q1

1. a combiner class and a reducer class can be used interchangeably when the problems are commutative and associative.

max(0, 20, 10, 25, 15) = max(max(0, 20, 10), max(25, 15)) = max(20, 25) = 25

Commutative: max(a, b) = max(b, a)

Associative: max (max(a, b), c) = max(a, max(b, c))

1. it is lazy evaluated. the data is not available or transformed until an action is executed that triggers the execution. reduce is action function

q2

class Mapper

method map(self, \_, \_):

emit(“departmentID,employeeID”, “salary”)

class Reducer

method reduce\_init(self):

current\_departmentID = “”

current\_dic = {}

method reduce(key, value):

departmentID, employeeID = key.split(“,”)

if departmentID != current\_departmentID:

if current\_departmentID != “”:

maximum\_salary = max(current\_dic.values())

for key1, value1 in current\_dic.item():

if (value1 == maximum\_salary):

emit(current\_departmentID, key1)

current\_dic = {}

current\_dic[employeeID] = value

current\_departmentID = departmentID

else:

current\_dic[employeeID] = value

method reduce\_final(self):

maximum\_salary = max(current\_dic.values())

for key1, value1 in current\_dic.item():

if (value1 == maximum\_salary):

emit(current\_departmentID, key1)

in JOBCONF, configure:

'mapreduce.map.output.key.field.separator': ',',

'mapreduce.partition.keypartitioner.options':'-k1,1',

'mapreduce.partition.keycomparator.options':'-k1,1



class Mapper

method map(self, key, list\_vertex):

for item in list\_vertex:

if key < item:

emit(“key,item”, “”)

else:

emit(“item,key”, “”)

class Reducer

method reduce\_init(self):

key\_dic = {}

method reduce(key, value):

vertex1, vertex2 = key.split(“,”)

if key not in key\_dic:

key\_dic[key] = []

for key1,value1 in key\_dic:

if key1.split(“,”)[0] == vertex1 or key1.split(“,”)[1] == vertex1:

value1.append(key)

else if key1.split(“,”)[0] == vertex2 or key1.split(“,”)[1] == vertex2:

value1.append(key)

method reduce\_final(self):

for key1,value1 in key\_dic:

value1 = list(set(value))

sorted(key\_dic)

for key1,value1 in key\_dic:

print(“key1: “ + “, ”.join(value1))

q3



see problemA.py



see problem.py

q4

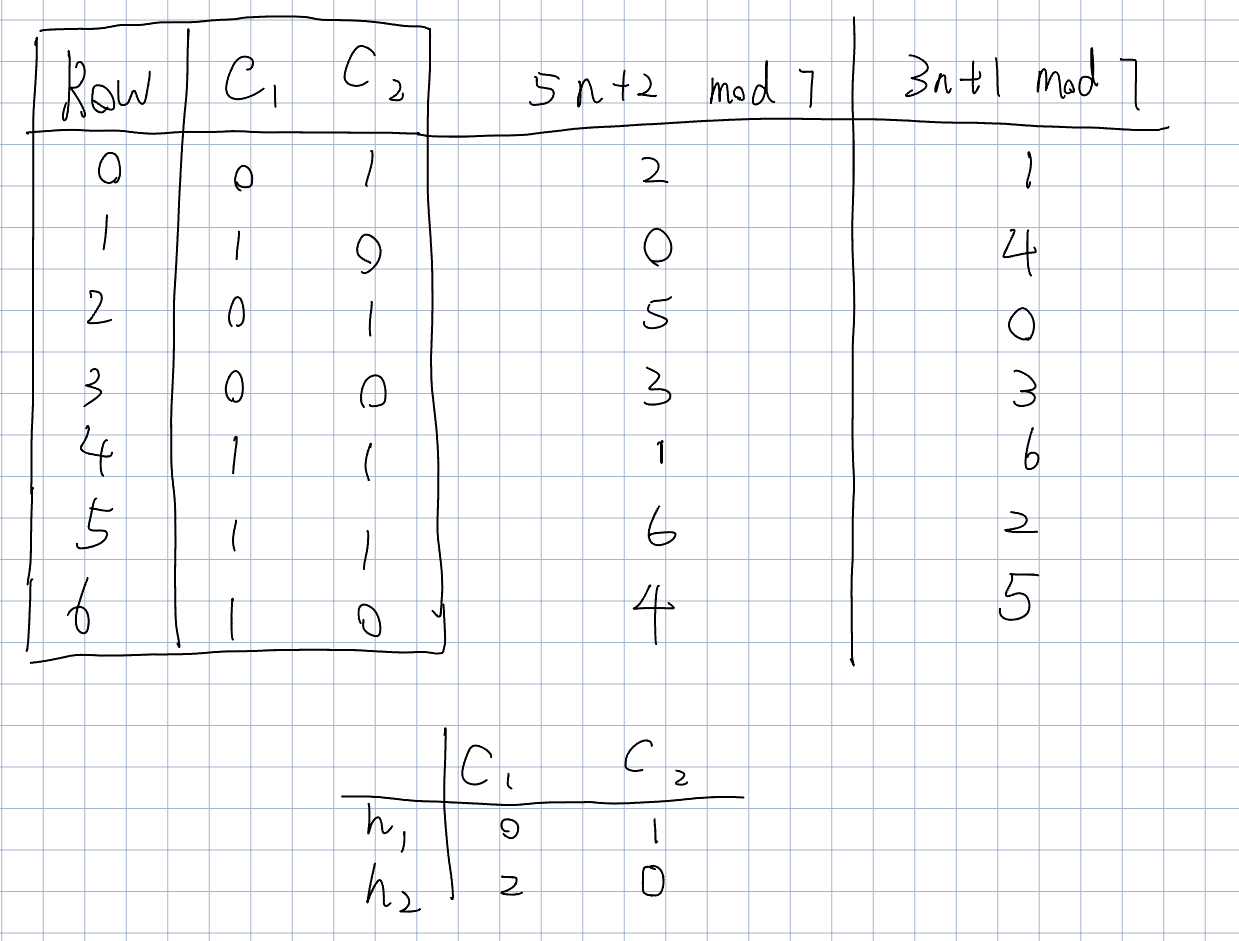
1. set of 2-shingles:

S(A) = ['th', 'he', 'e ', ' s', 'sk', 'ky', 'y ', ' i', 'is', 's ', ' d', 'da', 'ar', 'rk', 'k ', ' t', 'th', 'he', 'e ', ' m', 'mo', 'oo', 'on', 'n ', ' i', 'is', 's ', ' b', 'br', 'ri', 'ig', 'gh', 'ht']

S(B) = ['th', 'he', 'e ', ' m', 'mo', 'oo', 'on', 'n ', ' i', 'in', 'n ', ' t', 'th', 'he', 'e ', ' s', 'sk', 'ky', 'y ', ' i', 'is', 's ', ' b', 'br', 'ri', 'ig', 'gh', 'ht']

number of intersecting element = 22

jaccard similarity = 0.79

1. 

q5



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |

(ii)

“sql”is contained



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| H0 | 1 | 1 | 1 | 3 | 0 |
| H1 | 0 | 1 | 3 | 1 | 1 |
| H2 | 0 | 1 | 1 | 4 | 0 |

use the built CM-sketch to get the count for word “data”: 3

q6