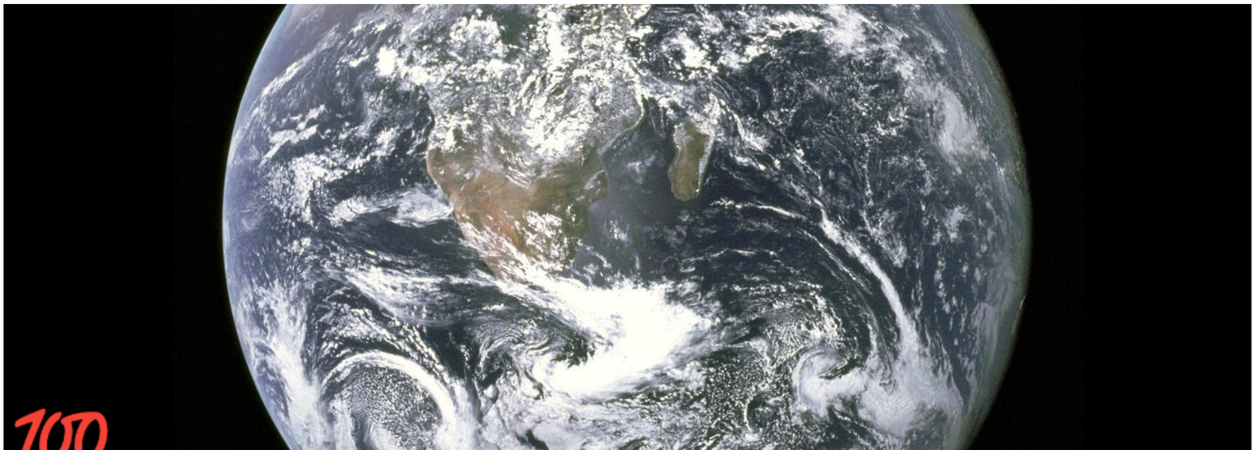


Mining Big Data Assignment 2

`Exercise 1` S-curve:

The code file:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/997f67dc-67f9-46ce-a1e1-9f93ded7c593/S-curve_plot.py

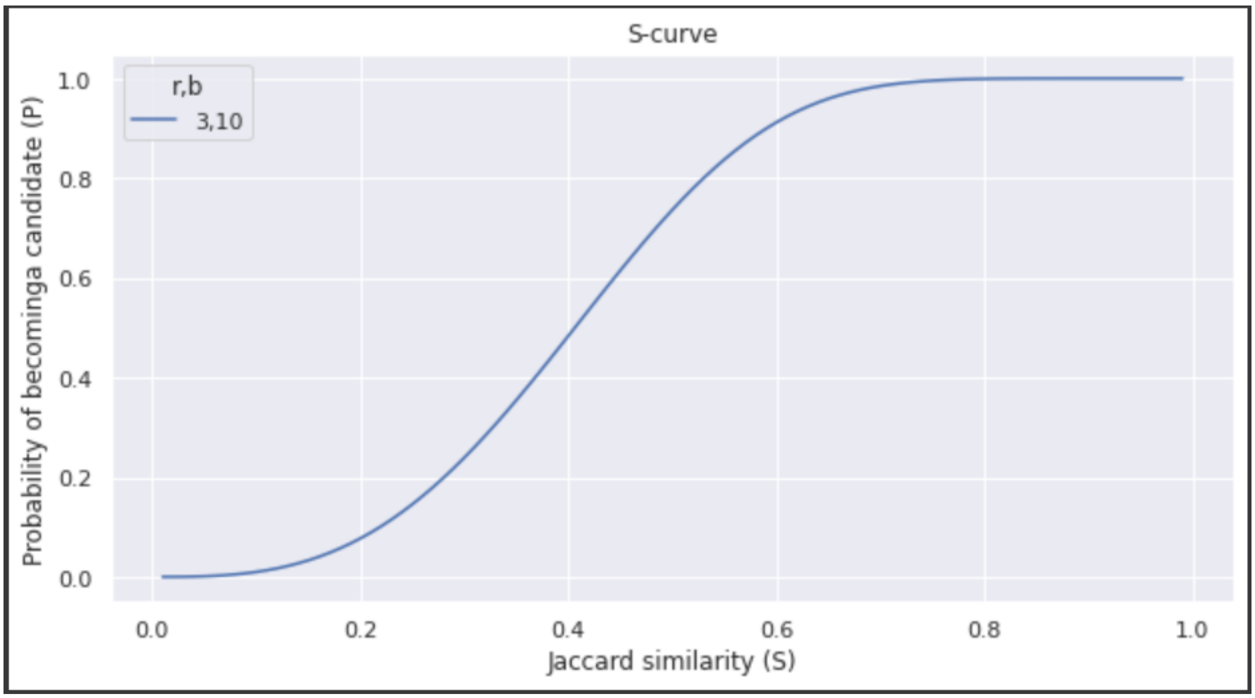
```
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1. r=3 and b=10:

```python
def probability(s, r, b):
    # s: similarity
    # r: rows (per band)
    # b: number of bands
    return 1 - (1 - s**r)**b

results = pd.DataFrame({
    'Jaccard similarity (S)': [],
    'Probability of becominga candidate (P)': [],
    'r,b': []
})

for s in np.arange(0.01, 1, 0.01):
        b = 10
        r = 3
        P = probability(s, r, b)
        results = results.append({
            'Jaccard similarity (S)': s,
            'Probability of becominga candidate (P)': P,
            'r,b': f"{r},{b}"
        }, ignore_index=True)
# plot line graph
sns.set(rc={'figure.figsize':(10,5)})
ax = sns.lineplot(data=results, x='Jaccard similarity (S)', y='Probability of becominga candidate (P)', hue='r,b', color='#965786')
ax.set(title="S-curve")
```

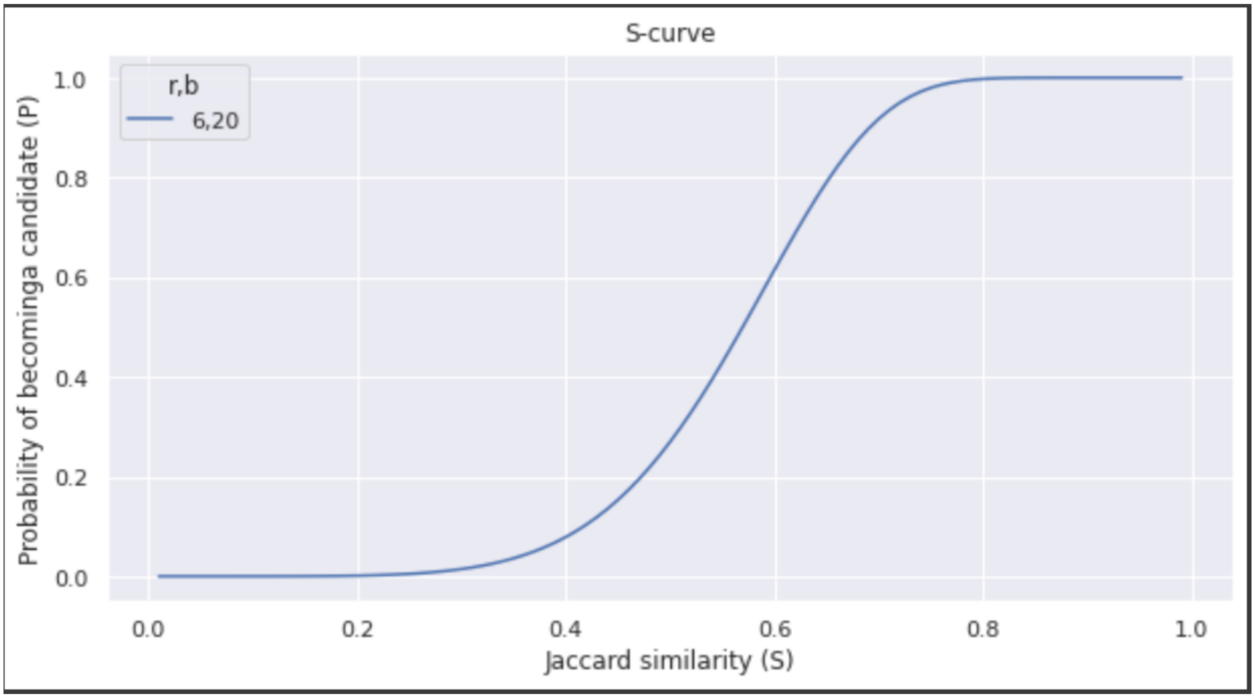


2. r=6 and b=20:

```python
def probability(s, r, b):
        # s: similarity
        # r: rows (per band)
        # b: number of bands
        return 1 - (1 - s**r)**b

results = pd.DataFrame({
        'Jaccard similarity (S)': [],
        'Probability of becominga candidate (P)': [],
        'r,b': []
})

for s in np.arange(0.01, 1, 0.01):
                b = 20
                r = 6
                P = probability(s, r, b)
                results = results.append({
                        'Jaccard similarity (S)': s,
                        'Probability of becominga candidate (P)': P,
                        'r,b': f"{r},{b}"
                }, ignore_index=True)
# plot line graph
sns.set(rc={'figure.figsize':(10,5)})
ax = sns.lineplot(data=results, x='Jaccard similarity (S)', y='Probability of becominga candidate (P)', hue='r,b', color='#965786')
ax.set(title="S-curve")
```

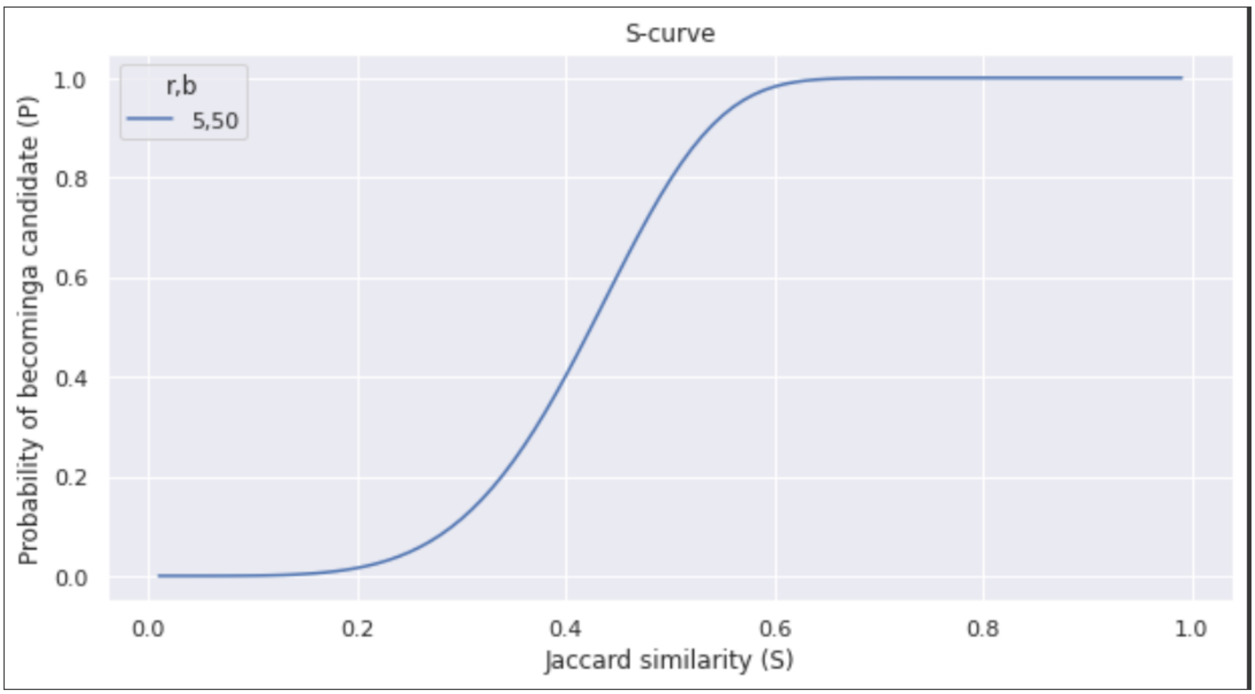


3. r=5 and b=50:

```
def probability(s, r, b):
        # s: similarity
        # r: rows (per band)
        # b: number of bands
        return 1 - (1 - s**r)**b

results = pd.DataFrame({
        'Jaccard similarity (S)': [],
        'Probability of becominga candidate (P)': [],
        'r,b': []
})

for s in np.arange(0.01, 1, 0.01):
                b = 50
                r = 5
                P = probability(s, r, b)
                results = results.append({
                        'Jaccard similarity (S)': s,
                        'Probability of becominga candidate (P)': P,
                        'r,b': f"{r},{b}"
                }, ignore_index=True)
# plot line graph
sns.set(rc={'figure.figsize':(10,5)})
ax = sns.lineplot(data=results, x='Jaccard similarity (S)', y='Probability of becominga candidate (P)', hue='r,b', color='#965786')
ax.set(title="S-curve")
```



`Exercise 2` Filtering Streams:

1. m = 2 billion members of S → 2 x 10^9

   n = 10 billion bits → 1 x 10^10

   false-positive rate when using three hash functions:

$(1-e^{-}km/n)k$ {where k = 3, m = 2 x 10^9, n = 1 x 10^10 }

$e^{-}km/n = 0.5488116$

$(1-e^{-}km/n)^k = (1-0.5488116)^3 =$ 0.09185

false-positive rate when using four hash functions:

$(1-e^{-}km/n)k$ {where k = 4, m = 2 x 10^9, n = 1 x 10^10 }

$e^{-}km/n = 0.449328$

$(1-e^{-}km/n)^k = (1-0.449328)^4 = 0.09195$

2.

$f(k) = (1 - e^{-km/n})^k$

The Stationary Points are at $\frac{df(k)}{dk} = 0$

Let $y = e^{-\frac{m}{n}}$, $\frac{m}{n} > 0$ $\therefore$ $\underline{0 < y < 1}$

$$\frac{d}{dx}((1-y^x)^x) =$$

$$\left(\ln(1-y^x) - \frac{x\left(x\frac{d}{dx}(y) + \ln(y)\, y\right) y^{x-1}}{1-y^x}\right)(1-y^x)^x$$

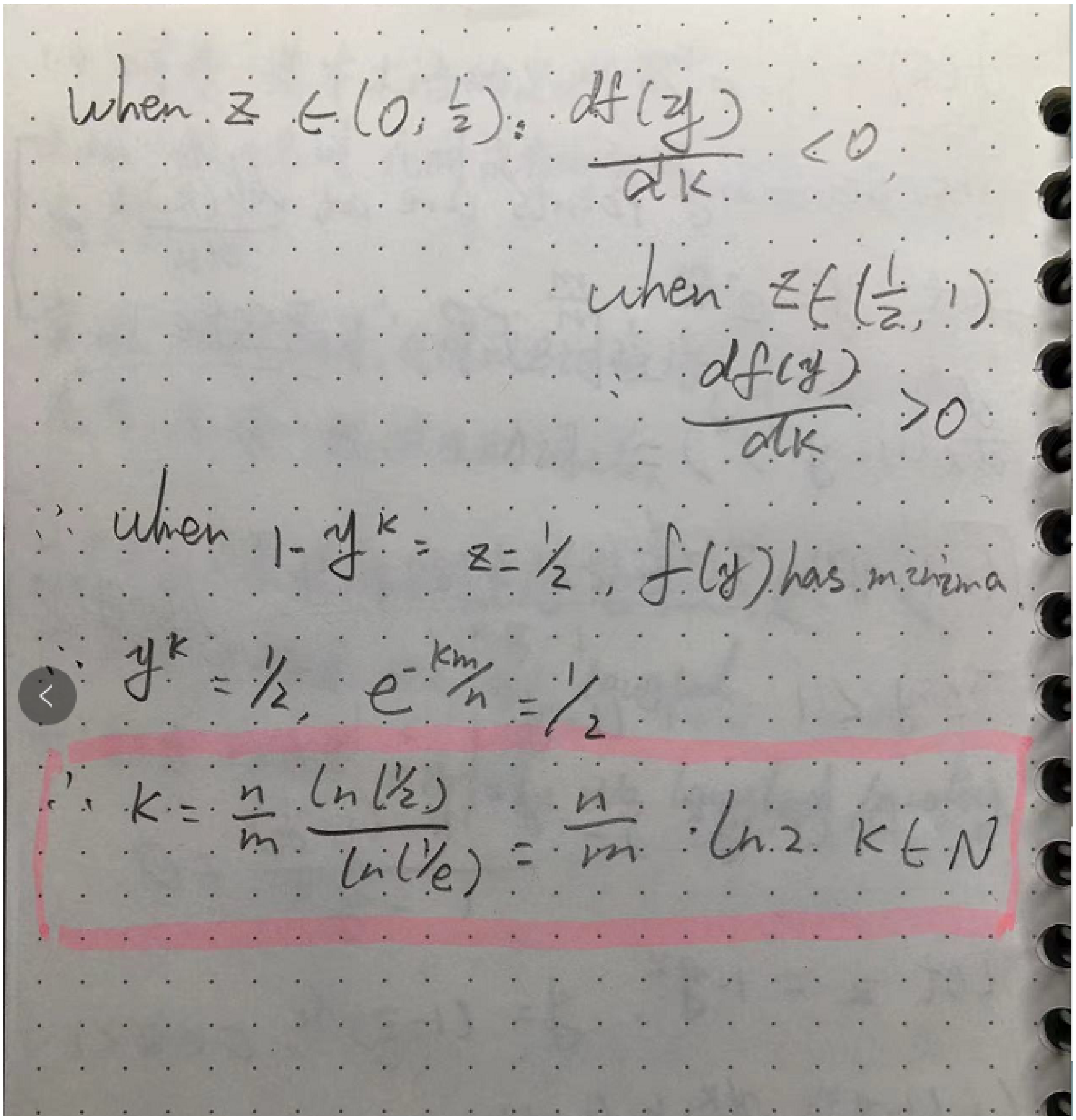
$\because y < 1$ $\therefore (1-y^k)^k > 1$

$$\boxed{\therefore \ln(1-y^k) - \frac{y^k \cdot k \cdot \ln(y)}{1-y^k} = 0}$$

Let $z = 1-y^k$, $y = (1-z)^{1/k}$, $0 < z < 1$

$$\ln(1-y^k) - \frac{y^k \cdot k \cdot \ln(y)}{1-y^k} = \ln(z) - \frac{(1-z)\cdot k \cdot \ln\left((1-z)^{\frac{1}{k}}\right)}{z}$$

$$= \ln(z) - \frac{(1-z)\cdot \ln(1-z)}{z} = 0.$$



The false-positive rate is minimised when the number of hash functions equal to n/m*ln2.

`Exercise 3` Map-Reduce: Friend Recommendation System:

Files:

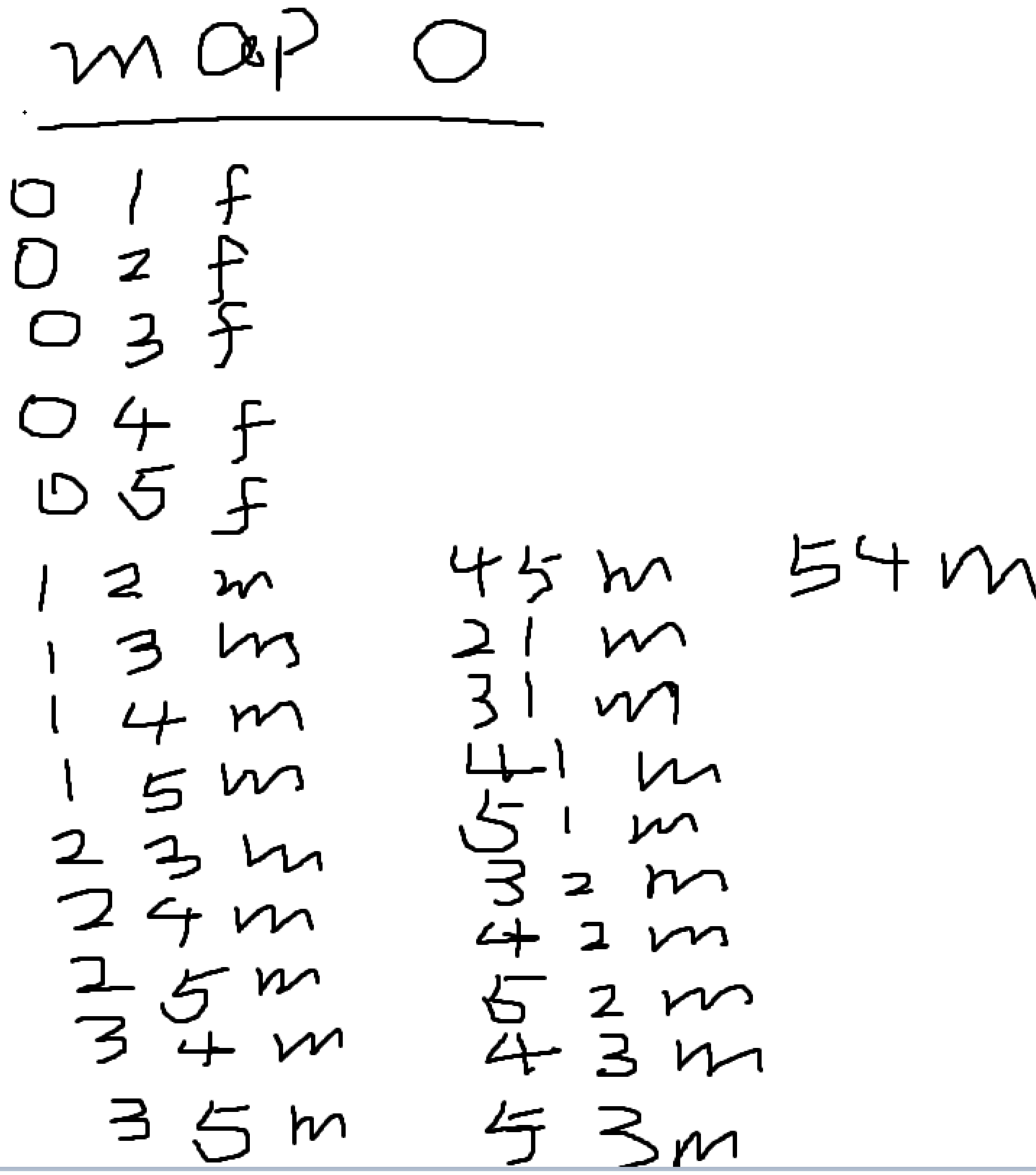
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/38abf1f2-126f-4481-930e-e8c67965bd9d/Friend-Recommendation-System.zip

Write-up:

The main idea for Map function is to identify the friends of each user and to label them based on whether they are friends or they got mutual friends.

user	friend
0	1, 2, 3, 4, 5
1	0, 2, 4
2	0, 1
3	0
4	0, 1
5	0

For example: In the above figure, the users are 0,1,2,3,4,5 and each user has their corresponding friends. The map function will first take one friend out of the user's friends list and mark that friend and user are in friend relationship. In user 0 case, map will first take friend 1 and mark that friend 1 and user 0 are in friend relationship. After marking each friends in user's friends list, map will then take two friends out of user's friends list and mark those two friends have mutual friend which is the user. In user 0 case, the map will take friend 1 and friend 2 and mark them have mutual friend which is user 0. In order to identify all the friends and be able to shuffle and sort them, the map will not only put friend 1 and friend 2 in this way, but it will also put friend 2 and friend 1 so that each friend can be the key in the reduce process.



In the figure above, we can see that all the friends and users' relationship been stated. "f" denote they are friends and "m" denote they have mutual friend. Number 0 and 1 has been used in the actual map implementation in which number 0 denote they have mutual friend and 1 denote they are friend.

reduce 0

0 1 f
0 2 f
0 3 f
0 4 f
0 5 f
0 2 m
0 4 m
0 1 m
0 1 m

After each key value pairs been outputted from map and been shuffled and sorted, the reduce will get all pairs based on the key value. Take the example from first figure, after all the users been putted into the map, this is what we would get in the reduce. The whole logic behind the reduce part is to count the number of mutual friends. From figure above we can see that, user 0 and user 2 have mutual friend, therefore, user 2 could be the potential friend for user 0. However, we can also see that user 2 is already the friend of user 0, hence, no recommendation of user 2 to user 0. The algorithm in reduce simply check each key value pairs, if the marker denote that they are friend already, then put a sign to say no recommendation for this user, if the marker denote that they have mutual friend, then increment counter by 1.

Finally, sort the user IDs in numerically ascending order if there are recommended users with the same number of mutual friends and sort the user in decreasing order based on the number of mutual friends they got.

Results:

User ID	Recommendations
924	439, 2409, 6995, 11860, 15416, 43748, 45881
8941	8943, 8944, 8940
8942	8939, 8940, 8943, 8944
9019	9022, 317, 9023
9020	9021, 9016, 9017, 9022, 317, 9023
9021	9020, 9016, 9017, 9022, 317, 9023
9022	9019, 9020, 9021, 317, 9016, 9017, 9023
9990	13134, 13478, 13877, 34299, 34485, 34642, 37941
9992	9987, 9989, 35667, 9991
9993	9991, 13134, 13478, 13877, 34299, 34485, 34642, 37941

`Exercise 4` Data streams:

Scenario 1:

Data stream consists of the integers 3, 1, 4, 6, 5, 9

Question 1: Hash function: $h(x) = (2x + 1) \bmod 32$
Question 2: Hash function: $h(x) = (3x + 7) \bmod 32$
Question 3: Hash function: $h(x) = 4x \bmod 32$

Calculate a hash for each element from a bag:

x_i	$h_1(x_i)$ {where $h_1(x_i) = (2x + 1) \bmod 32$}	$h_2(x_i)$ {where $h_2(x_i) = (3x + 7) \bmod 32$}	$h_3(x_i)$ {where $h_3(x_i) = 4x \bmod 32$}
3	7	16	12
1	3	10	4
4	9	19	16
6	13	25	24
5	11	22	20
9	19	34	4

write a binary representation of a hash value:

h1(xi)	binary(h1(xi))	h2(xi)	binary(h2(xi))	h3(xi)	binary(h3(xi))
7	00111	16	10000	12	01100
3	00011	10	01010	4	00100
9	01001	19	10011	16	10000
13	01101	25	11001	24	11000
11	01011	22	10110	20	10100
19	10011	34	00010	4	00100

Question 1: Hash function: h(x) = (2x + 1) mod 32

The maximum tail length is 0.

Use estimate 2^R for the number of distinct elements seen in the stream

2^0 = 1

The resulting estimate of the number of distinct elements is 1.

Question 2: Hash function: h(x) = (3x + 7) mod 32

The maximum tail length is 4.

Use estimate 2^R for the number of distinct elements seen in the stream

2^4 = 16

The resulting estimate of the number of distinct elements is 16.

Question 3: Hash function: h(x) = 4x mod 32

The maximum tail length is 4.

Use estimate 2^R for the number of distinct elements seen in the stream

2^4 = 16

The resulting estimate of the number of distinct elements is 16.

Scenario 2:

Data stream consists of the integers 4, 5, 6, 7, 10, 15.

– Question 4: Hash function: h(x) = (6x + 2) mod 32
– Question 5: Hash function: h(x) = (2x + 5) mod 32
– Question 6: Hash function: h(x) = 2x mod 32

Calculate a hash for each element from a bag:

xi	h4(xi) {where h4(xi) = (6x + 2) mod 32}	h5(xi) {where h5(xi) = (2x + 5) mod 32}	h6(xi) {where h6(xi) = 2x mod 32}
4	26	13	8
5	0	15	10
6	6	17	12
7	12	19	14
10	30	25	20
15	28	35	30

h4(xi)	binary(h4(xi))	h5(xi)	binary(h5(xi))	h6(xi)	binary(h5(xi))
26	11010	13	01101	8	01000
0	00000	15	01111	10	01010
6	00110	17	10001	12	01100
12	01100	19	10011	14	01110
30	11110	25	11001	20	10100
28	11100	35	100011	30	11110

Question 4: Hash function: h(x) = (6x + 2) mod 32

The maximum tail length is 2.

Use estimate 2^R for the number of distinct elements seen in the stream

2^2 = 4

The resulting estimate of the number of distinct elements is 4.

Question 5: Hash function: h(x) = (2x + 5) mod 32

The maximum tail length is 0.

Use estimate 2^R for the number of distinct elements seen in the stream

2^0 = 1

The resulting estimate of the number of distinct elements is 1.

Question 6: Hash function: h(x) = 2x mod 32

The maximum tail length is 3.

Use estimate 2^R for the number of distinct elements seen in the stream

2^3 = 8

The resulting estimate of the number of distinct elements is 8.

`Exercise 5` Summary of 3.6 and 3.7:

3.6 The Theory of Locality-Sensitive Functions

This part introduces the characteristics of the function family and the role of the function family. The family of function which can are combined and more effectively distinguish similar pairs.
Example of the family of functions: Minhash.
The family of function can efficiently generate candidate pairs. A family of functions needs to meet three conditions:

1. They must choose the close pair to be the candidate pair
2. Functions must be independent in statistical
3. The function must be able to identify candidate pairs within a short period of time and they must be composable to build functions for avoiding false positives and negative, in addition to that the combined function must take much less time than number of pairs.

3.6.1 Locality-Sensitive Functions:

This part is to introduce the definition of the function family and the conditions that the functions in the function family need to meet.

The definition of the family of functions is the function input two set elements {x, y} check if these two elements are candidate pairs.

The function is satisfying the following two conditions:

- If x and y are closer, the probability of $f(x) = f(y)$ is higher.
- Or if x and y are further apart, the probability of $f(x) = f(y)$ is lower.

For example, d is the distance measure, if $d1 < d2$ and then we can have a family function F (d1, d2, p1, p2)-sensitive

If $d(x, y) \leq d1$, the $Pr\,[f(x) = f(y)] \geq p1$
If $d(x, y) \geq d1$, the $Pr\,[f(x) = f(y)] \leq p1$

3.6.2 Locality-Sensitive Families for Jaccard Distance:

This part is about the relationship and use of function family and Jaccard distance.

After we define the family of functions, the minhash function is matching with it, and we set the Jaccard Distance as the distance measure:

-if d is the Jaccard distance, the $SIM(x, y) = 1 - d(x, y)$, because the probability that the hashes of x and y is equal to their Jaccard similarity
So, when 0 ≤ d1 < d2 ≤ 1 then we get minhash function (d1, d2, 1-d1, 1-d2)-sensitive family.

3.6.3 Amplifying a Locality-Sensitive Family:

This part mainly introduces the two operations of the function OR-construction and AND-construction.

Consists of r members in the locality-sensitive family F to build F'. So, we get (d1, d2, p1, p2) -sensitive family F and the set {f1, f2, ..., fr} in F'.

OR-construction make the new F' into (d1, d2, 1 – (1 – $p1)r$,1 – (1 $-p2r$)-sensitive, The OR-construction increases the probability, but if choose a suitable r, the F' can make the upper bound probability close to1, while the lower bound probability mostly has no effect.

AND-construction make the new F' into (d1, d2, $p1r$, $p2r$)-sensitive, The AND-construction reduces the probability, but if choose a suitable r, the F' can make the lower limit probability close to 0, while the upper limit probability mostly has no effect.

Compose constructions:
AND- and OR-construction can be used in combination, The advantage of this is that it can make p1 closer to 1 and p2 closer to 0.

The probability after AND-construction and OR-construction is1 $- (1 - pr)r$.

If consists of 4 members in the locality-sensitive family, then we have
Figure 3.1:

p	$1-(1-p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

Figure 3.11: Effect of the 4-way AND-construction followed by the 4-way OR-construction

The probability after OR-construction and AND-construction is$(1 - (1 - pr))r$.

Also, if consists of 4 members in the locality-sensitive family, then we have Figure 3.12

p	$\left(1-(1-p)^4\right)^4$
0.1	0.0140
0.2	0.1215
0.3	0.3334
0.4	0.5740
0.5	0.7725
0.6	0.9015
0.7	0.9680
0.8	0.9936

Figure 3.12: Effect of the 4-way OR-construction followed by the 4-way AND-construction

3.7 LSH Families for Other Distance Measures:

This chapter covers other distance measures like Jaccard distance: Hamming Distance, Random Hyperplanes and the Cosine Distance, Euclidean Distance and Euclidean Spaces.

3.7.1 LSH Families for Hamming Distance:

Definition of Hamming distance: the minimum number of replacements required to change one of the two equal-length strings s1 and s2 into the other The h (x, y) is the hamming distance between x, y. When x and y are the same at the i position of the vector fi(x)=fi(y) the probability of fi(x)=fi(y) is $1 - h(x, y)/d$ when d1<d2 the family of function is $(d1, d2, 1 - d1/d, 1 - d2/d)$ −sensitive.

The difference between Hamming distance and Jaccard is value range and size

1. The value range of Hamming is 0 to d-dimensional vectors, the Jaccard distance is between 0 to 1.
2. The size of Hamming distance function family is d, the minhash function have unlimited supply.
   The second difference will affect the s-curve, Because the size of b will directly affect the number of functions.

3.7.2 Random Hyperplanes and the Cosine Distance:

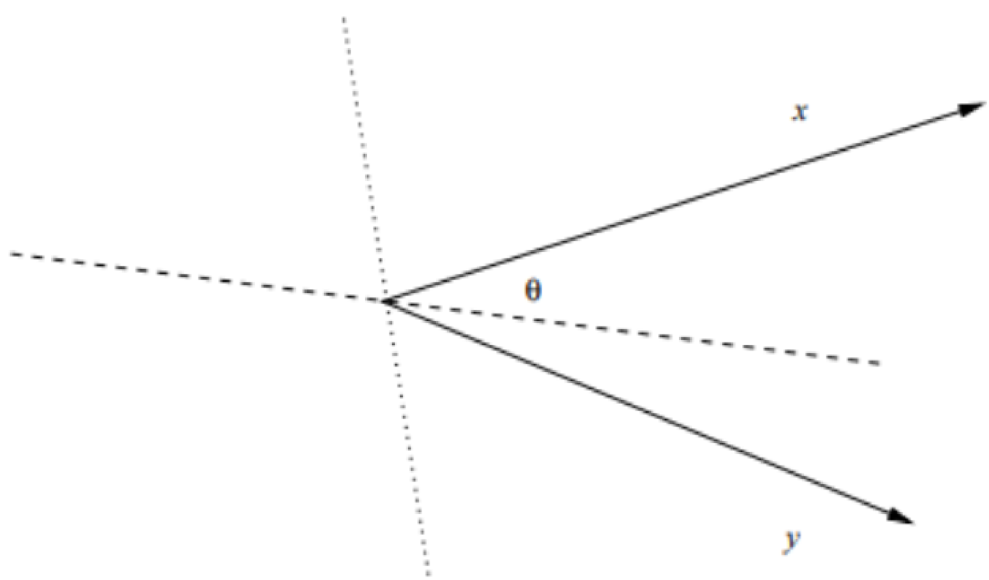


Figure 3.13: Two vectors make an angle θ

The cosine distance is the angle θ between the cosine distances of two vectors. The smaller the θ, the more similar the two vectors are. The hyperplane intersects the x and y planes and there is also a random vector v, the hyperplane is the set of points whose dot product with v is 0. Determining whether x and y are similar is by comparing the positive and negative of the inner product of two vectors x, y of similarity and a random vector v. The probability of f(x)= f(y) which means v.x and v.y have same sign which should be (180-θ)/180.

Calculate the cosine angle is:

inner product of x and y / (L2-norm of x * L2-norm of y),
Then apply the arc cos function to convert the result to an angle between
0 and 180 degrees

3.7.3 Sketches:

Euclidean distance is to calculate the probability that two vectors are in a bucket, where if d is the distance between two points and a is the width of the bucket.

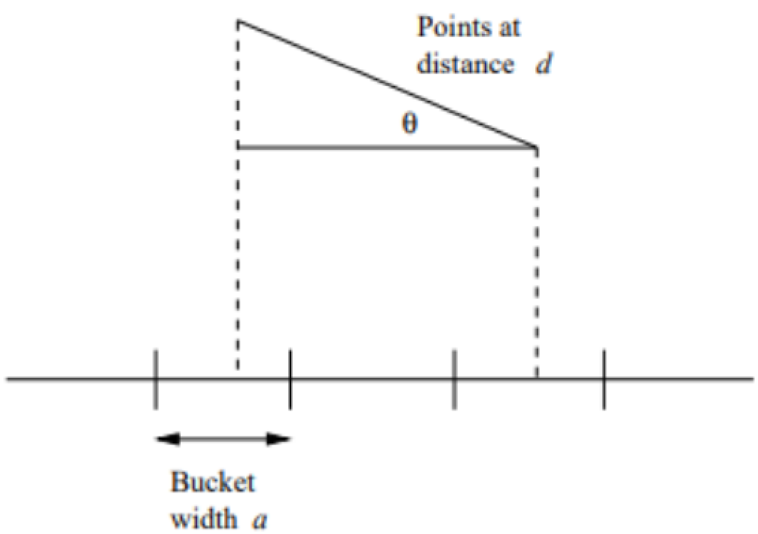


Figure 3.14: Two points at distance $d \gg a$ have a small chance of being hashed to the same bucket

if d=a/2 then there is at least a 50% chance that the two points will be in the same bucket and the closer the angle of θ is to 90 degrees, the higher the probability of being in the same bucket, if it is 90 degrees, then it must be in the same bucket. If d>>a then for the two points to be in the same bucket, θ must be close to 90 degrees. If d≥2a, the probability the two point in the same bucket are less than 1/3 So, the function family of Euclidean Distance should be (a/2, 2a , ½, 1/3) - sensitive.

3.7.5 More LSH Families for Euclidean Spaces:

The limitation of Euclidean distance is that the family function in 3.7.4 can only be in two-dimensional Euclidean space and need a stronger condition d1< 4d2. For any distance pair and any dimension of space satisfying d1<d2, there exists a family of hash functions with (d1, d2, p1, p2)-sensitive, we can know the p1 must more than the two point of d2 in the same bucket p2, but p1 and p2 are not easy to calculate if it does not have the stronger condition.