**Main()**

{

readInPhosphMatrix():

read in phosphorylation matrix

initialize activation matrix with random 0/1

combine activation matrix and phosphorylation matrix and save into combinedMatrix

readInIntervMatrix():

read in intervention matrix and save into intervMatrix

readInEdges():

read in edge information, save to edgeList

buildNetwork():

create nodeList

add parent and children to each node

calculateCPTforEachNode()

foreach node

node.calculateCPT ( combinedMatrix )

while true:

calculateJointProbForAllNodes ()

if converge : difference of jointProb < 0.001 or number of iterations > 1000

output combinedMatrix, activationMatrix

break

else

inferActivation()

}

-------------------------------------------------------------------------------------------------------------------------------------------

**Data Structure:**

**Class Node:**

int index

string name

char type

int[] children

int[] parents

double[] CPT1 : only save P(A=1|Pa), P(A=0|Pa)=1-P(A=1|Pa)

calculateCPT(combinedMatrix)

**Class Data:**

int[] interventMatrix

int[] combinedMatrix ( activation + phosphorylation )

string[] edgeList

Node[] nodeList

inferActivation()

calculateJointProbForAllNodes()

**------------------------------------------------------------------------------------------------------------------**

**Node::calculateCPT (combinedMatrix)**

{

foreach case

get the node and its parent value in combinedMatrix

count the combination of those column values:

N = numOfparents

If NodeValue ==0

Count0 [ ] ++

else

count1 [ ] ++

CPT1 = count1 / (count0 + count1)

}

**Data::inferActivation()**

{

Foreach case

Foreach activationNode A

If ( intervention A is 1)

Set inferA to 1

elseif ( intervention A is -1)

Set inferA to 0

else infer:

get the parent of A

find the value of A and A’s parent in combinedMatrix

lookup the CPT value of A : CPT\_A for A=1, A=0

get the children of A

Foreach children C

get the parent of C

find the value of C and C’s parent in combinedMatrix

lookup the CPT value of C : CPT\_C for A=0, A=1

inferA = 1 / { 1 + exp [ log ( P(A=0)/P(A=1) ) ] }

while log[P(A=0)] = log [CPT\_A(A=0)] + ∑ log [CPT\_C (A=0)]

log[P(A=1)] = log [CPT\_A(A=1)] + ∑ log [CPT\_C (A=1)]

log ( P(A=0)/P(A=1) ) = log[P(A=0)] - log[P(A=1)]

sample a random number r

if r > inferA set action of A to 0

else set A to 1

update A’s CPT

update A children’s CPT

}

**Data::calculateJointProbForAllNodes()**

{

foreach case

foreach protein

∑ log P(A|Pa)

}

Note: In code implementation, readinMatrices, readinEdges and buildNetwork are implemented in the constructor of Data class. In the main() code, it only creates an object of Data class.