PART 1

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
program ReadPractice
implicit none
integer, dimension(8,8) :: array1,array2,array3
character(len=8):: Names="ABCDEFGH" !declare unique node names
character,dimension(8)::Unique_Node_Names !initalize an array of unique node names
integer::i,counter=0,t=1,j,d=0,counter1,g,p,counter2=0
logical flag1
flag1=.TRUE.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

open(12,
)
read(12,*) array1
open(13,file="testdata1.dat",form="formatted",status="old", action="read")
read(13,*) array2 !creating a new array 2, with all the same values as array1.
                !really no easy way to do this in fortran, so did it the simplist way i know. fuck it.

open(unit=78,file="testoutput_kuhn_alkhairo_stylianou.dat",status="old")
write(78,*)'ADJACENCY MATRIX'
call printMatrix(array1,8,8)

write(78,*)" "


write(78,*)'ADJACENCY MATRIX MULTIPLIED BY ITSELF 5 TIMES'
do i =1,4
    array1= MATMUL(array1,array2) !multiplying the array by itself.
end do
call printMatrix(array1,8,8)
write(78,*)

!pre populating array3 with zeros
do 133 i=1,8
    do 134 j=1,8
        array3(i,j) = 0
    134 continue
133 continue
```

```
!call printMatrix(array3,8,8)
!print *, LEN(str(array1))

do while (flag1)
      !print *, '#####################################################'
      !call printMatrix(array3,8,8)
      array1=MATMUL(array1,array2)
      d = d + 1
      counter1 = 0
      do 43 i = 1,8
            do 57 j=1,8
                  if (array1(i,j).GE.1) then
                        array3(i,j)=1
                  endif
            57 continue
      43 continue
      do 555 g=1, 8
            do 888 p=1,8
                  if(array3(g,p).eq.1) then
                        counter1 = counter1 + 1
                  endif
                  if(counter1.eq.64) then
                        !print *,"COUTNER: ", counter1
                        !call printMatrix(Array3,8,8)
                        flag1 = .FALSE.
                  endif
            888 continue
      555 continue
end do
write(78,*)"DIAMETER OF THIS NETWORK ", d + 1

!call printMatrix(array3,8,8)




write(78,*)'UNIQUE NODE NAMES: '
do i =1,len(Names) !iterate through array of unique node names, and print out characters
      Unique_Node_Names(i)=Names(i:i)
      write(78,*) Unique_Node_Names(i)
end do
```

```
close(12)


end program ReadPractice

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine printMatrix(array, n, m) !passing the array with dimensions of 8,and 8

implicit none
integer, intent(in) :: array(n,m)
integer, intent(in) :: n,m
integer :: i, counter,j
character (len=8) :: header = "ABCDEFGH"
counter=0

write(78,*) "        A          B          C          D          E          F          G          H"
do i=1,n
	print *, array(i,:) !print the arrays out
	write(78,*) header(i:i), array(i,:)
end do
write(78,*) "  "
do 43 i= 1,n
	do 57 j=1,m !since array is a two d array, i and j need to be used in order to iterate through
it
		if (array(i,j).eq.1) then ! if either i or j equals 1, count the occurances
			counter = counter + 1
		endif
	57 continue
43 continue

write(78,*)'Number of data pairs: ', counter !print the occurances of 1
write(78,*)'Number of unique Nodes: ', n        !print the number of unique nodes, for us it's 8

close(1)
end subroutine printMatrix
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

OUTPUT FOR PART 1

ADJACENCY MATRIX

	A	B	C	D	E	F	G	H
A	0	0	1	0	0	0	1	0
B	0	0	1	0	1	0	0	0
C	1	1	0	0	0	1	0	0
D	0	0	0	0	1	0	0	0
E	0	1	0	1	0	1	0	1
F	0	0	1	0	1	0	0	0
G	1	0	0	0	0	0	0	1
H	0	0	0	0	1	0	1	0

Number of data pairs: 18
Number of unique Nodes: 8

ADJACENCY MATRIX MULTIPLIED BY ITSELF 5 TIMES

	A	B	C	D	E	F	G	H
A	0	0	21	0	23	0	13	0
B	0	0	29	0	36	0	15	0
C	21	29	0	15	0	29	0	22
D	0	0	15	0	21	0	8	0
E	23	36	0	21	0	36	0	29
F	0	0	29	0	36	0	15	0
G	13	15	0	8	0	15	0	14
H	0	0	22	0	29	0	14	0

Number of data pairs: 0
Number of unique Nodes: 8

DIAMETER OF THIS NETWORK 3
UNIQUE NODE NAMES:
A
B
C
D
E
F
G

H

PART 2 Program

```
program experimental
implicit none
Integer :: a,b,lineCount = 1,i,j, counter =
0,sizearray,counter2=0,x,y,counter3,counter4,r,a1,b1,a_num,b_num,indexcounter = 1
Integer :: Array1(1:3989), Array2(1:3989),UniqueArray(1:4000), UniqueArray2(1:176),
indexarray(1:180)
Integer, Dimension(176,176) :: FinalArray
logical InFile


InFile = .true.


open(unit=88, file="experimental_data.dat", status="old")

do while (InFile)

	read(unit=88,FMT=*) a,b
	if (a == 1) then
		InFile = .false.
	endif
	!print *, a,b
	Array1(linecount) = a
	Array2(linecount) = b

	linecount = linecount + 1
end do

!print *, "COLUMN B ####################################"
!call printMatrix(Array2,1,3989)

do 77 i = 1,3989,1   !Creating An Array of Unique Numbers from Array1 (ColumnA)
	!print *, Array1(i)
	counter = 0
	do 78 j = 1,4000
		if (Array1(i)==UniqueArray(j)) then
			counter = counter + 1
```

```
            endif
        78 continue
        if (counter == 0) then
            UniqueArray(indexcounter) = Array1(i)
            indexcounter = indexcounter + 1
            !print *, UniqueArray(i)
            !indexarray(indexcounter) = i
        endif
77 continue

!call printMatrix(UniqueArray,1,4000)

!print *, "COLUMN A ######################################"
!call printMatrix(Array1,1,3989)

do 1236 i = 1,176,1
        !print *, UniqueArray(i)
        UniqueArray2(i) = UniqueArray(i)
1236 continue

!call printMatrix(UniqueArray2,1,176)

do 65 x =1,176  ! Initializing the Final Array (176x176) with Zeros
        do 54 y=1,176
            FinalArray(x,y) = 0
        54 continue
        !print *, FinalArray(x,:)
65 continue

!Print out the Resulting Initialized Array
!call printMatrix(FinalArray,176,176)



do 108 r=1, 3989,1
        a_num = Array1(r) !first iteration: 10
        b_num = Array2(r) !first iteration: 20
        a1 = returnIndex(UniqueArray, a_num) !Finding the Index of a_num in Our Unique List
using a subroutine
        b1 = returnIndex(UniqueArray, b_num) !Finding the Index of b_num in our Unique List
using a subroutine
        FinalArray(a1,b1) = 1       !Setting that index to 1 in our FinalArray, first iteration (1,2) = 1
108 continue
```

```
call printMatrix(FinalArray,176,176)


!sizearray = size(UniqueArray)
!print *, sizearray

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
CONTAINS
integer function returnIndex(array, num) ! passing UniqueArray and a num to find the index of in
our UniqueArray
implicit none
integer, intent(in) :: array(1:176)
integer, intent(in) :: num
integer :: i, counter=1 ,j

do i=1,181
	if (array(i) == num) then  ! when we find the number we are searching break from counter
loop
		counter = i
		exit
	endif
end do
returnIndex = counter ! returning counter
end function returnIndex

end program experimental

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine printMatrix(array, n, m) !passing the array with dimensions of 8,and 8

implicit none
integer, intent(in) :: array(n,m)
integer, intent(in) :: n,m
integer :: i, counter,j

do i=1,n
	print *, array(i,:) !print the arrays out
end do
end subroutine printMatrix
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!integer function returnIndex(array, num) ! passing UniqueArray and a num to find the index of in
our UniqueArray
!implicit none
```

```
!integer, intent(in) :: array(1:3989)
!integer, intent(in) :: num
!integer :: i, counter=1 ,j
!
!do i=1,176
!      if (array(i) == num) then  ! when we find the number we are searching break from counter
loop
!            counter = i
!            exit
!      endif
!end do
!returnIndex = counter ! returning counter
!end function returnIndex
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

NOTE: After running this first program we will feed the output file to a modified ReadPractice program that is similar to Part 1.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
program ReadPractice
implicit none
integer, dimension(176,176) :: array1,array2,array3
character(len=8):: Names="ABCDEFGH" !declare unique node names
character,dimension(8)::Unique_Node_Names !initalize an array of unique node names
integer::i,counter=0,t=1,l,ll,j,d=0,counter1,g,p,counter2=0
logical flag1
flag1=.TRUE.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

open(56,file="expadjmatrix.dat", form="formatted",status="old", action="read")
read(56,*) array1
!read(56,*) array2
open(69,file="expadjmatrix1.dat",form="formatted",status="old", action="read")
read(69,*) array2 !creating a new array 2, with all the same values as array1.
              !really no easy way to do this in fortran, so did it the simplist way i know. fuck it.

open(unit=78,file="expadj_kuhn_alkhairo_stylianou.dat",status="old")
!call printMatrix(array1,176,176)

write(78,*)" "


!write(78,*)'ADJACENCY MATRIX MULTIPLIED BY ITSELF 5 TIMES'
```

```
!do i =1,4
l=11458
!array1= MATMUL(array1,array2) !multiplying the array by itself.
!end do


call printMatrix(array1,176,176)
write(78,*)

!pre populating array3 with zeros
do 133 i=1,176
      do 134 j=1,176
            array3(i,j) = 0
      134 continue
133 continue

!call printMatrix(array3,8,8)
l=l+1542
!print *, LEN(str(array1))

do while (flag1)

      array1=MATMUL(array1,array2)
      d = d + 1
      counter1 = 0
      do 43 i = 1,176
            do 57 j=1,176
                  if (array1(i,j).GE.1) then
                        array3(i,j)=1
                  endif
            57 continue
      43 continue
      do 555 g=1, 176
            do 888 p=1,176
                  if(array3(g,p).GE.1) then  !EQ
                        counter1 = counter1 + 1
                  endif
                  if(counter1.GE.l) then
                        flag1 = .FALSE.
                  endif
            888 continue
      555 continue
end do
```

```
write(78,*)"DIAMETER OF THIS NETWORK ", d + 1

!call printMatrix(array3,8,8)


close(78)


end program ReadPractice

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine printMatrix(array, n, m) !passing the array with dimensions of 8,and 8

implicit none
integer, intent(in) :: array(n,m)
integer, intent(in) :: n,m
integer :: i, counter=0,j



do i=1,n
	print *, array(i,:) !print the arrays out
end do
write(78,*) "  "
do 43 i= 1,n
	do 57 j=1,m !since array is a two d array, i and j need to be used in order to iterate through
it
		if (array(i,j).eq.1) then ! if either i or j equals 1, count the occurances
			counter = counter + 1
		endif
	57 continue
43 continue

write(78,*)'Number of data pairs: ', counter !print the occurances of 1
write(78,*)'Number of unique Nodes: ', n 	!print the number of unique nodes, for us it's 8

close(1)
end subroutine printMatrix
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

OUTPUT FOR PART 2 PROGAMS.

```
Number of data pairs:        3989
 Number of unique Nodes:          176
```