JIE MEI z5173405

Question 1

1. $L1(k,N) = k * B - 2 * L2(k,N) - 3 * L3(k,N) - 4 * L4(k,N) - 5 * L5(k,N)$

In the process of calculating L1(k,N), the total number of failed blocks is subtracted from the number of lost blocks of 2, 3, 4, 5 replicas.

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

In our calculation of L2(k,N), we first calculate that when k-1 dataNodes have failed, one replica has been lost and when the kth dataNodes failed, its second replica has also lost. Then add that the number of blocks with two replicas lost when k-1 dataNodes have failed. Finally, we need to subtract the number of blocks that lost 3 replicas when the kth dataNodes failed.

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

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

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

$L3(k,N)$, $L4(k,N)$ and $L5(k,N)$ like the $L2(k,N)$ calculation process.

2. In the code, I created a 2d array to store all the calculated data. I used the concept of dynamic programming to complete this problem. I need the result of the previous step for every calculation.

Sample code(java)

```java
class Q1 {
    public static void calculate(int l,int k,float array[][]) {
        if(l != 1) {
            array[l][k] = (6-l)*array[l-1][k-1]/(500-k+1)+array[l][k-1]-(6-l-1)*array[l][k-1]/(500-k+1);
        }
        else {
            array[l][k] = k*40000-2*array[2][k]-3*array[3][k]-4*array[4][k]-5*array[5][k];
```

```
        }
    }

    public static void main(String[] args) {
        float[][] array = new float[6][201];
        for(int i = 0;i < 6;i++) {
            for(int j = 0;j < 201;j++) {
                array[i][j] = 0;
            }
        }
        array[1][1] = 40000;
        for(int k = 2; k < 201;k++) {
            for(int l = 5; l > 0;l--) {
                calculate(l,k,array);
            }
        }
        System.out.println(array[5][200]);

    }
}
```

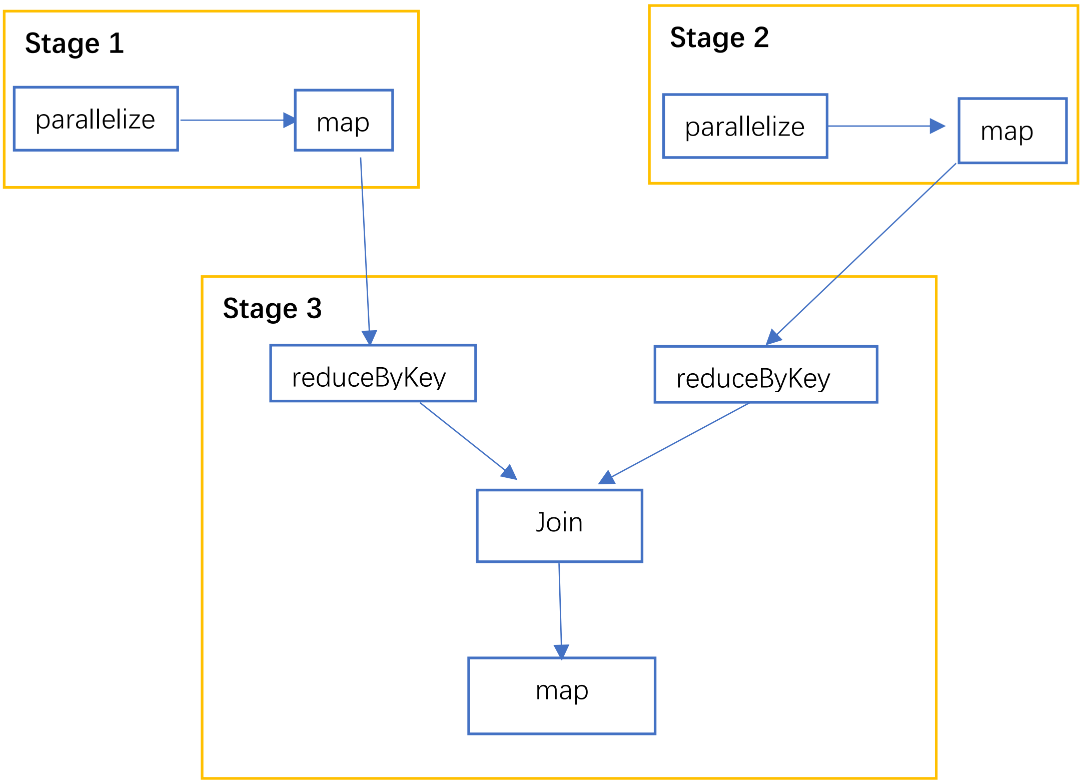
We can get the number of blocks that cannot be recovered under this scenario is

39736.77.

Question 2

1. The expected output should be that: [('Joseph', 165), ('Jimmy', 159), ('Tina', 155), ('Thomas', 167)]. Rdd_1 is the initialization raw_data. Rdd_2 is the attribute of each data only the name and score. Rdd_3 is to find the maximum score of each person. Rdd_4 is to find the minimum score for each person. Rdd_5 is rdd_3 join rdd_4, and the score column will have a minimum score and a maximum score. Rdd_6 add the maximum score and the minimum score together.

2. Above code snippet have 3 stages.



Since reduceByKey is shuffled during the transformation process, it will be divided into two stages before reduceByKey. Because the two reduceByKey operations divide the same key into the same partition, when the join operator is executed, the join operator performs the join operation on the same key, and no shuffle is generated. So other transformations are in same stage.

3. Two shuffle operations in the above code will cause inefficiency. Only one shuffle operation is needed to achieve the above results. I can replace twice reduceByKey with a groupByKey.

The code like that:

```
rdd_1 = sc.parallelize(raw_data)

rdd_2 = rdd_1.map(lambda x: (x[0],x[2])).groupByKey()

rdd_3 = rdd_2.map(lambda x: (x[0],max(x[1])+ min(x[1])))
```

rdd_3.collect()

Question 3

1. According to the question, we know the $\cos\theta\,(o,q) \geq 0.9.$ so we can get the $\theta \leq 25.84°$. Then we can know that: $\Pr[h_{(o)} = h_{(q)}] = 1 - \frac{\theta}{\pi} \geq 0.856$. We want to find any near duplicate with probability no less than 99% that should be: $1 - \left(1 - P_{(q,o)}^{k}\right)^{l} \geq 0.99$, we know the $k = 5\ and\ P_{(q,o)} = 0.856$, so we can get the $1 - (1 - 0.856^{5})^{l} \geq 0.99$. Finally, we can get the result $L \geq 8.$
2. From the $\cos\theta\,(o,q) < 0.8$, we can get the $\theta > 36.9°$. Then we can know that: $\Pr[h_{(o)} = h_{(q)}] = 1 - \frac{\theta}{\pi} < 0.795$. we still use the equation $1 - \left(1 - P_{(q,o)}^{k}\right)^{l}$ to get the maximum value of the probability of o to become a false positive of query q. we can get the $1 - (1 - 0.795^{5})^{10} = 0.978$. therefore, the maximum value of the probability of o to become a false positive of query q is 97.8%.