

COMP9313

Project 1

登登教育

-Elijah-

20T2

1

C2LSH 解析

2

实现思路
代码详解

3

效率优化

1

C2LSH 解析

Pseudo code of Candidate Generation in C2LSH

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

count(hashes_1, hashes_2, offset):

counter \leftarrow 0

for each $hash_1, hash_2$ **in** hashes_1, hashes_2:

if $|hash_1 - hash_2| \leq offset$:

counter \leftarrow counter + 1

return counter

假设 $\alpha m = 8$

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

HashCode(query)

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

Collision

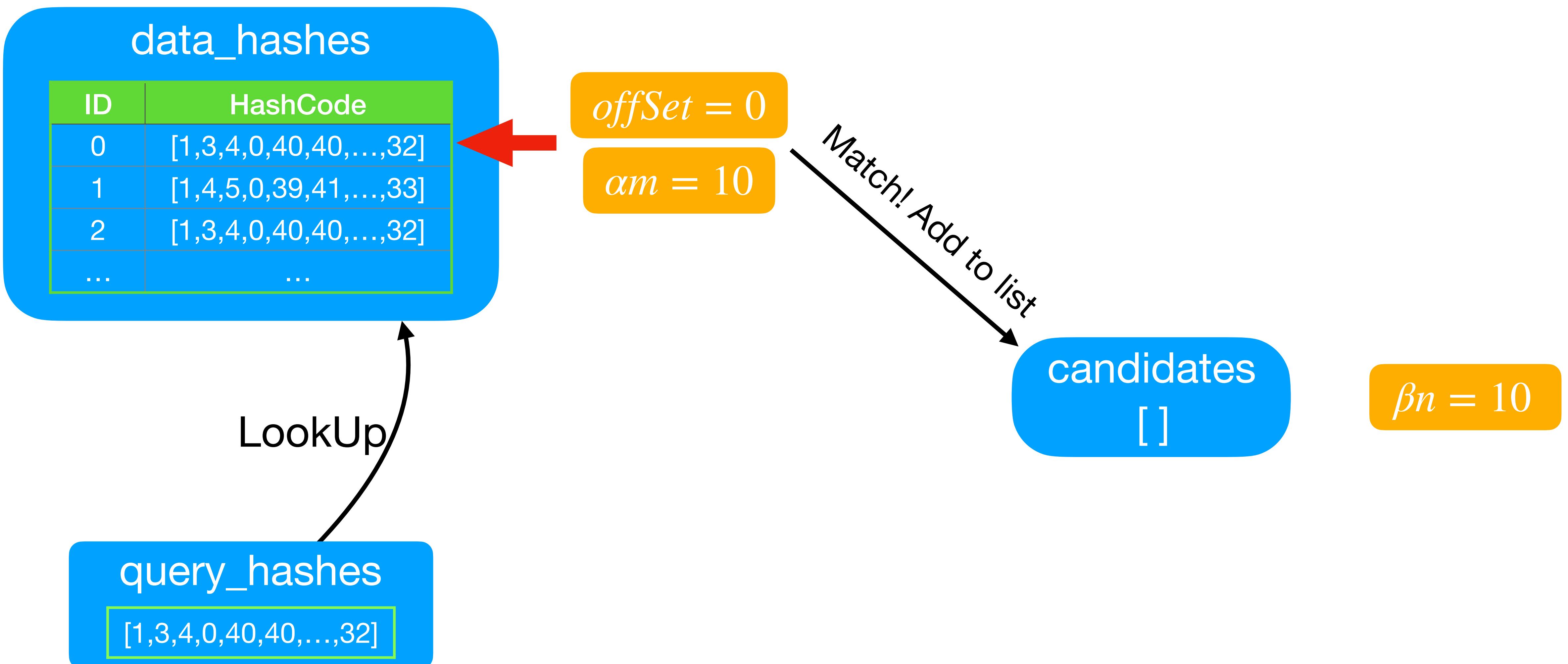
- Consider $h(o) = h(q) \pm 1$ if not enough candidates

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

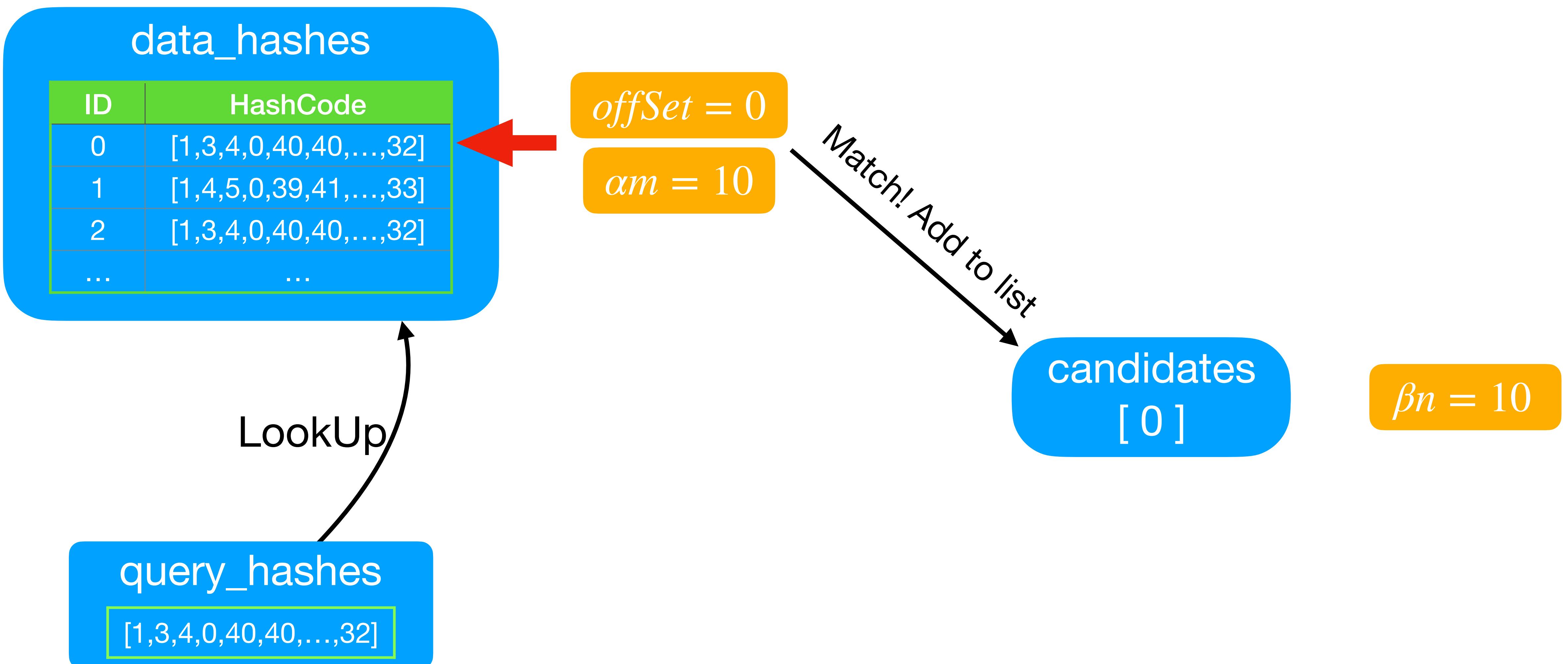
- Then $h(o) = h(q) \pm 2$ and so on...

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

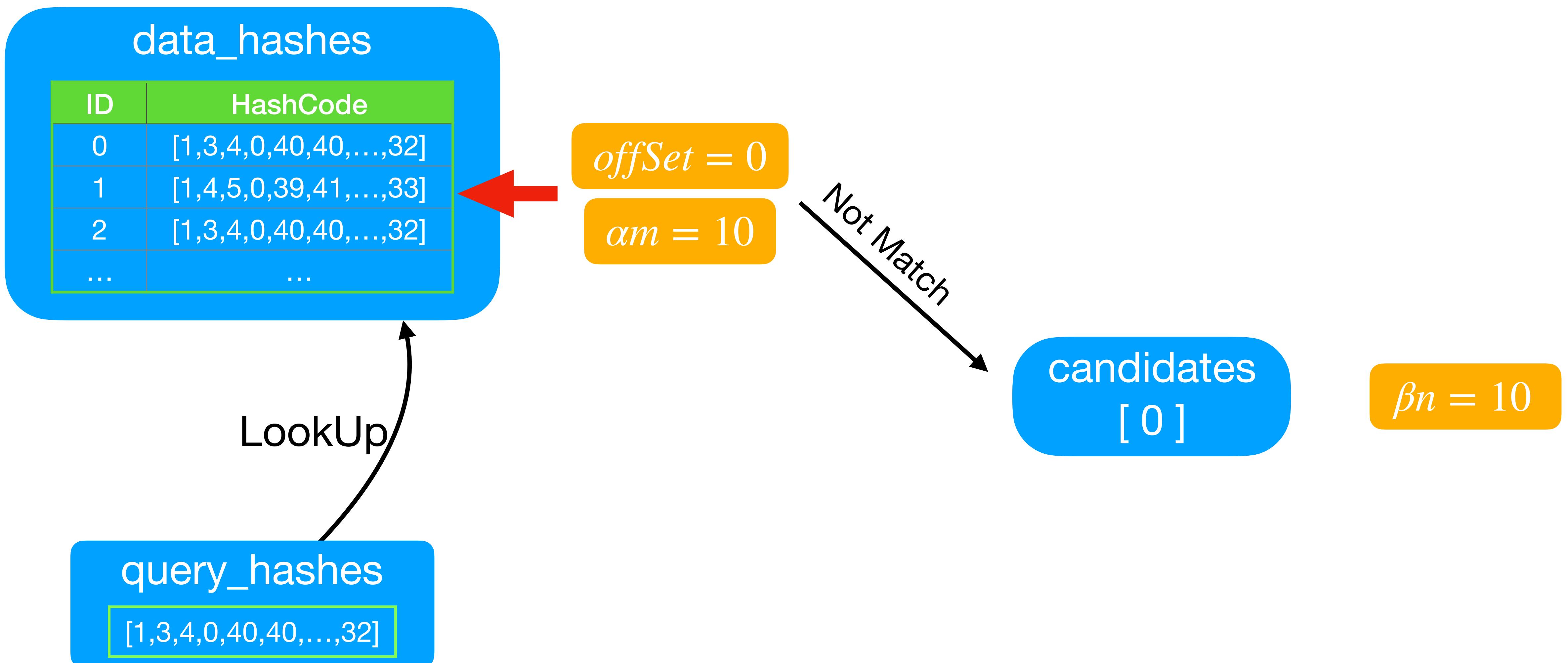
C2LSH 流程



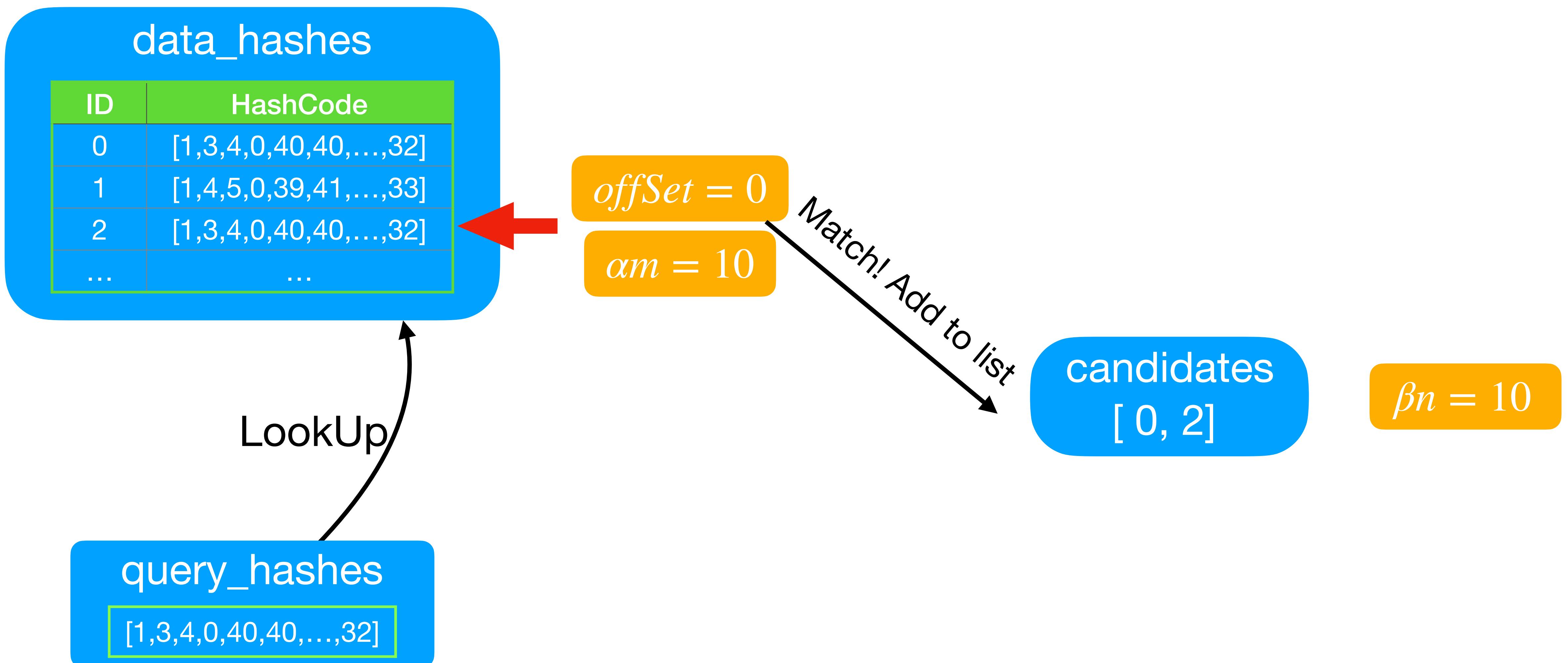
C2LSH 流程



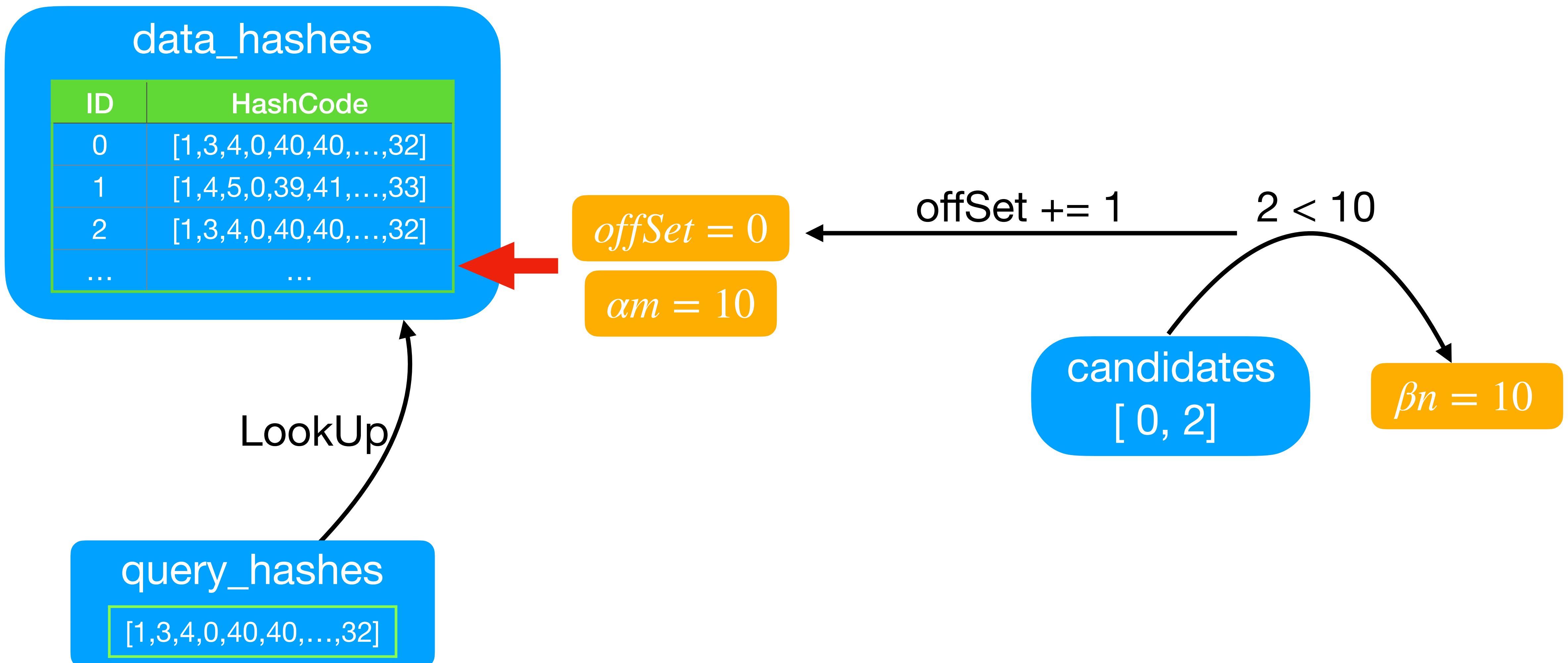
C2LSH 流程



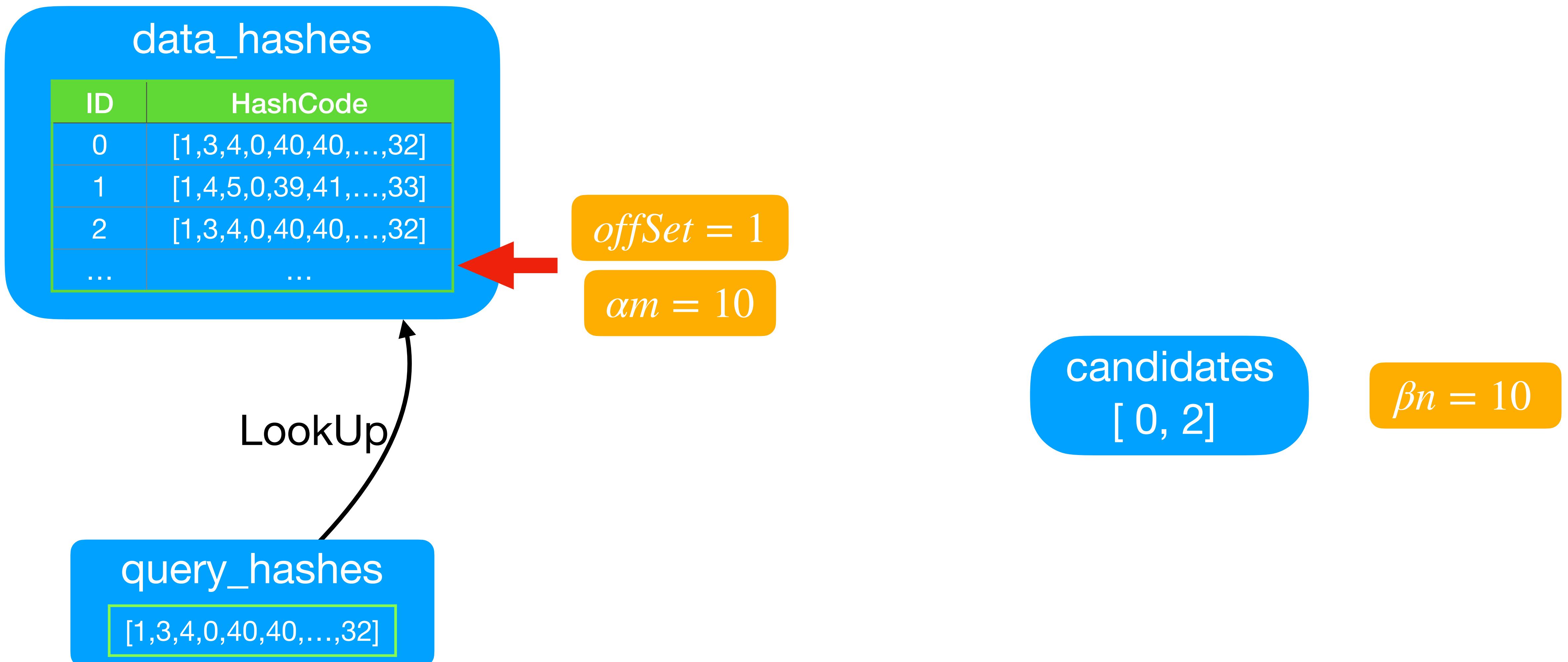
C2LSH 流程



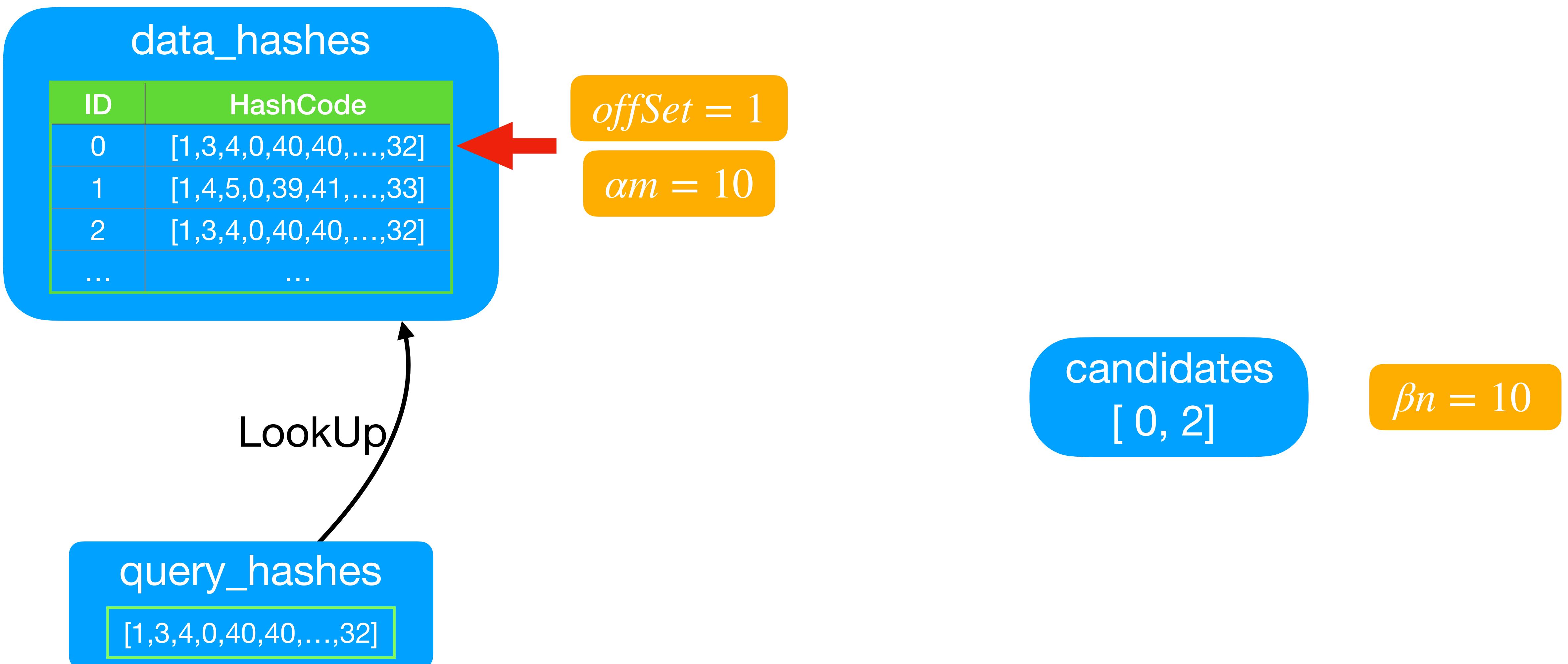
C2LSH 流程



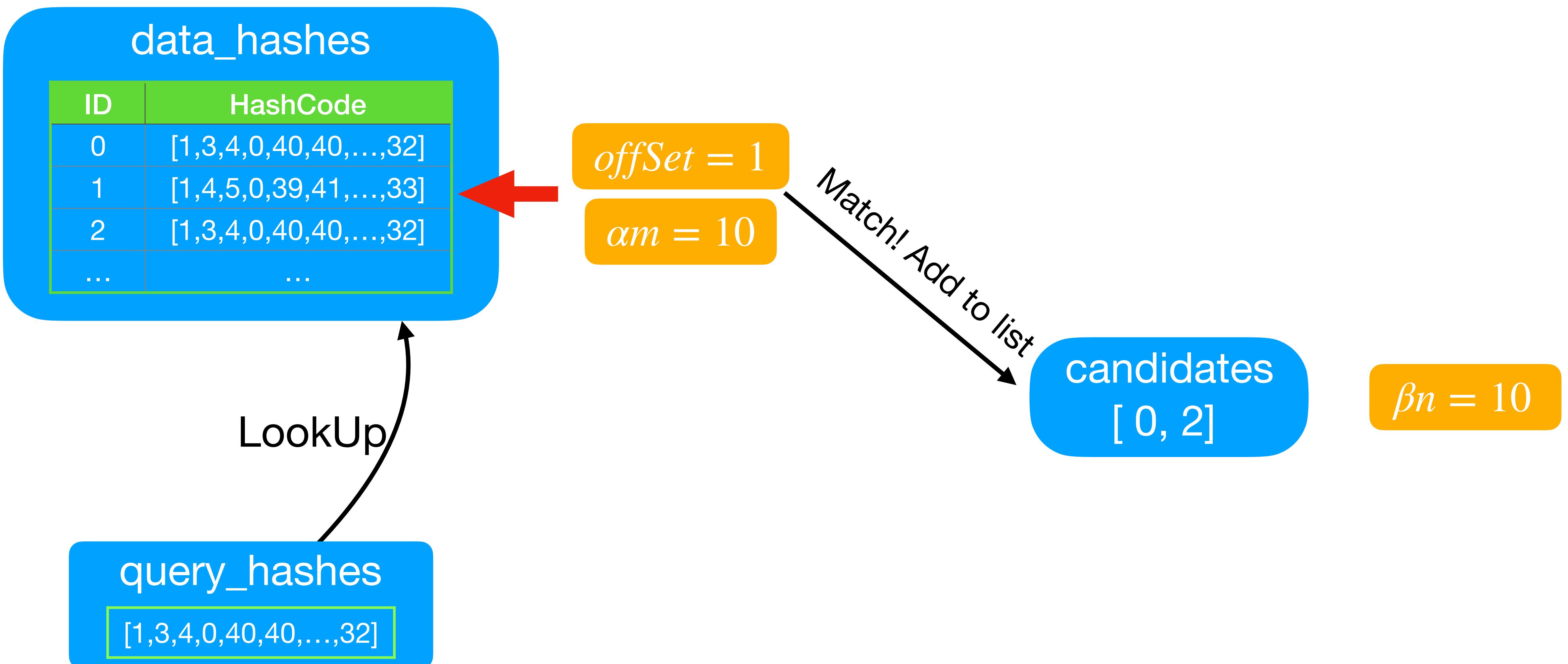
C2LSH 流程



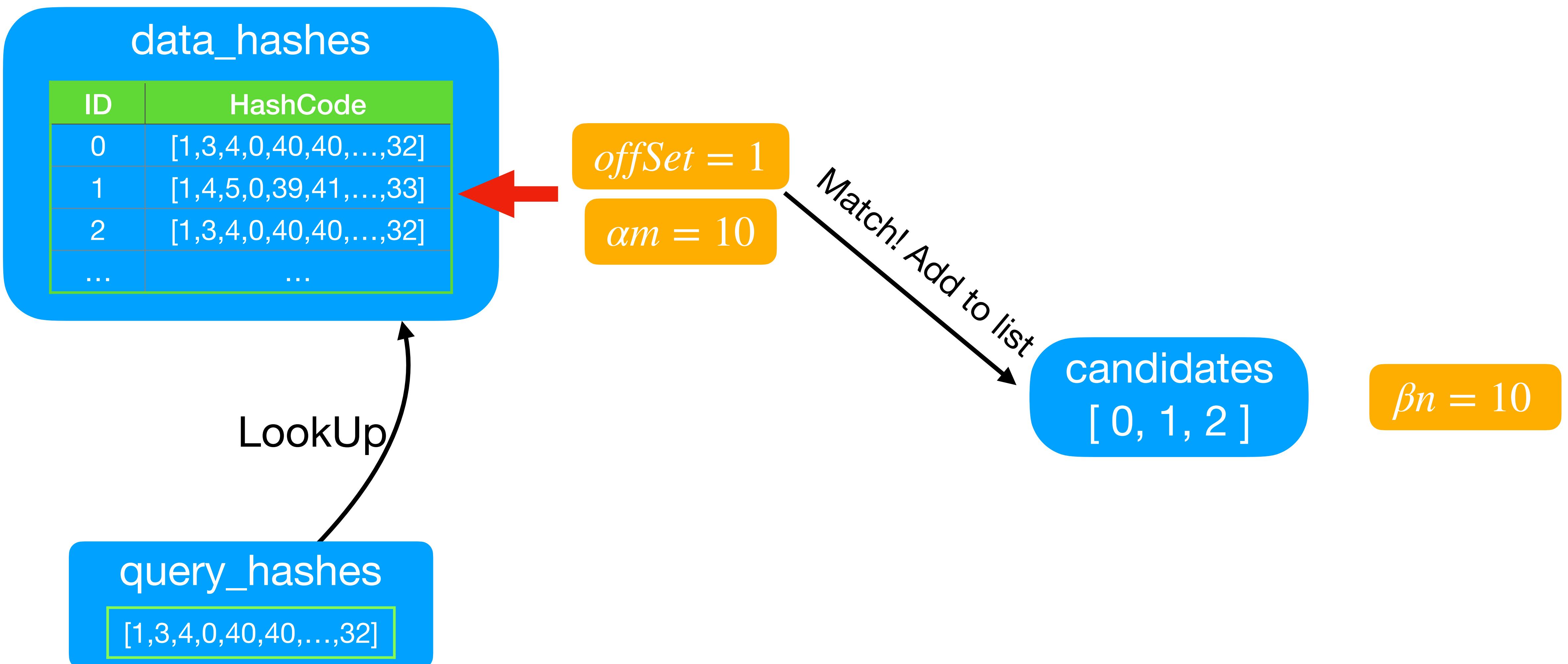
C2LSH 流程



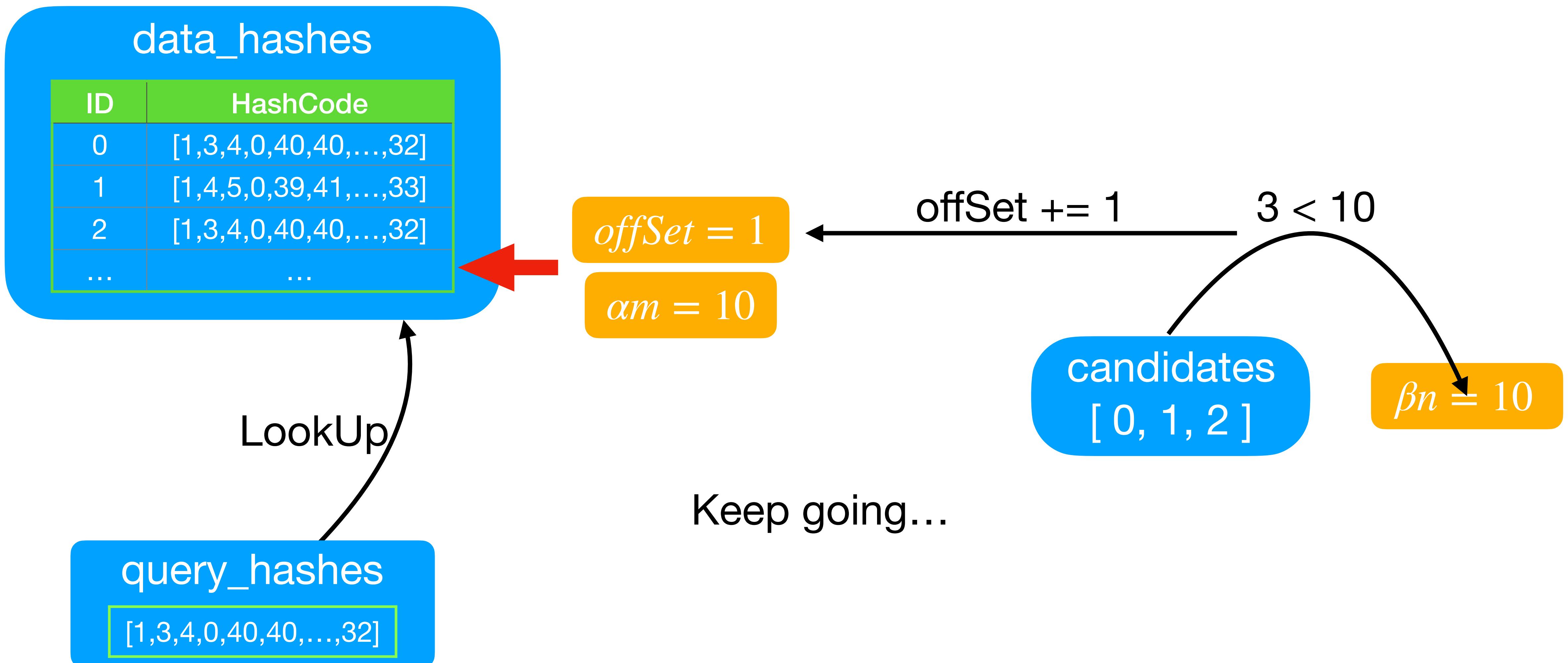
C2LSH 流程



C2LSH 流程



C2LSH 流程



2

实现思路
代码详解

如何使用 Spark RDD 实现这个算法

如何使用 Spark RDD 实现这个算法
当你想“遍历”RDD的时候就要想到 map()

如何使用 Spark RDD 实现这个算法

当你想“遍历”RDD的时候就要想到 map()

当你想筛选某些符合条件的元素的时候 就要想到 Filter()

如何使用 Spark RDD 实现这个算法

当你想“遍历”RDD的时候就要想到 map()

当你想筛选某些符合条件的元素的时候 就要想到 Filter()

当你想按 Key 聚合 RDD 的时候就要想到 groupByKey, combineByKey, reduceByKey...

如何使用 Spark RDD 实现这个算法

当你想“遍历”RDD的时候就要想到 map()

当你想筛选某些符合条件的元素的时候 就要想到 Filter()

当你想按 Key 聚合 RDD 的时候就要想到 groupByKey, combineByKey, reduceByKey...

当你想做扁平化的时候就要想到 flatMap()

如何使用 Spark RDD 实现这个算法

当你想“遍历”RDD的时候就要想到 map()

当你想筛选某些符合条件的元素的时候 就要想到 Filter()

当你想按 Key 聚合 RDD 的时候就要想到 groupByKey, combineByKey, reduceByKey...

当你想做扁平化的时候就要想到 flatMap()

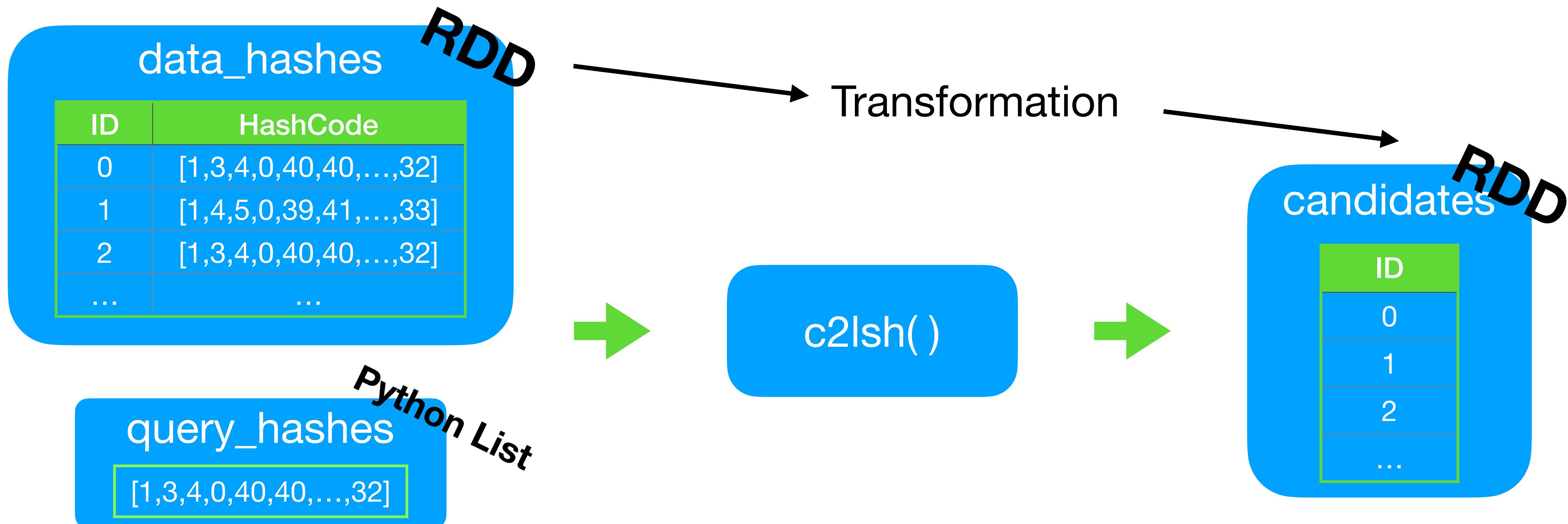
...



submission.py

```
| def c2lsh(data_hashes, query_hashes, alpha_m, beta_n):
```

C2LSH 实现思路



ifQualified 写好之后一个 filter 一个 map 解决战斗
Demo

3

效率优化

1. filter + map => flatMap

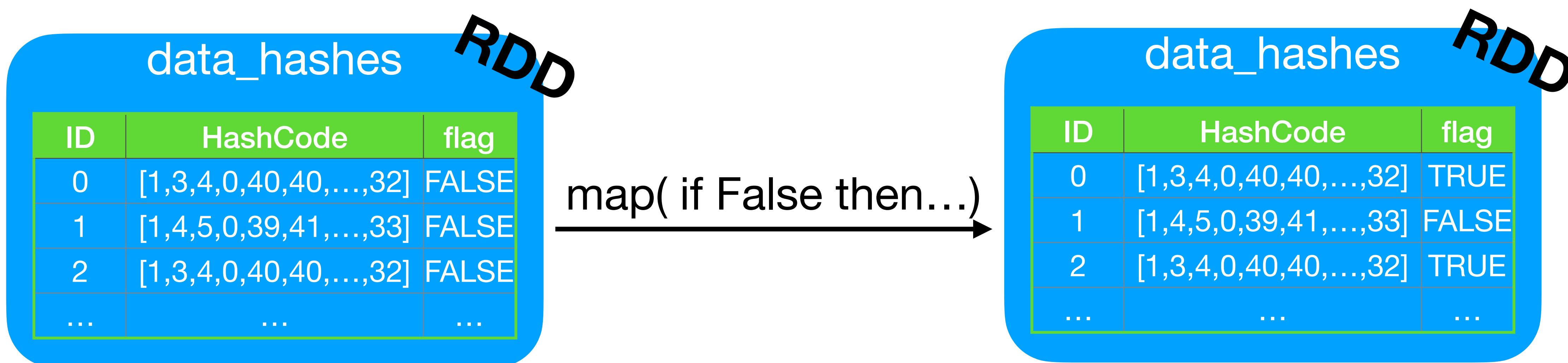
2. 把 $\text{abs}(\text{做差的计算提前算好})$

Demo

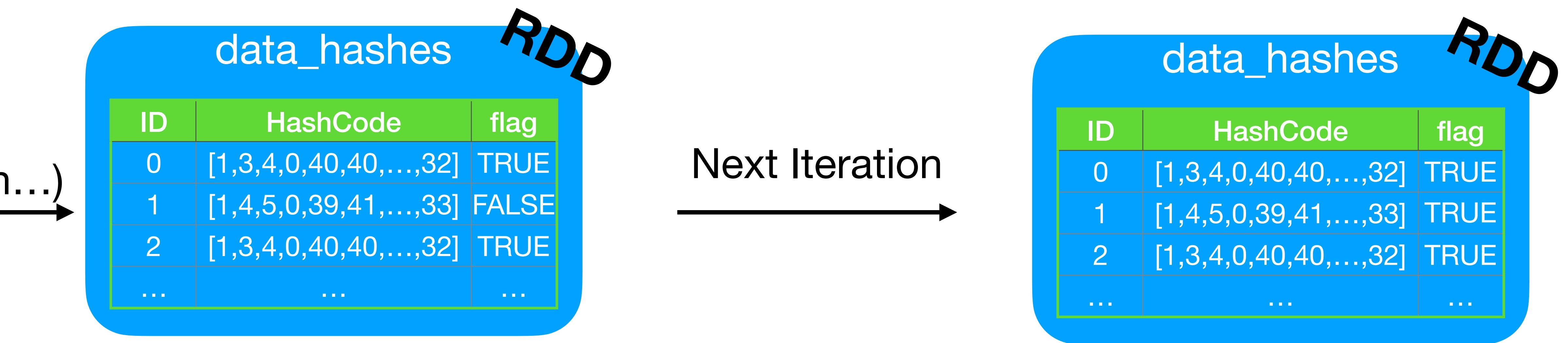
其他的实现思路：

加 flag 迭代 data_hash的版本

C2LSH DAG



C2LSH DAG



C2LSH DAG

data_hashes *RDD*

| ID | HashCode | flag |
|-----|------------------------|------|
| 0 | [1,3,4,0,40,40,...,32] | TRUE |
| 1 | [1,4,5,0,39,41,...,33] | TRUE |
| 2 | [1,3,4,0,40,40,...,32] | TRUE |
| ... | ... | ... |

get Id Only if True



candidates *RDD*

| ID |
|-----|
| 0 |
| 1 |
| 2 |
| ... |

但实测不加 flag 不迭代更新更快一点

多map了几步，加 RDD 复用

| testCase2 | |
|------------|------|
| 不迭代不加 Flag | 32s |
| 迭代， 加 Flag | 128s |

3. map => mapPartitions

4. Cache (重复使用，或者开始有分支的 RDD)

其他调优方案（有待测试）

1. 使用 `HashCode` 作为 Key 对 id 聚合
(此方法有 shuffle 不一定快取决于 `DataSet`)

data_hashes

| ID | HashCode |
|-----|------------------------|
| 0 | [1,3,4,0,40,40,...,32] |
| 1 | [1,4,5,0,39,41,...,33] |
| 2 | [1,3,4,0,40,40,...,32] |
| ... | ... |

groupBy(HashCode)



shuffle_hashes

| HashCode | IDs |
|------------------------|--------|
| [1,3,4,0,40,40,...,32] | [0, 2] |
| [1,4,5,0,39,41,...,33] | [1] |
| ... | ... |

add Flag



Candidates

| ID |
|-----|
| 0 |
| 1 |
| 2 |
| ... |

flatMap



Candidates

| IDs |
|--------|
| [0, 2] |
| [1] |
| ... |

shuffle_hashes

| HashCode | IDs | Flag |
|------------------------|--------|-------|
| [1,3,4,0,40,40,...,32] | [0, 2] | FALSE |
| [1,4,5,0,39,41,...,33] | [1] | FALSE |
| ... | ... | ... |

Demo

- `groupByKey`

2. groupByKey => combineByKey

3. 手动分区 repartition(12) (有 shuffle Performance 有待测试)

Demo

- 使用自定义累加器代替 shuffle (进阶需要 sc)
- 使用 broadCast 代替 join (进阶需要 sc)

testCase

同一个testCase 多跑几次，

等电脑凉的时候再跑，

同时建议跑的时候设置成 local[4]

3 个 testCase 都是 32 位的 hashCode，使用的 queryHash 都是 32 个 0

testCase0.pkl : 20w 数据都是 32 个 0

testCase1.pkl : 20w 数据，有 5w 是 32 个 0，另有 5w 是 32 个 [-10, 10] 内的随机数
另有 10w 个是 [20, 40] 内的随机数

testCase2.pkl : 100w 数据，有 5w 是 32 个 0，另有 5w 是 32 个 [-10, 10] 内的随机数
另有 90w 个是 [20, 40] 内的随机数

| | | alphaN = 10 Beta = 10 | alphaN = 10 Beta = 1 | alphaN = 10 Beta = 10w | alphaN = 10 Beta = 10w |
|---|-----------|--------------------------|-------------------------|---------------------------|---------------------------|
| | | toy | testCase0 | testCase1 | testCase2 |
| no flag no 迭代 | Version 1 | 1.8s | 2.87s | 9s | 29s |
| Shuffle once Flag 迭代 cache mapPartitions | Version 3 | 2.6s | 2.15s | 14s | 74s |

调优路漫漫其远兮。。。

不帮忙 debug, 如果有需要联系 Amy 1v1

Question?