

COMP9313 Big Data Management

Project 1 Report(Term2 2020)

Student Name: Zhenming Wang Student ID: z5140192

Part1: Understanding of Virtual Rehashing

This part of the knowledge is the basis for completing the project, so first show the bottom-level knowledge principle

Set the Collision number = 8

- At first consider $h(o) = h(q)$

q	1	1	1	1	1	1	1	1	1	1
o_1	1	0	2	-1	1	2	4	-3	0	-1

Only two number can match, the Collision number=2<8, So it need to relaxing conditions.

- Consider $h(o) = h(q) \pm 1$ if not enough candidates

q	1	1	1	1	1	1	1	1	1	1
o_1	1	0	2	-1	1	2	4	-3	0	-1

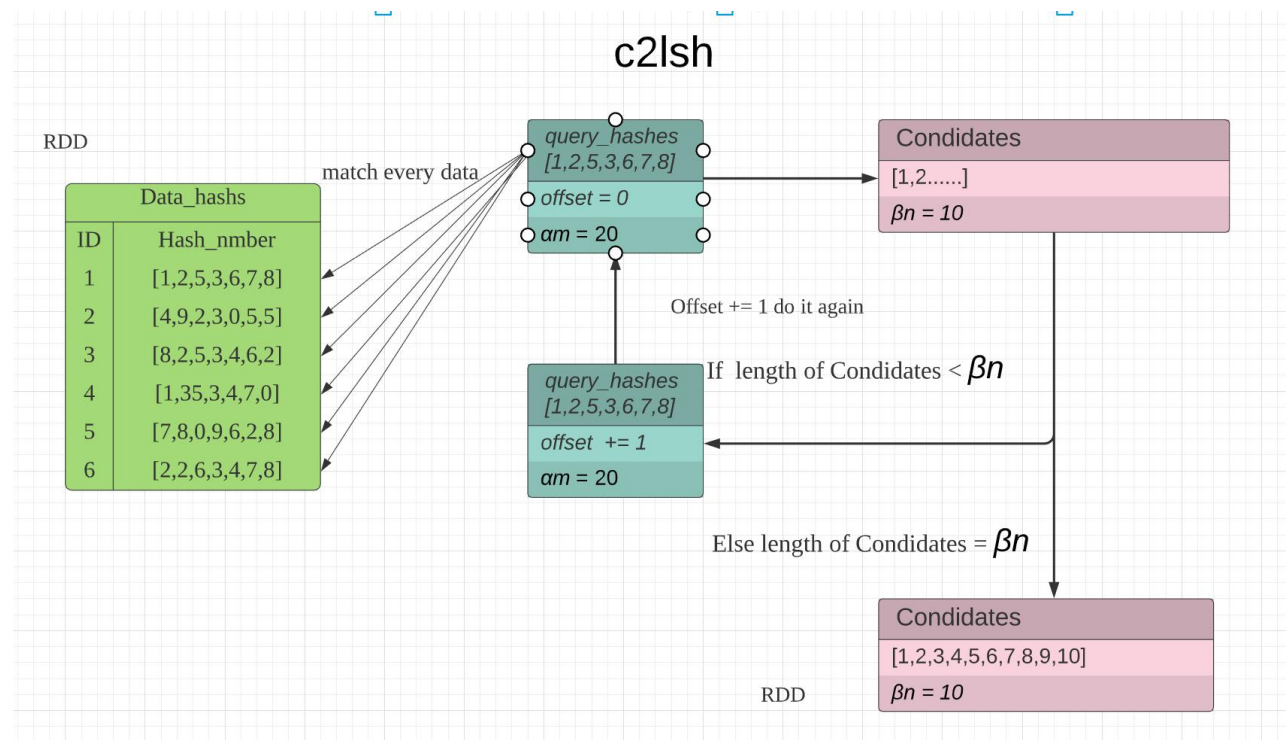
The Collision number=6<8, So it also need to relaxing conditions.

Then $h(o) = h(q) \pm 2$ and so on...

q	1	1	1	1	1	1	1	1	1	1
o_1	1	0	2	-1	1	2	4	-3	0	-1





Finally, Collision number=8 and Complete match,the task completed

Part 2: Implementation details of c2lsh()



Part 3: Show the evaluation result of my implementation using my own test cases.

I randomly created three test sets, the simple1 size is 2000000, the simple2 size is 400,000, and the simple3 size is 1000000.

 simple1.pkl	7/6/2020 6:05 AM	PKL File
 simple2.pkl	7/6/2020 6:19 AM	PKL File
 simple3.pkl	7/6/2020 6:23 AM	PKL File
 simpleQueryHash.pkl	7/6/2020 6:18 AM	PKL File

Simple1

```
running time: 2.3457112312316895
Number of candidate: 200000
set of candidate: {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, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 5
0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102,
103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 1
24, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 14
5, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 16
6, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 18
7, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 20
8, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 22
9, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 25
0, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 27
1, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 29
2, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 31
3, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 33
4, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 35
5, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 37
```

Simple2

```
running time: 12.928142309188843
Number of candidate: 100000
set of candidate: {0, 1, 7, 9, 10, 13, 14, 17, 18, 21, 23, 27, 30, 32, 33, 34, 36, 38, 43, 44, 47, 48, 4
9, 50, 51, 54, 58, 59, 61, 63, 64, 66, 67, 68, 69, 72, 73, 74, 77, 78, 80, 82, 83, 86, 88, 90, 91, 96, 98,
99, 101, 107, 108, 109, 112, 113, 117, 118, 119, 120, 121, 123, 127, 129, 130, 131, 132, 133, 137, 140, 14
1, 143, 144, 146, 147, 150, 151, 154, 155, 156, 157, 159, 161, 162, 165, 167, 169, 170, 171, 177, 178, 18
1, 183, 185, 187, 188, 189, 193, 194, 195, 196, 198, 200, 203, 205, 206, 211, 213, 214, 216, 217, 218, 22
0, 226, 227, 228, 229, 230, 231, 232, 233, 234, 237, 238, 239, 240, 242, 243, 245, 246, 247, 251, 253, 25
7, 259, 263, 264, 265, 266, 267, 270, 271, 272, 273, 275, 277, 278, 279, 282, 283, 284, 285, 287, 289, 29
2, 293, 294, 295, 296, 297, 298, 299, 301, 304, 308, 311, 312, 315, 319, 320, 322, 325, 326, 333, 334, 33
5, 336, 337, 340, 341, 342, 343, 344, 346, 349, 351, 352, 355, 358, 359, 360, 361, 362, 364, 365, 366, 37
0, 372, 373, 376, 377, 378, 380, 381, 382, 383, 384, 385, 386, 387, 388, 390, 391, 392, 393, 397, 398, 39
9, 401, 408, 412, 414, 415, 419, 422, 433, 436, 437, 445, 448, 449, 450, 452, 453, 454, 457, 458, 461, 46
2, 463, 464, 466, 467, 468, 469, 471, 473, 475, 476, 478, 481, 483, 484, 487, 488, 491, 497, 498, 503, 50
4, 509, 510, 511, 512, 517, 521, 522, 523, 525, 527, 528, 530, 532, 533, 537, 538, 540, 543, 544, 546, 54
8, 549, 550, 551, 552, 553, 557, 561, 563, 565, 567, 569, 570, 574, 577, 581, 586, 588, 597, 598, 599, 60
0, 603, 604, 605, 606, 609, 610, 611, 615, 616, 617, 621, 624, 625, 628, 632, 634, 635, 636, 639, 643, 64
4, 645, 647, 648, 649, 654, 660, 661, 662, 663, 664, 665, 668, 670, 671, 673, 674, 675, 676, 678, 679, 68
0, 681, 685, 686, 688, 693, 695, 696, 697, 699, 700, 701, 703, 704, 705, 706, 708, 710, 711, 712, 713, 71
```

Simple 3

```
running time: 48.36783480644226
Number of candidate: 100000
set of candidate: {12, 524302, 524303, 786447, 524305, 18, 786449, 20, 262173, 31, 524320, 35, 524325, 3
8, 524327, 524328, 786474, 43, 524332, 524335, 524336, 262193, 50, 524337, 524340, 786482, 786483, 786486,
524347, 62, 64, 524353, 262210, 786500, 262215, 524365, 262222, 80, 524377, 786526, 262242, 99, 524386, 78
6531, 786534, 786535, 105, 786540, 262254, 262257, 262263, 524409, 126, 127, 128, 524416, 786560, 262275,
786561, 786562, 136, 145, 786586, 155, 156, 262302, 786592, 161, 524454, 786599, 524458, 262315, 173, 2623
18, 524461, 786609, 524467, 786611, 786615, 262328, 262333, 786624, 262338, 524484, 786629, 262351, 208, 2
13, 262357, 786649, 219, 223, 524516, 786661, 230, 524520, 262378, 236, 262383, 262384, 786672, 786678, 24
7, 262392, 262393, 250, 786683, 262397, 262398, 256, 262401, 786697, 262410, 524558, 277, 786712, 262425,
262431, 292, 786724, 786733, 302, 786734, 786738, 307, 524596, 786743, 786746, 524609, 331, 262475, 26247
7, 334, 786766, 524627, 524632, 786778, 262491, 524636, 262494, 262495, 524638, 353, 262498, 262508, 26250
9, 786800, 371, 262519, 786808, 262521, 380, 262524, 262525, 786813, 786817, 786824, 393, 524684, 524685,
262542, 262543, 786829, 401, 786834, 786835, 407, 524695, 411, 524700, 414, 262561, 786850, 419, 524713, 2
62572, 429, 430, 431, 524719, 786867, 440, 442, 524731, 262588, 786876, 786879, 262596, 786889, 524747, 46
1, 262606, 524750, 262611, 786900, 524763, 482, 485, 786919, 524780, 524786, 262644, 524791, 262651, 52479
7, 524800, 262657, 514, 786946, 786953, 522, 262669, 786958, 527, 262673, 530, 786975, 786978, 524836, 54
9, 553, 262697, 262707, 524851, 566, 567, 568, 787008, 577, 262721, 262722, 787015, 585, 262732, 524877, 5
92, 593, 594, 787027, 524884, 524891, 524896, 524904, 620, 524910, 787060, 630, 631, 262774, 262775, 78706
```

As the test data increases, the running time becomes longer, but the overall running speed is not very bad.

Part 4 : Improve the efficiency of my implementation

All tests and improvements are completed in the toy data set test.

According to the pseudo-code given by lecture, I completed the initial code.

```
def c2lsh(data_hashes, query_hashes, alpha_m, beta_n):
    offset = -1
    num_rdd_candidates = -1
    list_candidates = []
    while num_rdd_candidates < beta_n:
        rdd_candidates = data_hashes.filter\
            (lambda e: checkmatch(e[1], query_hashes, alpha_m, offset))\
            .map(lambda e: e[0])
        num_rdd_candidates = rdd_candidates.count()

        list_rddcandidates = data_hashes.filter\
            (lambda e: list_candidates.append(e[0]))
        number = list_rddcandidates.count()
        if number < beta_n:
            offset += 1

    return rdd_candidates
```

```
def checkmatch(data_hashes, query_hashes, alpha_m, offset):
    counter = 0
    for i in range(len(data_hashes)):
        if (abs(data_hashes[i] - query_hashes[i]) <= offset):
            counter += 1
            if counter >= alpha_m:
                return counter
```

```
running time: 12.817864894866943
Number of candidate: 10
set of candidate: {0, 70, 40, 10, 80, 50, 20, 90, 60, 30}
```

In this code, build filter and map method complete RDD translated, there have twice RDD translate. Then in order to add the item in to candidates

and offset +=1, I did the RDD translated again. So in this code has three times RDD translated. The running time is 12.8 seconds.

However I found the number part 2 (red number) is not necessary. So I did the first change code. I found offset can follow while method, so I put the offset +=1 at the RDD translate front. Continue to use filter and map method translated RDD. There only has twice RDD translated. The result is better, running time is 6.7 seconds.

```
def c2lsh(data_hashes, query_hashes, alpha_m, beta_n):
    offset = -1
    num_rdd_candidates = -1
    while num_rdd_candidates < beta_n:
        offset += 1
        rdd_candidates = data_hashes.filter\
            (lambda e: checkmatch(e[1], query_hashes, alpha_m, offset))\
            .map(lambda e: e[0])
        num_rdd_candidates = rdd_candidates.count()
    return rdd_candidates

def checkmatch(data_hashes, query_hashes, alpha_m, offset):
    counter = 0
    for i in range(len(data_hashes)):
        if (abs(data_hashes[i] - query_hashes[i]) <= offset):
            counter += 1
            if counter >= alpha_m:
                return counter
```

```
running time: 6.77119255065918
Number of candidate: 10
set of candidate: {0, 70, 40, 10, 80, 50, 20, 90, 60, 30}
```

Secondly, using flatMap method replace the filter and map methods, because the flatMap method can achieve the same operation and get

same result. And there is only one RDD translated here. The running speed will definitely increase.

```
def c2lsh(data_hashes, query_hashes, alpha_m, beta_n):
    offset_ = -1
    num_rdd_candidates = -1
    while num_rdd_candidates < beta_n:
        offset_ += 1
        rdd_candidates = data_hashes.flatMap\
            (lambda e: [e[0]] if checkmatch(e[1], query_hashes, alpha_m, offset_) else [])
        num_rdd_candidates = rdd_candidates.count()
    return rdd_candidates

def checkmatch(data_hashes, query_hashes, alpha_m, offset):
    counter = 0
    for i in range(len(data_hashes)):
        if (abs(data_hashes[i] - query_hashes[i]) <= offset):
            counter += 1
            if counter >= alpha_m:
                return counter
```

running time: 5.214612722396851

Number of candidate: 10

set of candidate: {0, 70, 40, 10, 80, 50, 20, 90, 60, 30}

The result has indeed changed, but the effect is not particularly obvious.

The reason may be that the test set is not large enough. Below I use my simple3 testcases, one million to test the effect of these three improvements.

First

running time: 75.2646837234497

Number of candidate: 100000

set of candidate: {12, 524302, 524303, 786447, 524305, 18, 786449, 20, 262173, 31, 524320, 35, 524325, 38, 524327, 524328, 786474, 43, 524332, 524335, 524336, 262193, 50, 524337, 524340, 786482, 786483, 786486, 5243 47, 62, 64, 524353, 262210, 786500, 262215, 524365, 262222, 80, 524377, 786526, 262242, 99, 524386, 786531, 786534, 786535, 105, 786540, 262254, 262257, 262263, 524409, 126, 127, 128, 524416, 786560, 262275, 786561, 786562, 136, 145, 786586, 155, 156, 262302, 786592, 161, 524454, 786599, 524458, 262315, 173, 262318, 52446 1, 786609, 524467, 786611, 786615, 262328, 262333, 786624, 262338, 524484, 786629, 262351, 208, 213, 262357, 786649, 219, 223, 524516, 786661, 230, 524520, 262378, 236, 262383, 262384, 786672, 786678, 247, 262392, 262 393, 250, 786683, 262397, 262398, 256, 262401, 786697, 262410, 524558, 277, 786712, 262425, 262431, 292, 786 724, 786733, 302, 786734, 786738, 307, 524596, 786743, 786746, 524609, 331, 262475, 262477, 334, 786766, 524 627, 524632, 786778, 262491, 524636, 262494, 262495, 524638, 353, 262498, 262508, 262509, 786800, 371, 26251 9, 786808, 262521, 380, 262524, 262525, 786813, 786817, 786824, 393, 524684, 524685, 262542, 262543, 786829, 401, 786834, 786835, 407, 524695, 411, 524700, 414, 262561, 786850, 419, 524713, 262572, 429, 430, 431, 5247 19, 786867, 440, 442, 524731, 262588, 786876, 786879, 262596, 786889, 524747, 461, 262606, 524750, 262611, 7 86900, 524763, 482, 485, 786919, 524780, 524786, 262644, 524791, 262651, 524797, 524800, 262657, 514, 78694 6, 786953, 522, 262669, 786958, 527, 262673, 530, 786975, 786978, 524836, 549, 553, 262697, 262707, 524851, 566, 567, 568, 787008, 577, 262721, 262722, 787015, 585, 262732, 524877, 592, 593, 594, 787027, 524884, 5248 91, 524896, 524904, 620, 524910, 787060, 630, 631, 262774, 262775, 787063, 787067, 262781, 643, 262787, 5249 33, 262794, 262796, 655, 787092, 262806, 787094, 665, 524955, 668, 524966, 787117, 262841, 262842, 262844, 5 24991, 524997, 787143, 716, 262860, 718, 262862, 723, 262868, 525012, 262871, 787159, 731, 733, 734, 262877, 525029, 747, 262892, 787188, 759, 525050, 787196, 262912, 262915, 772, 262916, 787210, 525074, 262932, 78722 0, 793, 787227, 798, 262942, 787230, 262945, 787231, 803, 262955, 813, 787246, 525106, 787251, 525120, 26297

Second

running time: 71.4861068725586

Number of candidate: 100000

set of candidate: {12, 524302, 524303, 786447, 524305, 18, 786449, 20, 262173, 31, 524320, 35, 524325, 3 8, 524327, 524328, 786474, 43, 524332, 524335, 524336, 262193, 50, 524337, 524340, 786482, 786483, 786486, 524347, 62, 64, 524353, 262210, 786500, 262215, 524365, 262222, 80, 524377, 786526, 262242, 99, 524386, 78 6531, 786534, 786535, 105, 786540, 262254, 262257, 262263, 524409, 126, 127, 128, 524416, 786560, 262275, 786561, 786562, 136, 145, 786586, 155, 156, 262302, 786592, 161, 524454, 786599, 524458, 262315, 173, 2623 18, 524461, 786609, 524467, 786611, 786615, 262328, 262333, 786624, 262338, 524484, 786629, 262351, 208, 2 13, 262357, 786649, 219, 223, 524516, 786661, 230, 524520, 262378, 236, 262383, 262384, 786672, 786678, 24 7, 262392, 262393, 250, 786683, 262397, 262398, 256, 262401, 786697, 262410, 524558, 277, 786712, 262425, 262431, 292, 786724, 786733, 302, 786734, 786738, 307, 524596, 786743, 786746, 524609, 331, 262475, 26247 7, 334, 786766, 524627, 524632, 786778, 262491, 524636, 262494, 262495, 524638, 353, 262498, 262508, 26250 9, 786800, 371, 262519, 786808, 262521, 380, 262524, 262525, 786813, 786817, 786824, 393, 524684, 524685, 262542, 262543, 786829, 401, 786834, 786835, 407, 524695, 411, 524700, 414, 262561, 786850, 419, 524713, 2 62572, 429, 430, 431, 524719, 786867, 440, 442, 524731, 262588, 786876, 786879, 262596, 786889, 524747, 46 1, 262606, 524750, 262611, 786900, 524763, 482, 485, 786919, 524780, 524786, 262644, 524791, 262651, 52479 7, 524800, 262657, 514, 786946, 786953, 522, 262669, 786958, 527, 262673, 530, 786975, 786978, 524836, 54 9, 553, 262697, 262707, 524851, 566, 567, 568, 787008, 577, 262721, 262722, 787015, 585, 262732, 524877, 5 92, 593, 594, 787027, 524884, 524891, 524896, 524904, 620, 524910, 787060, 630, 631, 262774, 262775, 78706

Third

```
running time: 48.36783480644226
Number of candidate: 100000
set of candidate: {12, 524302, 524303, 786447, 524305, 18, 786449, 20, 262173, 31, 524320, 35, 524325, 3
8, 524327, 524328, 786474, 43, 524332, 524335, 524336, 262193, 50, 524337, 524340, 786482, 786483, 786486,
524347, 62, 64, 524353, 262210, 786500, 262215, 524365, 262222, 80, 524377, 786526, 262242, 99, 524386, 78
6531, 786534, 786535, 105, 786540, 262254, 262257, 262263, 524409, 126, 127, 128, 524416, 786560, 262275,
786561, 786562, 136, 145, 786586, 155, 156, 262302, 786592, 161, 524454, 786599, 524458, 262315, 173, 2623
18, 524461, 786609, 524467, 786611, 786615, 262328, 262333, 786624, 262338, 524484, 786629, 262351, 208, 2
13, 262357, 786649, 219, 223, 524516, 786661, 230, 524520, 262378, 236, 262383, 262384, 786672, 786678, 24
7, 262392, 262393, 250, 786683, 262397, 262398, 256, 262401, 786697, 262410, 524558, 277, 786712, 262425,
262431, 292, 786724, 786733, 302, 786734, 786738, 307, 524596, 786743, 786746, 524609, 331, 262475, 26247
7, 334, 786766, 524627, 524632, 786778, 262491, 524636, 262494, 262495, 524638, 353, 262498, 262508, 26250
9, 786800, 371, 262519, 786808, 262521, 380, 262524, 262525, 786813, 786817, 786824, 393, 524684, 524685,
262542, 262543, 786829, 401, 786834, 786835, 407, 524695, 411, 524700, 414, 262561, 786850, 419, 524713, 2
62572, 429, 430, 431, 524719, 786867, 440, 442, 524731, 262588, 786876, 786879, 262596, 786889, 524747, 46
1, 262606, 524750, 262611, 786900, 524763, 482, 485, 786919, 524780, 524786, 262644, 524791, 262651, 52479
7, 524800, 262657, 514, 786946, 786953, 522, 262669, 786958, 527, 262673, 530, 786975, 786978, 524836, 54
9, 553, 262697, 262707, 524851, 566, 567, 568, 787008, 577, 262721, 262722, 787015, 585, 262732, 524877, 5
92, 593, 594, 787027, 524884, 524891, 524896, 524904, 620, 524910, 787060, 630, 631, 262774, 262775, 78706
```

In the case of using a single core, the final code running time is shortened by 23s, the result is very good.