

PROJECT REPORT

**REPORT**

**ADJACENCY MATRICES**

**Group Members**

NUR NABILA BINTI ROSLI (CA13016)

NUR SYUHAIDAH BINTI ISMAIL (CB13006)

NOR AMIRAH BINTI ZAHARI (CA13014)

NATASHA SHAFIRA BINTI ATAN (CA13011)

**DATE :**

16 DECEMBER 2013

**EVALUATOR :**

DR MOHD SHAM BIN MOHAMAD

SUBJECT CODE : BUM2223

DISCRETE STRUCTURE & APPLICATIONS

**Semester I**

**Academic Session 2013/2014**

**Table of Content**

1. **Introduction**
   1. Problem Statement....................................................................................3
   2. Background Project...................................................................................4
2. **Research Methodology (Pseudocode)**
   1. Algorithm (Pseudocode)............................................................................5-6
3. **Result and Discussion**
   1. Project Discussion...............................................................................................7-8
   2. Source Code........................................................................................................9-11
   3. Constraints……………………………………………………………………...12
4. **Planning for Future Work**............................................................................................13
5. **Conclusion**……………………………………………………………………………..14
6. **Reference**……………………………………………………………………………....15
7. **Appendix**………………………………………………………………………………16-19

**ADJACENCY MATRICES**

**SUBJECT (**BUM2223**)**

DISCRETE STRUCTURE & APPLICATIONS

1. INTRODUCTION

Adjacency matrices represent which vertices of graph are adjacent to which vertices. Another matrix that represent for a graph is incidence matrix. Adjacency matrix of a finite graph G on vertices is the n×n matrix where the non-diagonal entry *aij* is the number of edges from vertex *i* to vertex *j*, and the diagonal entry *aii*, depending on the convention, is either once or twice the number of edges (loops) from vertex *i* to itself.

Undirected graphs often use the latter convention of counting loops twice, whereas directed graphs typically use the former convention. There exists a unique adjacency matrix for each isomorphism class of graphs (up to permuting rows and columns), and it is not the adjacency matrix of any other isomorphism class of graphs. Graph can be associate a matrix which graph store information about the graph in that particular matrix.

Matrix also can be used to obtain more detailed information about the graph. There is some cases of finite simple graph, the adjacency matrix is a (0,1)-matrix with zeros on its diagonal. If the graph is undirected, the adjacency matrix is symmetric.

Matrices and Determinants were discovered and developed in the eighteenth and nineteenth centuries. Initially, their development dealt with transformation of geometric objects and solution of systems of linear equations. Historically, the early emphasis was on the determinant, not the matrix. In modern treatments of linear algebra, matrices are considered first.

* 1. **Problem Statement**

In developing adjacency matrices there is a few problem that occurs which is user are not aware with the use of matrices, way to implement matrices in our daily life and how it plays an important role. Basically, matrices allow logical manipulation of very large sets of numbers at once. More rigorously, matrices allow the solution of linear equation system in real world problems. Without the use of matrices it will be hard for the user to develop and designing their project. Since using matrices theory it will help reduce the workload, and make work much easier.

* 1. **Background Project**

This project title is creating an adjacency matrix of a graph by using pseudocode c or c++. Application of matrices are found in most scientific fields, in computer graphics matrices are used for project in 3-dimensional image into 2-dimensional screen. In probability matrices are used in algorithm that ranks the page in Google Search. Matrix calculus generalizes classical analytical nations such as derivatives and exponential to higher dimensions.

Major branch of numerical analysis is devoted to the development of efficient algorithm for matrix computations, a subject that is centuries old and is today and expanding area of research. Matrix decomposition methods simplify computations, both theoretically and practically. Matrices are possible to handle large numbers of variables that came up in real world problems such as economics.

Matrices come up due to their other properties, linear function can be transform and represent by using matrix multiplication. Therefore, instead of using linear function by calculating f(x) user can used matrix multiplication and can get the same result as calculating linear function. There are many other uses in higher mathematics, derivatives of multivariable mapping are in a matrix form, solution of different equations utilizes it.

Matrices can be multiplied and added using some standard mathematical rules. The surprising thing is that these standard operations, when applied to the adjacency matrix of a graph, have a real interpretation in terms of the graph properties that already know.

Last but not least, matrix theory leads into many other areas of interest, including topic in abstract mathematics. Basically mathematician told that matrices have enormous application indeed are indispensable.

1. *RESEARCH METHODOLOGY (PSEUDOCODE)*

For this project, Research Methodology Method is use as a project methodology to achieve the quality of project. Research Methodology is a common used for the system in use to develop, maintain and replace information application.

(Refer figure 1)

**Figure 1:** Research Methodology

In research methodology, the purpose of using pseudocode is that it is easier for humans to understand than conventional programming language code, and that it is a compact and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

Since each programming language uses a unique syntax structure, understanding the code of multiple languages can be difficult. Pseudocode remedies this problem by using conventional syntax and basic english phrases that are universally understood. Therefore, pseudocode must be converted into a specific programming language if it is to become a usable application.

* 1. **Algorithm (Pseudocode)**

Read m (number of vertices) and n (number of edges)   
Create an m x m matrix and fill it with zeros  
Loop n times -   
   Read an edge (source vertex, destination vertex)   
   set the [source vertex] [destination vertex] element of the adjacency matrix to 1   
End Loop   
Print the matrix   
  
This assumes that the vertices are numbered sequentially from zero. If they're numbered from one, subtract one when using them as matrix indices.

1. RESULTS & DISCUSSIONS

By this project we will describe on output and final result from analysis of the system that we have been made. After the system has been implement, the testing phase, discussion, advantages, constraints in completing the system and assumption for future development of the application will describe smoothly. Hopefully, the carry out of the result can bring the benefits also the idea to upgrade and enhance system performance to be better. Moreover, the sustainable of the system as well as the interface design are a user-friendly and attractive to be used in the future use.

Generally, Adjacency Matrices is a program that is implementing to make print screen the output. For this system, we use programming language like C language to implement the coding to build function of the system. All of the main resource either in hardware and software that we use to achieve the target and requirement in implement of our system. The outputs from using these resources are work smoothly according to our planning. Somehow, there are some lack on the system that we must fixed and make improvement in the future in getting high possible performance.

However, we will describe more precise this project about the result after the testing phase, discussion, advantages, constraints in completing the system and assumption for future development of the application.

**3.1 Project Discussion**

The primary property of a graph to consider when deciding which data structure to use is the number of edges relative to the number of vertices in the graph. A graph where *E* is close to V2 is a *dense* graph, whereas a graph where *E =* alpha *V* and alpha is much smaller than *V* is a sparse graph. For dense graphs, the adjacency*-*matrixrepresentation is usually the best choice, whereas for sparse graphs the adjacency-list representation is a better choice. Also the edge-listrepresentation is a space efficient choice for sparse graphs that is appropriate in some situations.

## Adjacency Matrix Representation

An adjacency-matrix representation of a graph is a 2-dimensional *V x V* array. Each element in the array *auv* stores a Boolean value saying whether the edge *(u,v)* is in the graph. [Figure 1](http://www.boost.org/doc/libs/1_55_0/libs/graph/doc/graph_theory_review.html#fig:adj-matrix)depicts an adjacency matrix for the graph (minus the parallel edge *(b,y)*). The amount of space required to store an adjacency-matrix is *O (V2)*. Any edge can be accessed, added, or removed in *O (1)* time. To add or remove a vertex requires reallocating and copying the whole graph, an *O (V2)* operation. The [adjacency matrix](http://www.boost.org/doc/libs/1_55_0/libs/graph/doc/adjacency_matrix.html) class implements the BGL graph interface in terms of the adjacency-matrix data-structure.

|  |
| --- |
| http://www.boost.org/doc/libs/1_55_0/libs/graph/doc/figs/adj_matrix.gif |
| **Figure 2:** The Adjacency Matrix Graph Representation. |

**3.2 Source Code**

The factor PATHINSERTION produces a function that keeps track of the path by which we arrived at a given node. The function insert for a path between two nodes of a graph (ordered pairs of vertices associated to the edges). The vertices contain paths represented as listspoint where two or more straight lines meet. While edges is a line segment that joins two vertices. If so we return the path as result, if not we add every adjacent vertex to form a list of new paths, one step longer, which are placed in queue, we also record the fact that we have visited this node.

//Adjacency Matrices

//Section 03

//Nur Nabila Binti Rosli CA13016

//Nur Syuhaidah Binti Ismail CB13006

//Nor Amirah Binti Zahari CA13014

//Natasha Shafira Binti Atan CA13011

#include <stdio.h>

#include <stdlib.h>

# include <conio.h>

void main()

{

int option;

printf("Nur Nabila Binti Rosli CA13016 \n");

printf("Nur Syuhaidah Binti Ismail CB13006 \n");

printf("Nor Amirah Binti Zahari CA13014 \n");

printf("Natasha Shafira Binti Atan CA13011 \n");

do

{

printf("\n Represent a Graph using ");

printf("Adjacency Matrix method \n ");

printf("\n 1. Directed Graph ");

printf("\n 2. Un-Directed Graph ");

printf("\n 3. Exit ");

printf("\n\n Please Select Graph : ");

scanf("%d", &option);

switch(option)

{

case 1 : dir\_graph();

break;

case 2 : undir\_graph();

break;

case 3 : exit(0);

} // switch

}while(1);

} // main

int dir\_graph()

{

int adj\_mat[50][50];

int n;

int in\_deg, out\_deg, i, j;

printf("\n How Many Vertices ? : ");

scanf("%d", &n);

read\_graph (adj\_mat, n);

printf("\n Vertex \t In\_Degree \t Out\_Degree \t Total\_Degree ");

for (i = 1; i <= n ; i++ )

{

in\_deg = out\_deg = 0;

for ( j = 1 ; j <= n ; j++ )

{

if ( adj\_mat[j][i] == 1 )

in\_deg++;

} // for

for ( j = 1 ; j <= n ; j++ )

if (adj\_mat[i][j] == 1 )

out\_deg++;

printf("\n\n %5d\t\t\t%d\t\t%d\t\t%d\n\n",i,in\_deg,out\_deg,in\_deg+out\_deg);

} // for

return;

} // dir\_graph

int undir\_graph()

{

int adj\_mat[50][50];

int deg, i, j, n;

printf("\n How Many Vertices ? : ");

scanf("%d", &n);

read\_graph(adj\_mat, n);

printf("\n Vertex \t Degree ");

for ( i = 1 ; i <= n ; i++ )

{

deg = 0;

for ( j = 1 ; j <= n ; j++ )

if ( adj\_mat[i][j] == 1)

deg++;

printf("\n\n %5d \t\t %d\n\n", i, deg);

} // for

return;

} // undir\_graph

int read\_graph ( int adj\_mat[50][50], int n )

{

int i, j;

char reply;

for ( i = 1 ; i <= n ; i++ )

{

for ( j = 1 ; j <= n ; j++ )

{

if ( i == j )

{

adj\_mat[i][j] = 0;

continue;

} // if

printf("\n Vertices %d & Vertices %d are Adjacent ? (Y/N) :",i,j);

scanf("%c", &reply);

if ( reply == 'y' || reply == 'Y' )

adj\_mat[i][j] = 1;

else

adj\_mat[i][j] = 0;

} // for

} // for

return;

} // read\_graph

**Figure 3:** Adjacency Matrices Source Code

**3.4 Constraints**

We have faced with three (3) type of constraint during the process completion on current going work that leaves some effect on our project development. There are:

1. **Technical constraints**

There are got problem on connection of the network in the hostel KK3. Every work and research on going work needs network resource that hopefully the completion of project can be work and run smoothly.

1. **Time constraints**

Adjacency Matrices Program is developed in less time period around about a month. Less time given, we just can implement the basic function in this application. If the duration can be extended, there will be more successful requirement can be fulfil to fix the system.

1. **Development Constraint**

Unfamiliar with coding we use in develop this system. Usually, in class we just learn how to use the programming technique as my main resource code. But to get a good system, we need learn another programming language to create the system.

**Knowledge and experience constraints**

Development this project, a wide knowledge in a variety of aspects is very important to analyse. Lack of knowledge of using programming language is a constraint that is need avoided. Furthermore, developing this system also required good knowledge in implementation in how to manage the timing consuming.

1. SUGGESTION FOR FUTURE WORK

Future works for this project are we would like to further evaluate the utility of our application, especially in helping of the graph mining procedure. We would also like to study its performance on more graph models and more real graph datasets. User studies could also help to design a better application. Several properties of our reordering result might help to compare the similarities between different graphs.

We would like to study their potentials, and their meanings for a graph. How it can be used other than the node ordering also needs more studies. Our current reordering procedure is running in a single thread. Our current application is using vertices list and edges list as its input. Lastly, the output will display in matrices form.

1. CONCLUSION

We have presented an approach for this project and demonstrated its strengths and weaknesses with respect to finding the solution that are of importance to analysts the adjacency matrix in this project.

Adjacency Matrices Program is stand-alone application it is develop in learning environment. It application offer the solution that will encourage the virtualisation aspect that is used in graphical industry. Sometime what we have learned during this project, we have learned a lot of skill how to use variety of programming languages. We can apply all the knowledge that we have learn in the future (working). However, we admitted that the system that we implement is not follow full requirement that they want. In future, we need make improvement and fix them to be a good one.

Lastly, what we’ve learned from this project is about team work skills for group project. To work together successfully, group members trust one another enough to share their own ideas and feelings and also group members demonstrate support for one another as they accomplish their goals. Without team working, we will unable to finish this project on the time. While develop this project, we have facing some problem such as confusing with creating design interface of system. But we are finally doing our project on time and successfully.

Before we end our words, Adjacency Matrices Program has achieved the main objectives and scope based on what we have planned. Hopefully, Adjacency Matrices Program can be a good one in the future.

1. REFERENCES

[1] Additional advice and supportive materials for Adjacency Matrices System Project.

Dr Mohd Sham Bin Mohamad.

Faculty of Industry Sciences & Technology [FIST].

[2] Book Discrete\_Mathematics\_and\_Its\_Applications\_7th\_Edition\_Rosen.

[3] Math Pages for References

URL: http://www.mathpages.com/home/icombina.htm

[4] Math World

URL: http://mathworld.wolfram.com/topics/DiscreteMathematics.html

[5] Tutorial Matrices

URL: <http://www.thelearningpoint.net/computer-science/algorithms/>

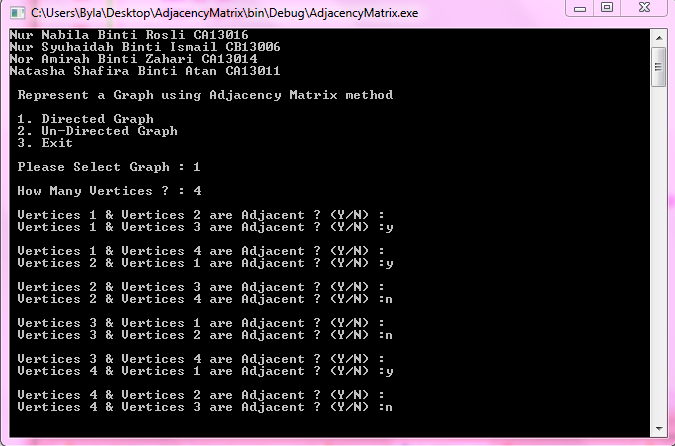
[6] Tutorial Programming Language

URL: <http://www.codeproject.com/>

1. APPENDIX

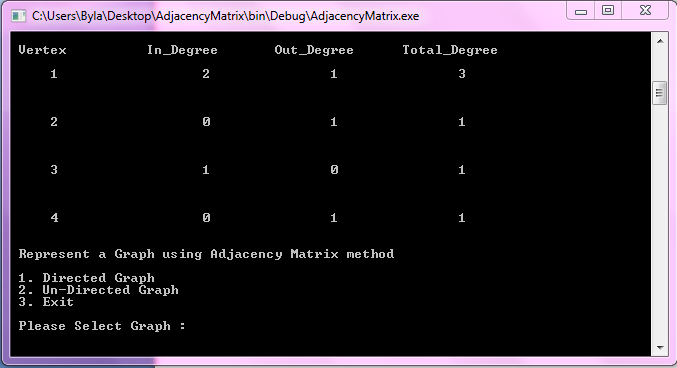
**APPENDIX**

**INPUT PRINT SCREEN**

****

**APPENDIX**

**OUTPUT PRINT SCREEN**

****