes very less time to find the candidates.

Changes made to the original/ greedy algorithm:
After some research, trial and error, I made RDD of differences of data_hashes and query_hashes and sorted the results. In the same RDD, if we take the values in alpha_mth column and sort them again, the beta_nth position is the offset to satisfy given alph_m and beta_n.

Below is the screenshot of given toy example with alpha_m and beta_n = 10. In the sorted differences RDD, the 9th row is highlighted and sorted again. The value of 9th element is the offset require.



	0	1	2	3	4	5	6	7	8	**9**	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
0	0	0	0	0	0	0	0	0	0	**0**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
1	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39		**alpha_m**	**beta_n**
2	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39		**10**	**10**
3	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
4	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
5	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
6	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
7	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
8	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
9	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
10	0	0	0	0	0	0	0	0	0	**1**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
11	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
12	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
13	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
14	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
15	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
16	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
17	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
18	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
19	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
20	0	0	0	0	0	0	0	0	1	**2**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
21	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
22	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
23	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
24	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
25	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
26	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
27	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
28	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
29	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
30	0	0	0	0	0	0	0	1	2	**3**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
31	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
32	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
33	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
34	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
35	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
36	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
37	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
38	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
39	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
40	0	0	0	0	0	0	1	2	3	**4**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
41	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
42	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
43	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
44	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
45	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
46	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
47	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
48	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
49	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
50	0	0	0	0	0	1	2	3	4	**5**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
51	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
52	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
53	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
54	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
55	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
56	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
57	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
58	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
59	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
60	0	0	0	0	1	2	3	4	5	**6**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
61	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
62	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
63	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
64	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
65	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
66	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
67	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
68	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
69	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
70	0	0	0	1	2	3	4	5	6	**7**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
71	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
72	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
73	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
74	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
75	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
76	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
77	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
78	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
79	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
80	0	0	1	2	3	4	5	6	7	**8**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
81	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
82	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
83	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
84	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
85	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
86	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
87	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
88	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
89	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
90	0	1	2	3	4	5	6	7	8	**9**	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20			
91	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
92	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
93	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
94	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
95	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
96	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
97	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
98	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			
99	20	20	20	20	20	20	20	20	20	**20**	20	20	20	20	20	20	20	20	20	20	20	20	39	39	39	39	39	39	39	39	39	39			

The above picture shows the instance of the **sorted** differences of **hash_data** and **query_data**. I have highlighted data in the alpha_m position (alpha_m =10-1).

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**9**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**

beta_nth position in alpha_mth sorted column

We can see that for the toy example, the required offset was 9.

How this approach is better than greedy algorithm?

In greedy algorithm, the entire data is iterated 2 times for each offset calculation till right offset for is calculated for a given data and other parameters i.e. O(m*n*p) where m is no. of keys, n is length of values list and p is the number of times the program was executed to till conditions were satisfied.

In my case, I traverse the dataset thrice. 1st time to calculate the difference between the data and query hashes, 2nd to get the elements in alpha_mth row and once after calculating the offset to get the ids which satisfy the conditions. The time complexity of the algorithm is O(m*n). Since 3 is a constant irrespective of offset, it can be neglected.

Implementation details of c2lsh().

```
def c2lsh(data_hashes, query_hashes, alpha_m, beta_n):
    processed_data = data_hashes.map(lambda data: [data[0], find_absdiff(data[1], query_hashes)])
    offset = calculate_offset(processed_data, alpha_m-1, beta_n-1)
    candidates = processed_data.map(lambda data: count_function(data[0], data[1],offset,alpha_m)).filter(lambda value: value != None)
    return candidates
```

So, as explained above, in `c2lsh` function, a temporary RDD(`processed_data`) is created which stores all the differences between data_hash and query_hashes respectively. To do this, map function is used which maps given data_hases as key and value (list of size m) is used.
Using lambda function, values along with query_hashes are sent to the function to perform the calculation.

Then this RDD, is passed to the `calculate_offset` function along with alpha_m and beta_n values to get the right offset.

Once the offset is calculated, the temporary RDD is iterated again, to check which of the ids satisfy alpha_m condition.

Finally, all the Ids that satisfy, the condition are returned.

Finding the right offset for a given data, alpha_m and beta_n:
Below is the function that calculates the offset.

```
def calculate_offset(transformed_data_hashes, alpha_m, beta_n):
    transformed_data_hashes = transformed_data_hashes.map(lambda data: get_element_from_pos(data[1], alpha_m)).sortBy(lambda value: value).zipWithIndex()
    transformed_data_hashes = transformed_data_hashes.filter(lambda data: data[1] == beta_n)
    offset = transformed_data_hashes.map(lambda data: data[0]).min()
    return offset
```

Here, the received RDD is again mapd, using map function to retrieve values in alpha_mth column and then sorted using sortBy function which in turn zipped with index to get the beta_nth value.

Now there is only one column of sorted values which were in alpha_m column with index. Using filter function, index corresponding to beta_n value can be found.

Finally, map (). min () function can be used to retrieve the value as an integer. Explicitly there is no need to use min since there is only one value.

Testing and results:

Below function was used to generate a random data.

```
def generate(dimension, count, seed, start=-1000, end=1000):
    random.seed(seed)
    data = [
        [
            random.randint(start, end)
            for _ in range(dimension)
        ]
        for i in range(count)
    ]
    query = [random.randint(start, end) for _ in range(dimension)]
    return data, query
```

Reference: this function is taken from the course forum piazza. (https://piazza.com/class/kamb1tqxe9h6np?cid=14).

Below random data was generated using the above function.

```
Data_Hashes:
[[6, 6, 0, 4, 8, 7, 6, 4, 7, 5],
 [9, 3, 8, 2, 4, 2, 1, 9, 4, 8],
 [9, 2, 4, 1, 1, 10, 5, 7, 8, 1],
 [5, 6, 5, 9, 10, 3, 8, 7, 7, 8],
 [4, 0, 8, 0, 1, 6, 10, 10, 0, 9],
 [7, 5, 3, 5, 1, 3, 9, 3, 3, 2],
 [8, 7, 1, 1, 5, 8, 7, 1, 4, 8],
 [4, 1, 8, 5, 8, 3, 9, 8, 9, 4],
 [7, 1, 9, 6, 5, 9, 3, 4, 2, 3],
 [2, 0, 9, 10, 4, 7, 1, 1, 10, 2]]

Query Hashes:
[2, 0, 1, 8, 10, 6, 8, 4, 8, 3]
```

With above data with alpha_m = 10, and beta_n = 10, Using the regular greedy / algorithm given in the lecture, I found the offset of the above dataset is 9 and wanted to confirm if the above described method returns offset 9.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	1	1	1	2	2	2	4	4	6
1	3	4	4	5	5	6	6	7	7	7
2	0	2	2	3	3	3	4	7	7	9
3	0	0	1	1	3	3	3	4	5	6
4	0	0	2	2	6	6	7	8	8	9
5	1	1	1	2	3	3	5	5	5	9
6	0	1	2	3	4	5	5	6	7	7

7	1	1	1	1	2	2	3	3	4	7
8	0	0	1	2	3	5	5	5	6	8
9	0	0	1	1	2	2	3	6	7	8

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
6	6	7	7	7	8	8	9	9	9

The offset calculated using above mentioned method is 9.

How about the larger data set ?
I created a dataset with 100 keys and list with 70000 values. And it took 3.9 seconds to return the values.

Additional Information:
Here are the other testcases taken from Piazza forum: (link: https://piazza.com/class/kamb1tqxe9h6np?cid=149)

```
# Test 1
alpha_m, beta_n = 10, 10
data, query = generate(10, 20000, 0, 0, 1000)
running time: 1.3994967937469482
Number of candidate:  10
set of candidate:  {9536, 5825, 3270, 15177, 15817, 9261, 4124, 10478, 10545, 1628}

# Test 2
alpha_m, beta_n = 10, 50
data, query = generate( 13, 200, 100, -50000, 50000)
running time: 1.2963991165161133
Number of candidate:  50
set of candidate:  {133, 6, 7, 134, 143, 144, 17, 22, 150, 24, 153, 154, 28, 158, 159, 32, 162, 35, 36, 38, 41, 44, 172, 46, 174, 50, 53, 54, 55, 184, 57, 59, 187, 61, 63, 192, 195, 68, 69, 72, 74, 87, 88, 93, 104, 108, 110, 119, 122, 127}


# Test 3
alpha_m, beta_n = 10, 64
data, query = generate2( 13, 9, 100, 0, 120)
running time: 1.2118151187896729
Number of candidate:  81
set of candidate:  {896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895}

# Test 4
alpha_m, beta_n = 13, 25
data, query = generate3( 13, 7, 100, 0, 120)
running time: 1.1820099353790283
Number of candidate:  35
set of candidate:  {644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678}

#Test 5
alpha_m, beta_n = 10, 50
data, query = generate( 90, 1_000, 120, -50_000, 50_000)
running time: 1.2760601043701172
Number of candidate:  50
set of candidate:  {1, 129, 131, 265, 650, 779, 905, 531, 660, 277, 915, 289, 548, 933, 550, 169, 299, 429, 174, 439, 56, 59, 832, 960, 962, 195, 71, 711, 967, 590, 463, 83, 600, 728, 602, 92, 609, 996, 230, 614, 871, 746, 499, 500, 373, 501, 247, 756, 758, 639}


# Test 6
# Credit to Anonymous Gear @149_f6
# https://piazza.com/class/kamb1tqxe9h6np?cid=149_f6
alpha_m, beta_n = 10, 500
data, query = generate( 13, 2_000, 100, -230_000, 50_000)
running time: 1.228196144104004
Number of candidate:  500
```

```
set of candidate:  {1, 3, 5, 22, 27, 29, 32, 38, 40, 45, 47, 56, 59, 60, 62, 68, 69, 73, 74, 82, 90,
95, 97, 101, 102, 108, 118, 124, 126, 131, 132, 134, 137, 138, 146, 148, 149, 151, 153, 160, 162, 164,
176, 183, 185, 186, 192, 194, 203, 205, 207, 212, 213, 214, 216, 219, 220, 221, 226, 236, 248, 257, 258,
262, 266, 269, 274, 276, 278, 280, 281, 283, 293, 298, 304, 305, 307, 310, 316, 317, 322, 325, 327, 328,
330, 333, 337, 347, 348, 351, 352, 353, 357, 358, 361, 363, 374, 375, 378, 385, 391, 402, 406, 408, 414,
415, 416, 420, 421, 423, 424, 428, 430, 431, 432, 436, 441, 447, 450, 455, 458, 461, 464, 466, 473, 474,
477, 479, 480, 491, 497, 501, 512, 531, 532, 544, 546, 552, 554, 556, 558, 559, 563, 564, 566, 567, 569,
581, 585, 587, 590, 591, 600, 611, 615, 618, 623, 628, 629, 634, 635, 644, 647, 652, 657, 663, 668, 673,
677, 681, 683, 697, 698, 701, 702, 705, 707, 711, 714, 717, 728, 729, 744, 748, 752, 753, 756, 761, 770,
772, 774, 777, 781, 785, 788, 789, 790, 792, 797, 799, 802, 804, 806, 807, 808, 811, 822, 826, 829, 830,
836, 837, 839, 840, 841, 845, 850, 851, 856, 857, 861, 862, 869, 870, 872, 873, 890, 891, 892, 895, 902,
922, 931, 934, 940, 942, 943, 949, 951, 956, 957, 959, 960, 962, 968, 969, 970, 971, 973, 974, 976, 977,
979, 982, 988, 992, 996, 1004, 1011, 1022, 1023, 1025, 1026, 1039, 1041, 1043, 1047, 1053, 1059, 1060,
1061, 1072, 1083, 1091, 1096, 1097, 1098, 1099, 1106, 1111, 1115, 1122, 1123, 1124, 1134, 1142, 1143,
1144, 1147, 1148, 1155, 1159, 1167, 1174, 1182, 1185, 1191, 1192, 1197, 1198, 1204, 1206, 1207, 1210,
1212, 1222, 1224, 1227, 1238, 1243, 1244, 1247, 1248, 1250, 1251, 1255, 1262, 1263, 1267, 1272, 1275,
1278, 1283, 1289, 1290, 1296, 1300, 1301, 1304, 1310, 1317, 1318, 1325, 1326, 1329, 1336, 1339, 1341,
1344, 1345, 1346, 1348, 1351, 1353, 1358, 1362, 1364, 1367, 1368, 1370, 1371, 1377, 1387, 1397, 1399,
1400, 1406, 1407, 1410, 1422, 1426, 1442, 1444, 1448, 1451, 1453, 1456, 1461, 1464, 1474, 1482, 1488,
1499, 1500, 1513, 1517, 1522, 1523, 1524, 1525, 1526, 1530, 1538, 1548, 1557, 1559, 1561, 1562, 1563,
1564, 1565, 1566, 1567, 1573, 1574, 1575, 1582, 1589, 1590, 1591, 1593, 1596, 1597, 1601, 1604, 1605,
1611, 1619, 1626, 1627, 1628, 1631, 1632, 1634, 1642, 1644, 1647, 1655, 1656, 1660, 1662, 1664, 1673,
1676, 1683, 1689, 1693, 1696, 1697, 1698, 1705, 1708, 1712, 1726, 1728, 1730, 1735, 1745, 1750, 1753,
1768, 1772, 1775, 1782, 1792, 1796, 1799, 1801, 1803, 1807, 1808, 1818, 1820, 1821, 1827, 1842, 1843,
1844, 1849, 1850, 1858, 1863, 1864, 1867, 1868, 1871, 1872, 1875, 1878, 1880, 1886, 1892, 1894, 1908,
1909, 1920, 1925, 1926, 1940, 1941, 1943, 1945, 1948, 1950, 1951, 1952, 1953, 1956, 1958, 1959, 1963,
1968, 1971, 1972, 1974, 1981, 1983, 1985, 1990, 1998}


# Test 7
# Credit to Anonymous Gear @149_f6
# https://piazza.com/class/kamb1tqxe9h6np?cid=149_f6
alpha_m, beta_n = 10, 500
data, query = generate( 15, 70_000, 140, -500_000, 500_000)
running time: 1.7152910232543945
Number of candidate:  500
set of candidate:  {63494, 26634, 55308, 43024, 53264, 57360, 18461, 43039, 32801, 34850, 38948, 53294,
18489, 67652, 38984, 43082, 8270, 12367, 63568, 55378, 36947, 67670, 55394, 10344, 28779, 14444, 116,
63604, 20600, 32890, 34939, 59514, 36995, 34949, 134, 69769, 34954, 47242, 69772, 41102, 69785, 2208,
41121, 53410, 32933, 2222, 65712, 43186, 10425, 30906, 65721, 10432, 65733, 32966, 16598, 67798, 57560,
41177, 43230, 41187, 20709, 18664, 22762, 57581, 2289, 63739, 22784, 8456, 10507, 20754, 43289, 47385,
20766, 51487, 16672, 55584, 22831, 31026, 39230, 39231, 53568, 18759, 53576, 59721, 49483, 332, 63821,
335, 53593, 6491, 14683, 20829, 39264, 22882, 2406, 67947, 69998, 372, 55674, 65923, 59781, 63877, 4489,
29065, 2452, 53658, 57757, 37278, 49573, 51625, 68010, 43438, 2486, 8631, 53691, 47548, 45506, 68034,
61894, 47559, 41416, 35289, 14815, 66016, 10725, 486, 57836, 12785, 25073, 27131, 55811, 27141, 41484,
45586, 64018, 31256, 68120, 31261, 37411, 6693, 21035, 14893, 6702, 21040, 563, 566, 27190, 31304,
53835, 49742, 66128, 37459, 43609, 62046, 27233, 53857, 16996, 37483, 6764, 4745, 64139, 25228, 31372,
62097, 49810, 58004, 25237, 2716, 41632, 47781, 23207, 55976, 39596, 51885, 19122, 62132, 10944, 43720,
10953, 8906, 717, 25310, 8933, 49894, 45799, 2798, 19185, 39669, 35575, 66300, 33540, 49926, 27402,
2840, 793, 11035, 47899, 41761, 39716, 35657, 37711, 56152, 21338, 47968, 19302, 21353, 64361, 25453,
52082, 47988, 54149, 4998, 39816, 17289, 17293, 66455, 56218, 5020, 39850, 939, 11182, 13231, 58293,
39869, 11212, 64465, 43989, 3032, 21466, 58330, 58332, 15326, 31722, 13295, 17393, 50162, 41971, 25590,
64506, 11271, 33799, 56327, 60435, 29717, 39957, 50203, 3105, 21547, 37933, 58424, 11323, 31804, 3134,
50245, 44103, 64583, 27724, 13391, 29790, 58463, 64609, 3180, 11380, 19583, 23693, 15502, 64661, 27805,
31904, 60582, 60586, 7345, 13503, 50377, 40141, 27859, 52437, 68825, 54491, 60635, 34013, 3295, 38111,
48356, 54501, 21743, 40183, 11514, 29950, 3333, 64778, 25868, 36108, 5394, 66836, 3352, 11545, 62747,
52512, 5413, 27944, 44335, 54575, 5426, 56632, 1338, 11580, 15684, 50503, 48456, 1363, 21844, 15707,
40290, 66918, 40296, 64873, 48491, 34165, 5499, 50562, 34179, 3463, 9608, 32135, 1420, 25997, 28044,
30093, 58773, 1433, 13724, 52639, 46498, 60841, 17835, 26033, 56753, 11699, 34230, 23996, 7614, 17855,
17859, 7626, 36300, 26061, 52692, 26074, 56800, 69088, 52707, 48614, 5609, 48622, 38382, 38385, 1529,
1532, 28160, 46593, 11780, 54788, 26118, 54797, 22032, 17941, 13853, 40477, 65059, 60965, 5671, 52776,
30260, 56886, 13885, 46653, 58941, 38465, 42562, 69188, 50761, 3660, 48716, 30286, 17999, 11857, 40534,
18012, 52839, 22130, 18035, 9850, 28284, 40572, 48766, 16001, 63112, 28301, 1682, 20116, 59042, 18089,
34474, 1708, 22195, 38579, 1722, 69309, 48832, 52932, 50890, 24270, 16080, 50900, 52949, 38615, 20185,
67295, 9956, 34532, 26342, 63209, 67310, 44788, 32501, 59128, 65273, 12026, 24322, 7939, 48899, 32521,
20238, 67346, 16153, 16159, 59175, 63273, 69417, 18230, 26424, 59199, 69439, 24388, 20294, 3911, 24391,
34636, 61261, 67405, 8017, 5971, 28501, 40793, 14181, 67429, 22377, 69484, 26477, 12142, 57199, 61300,
32629, 16246, 51062, 22393, 69499, 10114, 38788, 46982, 55174, 65414, 20366, 3983, 14243, 63405, 38837,
42934, 26559, 18368, 22468, 10182, 63439, 69586, 30676, 32724, 40920, 53210, 24539, 59358, 65508, 51177,
2026, 22508, 47087, 38897, 12275, 59385, 51199}
```

Note:
I could not install specified software on CSE Machines. However, I have installed all the software required for this course on my local machine. These tests were performed on my person computer (processor: 2.4 GHz Quad-Core Intel Core i5 with 16GB RAM).