// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Project #4 functions to add  
// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
public int outdegree(int K)  
{  
 // Number of arrows FROM node K of digraph  
 // Nodes are numbered 0,1,2,...,SIZE-1  
 int rowsum = 0;  
 for(int r = 0; r < SIZE; r++)  
 rowsum = rowsum + M[K][r];  
 return rowsum;  
}  
  
public BMat meet(BMat M2)  
{  
 BMat temp = new BMat(SIZE);  
  
 // join the member variable M with M2  
 for(int i=0; i<SIZE; i++)  
 {  
 for(int j=0; j<SIZE; j++)  
 {  
 if(this.M[i][j] == 1 && M2.M[i][j] == 1)  
 {  
 temp.M[i][j] = 1;  
 }  
 else  
 {  
 temp.M[i][j] = 0;  
 }  
 }  
 }  
 return temp;  
}  
  
public BMat transpose()  
{   
 BMat transposed = new BMat(SIZE);  
  
 for(int i=0; i<SIZE; i++)  
 {  
 for(int j=0; j<SIZE; j++)  
 {   
 transposed.M[i][j] = this.M[j][i];  
 }  
 }  
  
 return transposed;  
}  
  
public BMat product(BMat M2)  
{  
 // logical AND  
 BMat BMatProduct = new BMat(SIZE);  
 for(int i=0; i<SIZE; i++)  
 {  
 for(int j=0; j<SIZE; j++)  
 {  
 BMatProduct.M[i][j] = this.M[i][j] \* M2.M[j][i];  
 }  
 }  
 return BMatProduct;  
}  
  
public BMat tclosure()  
{  
 BMat Work = new BMat(this.M);  
 BMat T = new BMat(SIZE);  
  
 int[] row = new int[SIZE];  
 int[] col = new int[SIZE];  
  
 for(int i=0; i<SIZE; i++)  
 {  
 // set row and columns to work with  
 for(int j=0; j<SIZE; j++)  
 {  
 row[j] = Work.M[i][j];  
 }  
 for(int j=0; j<SIZE; j++)  
 {  
 col[j] = Work.M[j][i];  
 }  
  
 // build T.M  
 for(int k=0; k<SIZE; k++)  
 {  
 for(int m=0; m<SIZE; m++)  
 {  
 T.M[m][k] = row[k] \* col[m];  
 }  
 }  
  
 // W = W v T  
 // join the member variable M with M2  
 for(int p=0; p<SIZE; p++)  
 {  
 for(int q=0; q<SIZE; q++)  
 {  
 if(T.M[p][q] == 1 || Work.M[p][q] == 1)  
 {  
 Work.M[p][q] = 1;  
 }  
 else  
 {  
 Work.M[p][q] = 0;  
 }  
 }  
 }  
 }  
 return Work;  
}  
  
public int rootnode()  
{  
 int[] col = new int[SIZE];  
 int index = 0;  
 int count = 0;  
 boolean colZero;  
  
 for(int i=0; i<SIZE; i++)  
 {  
 colZero = true;  
  
 // fill col array with i M collumn  
 for(int j=0; j<SIZE; j++)  
 {  
 col[j] = M[j][i];  
 }  
  
 // check for any 1s  
 for(int j=0; j<SIZE; j++)  
 {  
 if(col[j] != 0)  
 {  
 colZero = false;  
 }  
 }  
  
 // if it didn't find any 1s  
 if(colZero == true)  
 {  
 index = i;  
 count++;  
 }  
 }  
  
 // if it found more than one root candidate  
 if(count != 1)  
 {  
 return -1;  
 }  
 else  
 {  
 return index;  
 }  
}

// Test program on next page

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Project #4 test program

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
public class P4Test  
{  
 public static void main(String[] args)  
 {  
 // Boolean matrix definitions  
 int A[][] = new int[][]  
 {{1, 1, 0, 0, 1},  
 {1, 0, 1, 0, 0},  
 {0, 0, 0, 0, 0},  
 {1, 0, 0, 0, 0},  
 {0, 0, 1, 0, 1}};  
  
 int B[][] = new int[][]  
 {{0, 1, 0, 0, 1},  
 {0, 1, 1, 0, 0},  
 {1, 0, 1, 0, 0},  
 {1, 0, 0, 0, 0},  
 {0, 1, 0, 0, 1}};  
  
 int C[][] = new int[][]  
 {{0, 1, 0, 0, 0},  
 {0, 0, 1, 0, 0},  
 {0, 0, 0, 1, 0},  
 {1, 0, 0, 0, 1},  
 {0, 1, 0, 0, 0}};  
  
 int D[][] = new int[][]  
 {{1, 1, 0, 0, 0, 0},  
 {1, 1, 1, 0, 0, 0},  
 {0, 1, 1, 1, 0, 0},  
 {0, 0, 1, 1, 0, 0},  
 {0, 0, 0, 0, 0, 1},  
 {0, 0, 0, 0, 1, 1}};  
  
 int E[][] = new int[][]  
 {{0, 1, 1, 0, 0, 1},  
 {0, 1, 1, 0, 0, 1},  
 {0, 0, 1, 0, 0, 1},  
 {0, 0, 0, 0, 1, 1},  
 {0, 0, 0, 1, 1, 1},  
 {0, 0, 0, 0, 0, 0}};  
  
 int F[][] = new int[][]  
 {{0, 0, 0, 0, 1, 0, 1, 0, 0},  
 {1, 0, 0, 1, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 1, 0, 0, 0, 0, 1, 1},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 1, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0}};  
  
 int G[][] = new int[][]  
 {{0, 0, 0, 1, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {1, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 1, 0, 0, 1, 0, 0, 0},  
 {0, 1, 0, 0, 0, 0, 1, 0, 1},  
 {0, 0, 0, 0, 0, 0, 0, 1, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0}};  
  
 BMat BMA = new BMat(A);  
 BMat BMB = new BMat(B);  
 BMat BMC = new BMat(C);  
 BMat BMD = new BMat(D);  
 BMat BME = new BMat(E);  
 BMat BMF = new BMat(F);  
 BMat BMG = new BMat(G);  
   
 //////////////////////////////  
 // part a //  
 //////////////////////////////  
 BMat firstPriority = new BMat(5);  
 BMat secondPriority = new BMat(5);  
 BMat thirdPriority = new BMat(5);  
 BMat fourthPriority = new BMat(5);  
 BMat fifthPriority = new BMat(5);  
   
 System.out.println("a: W = (C' ^ (A v B)) ^ B'");  
   
 // (A v B)  
 firstPriority = BMA.join(BMB);  
   
 // C'  
 secondPriority = BMC.complement();  
   
 // (C' ^ (A v B))  
 thirdPriority = secondPriority.meet(firstPriority);  
   
 // B'  
 fourthPriority = BMB.complement();  
   
 // (C' ^ (A v B)) ^ B'  
 fifthPriority = thirdPriority.meet(fourthPriority);  
 fifthPriority.show();  
   
   
   
   
 //////////////////////////////  
 // part b //  
 //////////////////////////////   
   
 firstPriority = new BMat(5);   
 secondPriority = new BMat(5);  
 thirdPriority = new BMat(5);  
 fourthPriority = new BMat(5);  
 fifthPriority = new BMat(5);  
   
 System.out.println("\nb: W = (B^T O B) ^ (C v C^T)");  
   
 // B^T  
 firstPriority = BMB.transpose();  
   
 // (B^T O B)  
 secondPriority = firstPriority.product(BMB);  
   
 // C^T  
 thirdPriority = BMC.transpose();  
   
 // (C v C^T)  
 fourthPriority = BMC.join(thirdPriority);  
   
 // (B^T O B) ^ (C v C^T)  
 fifthPriority = secondPriority.meet(fourthPriority);  
   
 fifthPriority.show();  
   
   
   
 //////////////////////////////  
 // part C //  
 //////////////////////////////   
   
 firstPriority = new BMat(5);   
 secondPriority = new BMat(5);  
 thirdPriority = new BMat(5);  
 fourthPriority = new BMat(5);  
 fifthPriority = new BMat(5);  
   
   
 System.out.println("\nc: W = OC^18");  
 firstPriority = BMC.product(BMC); // 2  
   
 for(int i=0; i<16; i++)  
 {  
 firstPriority = firstPriority.product(firstPriority);  
 }  
 firstPriority.show();  
   
 //////////////////////////////  
 // part D //  
 //////////////////////////////   
   
 firstPriority = new BMat(6);   
 secondPriority = new BMat(6);  
 thirdPriority = new BMat(6);  
 fourthPriority = new BMat(6);  
 fifthPriority = new BMat(6);  
  
 System.out.println("\nd: W = (D v E)T ^ (DT v ET)");  
   
 // (D v E)  
 firstPriority = BMD.join(BME);  
   
 // (D v E)T  
 firstPriority = firstPriority.transpose();  
   
 // DT  
 secondPriority = BMD.transpose();  
   
 // ET  
 thirdPriority = BME.transpose();  
   
 // (DT v ET)  
 fourthPriority = secondPriority.join(thirdPriority);  
   
 // (D v E)T ^ (DT v ET)  
 fifthPriority = firstPriority.meet(fourthPriority);  
   
 fifthPriority.show();  
   
   
 //////////////////////////////  
 // part E //  
 //////////////////////////////   
   
 firstPriority = new BMat(6);   
 secondPriority = new BMat(6);  
 thirdPriority = new BMat(6);  
 fourthPriority = new BMat(6);  
 fifthPriority = new BMat(6);  
 BMat sixthPriority = new BMat(6);  
 BMat seventhPriority = new BMat(6);  
   
 System.out.println("\ne: W = OD1 v OD2 v OD3 v OD4");  
   
 // OD1 = D O D  
 firstPriority = BMD.product(BMD);  
   
 // OD2 = D O D O D  
 secondPriority = BMD.product(BMD);  
 secondPriority = secondPriority.product(BMD);  
   
 // OD3 = D O D O D O D  
 thirdPriority = BMD.product(BMD);  
 thirdPriority = thirdPriority.product(BMD);  
 thirdPriority = thirdPriority.product(BMD);  
   
 // OD4 = D O D O D O D O D  
 fourthPriority = BMD.product(BMD);  
 fourthPriority = fourthPriority.product(BMD);  
 fourthPriority = fourthPriority.product(BMD);  
 fourthPriority = fourthPriority.product(BMD);  
   
 // OD1 v OD2  
 fifthPriority = firstPriority.join(secondPriority);  
   
 // OD1 v OD2 v OD3  
 sixthPriority = fifthPriority.join(thirdPriority);  
   
 // OD1 v OD2 v OD3 v OD4  
 seventhPriority = sixthPriority.join(fourthPriority);  
   
 seventhPriority.show();  
   
   
   
   
 //////////////////////////////  
 // part F //  
 //////////////////////////////   
   
 System.out.println("\nf: X = maximum out-degree of all nodes in D");  
 System.out.println(" 0: " + BMD.outdegree(0));  
 System.out.println(" 1: " + BMD.outdegree(1));  
 System.out.println(" 2: " + BMD.outdegree(2));  
 System.out.println(" 3: " + BMD.outdegree(3));  
 System.out.println(" 4: " + BMD.outdegree(4));  
 System.out.println(" 5: " + BMD.outdegree(5));  
  
   
 //////////////////////////////  
 // part G //  
 //////////////////////////////   
   
 System.out.println("\ng: W = symmetric closure of D. Is D symmetric?");  
 firstPriority = new BMat(6);  
   
 firstPriority = BMD.sclosure();  
   
 firstPriority.show();  
 System.out.println(" Yes D is symmetric");  
 System.out.println(" The diagonal does not matter and matrix is mirrored about that axis");   
   
   
   
 //////////////////////////////  
 // part H //  
 //////////////////////////////   
   
 System.out.println("\nh: W = transitive closure of E. Is E transitive?");  
 firstPriority = new BMat(6);  
 firstPriority = BME.tclosure();  
 firstPriority.show();  
 System.out.println(" I think it is transitive, I drew out the digraph and it looks like every relation fits,");  
 System.out.println(" but I'm not sure if that means the entire matrix is transitive,");  
 System.out.println(" It was never explained in the videos and I can't seem to find an answer");  
   
 //////////////////////////////  
 // part I //  
 //////////////////////////////  
   
 System.out.println("\ni: Show that matrix F represents a tree");  
   
 if(BMF.rootnode() == -1)  
 {  
 System.out.println(" Matrix F does not have a root node");  
 BMF.show();  
 }  
 else  
 {  
 System.out.println(" Matrix F has only 1 root node candidate\n which is node " + firstPriority.rootnode());   
 System.out.println(" v");  
 BMF.show();  
 System.out.println(" ^");  
   
 for(int i=0; i<9; i++)  
 {  
 System.out.println("Node " + i + " is out-degree " + BMF.outdegree(i) + " and in-degree " + BMF.indegree(i));  
 }  
   
 }  
   
 /////////////////////////////  
 // part J //  
 /////////////////////////////  
   
 System.out.println("\nj: Show that matrix G does not represent a tree");  
   
 if(BMG.rootnode() == -1)  
 {  
 System.out.println(" Matrix G does not have a root node");  
 BMG.show();  
 }  
 else  
 {  
 System.out.println(" Matrix G has only 1 root node candidate\n which is node " + BMG.rootnode());   
 System.out.println(" v");  
 BMF.show();  
 System.out.println(" ^");  
   
 for(int i=0; i<9; i++)  
 {  
 System.out.println("Node " + i + " is out-degree " + BMG.outdegree(i) + " and in-degree " + BMG.indegree(i));  
 }  
 }   
 } // end main  
} // end class

// outputs on next page

Output from test program:

(Copied from jgrasp)

§ a: W = (C' ^ (A v B)) ^ B'  
ÏÏ§Ï 1 0 0 0 0  
ÏÏ§Ï 1 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 1 0 0  
ÏÏ§Ï  
ÏÏ§Ïb: W = (B^T O B) ^ (C v C^T)  
ÏÏ§Ï 0 0 0 1 0  
ÏÏ§Ï 1 0 0 0 1  
ÏÏ§Ï 0 1 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï  
ÏÏ§Ïc: W = OC^18  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0  
ÏÏ§Ï  
ÏÏ§Ïd: W = (D v E)T ^ (DT v ET)  
ÏÏ§Ï 1 1 0 0 0 0  
ÏÏ§Ï 1 1 1 0 0 0  
ÏÏ§Ï 1 1 1 1 0 0  
ÏÏ§Ï 0 0 1 1 1 0  
ÏÏ§Ï 0 0 0 1 1 1  
ÏÏ§Ï 1 1 1 1 1 1  
ÏÏ§Ï  
ÏÏ§Ïe: W = OD1 v OD2 v OD3 v OD4  
ÏÏ§Ï 1 1 0 0 0 0  
ÏÏ§Ï 1 1 1 0 0 0  
ÏÏ§Ï 0 1 1 1 0 0  
ÏÏ§Ï 0 0 1 1 0 0  
ÏÏ§Ï 0 0 0 0 0 1  
ÏÏ§Ï 0 0 0 0 1 1  
ÏÏ§Ï  
ÏÏ§Ïf: X = maximum out-degree of all nodes in D  
ÏÏ§Ï 0: 2  
ÏÏ§Ï 1: 3  
ÏÏ§Ï 2: 3  
ÏÏ§Ï 3: 2  
ÏÏ§Ï 4: 1  
ÏÏ§Ï 5: 2  
ÏÏ§Ï

§  
ÏÏ§Ïg: W = symmetric closure of D. Is D symmetric?  
ÏÏ§Ï 1 1 0 0 0 0  
ÏÏ§Ï 1 1 1 0 0 0  
ÏÏ§Ï 0 1 1 1 0 0  
ÏÏ§Ï 0 0 1 1 0 0  
ÏÏ§Ï 0 0 0 0 0 1  
ÏÏ§Ï 0 0 0 0 1 1  
ÏÏ§Ï Yes D is symmetric  
ÏÏ§Ï The diagonal does not matter and matrix is mirrored about that axis  
ÏÏ§Ï  
ÏÏ§Ïh: W = transitive closure of E. Is E transitive?  
ÏÏ§Ï 0 1 1 0 0 1  
ÏÏ§Ï 0 1 1 0 0 1  
ÏÏ§Ï 0 0 1 0 0 1  
ÏÏ§Ï 0 0 0 1 1 1  
ÏÏ§Ï 0 0 0 1 1 1  
ÏÏ§Ï 0 0 0 0 0 0  
ÏÏ§Ï I think it is transitive, I drew out the digraph and it looks like every relation fits,  
ÏÏ§Ï but I'm not sure if that means the entire matrix is transitive,  
ÏÏ§Ï It was never explained in the videos and I can't seem to find an answer  
ÏÏ§Ï  
ÏÏ§Ïi: Show that matrix F represents a tree  
ÏÏ§Ï Matrix F has only 1 root node candidate  
ÏÏ§Ï which is node 0  
ÏÏ§Ï v  
ÏÏ§Ï 0 0 0 0 1 0 1 0 0  
ÏÏ§Ï 1 0 0 1 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 1 0 0 0 0 1 1  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 1 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï ^  
ÏÏ§ÏNode 0 is out-degree 2 and in-degree 1  
ÏÏ§ÏNode 1 is out-degree 2 and in-degree 1  
ÏÏ§ÏNode 2 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 3 is out-degree 3 and in-degree 1  
ÏÏ§ÏNode 4 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 5 is out-degree 1 and in-degree 0  
ÏÏ§ÏNode 6 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 7 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 8 is out-degree 0 and in-degree 1  
ÏÏ§Ï

§

§  
ÏÏ§Ïj: Show that matrix G does not represent a tree  
ÏÏ§Ï Matrix G has only 1 root node candidate  
ÏÏ§Ï which is node 4  
ÏÏ§Ï v  
ÏÏ§Ï 0 0 0 0 1 0 1 0 0  
ÏÏ§Ï 1 0 0 1 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 1 0 0 0 0 1 1  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 1 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï 0 0 0 0 0 0 0 0 0  
ÏÏ§Ï ^  
ÏÏ§ÏNode 0 is out-degree 1 and in-degree 1  
ÏÏ§ÏNode 1 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 2 is out-degree 1 and in-degree 1  
ÏÏ§ÏNode 3 is out-degree 2 and in-degree 1  
ÏÏ§ÏNode 4 is out-degree 3 and in-degree 0  
ÏÏ§ÏNode 5 is out-degree 1 and in-degree 1  
ÏÏ§ÏNode 6 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 7 is out-degree 0 and in-degree 1  
ÏÏ§ÏNode 8 is out-degree 0 and in-degree 1

Digraphs on next page









