

Clare Xinyu Xu  
Z5175081

## COMP9313 ASSIGN1

### Q1: HDFS

1. R here is the total # of block replicas R which is already the result after times five.  
In that case, blocks = R/N.

$$L1(k, N) = K * R/N - 2 * L2(K, N) - 3 * L3(K, N) - 4 * L4(K, N) - 5 * L5(K, N)$$

$$L2(K, N) = 4 * \frac{L1(K-1, N)}{(N-K+1)} + L2(K-1, N) - 3 * \frac{L2(K-1, N)}{(N-K+1)}$$

$$L3(K, N) = 3 * \frac{L2(K-1, N)}{(N-K+1)} + L3(K-1, N) - 2 * \frac{L3(K-1, N)}{(N-K+1)}$$

$$L4(K, N) = 2 * \frac{L3(K-1, N)}{(N-K+1)} + L4(K-1, N) - \frac{L4(K-1, N)}{(N-K+1)}$$

$$L5(K, N) = \frac{L4(K-1, N)}{(N-K+1)} + L5(K-1, N)$$

2. For N= 500, R=20,000,000 and k = 200

```
In [34]: 1 #Written by Clare Xinyu Xu
2 #z5175081
3 N = 500
4 R = 20000000
5 K = 200
6 B = R/N
```

```
In [35]: 1 L1_arr = [-1]*N
2 L2_arr = [-1]*N
3 L3_arr = [-1]*N
4 L4_arr = [-1]*N
5 L5_arr = [-1]*N
```

```
In [36]: 1 def L1(K,N):
2     global L1_arr,B
3
4     if K <=0:
5         return 0
6     if L1_arr[K-1] != -1:
```

```
1 def L2(K,N):
2     global L2_arr
3
4     if K <=0:
5         return 0
6     if L2_arr[K-1] != -1:
7         return L2_arr[K-1]
8     ans = 4 * L1(K-1,N)/(N-K+1) + L2(K-1, N) - 3 * L2(K-1,N)/(N-K+1)
9     L2_arr[K-1] = ans
10    return ans
```

```
1 def L3(K,N):
2     global L3_arr
3     if K <=0:
4         return 0
5     if L3_arr[K-1] != -1:
6         return L3_arr[K-1]
7     ans = 3 * L2(K-1,N)/(N-K+1) + L3(K-1, N) - 2 * L3(K-1,N)/(N-K+1)
8     L3_arr[K-1] = ans
9     return ans
```

```
1 def L4(K,N):
2     global L4_arr
3
4     if K <=0:
5         return 0
6     if L4_arr[K-1] != -1:
7         return L4_arr[K-1]
8     ans = 2 * L3(K-1,N)/(N-K+1) + L4(K-1, N) - L4(K-1,N)/(N-K+1)
9     L4_arr[K-1] = ans
10    return ans
```

```
1 def L5(K,N):
2     global L5_arr
3
4     if K <=0:
5         return 0
6     if L5_arr[K-1] != -1:
7         return L5_arr[K-1]
8     ans = L4(K-1,N)/(N-K+1) + L5(K-1, N)
9     L5_arr[K-1] = ans
10    return ans
```

```
1 L1(K,N)
1036781.3821642093
```

```
1 L2(K,N)
1389356.8690281338
```

```
1 L3(K,N)
923129.7317703705
```

```
1 L4(K,N)
304107.9551149882
```

```
1 L5(K,N)
39736.77280169178
```

In [29]: `1 max(L1_arr)`  
 Out[29]: 1646639.4029663643

In [30]: `1 max(L2_arr)`  
 Out[30]: 1389356.8690281338

In [31]: `1 max(L3_arr)`  
 Out[31]: 923129.7317703705

In [32]: `1 max(L4_arr)`  
 Out[32]: 304107.9551149882

In [33]: `1 max(L5_arr)`  
 Out[33]: 39736.77280169178

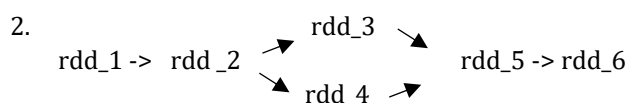
I put the above  $Li(K,N)$  ( $i=1..5$ ) and calculate the  $Li\_arr$ . Our final result is the  $Li\_arr$  since if it has been calculated before, it will be return the value with ans.

My  $Li\_arr$  result is shown as the screenshot. L1 is 1646639, L2 is 1389357, L3 is 923130, L4 is 304108 and L5 is 39737.

Since the question is to ask to compute the number of blocks that cannot be recovered. In this case, it is the value L5 which is 39737.

Q2: Spark

1. [('Joseph', 165), ('Jimmy', 159), ('Tina', 155), ('Thomas', 167)]



There are three stages in total according to the result calculated by the computer via <http://localhost:4040/stages/>.

#### Completed Stages (3)

Stage Id	Description	Submitted	Duration
2	<a href="#">collect at &lt;ipython-input-3-6f9348be7110&gt;:7</a> <small>+details</small>	2020/07/26 12:37:26	2 s
1	<a href="#">reduceByKey at &lt;ipython-input-3-6f9348be7110&gt;:3</a> <small>+details</small>	2020/07/26 12:37:24	0.7 s
0	<a href="#">reduceByKey at &lt;ipython-input-3-6f9348be7110&gt;:4</a> <small>+details</small>	2020/07/26 12:37:24	2 s

Stage 1: rdd\_1, rdd\_2, rdd\_3

Stage 2: rdd\_4

Stage 3: rdd\_5, rdd\_6

Reason: Rdd\_1, rdd\_2 and rdd\_3 is in the first stage. First of all, map is a narrow transformation. Then shuffle Rdd is created by ReduceByKey wide transformation so after it is a new stage by which means rdd 1 to 3 is in the same stage. Similarly, another ReduceByKey generate shuffle Rdd after which is the stage 2. Followed by Join here is to join data together by same key with same amounts that doesn't create the shuffle. Therefore, together with the map and collect stay in the stage 3.

Clare Xinyu Xu  
Z5175081

3. The reason for the implementation is inefficient is due to the massive numbers of transformations. Also, there are two wide transformations in the process in total, which contained two combineByKey extend the time.

One idea is to decrease the number of stages to make less of the wide transformation. What I did is only to use one combinByKey instead of two reduceByKey. Then we concatenate the scores of each person together and make it into a score list according to the different names. While we generate the list as the following format [('Joseph', [83, 74, 91, 82]), ('Jimmy', [69, 62, 97, 80]), ('Tina', [78, 73, 68, 87]), ('Thomas', [87, 93, 91, 74])], we can find the max and min score for each student and sum them up. Therefore, the total number of stages has been decreased to two only, and the number of rdd is also only two.

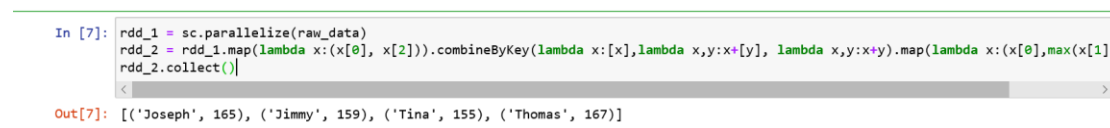
By what we did, the algorithm is more efficient.Code:

```
rdd_1 = sc.parallelize(raw_data)

rdd_2 = rdd_1.map(lambda x:(x[0], x[2])).combineByKey(lambda x:[x],lambda x,y:x+[y],
lambda x,y:x+y).map(lambda x:(x[0],max(x[1])+min(x[1])))

rdd_2.collect()
```

Screenshot:



```
In [7]: rdd_1 = sc.parallelize(raw_data)
rdd_2 = rdd_1.map(lambda x:(x[0], x[2])).combineByKey(lambda x:[x],lambda x,y:x+[y],
lambda x,y:x+y).map(lambda x:(x[0],max(x[1])+min(x[1])))
rdd_2.collect()

Out[7]: [('Joseph', 165), ('Jimmy', 159), ('Tina', 155), ('Thomas', 167)]
```

Q3: LSH

1. According to the formula in the slides, the probability of o is a nearest neighbor candidate of q is

$$p = 1 - (1 - p_{q,o}^k)^l$$

While  $P \geq 99\%$  and  $k = 5$  here

We also notice that the image o is a near duplicate to q if  $\cos(\theta(o,q)) \geq 0.9$ .

Therefore,  $P_{q,o} = P(\text{dist}(q,o)) = P(h(o) = h(q)) = 1 - \frac{\theta}{\pi} \geq 1 - \frac{\cos^{-1} 0.9}{\pi} = 85.64\%$

$$p_{q,o}^k = (85.84\%)^5 \approx 46.067\%$$

$$P \geq 99\% = (1 - (1 - 46.067\%)^l)$$

$$l \geq 8$$

Therefore, at least 8 hash tables the LSH scheme require to guarantee the possibility.

$$2. P = 1 - (1 - p_{q,o}^k)^l$$

Image o will be false positive when  $\text{dist}(o,q) > r_2$  which equals to  $\Pr[h(o) = h(q)] \leq p_2$

$$\text{Thus } P_{q,o} = P(h(o) = h(q)) = 1 - \frac{\theta}{\pi} = 1 - \frac{\cos^{-1} 0.8}{\pi} \approx 79.51\%$$

$$P = 1 - (1 - (79.51\%)^5)^{10} \approx 97.8\%$$

$\Pr[h(o) = h(q)] = \text{similarity}(o,q) = 79.51\%$ . By which means an image o with  $\cos(\theta(o,q)) < 0.8$  is already not a near duplicate. Thus, we are just calculating the probability of LSH calling it a near duplicate. Therefore, approximately 97.8% pairs of (o,q) with similarity 79.51% end up becoming candidate pairs. They are false positives since we will have to examine them due to the candidate pairs but then it will turn out that they are not near duplicate.

Thus, maximum value of the probability of o to become a false positive of query q is 97.8%.