# Maharishi University of Management
## DEPARTMENT OF COMPUTER SCIENCE
### CS 522 Big Data
### Midterm Exam

10/10

**[4 Points, Qn. 1]** Define an array to be nice if **whenever it contains 4, it does NOT contains 3.** Examples:

[7, 6, 2, 3, 1] //nice. 4 is not there.
[7, 6, 2, 4, 1] // nice. 4 is there, but 3 is not there
[3, 6, 2, 3, 4] //NOT nice. 4 is there and 3 is also there.
[3, 4, 2, 3, 4, 7, 4] //NOT nice. 4 is there and 3 is also there.
[1, 6, 2, 8, 2, 9] //nice. 4 is not there.

if (a & 3) not nice

Explain how map and fold (of functional programming) can be used to determine whether or not **an array is nice.** That is, draw state diagram (2 points) write pseudo-code for functions f and g, specify initial value and interpretation for final value. (2 points)

**[ 6 Points Qn. 2]**

Q1. Write an in-mapper combiner **algorithm** modifying **stripe approach for co-occurrence matrix.**

Q2. Assume that there are two Input-splits and two reducers. Note that Input-split 1 and Reducer 1 are on the same machine. Input-split 2 and Reducer 2 are on the same machine.

Further, let the partitioner assign all words less than letter 'k' to Reducer 1 and everything else to Reducer 2. Two input-splits are:

Input-Split 1 : [{cat mat rat cat mat}{cat bat mat bat rat}]

Input-Split 2 : [{bat rat bat mat rat}{cat rat bat rat bat}]

Let the **neighborhood of X, N(X)** be **next two terms.**

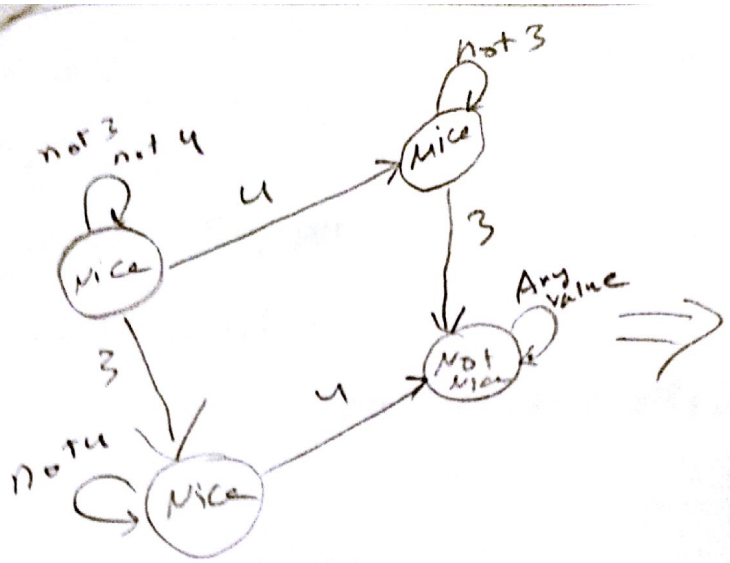
Example: Let Data block be [a b c a d e]

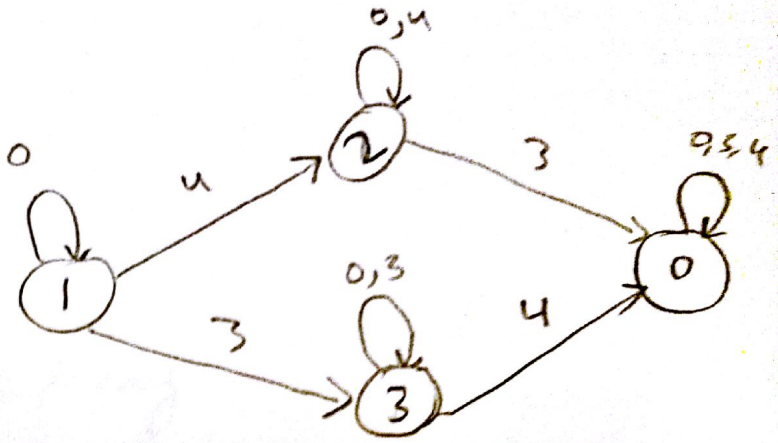
N(a) = {b, c}, N(b) = {c, a}, N(c) = {a, d}, N(a) ={d, e}, N(d) = {e}, N(e) = {}.

a. Illustrate stripe approach to compute co-occurrence matrix (with no combiner, **no** in-mapper combining).

b. Illustrate stripe approach to compute co-occurrence matrix with in-mapper combining. (That is, apply your algorithm Q1).

*Note: Illustrate an algorithm means, show mapper outputs, show reducer inputs and reducer outputs.*




```
int f (x)
    if (x=4 || x=3)
        return x
    else
        return 0
```

4

```
nt  g (x, y)
    if (x=0)
      return x;

    if (x=3 && y=4)
      return 0;
    else if (x=2 && y=3)
      return 0;
    else if (x=1 && y=4)
      return 2;
    else if (x=1 && y=3)
      return 3;
    return x;
```

initial
x=1

1, 2, 3 ⟶ Nice

0 ⟶ Not Nice

art #2

Q1)

```
Class Mapper

    method Initialize
        H = new Associative Array

    method map (docId id, doc d)
        for each w in d do

            for each t in neighborhood of (w) do
                Hw = H{w}
                if (Hw = Null)
                    Hw = new Associative Array
                    H{w} = Hw
                if (Hw{t} = NULL)
                    Hw{t} = 1
                else
                    Hw{t}++;

    method close

        for each string w in H do
            emit (string w, stripe H{w})
```

Part 2

Q1)

Class Reduce

```
method Reduce(w, stripes [H1, H2 ----])
    Ht = new Associative Array
    for each stripe H in [H1, H2 ---] do
        Sum(Ht, H)

    emit (string w, stripe Ht)
```

Q2) b)

| cat mat rat cat mat<br>cat bat mat bat rat | bat rat bat mat rat<br>cat rat bat rat bat | Input Splits |
|---|---|---|
| (cat, mat rat bat: 3 1 1)<br>(mat, rat cat bat: 2 1 1)<br>(rat, cat mat: 1 1)<br>(bat, mat bat rat: 1 1 1) | (bat, rat bat mat: 3 2 1)<br>(rat, bat mat rat: 3 1 1)<br>(mat, rat: 1)<br>(cat, rat bat: 1 1) | Mapper Output |
| (bat, [mat bat rat: 1 1 1, rat bat mat: 3 2 1])<br>(cat, [mat rat bat: 3 1 1, rat bat: 1 1]) | (mat, [rat cat bat: 2 1 1, rat: 1])<br>(rat, [rat mat: 1 1, bat mat rat: 3 1 1]) | Reducer Input |
| (bat, mat bat rat: 2 3 4)<br>(cat, mat rat bat: 3 2 2) | (mat, rat cat bat: 3 1 1)<br>(rat, cat mat bat rat: 1 2 3 1) | Reducer output |