King Abdulaziz University

Faculty of Engineering

Course ID: EE305

Instructer: Dr. Emad Khalaf

Section: AA

BINARY RELATIONS

Team Members	
Saad Ali Sadaqah Al-Jehani	1935151
Khalid Mamdouh Al-Dahasi	1935129
Mohammed Ba Othman	1936005

DECEMBER 5, 2021

KING ABDULAZIZ UNIVERSITY
Faculty Of Engineering

Table of Contents

Introduction ____ 3

Binary Relations ____ 3

Work ____ 3

- ❖ **Problem definition** ____ 3
- ❖ **Matrix Composition (Boolean Product):** ____ 4
- ❖ **Relation Properties** ____ 5
  - *Reflexive* ____ 5
  - *Irreflexive* ____ 5
  - *Symmetric* ____ 6
  - *Antisymmetric* ____ 6
  - *Asymmetric* ____ 7
  - *Transitive* ____ 8
  - *Equivalence* ____ 8
- ❖ **Matrix Operations** ____ 8
  - *Matrix of Relation R on any given order* ____ 8
  - *Matrix of infinite order* ____ 9
  - *Compliment* ____ 10
  - *Inverse* ____ 10
  - *Union* ____ 11
  - *Intersection* ____ 11
  - *In degrees and outdegrees table* ____ 12
  - *VERT TAIL HEAD NEXT Table* ____ 12

Output ____ 14

Conclusion ____ 20

References: ____ 21

Table of Figures:

Figure 1 PSEUDOCODE of Boolean product 4
Figure 2 PSEODUCODE of Reflexive Relations 5
Figure 3 PSEODUCODE of Irreflexive Relations 5
Figure 4 PSEODUCODE of Symmetric Relations 6
Figure 5 PSEODUCODE of Antisymmetric Relations 7
Figure 6 PSEODUCODE of Asymmetric Relations 7
Figure 7 PSEODUCODE of Transitive Relations 8
Figure 8 PSEODUCODE of Equivalence Relations 8
Figure 9 PSEODUCODE of Relations of different orders 9
Figure 10 PSEODUCODE of Relations of infinite order 9
Figure 11 PSEODUCODE of Compliment Operation 10
Figure 12 PSEODUCODE of Inverse Operation 10
Figure 13 PSEODUCODE of Union Operation 11
Figure 14 PSEODUCODE of Intersection Operation 11
Figure 15 PSEODUCODE of In degrees and Out degrees Table 12
Figure 16 PSEODUCODE of VERT TAIL HEAD NEXT Table (1) 13
Figure 17 PSEODUCODE of VERT TAIL HEAD NEXT Table (2) 13
Figure 18 User Relations Prompt 14
Figure 19 Relations Matrix and Relation Properties and operations 15
Figure 20 output of the inverse 16
Figure 21 output of the compliment 16
Figure 22 output of MR infinite 16
Figure 23 output of the power of relations 16
Figure 24 prompt of Matrix S 17
Figure 25 output of the union of matrix R and S 17
Figure 26 output of the intersect of matrix R and S 18
Figure 27 output of the Boolean product of matrix R and S 19
Figure 28 output of the in degrees and out degrees table 20
Figure 29 output of the VERT table 20
Figure 30 output of the termination of the program 20

Introduction:

Discrete mathematics is the science of finite mathematical structures, such as simple integers structures and logic structures. It mainly deals with finite sets [1]. This type of math science has gathered scientists' and researchers' attention in recent decades due to its many applications in things such as programming languages, networks, cryptography and encoding data, data structures, and many other modern technologies. It's not an overstatement to say that discrete math is the core of all technology that is around us today [2]. Discrete mathematics was there since ancient times. Records of Hindus using permutations on numbered sets suggest evidence of the applications of discrete mathematics in the 6^{th} century. a major part of discrete mathematics, the graph theory was also researched since the 18^{th} century. also, cryptology was mainly used in World War II which led to developments in theoretical computer science which evolved to be the foundations of digital computer science [3]. But this report will be on a certain subject in discrete mathematics, which is Binary Relations.

Binary Relations:

Binary relations are the association between elements of either two or one set. It can be projected as a set consisting of related pairs (x,y) where x is the input or the domain and where y is the output or the range. The notation x R y means that x is related to y by R, where R can be the relation that links x and y. A binary relation is used in many types of mathematics to project a variety of concepts. Such as arithmetic models like the greater than and the divides relations. Or Geometry models like the relation of x is parallel to y. it's not an exaggeration to say that 90% of discrete math is about some type of relations or functions that can be considered a relation. And therefore, it can be said that relations are applicable in most applications where discrete math is. Such as data structures and relational databases. Or simply any application where functions are involved [5]. Binary relation was introduced by De Morgan in 1860 and then developed by Pierce and Schroder to then be further developed to the modern science it now is. Although binary relation was being discussed since the 4^{th} century by Aristotle, De Morgan is the one who introduced the calculus of binary relations. This science then was neglected until 1941 it was revived by Tarski. Then it would be used in most applications of computer science and to model most forms of math.[4]

Work:

- **Problem definition:** The aim of this project is determining the relation properties of sets and they are: reflexive, Irreflexive, symmetric, antisymmetric, asymmetric, transitive and equivalence. Also, the project is aimed to do different operations and they are: finding the relations matrix of different lengths of a path, finding the relation matrix R^{∞}, finding the compliment and inverse of the relations matrix, finding the union and intersect of the relations matrix and an input, finding the composition of the relations matrix and an input matrix, construct the in-degree and out-degree table of the relations matrix, and finally construct the VERT TAIL HEAD NEXT table of the relations matrix. The definitions used are from Kolman, Bernard, Robert C Busby, and Sharon Cutler Ross. 2010. Discrete Mathematical Structures.

- **Matrix Composition (Boolean Product):** if a R b and b R c, then composition of R with itself will have a R c. The program will compare each row of the first matrix with each column in the second matrix, and if $r_{ij} = 1$ and $r_{ji} = 1$. Then the element at the cross section between them will equal 1. The program will use this method to find if the relations matrix is transitive. And to find $M_R \times M_S$ or $M_S \times M_R$, while M_S is an input.

```
DEFINE FUNCTION booleanProd(u, v):

    SET rows TO len(u)

    SET cols TO len(v)

    SET s TO []

    FOR i IN range(rows):

        SET col TO []

        FOR j IN range(cols):

            col.append(0)

        s.append(col)

    FOR i IN range(len(u)):

        s[i].clear()

        FOR j IN range(len(v)):

            SET isEqual TO False

            FOR k IN range(len(u)):

                IF u[i][k] EQUALS 1 and v[k][j] EQUALS 1:

                    SET isEqual TO True

            IF isEqual:

                s[i].append(1)

            IF not isEqual:

                s[i].append(0)

    RETURN s
```

Figure 1 PSEUDOCODE of Boolean product

❖ Relation Properties:

- *Reflexive*: a relation R on a set A is reflexive if (a,a) ∈ R for all a ∈ A. Based on this definition the program will see the Relations matrix $M_R = [r_{ij}]$. It will check if $r_{ii} = 1$ then, the Relation Matrix will be reflexive.

```
DEFINE FUNCTION Reflexive(number_of_rows, r):

    SET is_reflexive TO False

    FOR i IN range(number_of_rows):

        IF r[i][i] EQUALS 1:

            SET is_reflexive TO True

        ELSE:

            SET is_reflexive TO False

            break

    RETURN is_reflexive
```

Figure 2 PSEODUCODE of Reflexive Relations

- *Irreflexive*: a relation R on a set A is Irreflexive if $a \not R a$ for every a ∈ A. Based on this definition the program will see the Relations matrix $M_R = [r_{ij}]$ It will check if $r_{ii} = 0$ for elements of A. then, the Relation Matrix will be Irreflexive.

```
DEFINE FUNCTION Irreflexive(number_of_rows, r):

    SET is_irreflexive TO False

    FOR i IN range(number_of_rows):

        IF r[i][i] EQUALS 0:

            SET is_irreflexive TO True

        ELSE:

            SET is_irreflexive TO False

            break

    RETURN is_irreflexive
```

Figure 3 PSEODUCODE of Irreflexive Relations

- ***Symmetric*:** A relation R on a set A is symmetric if whenever a R b, then b R a. based on this definition the program will check every element in $M_{R\,=}$ [r_{ij}]. If r_{ij} = 1 and r_{ji} = 1 then, this relation matrix is symmetric.

```
DEFINE FUNCTION Symmetric(number_of_rows, r):

    SET is_symmetric TO False

    SET temp TO 1

    FOR i IN range(number_of_rows):

        IF temp EQUALS 1:

            FOR j IN range(number_of_rows):

                IF r[i][j] EQUALS 1 and r[j][i] EQUALS 1:

                    SET is_symmetric TO True

                IF (r[i][j] EQUALS 1 and r[j][i] EQUALS 0) or (r[i][j] EQUALS 0 and r[j][i] EQUALS 1):

                    SET is_symmetric TO False

                    SET temp TO 0

                    break

        ELSE:

            break

    RETURN is_symmetric
```

Figure 4 PSEODUCODE of Symmetric Relations

- ***Antisymmetric*:** A relation R on a set A is antisymmetric if whenever a R b and b R a, then a = b. The contrapositive of this definition is that R is antisymmetric if whenever a ≠ b, then a $\not R$ b or b $\not R$ a. based on the definition the program will check in $M_{R\,=}$ [r_{ij}]. If r_{ij} = 1 and r_{ji} = 1 and if i ≠ j. And if it was True the relation would not be antisymmetric. And if it was false the relation would be antisymmetric.

```
DEFINE FUNCTION AntiSymmetric(number_of_rows, r):

    SET is_anti_symmetric TO False

    FOR i IN range(number_of_rows):

        FOR j IN range(number_of_rows):

            IF i != j:

                IF r[i][j] EQUALS 1 and r[j][i] EQUALS 1:

                    SET is_anti_symmetric TO False

                    break

                ELSE:

                    SET is_anti_symmetric TO True

    RETURN is_anti_symmetric
```

Figure 5 PSEODUCODE of Antisymmetric Relations

- ***Asymmetric*:** A relation R on a set A is **asymmetric** if whenever a R b, then b $\not{R}$ a. according to the definition of the antisymmetric relation if whenever a ≠ b, then a $\not{R}$ b or b $\not{R}$ a the relation is antisymmetric. Furthermore, a relation R on a set A is Irreflexive if a $\not{R}$ a for every a ∈ A. Thus, a relation would be Asymmetric if it was both Antisymmetric and Irreflexive.

```
DEFINE FUNCTION ASymmetric(is_irreflexive, is_anti_symmetric):

    IF is_irreflexive EQUALS True and is_anti_symmetric EQUALS True:

        SET is_a_symmetric TO True

    ELSE:

        SET is_a_symmetric TO False

    RETURN is_a_symmetric
```

Figure 6 PSEODUCODE of Asymmetric Relations

- ***Transitive*: ** a relation R on a set A is transitive if whenever a R b and b R c, then a R c. It is also defined that if $(M_R)^2 = M_R$, then the relation is transitive.

```
DEFINE FUNCTION Transitive(r):

    IF r EQUALS booleanProd(r, r):

        SET is_transitive TO True

    ELSE:

        SET is_transitive TO False

    RETURN is_transitive

```

Figure 7 PSEODUCODE of Transitive Relations

- ***Equivalence*: ** A relation R on a set A is called an equivalence relation if it is reflexive, symmetric, and transitive.

```
DEFINE FUNCTION Equivalence(is_transitive, is_reflexive, is_symmetric):

    IF is_transitive EQUALS True and is_reflexive EQUALS True and is_symmetric EQUALS True:

        SET is_equivalence TO True

    ELSE:

        SET is_equivalence TO False

    RETURN is_equivalence
```

Figure 8 PSEODUCODE of Equivalence Relations

❖ Matrix Operations:

- *Matrix of Relation R on any given order:* If **M** is a Boolean Matrix, then we define **M**^2 as **M** (Boolean Product) **M** and **M**^3 as **M** (Boolean Product) **M** (Boolean Product) **M** and so on. Based on this definition, the program will first calculate all the powers of **M** up to the Matrix order, then the user will be asked to enter the power of **M** that is wanted, after that the program will divide the entered number by the Matrix order and take the remainder as an index to find the right Matrix calculated in the first step.

```
SET Power TO [RelationsMatrix]



FOR i IN range(1, MatrixOrder):

    Power.append(booleanProd(RelationsMatrix, Power[i - 1]))



IF userOpInput EQUALS "1":

    SET n TO int(INPUT("\nEnter The Number of The Power of MR You Want: "))

    SET order TO n%MatrixOrder



    OUTPUT("Matrix R^", n, "= \n")

    FOR i IN range(len(Power[order - 1])):

        FOR j IN range(len(Power[order - 1][i])):

            IF Power[order - 1][i][j] EQUALS 1:

                OUTPUT("\033[32m", Power[order - 1][i][j], "\033[0m", end=' ')

            ELSE:

                OUTPUT("\033[31m", Power[order - 1][i][j], "\033[0m", end=' ')

        OUTPUT()
```

Figure 9 PSEODUCODE of Relations of different orders

- *Matrix of infinite order:* To compute R^{∞}, we need all ordered pairs of vertices for which there is a path of any length from the first vertex to the second. Therefore, this can be the sum of R various powers.
  And to sum these matrices we will use the union operation function:
  $M_{R\infty} = M_R \vee M_R^2 \vee M_R^3 \vee \cdots$.

```
SET powerInf TO MatrixIdentify(MatrixOrder)

FOR i IN range(MatrixOrder):

    SET powerInf TO Union(powerInf, Power[i])

SET OUTPUT("\nMatrix R^inf TO \n")

OUTPUTMatrix(powerInf)
```

Figure 10 PSEODUCODE of Relations of infinite order

- *Compliment:* If **M** is a Boolean Matrix, then we define the compliment of **M** as the Matrix obtained from **M** by replacing every (0) with 1 and every (1) with 0. Based on this definition, the compliment function will check every r[i][j] in Matrix **R**, if r[i][j] = 1 it will replace it with 0 and if r[i][j] = 0 it will replace it with 1.

```
DEFINE FUNCTION Compliment(r):

    SET r_compliment TO MatrixIdentify(len(r))

    FOR i IN range(len(r)):

        FOR j IN range(len(r)):

            IF r[i][j] EQUALS 1:

                SET r_compliment[i][j] TO 0

            ELSE:

                SET r_compliment[i][j] TO 1



    RETURN r_compliment
```

Figure 11 PSEODUCODE of Compliment Operation

- *Inverse:* For a relation **R**, **R**−1 is defined as *b* ***R***−1 *a* if and only if *a* ***R*** *b*. Based on this definition, the inverse function will go through every r[i][j] in Matrix **R** and replace it with r[j][i] regardless of its value

```
DEFINE FUNCTION Inverse(r):

    SET r_inverse TO MatrixIdentify(len(r))

    FOR i IN range(len(r)):

        r_inverse[i].clear()

        FOR j IN range(len(r)):

            r_inverse[i].append(r[j][i])



    RETURN r_inverse
```

Figure 12 PSEODUCODE of Inverse Operation

- *Union*: If R and S are relations, then the union R ∪ *S* means the there is a R b or a S b. Based on this definition, the union function will take two Matrices and have a condition that if r[i][j] = 1 or s[i][j] = 1, then r ∪ s[i][j] = 1.

```
DEFINE FUNCTION Union(r, s):
    SET union_matrix TO MatrixIdentify(len(r))
    FOR i IN range(len(r)):
        FOR j IN range(len(r)):
            IF r[i][j] EQUALS 1 or s[i][j] EQUALS 1:
                SET union_matrix[i][j] TO 1
            ELSE:
                SET union_matrix[i][j] TO 0
    RETURN union_matrix
```

Figure 13 PSEODUCODE of Union Operation

- *Intersection:* If R and S are relations, then the intersection R ∩ *S* means the there is a R b and a S b. Based on this definition, the intersection function will take two Matrices and have a condition that if r[i][j] = 1 and s[i][j] = 1, then r ∩ s[i][j] = 1.

```
DEFINE FUNCTION Intersect(r, s):
    SET intersect_matrix TO MatrixIdentify(len(r))
    FOR i IN range(len(r)):
        FOR j IN range(len(r)):
            IF RelationsMatrix[i][j] EQUALS 1 and s[i][j] EQUALS 1:
                SET intersect_matrix[i][j] TO 1
            ELSE:
                SET intersect_matrix[i][j] TO 0
        OUTPUT()
```

Figure 14 PSEODUCODE of Intersection Operation

- *<u>In degrees and outdegrees table</u>***:** the in degrees out degrees table will calculate the number of inputs and outputs for all elements in the set. The in degrees would be sum of elements in the same column, while the out degrees would be the sum of elements in the same row.

```
DEFINE FUNCTION degreesTable(set_size):

    SET degrees_table TO []

    FOR i IN range(2):

        SET col TO []

        FOR j IN range(set_size):

            col.append(0)

        degrees_table.append(col)



    FOR i IN range(set_size):

        FOR j IN range(set_size):

            IF RelationsMatrix[i][j] EQUALS 1:

                SET degrees_table[0][j] TO degrees_table[0][j] + 1



    FOR i IN range(set_size):

        FOR j IN range(set_size):

            IF RelationsMatrix[j][i] EQUALS 1:

                SET degrees_table[1][j] TO degrees_table[1][j] + 1

    RETURN degrees_table
```

Figure 15 PSEODUCODE of in-degrees and out-degrees Table

- *<u>VERT TAIL HEAD NEXT Table:</u>* The Vertex table is a type of data structure that is called a linked list. The table will consist of the Vertex array which has a starting edge index for each of the vertices in set A. and there are the Tail and Head arrays which have the two linked vertices for each edge. And the Next array has the pointer for the next edge. This function takes the set A and relation R as input. Then it will initialize the Table, vertex, tail, and head arrays and append the edges' vertices to Head and Tail to then be appended in the Table. Vertex will append the first edge it finds for each vertex. An algorithm will append to each edge what's the next edge is after it. this is done by looking for an edge starting with the same vertex but with the conditions that it's not the same edge, the edge isn't the starting edge for the vertex, and is currently (in the loop iteration) has no next edge (to avoid loops).

```
DEFINE FUNCTION VTHNtable(A, R):

    SET Table TO []

    SET Vertex TO []

    FOR i IN range(len(A)):

        Vertex.append(0)

    SET Tail TO []

    SET Head TO []



    # Making a list FOR heads and tails

    FOR i IN range(len(R)):

        Tail.append(R[i][0])

    FOR i IN range(len(R)):

        Head.append(R[i][1])



    # Making the Main list called Table that will be OUTPUTed

    FOR i IN range(len(R)):

        Table.append([Tail[i], Head[i]])

    random.shuffle(Table)

    FOR i IN range(len(R)):

        Table[i].insert(0, i + 1)
```

Figure 16 PSEODUCODE of VERT TAIL HEAD NEXT Table (1)

```
    # Making the Vertix list

    FOR i IN range(len(A)):

        FOR j IN range(len(R)):

            IF A[i] EQUALS Table[j][1]:

                SET Vertex[i] TO Table[j][0]

                break



    # adding new coloumn to Taple that will be the 'Next' Column

    FOR i IN range(len(R)):

        Table[i].append(0)



    # algorithim to find the next element

    FOR i IN range(len(R)):

        FOR j IN range(len(R)):

            IF Table[i][1] EQUALS Table[j][1] and Table[j][0] not IN Vertex and Table[j][3] EQUALS 0 and i != j:

                SET Table[i][3] TO Table[j][0]

                break
```


Figure 17 PSEODUCODE of VERT TAIL HEAD NEXT Table (2)

Output:

```
**************************************** Program Started ****************************************
Enter Set A Size: 3

Enter the element at index 0
a
Enter the element at index 1
b
Enter the element at index 2
c

Would you like to enter the set R (1) or, the relations matrix (2) :
2

Enter the Element at R '( 1 , 1 )' :
1
Enter the Element at R '( 1 , 2 )' :
1
Enter the Element at R '( 1 , 3 )' :
1

Enter the Element at R '( 2 , 1 )' :
0
Enter the Element at R '( 2 , 2 )' :
0
Enter the Element at R '( 2 , 3 )' :
1

Enter the Element at R '( 3 , 1 )' :
0
Enter the Element at R '( 3 , 2 )' :
0
Enter the Element at R '( 3 , 3 )' :
1
```

Figure 16 User Relations Prompt

```
**************************************** Relation Matrix **************************************

Matrix R =


 1   1   1
 0   0   1
 0   0   1

************************************** Relation Proporties **************************************

is Matrix R Reflexive:    False
is Matrix R Irreflexive:  False
is Matrix R Symetric:     False
is Matrix R ASymertric:   False
is Matrix R AntiSymtric:  True
is Matrix R Transitive:   True

Matrix R is not an Equivalence Relation
**************************************** Matrix Operations ****************************************

What operation do you want to find?
1)  find the Relation R on any given order
2)  find M R^inf
3)  find the compliment of the Matrix
4)  find the inverse of the matrix
5)  find the union of Matrix R and Matrix S(input)
6)  find the intersect of Matrix R and Matrix S(input)
7)  find the boolean product of Matrix R and Matrix S(input)
8)  find the in degrees and out degrees table
9)  find the VERT TAIL HEAD NEXT table

0)  Exit the program

Enter the index of the desired operation:
```

Figure 17 Relations Matrix and Relation Properties and operations

```
Enter the index of the desired operation: 1

Enter The Number of The Power of MR You Want: 2
Matrix R^ 2 =

 1   1   1
 0   0   1
 0   0   1

*********************************************************************************************
```

Figure 21 output of the power of relations

```
Matrix R^inf =


 1   1   1
 0   0   1
 0   0   1


**********************************************************************************************
```

Figure 20 output of MR infinite

```
Enter the index of the desired operation: 3

Compliment of Matrix R =

 0   0   0
 1   1   0
 1   1   0


*********************************************************************************************
```

Figure 19 output of the compliment

```
Enter the index of the desired operation: 4

Inverse of Matrix R =

 1   0   0
 1   0   0
 1   1   1


********************************************************************************************
```

Figure 18 output of the inverse

```
Enter the index of the desired operation: 5

Would you like to enter the set S (1) or, the relations matrix (2) :
2

Enter the Element at S '( 1 , 1 )' :
1
Enter the Element at S '( 1 , 2 )' :
1
Enter the Element at S '( 1 , 3 )' :
1

Enter the Element at S '( 2 , 1 )' :
0
Enter the Element at S '( 2 , 2 )' :
0
Enter the Element at S '( 2 , 3 )' :
0

Enter the Element at S '( 3 , 1 )' :
0
Enter the Element at S '( 3 , 2 )' :
0
Enter the Element at S '( 3 , 3 )' :
0
```

Figure 22 Prompt of Matrix S

```
 1   1   1
 0   0   1
 0   0   1

Matrix S =

 1   1   1
 0   0   0
 0   0   0


Matrix R union S =

 1   1   1
 0   0   1
 0   0   1


*****************************************************************************************************
```

Figure 23 output of the union of matrix R and S

```
Enter the index of the desired operation: 6

Would you like to enter the set S (1) or, the relations matrix (2) :
1

Enter Set S Size: 3

Enter the related pair number 1 (example, (a,b)) :
(a,a)

Enter the related pair number 2 (example, (a,b)) :
(a,c)

Enter the related pair number 3 (example, (a,b)) :
(c,c)


Set S:  [['a', 'a'], ['a', 'c'], ['c', 'c']]



Matrix R =

 1   1   1
 0   0   1
 0   0   1

Matrix S =

 1   0   1
 0   0   0
 0   0   1

Matrix R intersect S =

 1   0   1
 0   0   0
 0   0   1


************************************************************************************************
```

Figure 24 output of the intersect of matrix R and S

```
Enter the index of the desired operation: 7

Would you like to enter the set S (1) or, the relations matrix (2) :
1

Enter Set S Size: 4

Enter the related pair number 1 (example, (a,b)) :
(a,a)

Enter the related pair number 2 (example, (a,b)) :
(a,c)

Enter the related pair number 3 (example, (a,b)) :
(c,a)

Enter the related pair number 4 (example, (a,b)) :
(c,c)


Set S:  [['a', 'a'], ['a', 'c'], ['c', 'a'], ['c', 'c']]
Would you to find MatrixR X MatrixS (1) or MatrixS X MatrixR (2) ?
1

Matrix R =

 1   1   1
 0   0   1
 0   0   1

Matrix S =

 1   0   1
 0   0   0
 1   0   1

the Boolean product of MatrixR on MatrixS =

 1   0   1
 1   0   1
 1   0   1


***********************************************************************************************
```

Figure 25 output of the Boolean product of matrix R and S

```
Enter the index of the desired operation: 8
in degrees and out degrees table:

in degree:
 1   1   3
out degree:
 3   1   1

**********************************************************************************************
```

Figure 26 output of the in degrees and out degrees table

```
Set R:

[('a', 'a'), ('a', 'b'), ('a', 'c'), ('b', 'c'), ('c', 'c')]

|Vertex|        Index          |          Tail          |          Head          |          Next

   a|01|        01             |           a            |           a            |           2
   b|05|        02             |           a            |           b            |           4
   c|03|        03             |           c            |           c            |           0
                04             |           a            |           c            |           0
                05             |           b            |           c            |           0
```

Figure 27 output of the VERT table

```
Enter the index of the desired operation: 0



**************************************** Program Ended ****************************************

Process finished with exit code 0
```

Figure 28 output of the termination of the program

Conclusion:

Discrete math is a very important science for this digital age we live in. Furthermore, the subject of relations is a subject that is a major part of this science. As it's the very foundation of models for any type of math's. It's also used frequently in computer science and the concept of data structures. In this report, We Made a program that is an application of the subject of Binary relations. We wrote algorithms and functions for identifying properties on relations and for several operations on relations. And finally, we made an algorithm to find the VERT table which is a type of data structure. After this report, we can work confidently with Relations and how to apply them in data structures.

References:

1. Franklin, James. 2017. "Discrete and Continuous: A Fundamental Dichotomy in Mathematics." *Journal of Humanistic Mathematics* 7 (2): 355–78. https://doi.org/10.5642/jhummath.201702.18.
2. Hart, Eric. n.d. "DISCRETE MATHEMATICS for ALL Overview of Discrete Mathematics in Prekindergarten through Grade 12 Guest Editorial." http://www2.southeastern.edu/orgs/LATM/Vol4Num2/Guest_Editorial.pdf.
3. "Discrete Mathematics: Past, Present and Future - Computer Science and Discrete Mathematics (CSDM) | Institute for Advanced Study." 2008. Www.ias.edu. September 4, 2008. https://www.ias.edu/math/csdm/dm.
4. Pratt, Vaughan. n.d. "Origins of the Calculus of Binary Relations." http://boole.stanford.edu/pub/ocbr.pdf.
5. Kolman, Bernard, Robert C Busby, and Sharon Cutler Ross. 2010. Discrete Mathematical Structures = 离散数学结构 / Monograph. Beijing: Higher Education Press.