

KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

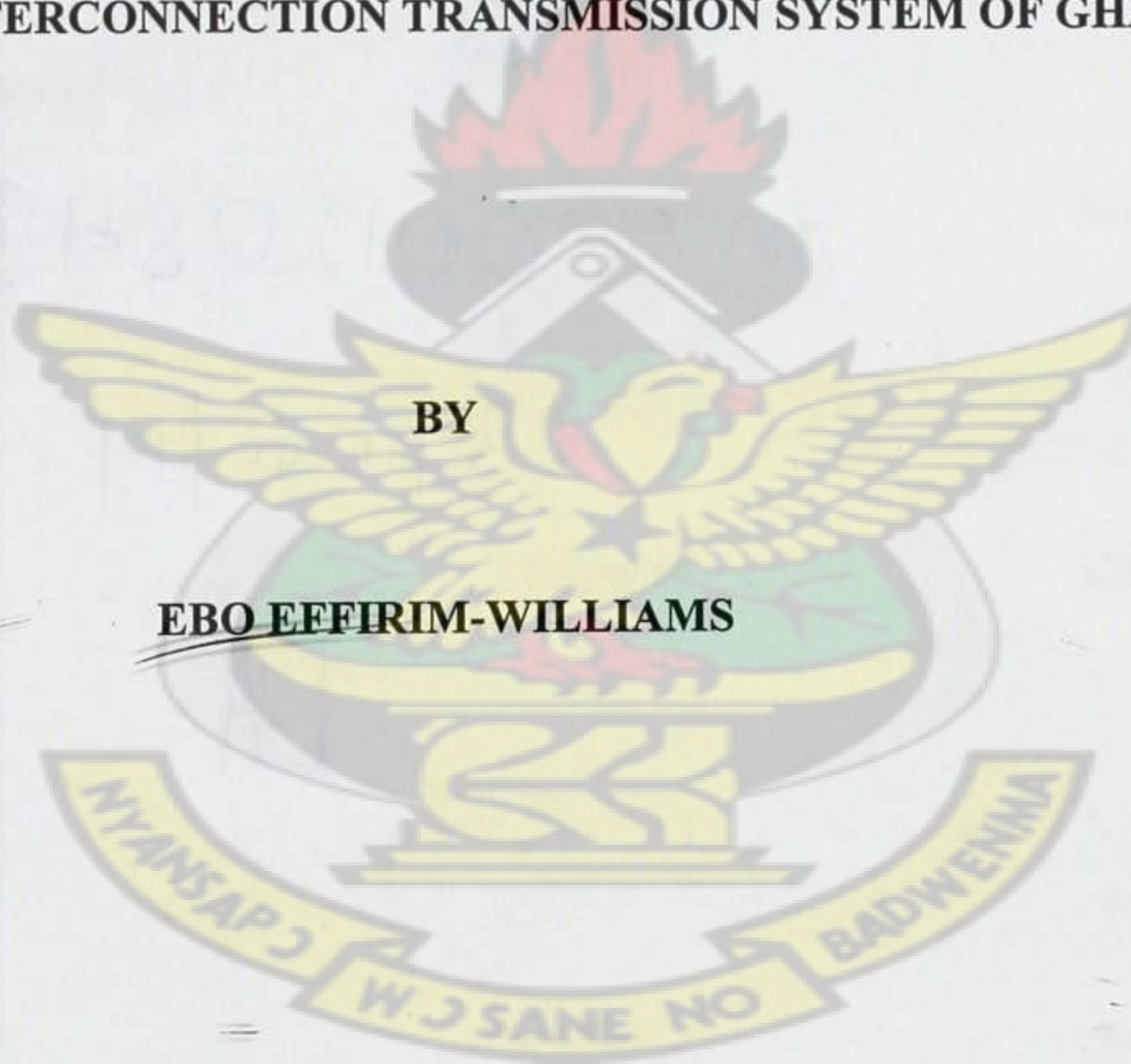
INSTITUTE OF DISTANCE LEARNING

KNUST

**MINIMUM SPANNING TREE AND THE SHORTEST PATH DISTANCE OF THE
NATIONAL INTERCONNECTION TRANSMISSION SYSTEM OF GHANA**

BY

EBO EFFIRIM-WILLIAMS



**A THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF
MASTER OF SCIENCE IN INDUSTRIAL MATHEMATICS**


JUNE, 2013

DECLARATION

I hereby declare that this submission is my own work towards the award of Master of Science degree and that, to the best of my knowledge, it contains no material previously published by another person nor material which has been accepted for award of any other degree of the university, except due acknowledgement has been made in the text.

Effirim-Williams, Ebo, PG 6319411

Student's Name & ID

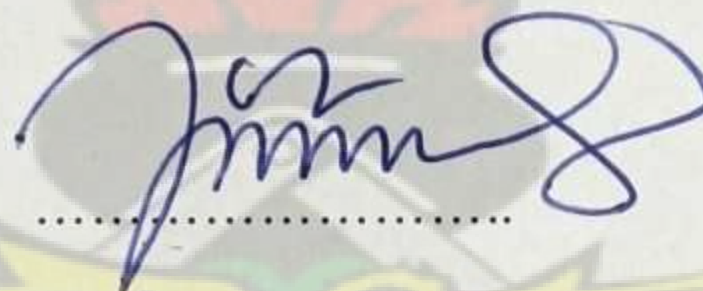


Signature

03-10-2013

Date

Certified by



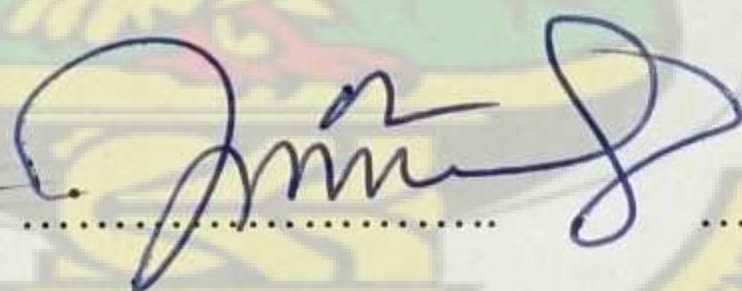
Prof. S. K. Amponsah

Signature

Date

(Supervisor)

Certified by



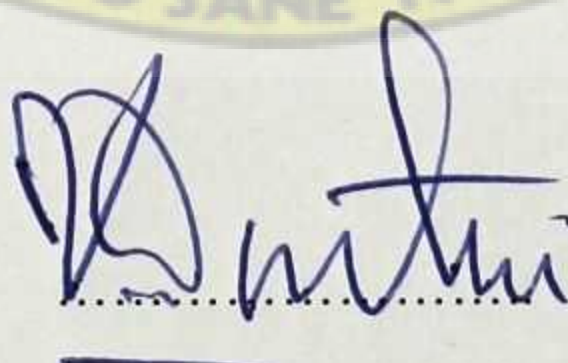
Mr. K. F. Darkwa

Signature

Date

(Head of Department)

Certified by



Prof. I. K. Dontwi

Signature

Date

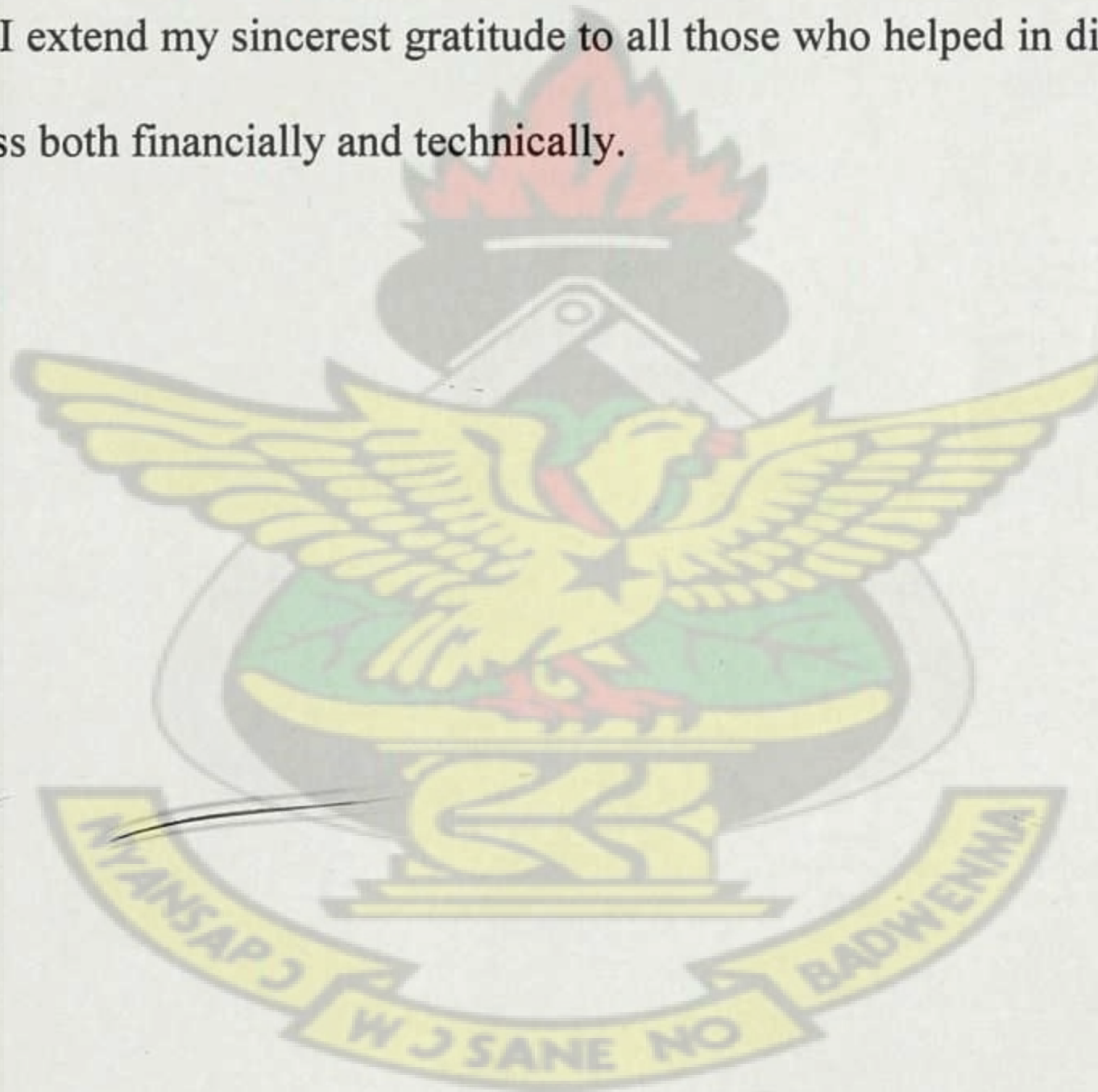
(Dean, IDL)

ACKNOWLEDGEMENT

First and foremost I am grateful to the Almighty God for His grace and mercies which has seen me through my life.

The study would not have been possible without the efforts of my supervisor, Prof. S. K. Amponsah. His dedication and valuable guidance made this work possible. I also wish to express my gratitude to all the lecturers in the Mathematics department.

Lastly but not the least, I extend my sincerest gratitude to all those who helped in diverse way to make this thesis a success both financially and technically.



DEDICATION

To the Glory of God, I dedicate this work to my family.

KNUST



ABSTRACT

The energy sector in Ghana has been undergoing deregulation with the aim of increasing power generation and the effective and efficient transmission of power. There is the need therefore to find out ways power can be transmitted effectively but in a cost effective manner. In this study a minimum spanning tree was found for National Interconnected Transmission System. The shortest path distance between the substations in the transmission network was also found. Prim's Algorithm was used to determine the minimum spanning tree, whiles Floyd Warshall's Algorithm was used to determine the shortest path distances. The study reveals that a total minimum distance of two thousand, five hundred and forty two kilometers (2542 km) will be covered in connecting the forty seven (47) 161kv substations on the National Interconnected Transmission System. It was recommended that the Ghana Grid Company Ltd adopt this study in their operations to reduce the material cost in constructing grid lines. It will also help them to determine the shortest route to use in case of emergency at their substation to enhance response time.

TABLE OF CONTENT

CONTENT	PAGE
DECLARATION.....	i
DEDICATION.....	ii
ACKNOWLEDGEMENT	iii
ABSTARCT.....	iv
CONTENTS.....	v
LIST OF TABLES.....	viii
LIST OF FIGURES.....	ix
CHAPTER 1	1
1.0 Introduction	1
1.1 Background to the Study.....	1
1.1.1 Transmission.....	2
1.1.2 Transmission Grid.....	2
1.1.3 Ghana Grid Company Ltd.....	3
1.1.4 Functions.....	3
1.1.5 Mode of Power Transmission.....	4
1.2 Problem Statement	4
1.3 Objectives of Study.....	5

1.4	Methodology.....	5
1.5	Justification/Significance of Study.....	6
1.6	Scope of Study.....	6
1.7	Limitation of Study	6
1.8	Organization of Thesis.....	7
1.9	Summary.....	7
CHAPTER 2		9
LITERATURE REVIEW		
2.0	Introduction	8
2.1	Review of Related Literature.....	8
2.2	Summary.....	25
CHAPTER 3.....		26
METHODOLOGY		
3.0	Introduction.....	26
3.1	Mathematical Model.....	26
3.1.1	Graph Theory.....	26
3.1.2	Networks.....	29
3.1.3	Spanning Tree.....	30

3.1.4	Minimum Spanning Tree.....	31
3.1.5	Algorithm For Solving MST Problem.....	32
3.2	Floyd-Warshall Algorithm.....	40
3.3	Summary.....	49
CHAPTER 4.....		50
DATA ANALYSIS AND RESULTS		
4.0	Introduction.....	50
4.1	Data Collection.....	50
4.2	Data Analysis.....	52
4.2.1	Prim’s Algorithm.....	52
4.2.2	Floyd Warshall’s Algorithm.....	56
4.3	Summary.....	56
CHAPTER 5.....		57
CONCLUSION AND RECOMMENDATIONS		
5.0	Summary.....	57
5.1	Conclusion.....	58
5.2	Recommendation.....	58
	References.....	59
	Appendix I.....	64
	Appendix II.....	69

LIST OF TABLES

Table 3.1:	Node Distance Matrix 1.....	37
Table 3.2:	Solution Matrix 1.....	37
Table 3.3:	Solution Matrix 2.....	38
Table 3.4:	Solution Matrix 3.....	38
Table 3.5:	Solution Matrix 4.....	39
Table 3.6:	Solution Matrix 5.....	39
Table 3.7:	Solution Matrix 6.....	39
Table 3.8:	Distance Matrix 1.....	41
Table 3.9:	Distance Matrix 2.....	41
Table 3.10:	Distance Matrix 3.....	43
Table 3.11:	Distance Matrix 4.....	44
Table 3.12:	Distance Matrix 5.....	46
Table 3.13:	Distance Matrix 6.....	47
Table 3.14:	Distance Matrix 7.....	48
Table 4.1:	Distance Matrix of Substations	77
Table 4.2:	Minimum Spanning Tree Matrix.....	78
Table 4.3:	Minimum Spanning Tree Table.....	52
Table 4.4:	Shortest Path Matrix.....	79

LIST OF FIGURES

Figure 3.1:	A Connected Graph.....	28
Figure 3.2:	A Connected Undirected Graph.....	31
Figure 3.3:	Four of the Spanning Tree of Graphs.....	31
Figure 3.4:	A Connected, Undirected Graph.....	31
Figure 3.5:	A Minimum-Cost Spanning Tree.....	32
Figure 3.6:	Hypothetical Network.....	36
Figure 3.6.1:	Partial.....	38
Figure 3.6.2:	Partial Connection.....	38
Figure 3.6.3:	Partial Connection 1.....	39
Figure 3.6.4:	Partial Connection Network.....	39
Figure 3.6.5:	Minimum Spanning Tree (MST).....	40
Figure 3.7:	Diagram to Show Connection of Points.....	41
Figure 4.1:	Network of the National Interconnected Transmission System.....	51
Figure 4.2:	Minimum Spanning Tree of the National Interconnected Transmission System.....	55

CHAPTER ONE

1.0 INTRODUCTION

This chapter will introduce the study and put forward the background. The problem statement, the objectives and the significance of the study will also be looked at.

Energy is the lifeblood of the global economy. The world today has become very industrialized with the production of new technological equipment which needs power to run them. Every economy needs energy to run its industry that produces goods and services, energy to light and beautify its cities, energy to entertain its people and for their basic amenities. The dependence on energy has become increasingly paramount hence the need for good planning in the sector to meet the energy needs for both domestic and industrial purposes. Energy efficiency is emerging as a key policy solution to address high energy cost and the treatment of climate change (Howland et al., 2009).

In Ghana the energy sector has been undergoing deregulation with the aim of increasing our power generation and the effective and efficient transmission of power. There is the need therefore to find out ways power can be transmitted effectively but in a cost effective manner.

1.1 BACKGROUND TO STUDY

In Ghana, the power sector is categorized in three (3) folds: Generation, Transmission and Distribution. In time past the Volta River Authority (VRA) was solely responsible for power generation and transmission of bulk power to the Electricity Company of Ghana (ECG) for distribution to various consumers. Reforms in the energy sector saw the birth of other power

generation companies like Asogli Power. It has also seen the formation of the Ghana Grid Company Limited (GRIDCo), which has been solely charged with the responsibility of transmission of bulk power to the ECG and Northern Electricity Department (NED) who distribute to various consumers.

1.1.1 TRANSMISSION

Transmission/Electric Power Transmission has been defined in many ways suitable to the use to which it is needed. The Farlex Dictionary defined Electric Power Transmission as 'The transport of generator-produced electrical energy to loads'. According to the Wikipedia free encyclopedia, 'Electric-power transmission is the bulk transfer of electrical energy, from generating power plants to electrical substations located near demand centers. This is distinct from the local wiring between high voltage substations and customers, which is typically referred to as electric power distribution'.

1.1.2 TRANSMISSION GRID

Transmission/Power Grid is defined by the Energy Dictionary (www.EnergyVortex.com) 'as a system of interconnected generating facilities, transmission corridors and power lines that provide energy to a group of customer. This term can refer to anything from a network that serves a single suburb or section of a city to a nation's entire power distribution system. Typically grid refers only to high-voltage transmission network that transport large volumes of energy from production facilities to urban area, industrial sites and end-use customers.

In Ghana, the Ghana Grid Company Limited (GRIDCo) is the Company responsible for transmission and the control of the transmission grid (National Interconnected Transmission

System). Its establishment is intended to develop and promote competition in Ghana's wholesale power market by providing transparent, non-discriminatory and open access to the transmission grid for all the participants in the power market particularly, power generators and bulk consumers and thus bringing efficiency in power delivery.

1.1.3 GHANA GRID COMPANY LIMITED (GRIDCo)

GRIDCo was established in accordance with the Energy Commission Act, 1997 (Act 154) and the Volta River Development (Amendment) Act, 2005 Act 692, which provided for the establishment and exclusive operation of the National Interconnected Transmission System (NITS) by an independent utility and the separation of the transmission functions of the Volta River Authority (VRA) from its other activities within the framework of the Power Sector Reform. GRIDCo was incorporated on December 15, 2006 as a private limited liability Company under the Companies Code, 1963 Act 179 and granted a certificate to commence business on December 18, 2006. The Company became operational on August 1, 2008 following the transfer of the staff and the power transmission assets from VRA to GRIDCo.

1.1.4 FUNCTIONS

The functions of the Ghana Grid Company Ltd (GRIDCo) are:

- Undertake economic dispatch and transmission of electricity from wholesale suppliers (generating companies) to bulk customer, which include the Electricity Company of Ghana (ECG), Northern Electricity Department (NED) and the Mines;
- Provide fair and non-discriminatory transmission services to all power market participants;

- Acquire, own and manage assets, facilities and systems required to transmit electrical energy;
- Provide metering and billing services to bulk customers;
- Carry out transmission system planning and implement necessary investment to provide the capacity to reliably transmit electric energy; and manage the Wholesale Power Market.

KNUST

1.1.5 MODE OF POWER TRANSMISSION

GRIDCo owns and operates over 4,000 km of high voltage (comprising mainly 161kV) transmission lines across the country. These lines carry power from various generating stations to transformer substations. At these substations the power is stepped down to lower voltages including 34.5kV, and 11kV for the major bulk customers and the distribution companies.

1.2 PROBLEM STATEMENT

There are several construction and reconstruction of substations and grid lines that are added to the National Grid from time to time. There are also some system challenges which needs the urgent attention of men on cite to maintain continuous transmission of power. There is also routine maintenance works performed on the various substations and Grid lines. There are several routes which can connect these substations and the grid lines. There is the need therefore to find out which of the routes will give us minimum connectivity and will be more economical in terms of cost of materials used in constructing grid lines, travelling distance and time it takes to cover such distance. There is also the need to find the shortest path from one point of the

network to the other to easily access, any time there is the need to respond to system challenges or any emergency.

On this background the research has been undertaken to find the minimum spanning tree route for the National Interconnection System (NITS), and also find the shortest path from any point of the network to the other.

1.3 OBJECTIVES OF STUDY

The objectives of study are:

- (i) to find using Prim's Algorithm, a minimum spanning tree route for the National Interconnection Transmission System (National Grid)
- (ii) to determine using Floyd Warshall Algorithm, to find the shortest path that connect one point of the transmission network to the other.

1.4 METHODOLOGY

Data for the study was collected from the Ghana Grid Company Limited (GRIDCo), and the Center for Remote Sensing and Geographic Information System (CERGIS), University of Ghana.

The data involved information about the various substations and grid lines within the National Grid. ArcGIS software, MATLAB and Microsoft Excel Spreadsheet were used in getting high level accuracy in the distance calculation and the determination of the minimum spanning by implementing the Prim's Algorithm, and the shortest path by implementing Floyd's Warshall Algorithm.

1.5 JUSTIFICATION/SIGNIFICANCE OF STUDY

Finding the minimum spanning tree route will help reduce cost in terms of resources used in constructing grid lines, and cost associated with traveling since distance to be covered will be reduced. Finding the shortest path between any given locations within the network will also reduce travelling time and the cost associated with it. This will improve access to the various substations and the grid lines across the country and reduce the cost associated with carrying out routine maintenance works. It will also reduce response time in case of an emergency at any location within the network. These will however, help in reducing transmission losses.

1.6 SCOPE OF THE STUDY

Electric power is transmitted to at least every corner of the country. The National Interconnected Transmission System is designed to cover the length of the entire country. The study covers the entire country. Data is collected from all the ten (10) regions of Ghana, where there is a substation and access to electric power.

1.7 LIMITATIONS OF STUDY

The limitations of the study are:

- (i) Wide area coverage (across the country)
- (ii) Limited time to complete the study

1.8 ORGANIZATION OF THESIS

In Chapter 1, we considered the introduction of the study. This consists of the background of the study with emphasis on electric power transmission, problem statement and the objectives of study. The justification and methodology were also put forward. Chapter 2 presents relevance and adequate literature on the subject matter. Chapter 3 presents methodology for solving the minimum spanning tree and shortest path problem. Chapter 4 will put forward data collection and analysis. Chapter 5 which is the last chapter of the study will consider the conclusions and recommendation of the study.

1.9 SUMMARY

In this chapter we considered the introduction and the background of the study. We also considered the problem statement and the objectives of the study. The justification and methodology was also put forward.

In the next chapter, we shall put forward reviewed literature related to the study.

CHAPTER 2

LITERATURE REVIEW

2.0 INTRODUCTION

In this chapter, we will review related literature to the study as it will help identify the theory behind our study and guide in the research process. A lot of studies have been conducted on Minimum Spanning Tree and Shortest Path Problems. Minimum Spanning Tree is of high importance in optimization (Zhou and Gen (1999)). Some have been finding ways of improving the algorithms in computing these problems while others compared the algorithms.

2.1 REVIEW OF RELATED LITERATURE

Moret et al., (1991) compared algorithms for the construction of a minimum spanning tree through large scale experimentation on randomly generated graphs of different structures and different densities. In order to extrapolate with confidence, the authors used graphs with up to 130,000 nodes (sparse) or 750,000 edges (dense). Algorithms included in their experiments were Prim's algorithm (implemented with a variety of priority queues), Kruskal's algorithm (using presorting or demand sorting), Cheriton and Tarjan's algorithm, and Fredman and Tarjan's algorithm. The authors also ran a large variety of tests to investigate low-level implementation decisions for the data structures, as well as to enable them to eliminate the effect of compilers and architectures. Within the range of sizes used, Prim's algorithm, using pairing heaps or sometimes binary heaps, is clearly preferable. While versions of Prim's algorithm using efficient implementations of Fibonacci heaps or rank-relaxed heaps often approach and (on the densest graphs) sometimes exceed the speed of the simpler implementations, the code for binary or

pairing heaps is much simpler, so that these two heaps appear to be the implementation of choice.

Some conclusions regarding implementation of priority queues also emerge from their study: in the context of a greedy algorithm, pairing heaps appear faster than other implementations, closely followed by binary, rank-relaxed and Fibonacci heaps, the latter two implemented with sacks, while splay trees finish a decided last.

According to Wei et al., (2011), Minimum Spanning Tree is a classical problem in graph theory that plays a key role in a broad domain of applications. The authors proposed a minimum spanning tree algorithm using Prim's approach on Nvidia GPU under CUDA architecture. By using new developed GPU-based Min-Reduction data parallel primitive in the key steps of the algorithm, higher efficiency is achieved. Experimental results show that the authors obtain about two times speedup on Nvidia GTX260 GPU over the CPU implementation and three times speed-up over non-primitives GPU implementation.

Harish and Narayanan (2007) studied "Accelerating large graph algorithms on the GPU using CUDA." The authors saw that large graphs involving millions of vertices are common in many practical applications and are challenging to process. Practical-time implementations using high-end computers are reported but are accessible only to a few. Graphics Processing Units (GPUs) of today have high computation power and low price. They have a restrictive programming model and are tricky to use. The G80 line of Nvidia GPUs can be treated as a SIMD processor array using the CUDA programming model. The authors presented a few fundamental algorithms – including breadth first search, single source shortest path, and all-pairs shortest path

– using CUDA on large graphs. They can compute the single source shortest path on a 10 million vertex graph in 1.5 seconds using the Nvidia 8800GTX GPU costing \$600. In some cases optimal sequential algorithm is not the fastest on the GPU architecture. GPUs have great potential as high-performance co-processors.

Dussert et al., (1986) developed a new approach for studying order and disorder in sets of particles. The authors approach is based on a graph constructed from the set of points locating the positions of the particles. This graph, which is called the minimal spanning tree, allowed them to deduce two parameters, namely, the average edge length m and the standard deviation σ , which are characteristic of the repartition to be studied. The method is applied to particles of an aggregated lithium thin film deposited on a di-electric substrate. These particles are found to be partially ordered. The use of a diagram involving both m and σ turns out to be a powerful tool for the determination of the degree of order in very various systems.

In a related work Dusser et al., (1987) developed a new approach to study order and disorder in biological membranes and more generally in biological structures. It is based on a graph constructed on the set points representing the position of particles. From this graph, which is called the minimal spanning tree, it is possible to deduce two parameters, namely the average length m and the standard deviation σ , which are characteristic of the repartition to be studied. The use of a diagram involving both m and σ makes it possible to determine the degree of order by taking a simple reading in the (m, σ) plane.

Gower and Ross (1969) explained Minimum Spanning Trees (MST) and single linkage cluster analysis (SLCA). It is shown that all the information required for the SLCA of a set of points is contained in their MST. Known algorithms for finding the MST are discussed. They are efficient even when there are very many points; this makes a SLCA practicable when other methods of cluster analysis are not. The relevant computing procedures are published in the Algorithm section of the same issue of Applied Statistics. The use of the MST in the interpretation of vector diagrams arising in multivariate analysis is illustrated by an example.

Vandervalk et al., (2009) described the application of two well-known graph algorithms, Edmonds' algorithm and Prim's algorithm, to the problem of optimizing distributed SPARQL queries. In the context of their paper, "distributed SPARQL query" is a SPARQL query which is resolved by contacting any number of remote SPARQL endpoints. Two optimization approaches were described. In the first approach, a static query plan is computed in advance of query execution, using one of two standard graph algorithms for finding minimum spanning trees (Edmonds' algorithm and Prim's algorithm). In the second approach, the planning and execution of the query are interleaved, so that as each potential solution is expanded it is permitted to follow an independent query plan. Their optimization approach requires basic statistics regarding RDF predicates which must be obtained prior to the user's query, through automated querying of the remote SPARQL endpoints.

Graham and Hell (1985) in their study "On the history of the minimum spanning tree problem" were of the view that, it was standard practice among authors discussing the minimum spanning tree problem to refer to the work of Kruskal (1956) and Prim (1957) as the sources of the

problem and its first efficient solutions, despite the citation by both of Boruvka (1926) as a predecessor. In fact, there are several apparently independent sources and algorithmic solutions of the problem. They have appeared in Czechoslovakia, France, and Poland, going back to the beginning of this century. The authors explored and compare these works and their motivations, and relate them to the most recent advances on the minimum spanning tree problem.

According to Michailidis (2005), a minimum spanning tree (MST) connects all the vertices of a weighted graph by existing edges whose total weight is minimum. It has been the object of intensive study due to its importance in combinatorial optimization. The author discussed the formulation of the MST problem, present some of the most commonly used algorithms for its construction, and discuss several of its applications in statistics, probability, and data analysis.

Zhou and Gen (1999) in their work on “Genetic algorithm approach on multi-criteria minimum spanning tree (MST) problem” came out that the multi-criteria MST (mc-MST) is a more realistic representation of the practical problem in the real world, but it is difficult for the traditional network optimization technique to deal with. A genetic algorithm (GA) approach was developed to deal with this problem. Without neglecting its network topology, the proposed method adopts the Prüfer number as the tree encoding and applies the Multiple Criteria Decision Making (MCDM) technique and non-dominated sorting technique to make the GA search give out all Pareto optimal solutions either focused on the region near the ideal point or distributed all along the Pareto frontier. Compared with the enumeration method of Pareto optimal solution, the numerical analysis shows the efficiency and effectiveness of the Genetic Algorithm (GA) approach on the mc-MST problem.

Borůvka presented in 1926 the first solution of the Minimum Spanning Tree Problem (MST) which is generally regarded as a cornerstone of Combinatorial Optimization, Nešetřil et al., (2001). The authors presented the first English translation of both of his pioneering works. This was followed by the survey of development related to the MST problem and by remarks and historical perspective. Out of many available algorithms to solve MST the Borůvka's algorithm is the basis of the fastest known algorithms.

According to Ahuja et al., (2001), the capacitated minimum spanning tree (CMST) problem is to find a minimum cost spanning tree with an additional cardinality constraint on the sizes of the sub trees incident to a given root node. The CMST problem is an NP-complete problem, and existing exact algorithms can solve only small size problems. Currently, the best available heuristic procedures for the CMST problem are tabu search algorithms. These algorithms use two-exchange neighborhood structures that are based on exchanging a single node or a set of nodes between two sub trees. The authors however, generalize their neighborhood structures to allow exchanges of nodes among multiple sub trees simultaneously; they referred to such neighborhood structures as multi-exchange neighborhood structures. The first multi-exchange neighborhood structure allows exchanges of single nodes among several sub trees. The second multi-exchange neighborhood structure allows exchanges that involve multiple sub trees. The size of each of these neighborhood structures grows exponentially with the problem size without any substantial increase in the computational times needed to find improved neighbors. Other approach, which is based on the cyclic transfer neighborhood structure due to Thompson and Psaraftis and Thompson and Orlin transforms a profitable exchange into a negative cost subset-

disjoint cycle in a graph, called an improvement graph, and identifies these cycles using variants of shortest path label-correcting algorithms. Their computational results with GRASP and tabu search algorithms based on these neighborhood structures reveal that:

- (i) for the unit demand case our algorithms obtained the best available solutions for all benchmark instances and improved some; and
- (ii) for the heterogeneous demand case our algorithms improved the best available solutions for most of the benchmark instances with improvements by as much as 18%. The running times our multi-exchange neighborhood search algorithms are comparable to those taken by two-exchange neighborhood search algorithms.

Graph theoretical arguments are used to show that the hierarchical clustering scheme induced by Tamura's N-step fuzzy relation f is contained in the maximal single linkage hierarchy, Dunn (1974). A method of computing f is proposed, based upon Prim's algorithm for generating maximal spanning trees and a result reported by Hu on maximal capacity routes in maximal spanning trees. It is shown that this procedure is superior to Tamura's generalized matrix multiplication algorithm with regard to both computing time and storage requirements.

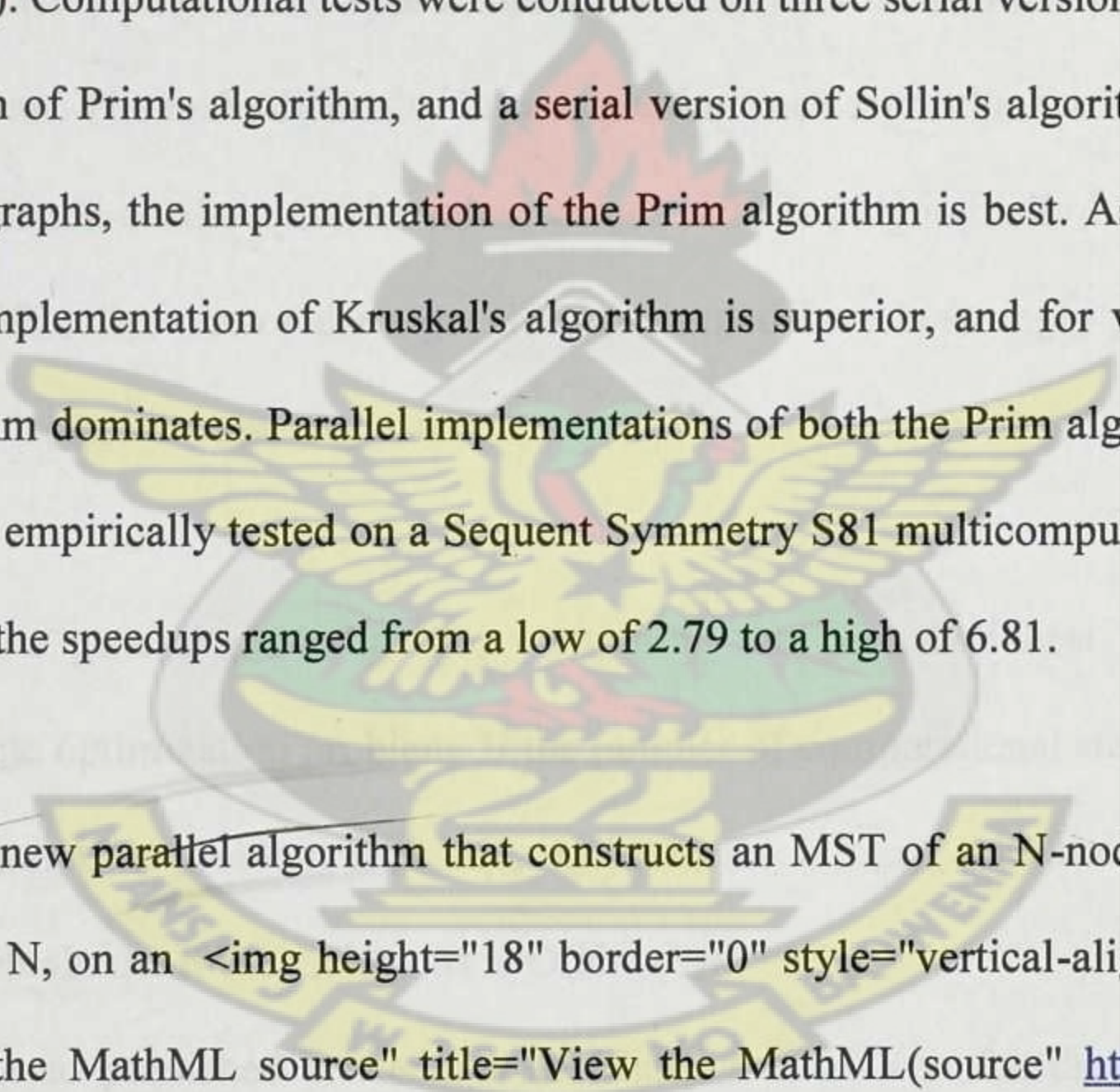
Kershenbaum and Van Slyke (1972) researched on "Computing minimum spanning trees efficiently". Their motivation was that, a ubiquitous problem in mathematical programming is the calculation of minimum spanning trees. Minimum spanning tree algorithms find application in such diverse areas as: least cost electrical wiring, minimum cost connecting communication and transportation networks, network reliability problems, minimum stress networks, clustering and numerical taxonomy, algorithms for solving traveling salesman problems, and multi-terminal

network flows. It is therefore important to know how to carry out these computations as efficiently as possible. The problem is to find a spanning sub tree of a given connected network which has minimum total length. Kruskal in 1956 showed that a "greedy" algorithm could be used; that is, if one looks at arcs in order of increasing length the first tree that can be formed is a minimum spanning tree. Shortly thereafter Prim and Dijkstra suggested another algorithm which appeared to be more efficient. Recent work suggests that a suitable implementation of Kruskal's Algorithm is computationally more efficient in a number of interesting cases, in particular when the network under consideration is sparse. A modification of Kruskal's Algorithm for the solution to the MST problem is presented and is compared with Prim's Algorithm. Prim's Algorithm is shown to have an upper bound on the number of calculations on the order of NN^2 , when applied to a network with NN nodes, regardless of the number of arcs in the network. The modification of Kruskal's Algorithm is shown to have an upper bound on the order of $NA \log_2 NA$ calculations, where NA is the number of arcs in the network. Thus for sparse networks a dramatic reduction in execution time can be obtained by the use of Kruskal's Algorithm. The effect is enhanced by the fact that Prim's Algorithm achieves its upper bound while the Kruskal modification, in general, does not. Modifications to both Prim's and Kruskal's Algorithms are introduced which give significant improvements for the complete range of sparseness. The relative merit of each algorithm is a function of sparseness of the network, the form in which the problem data is represented, the tradeoff between computation speed and storage requirements, the amount of time one wants to spend in coding, as well as many other factors.

Stoer and Wagner (1994) presented an algorithm for finding the minimum cut of an edge-weighted graph. It is simple in every respect. It has a short and compact description, is easy to

implement and has a surprisingly simple proof of correctness. Its runtime matches that of the fastest algorithm known. The runtime analysis is straight-forward. In contrast to nearly all approaches so far, the algorithm uses no flow techniques. Roughly speaking the algorithm consists of about $|V|$ nearly identical phases each of which is formally similar to Prim's minimum spanning tree algorithm.

An investigation to empirically test parallel algorithms for finding minimal spanning trees was done by Barr et al., (1989). Computational tests were conducted on three serial versions of Prim's algorithm, a serial version of Prim's algorithm, and a serial version of Sollin's algorithm. It was found that for complete graphs, the implementation of the Prim algorithm is best. As the graph density is reduced, the implementation of Kruskal's algorithm is superior, and for very sparse graphs, the Sollin algorithm dominates. Parallel implementations of both the Prim algorithm and the Sollin algorithm were empirically tested on a Sequent Symmetry S81 multicomputer. In tests involving ten processors, the speedups ranged from a low of 2.79 to a high of 6.81.

Bentley (1980) studied a new parallel algorithm that constructs an MST of an N-node graph in time proportional to $N \lg N$, on an  `<img height="18" border="0" style="vertical-align:bottom" width="162" alt="View the MathML source" title="View the MathML(source" http://ars.els-cdn.com/content/image/1-s2.0-0196677480900048-s11.gif)` computing system. The primary theoretical contribution of his paper is the new algorithm, which is an improvement over Sollin's parallel MST algorithm in several ways. On a more practical level, this algorithm is appropriate for implementation in VLSI technology.

Gonina et al., (2007) described parallel implementation of Prim's algorithm for finding a minimum spanning tree of a dense graph using MPI. The authors' algorithm uses a novel extension of adding multiple vertices per iteration to achieve significant performance improvements on large problems (up to 200,000 vertices). The authors describe several experimental results on large graphs illustrating the advantages of our approach on over a thousand processors.

Hassin (1992) studied two fully polynomial approximation schemes for the shortest path problem with an additional constraint. The main difficulty in constructing such algorithms arises since no trivial lower and upper bounds on the solution value, whose ratio is polynomially bounded, are known. In spite of this difficulty, one of the algorithms presented here is strongly polynomial. Applications to other problems were also discussed.

Lawler (1972) presented a general procedure for computing the best, 2nd best,...,Kth best solutions to a given discrete optimization problem. If the number of computational steps required to find an optimal solution to a problem with $n(0, 1)$ variables is $c(n)$, then the amount of computation required to obtain the k best solutions is $O(Knc(n))$.

The procedure specializes to published procedures of Murty and of Yen for the assignment problem and the shortest path problem, respectively. A method was presented for reducing the required amount of storage by a factor of n , compared with the algorithms of Murty and of Yen. It is shown how the K shortest (loopless) paths in an n -node network with positive and negative arcs can be computed with an amount of computation which is $O(Kn^3)$. This represents an improvement by a factor of n , compared with Yen's algorithm.

Chabini (1998) provided a solution for what appears to be a 30-year-old problem dealing with the discovery of the most efficient algorithms possible to compute all-to-one shortest paths in discrete dynamic networks. This problem lies at the heart of efficient solution approaches to dynamic network models that arise in dynamic transportation systems, such as intelligent transportation systems (ITS) applications. The all-to-one dynamic shortest paths problem and the one-to-all fastest paths problems are studied. Early results are revisited and new properties are established. The complexity of these problems is established, and solution algorithms optimal for run time are developed. A new and simple solution algorithm is proposed for all-to-one, all departure time intervals, shortest paths problems. It is proved, theoretically, that the new solution algorithm has an optimal run time complexity that equals the complexity of the problem. Computer implementations and experimental evaluations of various solution algorithms support the theoretical findings and demonstrate the efficiency of the proposed solution algorithm. The findings should be of major benefit to research and development activities in the field of dynamic management, in particular real-time management, and to control of large-scale ITSs.

Pallottino (2006) presented a new unified approach for shortest-path problems. Based on this approach, a computational method was developed which consists of determining shortest paths on a finite sequence of partial graphs defined as the "growth of the original graph." It was shown that the proposed method allowed to interpret within the same framework several different well-known algorithms, such as those of D'Esopo-Pape, Dijkstra, and Dial, and leads to a uniform analysis of their computational complexity. The author also stresses the existing parallelism between the proposed method and the matrix-multiplication methods of Floyd-Warshall, and Dantzig.

According to Han et al., (2006), a recent trend in computing are domain-specific program generators, designed to alleviate the effort of porting and re-optimizing libraries for fast-changing and increasingly complex computing platforms. Examples include ATLAS, SPIRAL, and the codelet generator in FFTW. Each of these generators produces highly optimized source code directly from a problem specification. The authors extend this list by a program generator for the well-known Floyd-Warshall algorithm that solves the all-pairs shortest path problem, which is important in a wide range of engineering applications. As the first contribution, the authors derive variants of the FW algorithm that make it possible to apply many of the optimization techniques developed for matrix-matrix multiplication. The second contribution is the actual program generator, which uses tiling, loop unrolling, and SIMD vectorization combined with a hill climbing search to produce the best code (float or integer) for a given platform. Using the program generator, the authors demonstrate a speedup over a straightforward single-precision implementation of up to a factor of 1.3 on Pentium 4 and 1.8 on Athlon 64. Use of 4-way vectorization further improves the performance by another factor of up to 5.7 on Pentium 4 and 3.0 on Athlon 64. For data type short integers, 8-way vectorization provides a speed-up of up to 4.6 on Pentium 4 and 5.0 on Athlon 64 over the best scalar code.

Katz and Kider (2008) described a shared memory cache efficient GPU implementation to solve transitive closure and the all-pairs shortest-path problem on directed graphs for large datasets. The proposed algorithmic design utilizes the resources available on the NVIDIA G80 GPU architecture using the CUDA API. The authors' solution generalizes to handle graph sizes that are inherently larger than the DRAM memory available on the GPU. Experiments demonstrated that the authors' method is able to significantly increase processing large graphs making our

method applicable for bioinformatics, internet node traffic, social networking, and routing problems.

Handler and Zang (1980) developed a Lagrangian relaxation algorithm for the problem of finding a shortest path between two nodes in a network, subject to a knapsack-type constraint. For example, the authors may wish to find a minimum cost route subject to a total time constraint in a multimode transportation network. Furthermore, the problem, which is shown to be at least as hard as NP-complete problems, is generic to a class of problems that arise in the solution of integer linear programs and discrete state/stage deterministic dynamic programs. One approach to solving the problem is to utilize a k th shortest path algorithm, terminating with the first path that satisfies the constraint. This approach is impractical when the terminal value of k is large. Using Lagrangian relaxation we propose a method that is designed to reduce this value of k . Computational results indicate orders of magnitude savings when the approach is applied to large networks.

Ahuja et al., (2011) investigated efficient implementations of Dijkstra's shortest path algorithm. The authors proposed a new data structure, called the redistributive heap, for use in this algorithm. On a network with n vertices, m edges, and nonnegative integer arc costs bounded by C , a one-level form of redistributive heap gives a time bound for Dijkstra's algorithm of $O(m + n \log C)$. A two-level form of redistributive heap gives a bound of $O(m + n \log C / \log \log C)$. A combination of a redistributive heap and a previously known data structure called a Fibonacci heap gives a bound of $O(m + n \sqrt{\log C})$. The best previously known bounds are $O(m + n \log n)$

using Fibonacci heaps alone and $O(m \log \log C)$ using the priority queue structure of Van Emde Boas, Kaas, and Zijlstra.

Okada and Soper (2000) concentrated on a shortest path problem on a network in which a fuzzy number, instead of a real number, is assigned to each arc length. Introducing an order relation between fuzzy numbers based on “fuzzy min”, a non-dominated path or Pareto Optimal path from the specified node to every other node is defined. An algorithm for solving the problem is developed on the basis of the multiple labeling method for a multi-criteria shortest path. As a result, a number of non-dominated paths can be obtained and is offered to a decision maker. However, a number of non-dominated paths derived from large scale network may be too numerous for him to choose a preferable path. Due to this situation, we propose a method to reduce the number of paths according to a possibility level. The proposed algorithm is numerically evaluated on large scale random networks.

According to Martins (1984), Multi-criteria shortest path problems have not been treated intensively in the specialized literature, despite their potential applications. In fact, a single objective function may not be sufficient to characterize a practical problem completely. For instance, in a road network several parameters (as time, cost, distance, etc.) can be assigned to each arc. Clearly, the shortest path may be too expensive to be used. Nevertheless the decision-maker must be able to choose some solution, possibly not the best for all the criteria.

The authors present two algorithms for this problem. One of them is an immediate generalization of the multiple labeling scheme algorithm of Hansen for the bi-criteria case. Based on this algorithm, it is proved that any pair of non-dominated paths can be connected by non-dominated

paths. This result is the support of an algorithm that can be viewed as a variant of the simplex method used in continuous linear multi-objective programming. A small example is presented for both algorithms.

Beasley and Christofides (1989) examined an integer programming formulation of the resource constrained shortest path problem. This is the problem of a traveler with a budget of various resources who has to reach a given destination as quickly as possible within the resource constraints imposed by his budget. A lagrangean relaxation of the integer programming formulation of the problem into a minimum cost network flow problem (which in certain circumstances reduces to an unconstrained shortest path problem) is developed which provides a lower bound for use in a tree search procedure. Problem reduction tests based on both the original problem and this lagrangean relaxation are given. Computational results are presented for the solution of problems involving up to 500 vertices, 5000 arcs, and 10 resources.

Feillet et al., (2004) proposed a solution procedure for the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). A relaxed version of this problem in which the path does not have to be elementary has been the backbone of a number of solution procedures based on column generation for several important problems, such as vehicle routing and crew pairing. In many cases relaxing the restriction of an elementary path resulted in optimal solutions in a reasonable computation time. However, for a number of other problems, the elementary path restriction has too much impact on the solution to be relaxed or might even be necessary. The authors proposed an exact solution procedure for the ESPPRC, which extends the classical label correcting algorithm originally developed for the relaxed (non-elementary) path version of this

problem. The authors presented computational experiments of this algorithm for our specific problem and embedded in a column generation scheme for the classical Vehicle Routing Problem with Time Windows. © 2004 Wiley Periodicals, Inc. NETWORKS, Vol. 44(3), 216–229 2004.

Ioachim et al., (1998) presented an optimal dynamic programming algorithm, the first such algorithm in the literature to solve the shortest path problem with time windows and additional linear costs on the node service start times. To optimally solve this problem, the authors proposed a new dynamic programming algorithm which takes into account the linear node costs. This problem has numerous applications: Two examples are job-shop scheduling and aircraft routing and scheduling. To underline the efficiency of the proposed method, we compare it with an approach based on partial discretization of the time windows. It clearly outperformed the discretization approach on test problems with wide time windows and many nodes with negative costs. © 1998 John Wiley & Sons, Inc. Networks 31: 193–204, 1998.

According to Borgwardt and Kriegel (2005), Data mining algorithms are facing the challenge to deal with an increasing number of complex objects. For graph data, a whole toolbox of data mining algorithms becomes available by defining a kernel function on instances of graphs. Graph kernels based on walks, sub trees and cycles in graphs have been proposed so far. As a general problem, these kernels are either computationally expensive or limited in their expressiveness. The authors tried to overcome this problem by defining expressive graph kernels which are based on paths. As the computation of all paths and longest paths in a graph is NP-hard, it was proposed that propose graph kernels based on shortest paths. These kernels are computable in

polynomial time, retain expressivity and are still positive definite. In experiments on classification of graph models of proteins, our shortest-path kernels show significantly higher classification accuracy than walk-based kernels.

Pettie and Ramachandran (2002) presented an undirected All-Pairs Shortest Paths (APSP) algorithm which runs on a pointer machine in time $O(mn\alpha(m,n))$ while making $O(mn\log \alpha(m,n))$ comparisons and additions, where m and n are the number of edges and vertices, respectively, and $\alpha(m, n)$ is Tarjan's inverse-Ackermann function. This improves upon all previous comparison & addition-based APSP algorithms when the graph is sparse, i.e., when $m = o(n \log n)$. At the heart of our APSP algorithm is a new single-source shortest paths algorithm which runs in time $O(m\alpha(m, n) + n \log \log r)$ on a pointer machine, where r is the ratio of the maximum-to-minimum edge length. So long as $r < 2^{n^{o(1)}}$ this algorithm is faster than any implementation of Dijkstra's classical algorithm in the comparison-addition model. For directed graphs the authors gave an $O(m + n \log r)$ -time comparison & addition-based SSSP algorithm on a pointer machine. Similar algorithms assuming integer weights or the RAM model were given earlier.

2.3 SUMMARY

In this chapter we looked at the relevant studies, literature and related works to this study. In the next chapter we shall look at the theories and the related methodology of the proposed model.

KNUST



CHAPTER 3

METHODOLOGY

3.0 INTRODUCTION

This chapter will look at the methodology that will be used in this study. Procedure used in collecting and analyzing data. Mathematical theories and instruments used will also be looked at in this chapter.

3.1 MATHEMATICAL MODEL

The first step towards solving instances of such large sizes is to find a mathematical formulation of the combinatorial problem such that every solution of the real problem corresponds to a solution of the mathematical model, and vice versa. A graph consists of points and lines connecting pairs of points. The graph associated with the problem is constructed as follows. Each substation in the study is declared to be a point. We connect a pair of points by a line segment if there is a direct link between the associated substations. Moreover, with each line segment we associated a length (weight) which corresponds to the distance it takes to travel between the two points that the line connects. Every roundtrip of the substation corresponds to some subset of the lines.

3.1.1 GRAPH THEORY

Graph theory is simply the study of graphs. Graph has several definitions depending on the researcher or writer. A graph is called regular if all its vertices have the same degree.

A graph is a combination of vertices or nodes and edges which connect in some fashion. These graphs are either directed or undirected based on their orientation. If the edges of the graph are

represented with ordered pairs of vertices, then the graph G is called directed or oriented, otherwise if the pairs are not ordered, it is called undirected or non-oriented graph. If two vertices connected by an edge $ek = (vi, vj)$ are called end vertices or ends of ek . In the directed graph, the vertex vi is called the source, and vj the target vertex of edge ek . The elements of the edge set E are distinct i.e., more than one edge can join the same vertices. Edges having the same end vertices are called parallel edges.

If $ek = (vi, vj)$, i.e., the end vertices are the same, then ek is called a self-loop. A graph G containing parallel edges and or self-loops is a multi-graph. A graph having no parallel edges and self-loops is called a simple graph. The number of vertices in G is called its order, written as $|V|$; its number of edges is given as $|E|$. A graph of order zero (0) is called an empty graph, and of order one is simply called trivial graph. A graph is finite or infinite based on its order. Two vertices vi and vj are neighbours or adjacent if they are the end vertices of the same edge $ek = (vi, vj)$. Two edges ei and ej are adjacent if they have an end vertex in common, say vi , i.e., $ei = (vi, vj)$ and $ej = (vi, vm)$. Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. $G' = (V', E')$ is a sub-graph of G ($G' \subseteq G$) if $V' \subseteq V$ and $E' \subseteq E$, i.e., the graph G contains graph G' . If $G' \subseteq G$ and V' spans all of G , i.e., $V' = V$ then G' is a spanning sub graph of G . Let $G = (V, E)$ be a graph with sets $V = \{v1, v2, v3, \dots\}$ and $E = \{1, 2, 3, \dots\}$.

A walk in a graph G is a finite nonempty alternating sequence $v0, v2, v3, \dots, vk-1, ek, vk$ of vertices and edges in G such that $ei = (vi, vi+1)$ for all $1 \leq i \leq k$.

A walk is a trail if all its edges are distinct. A trail is closed if its end vertices are the same, otherwise it is opened. A simple walk is a walk in which no edge is repeated. The length of a walk is its number of edges in the walk. A walk is closed when the first and last vertices, $v0$ and vn are the same. The degree or valency of the vertex V given by $d(V)$ is the number of edges that

have V as an endpoint. If $d(V) = 0$, then V is called an isolated vertex while a vertex of degree 1 is called pendant. The edge incident with a pendant vertex is called a pendant edge. A path is a walk in which no vertex is repeated. Closed walks are also called circuits. A cycle of length n is a closed walk of length n , $n \geq 3$, in which the vertices v_0, v_1, \dots, v_{n-1} are all different. A graph that contains no cycles at all is called acyclic; a connected acyclic graph is called a tree acyclic graphs are called forests. If a graph represents a road system, a common weight is the length of the corresponding stretch of road. Weights also often represent costs or durations. The weight of a path P is the sum of the weights of the edges in P . A cycle that passes through every vertex in a graph is called a Hamilton cycle and a graph with such a cycle is called Hamiltonian. A Hamilton path is a path that contains every vertex. A vertex V is called a cut-point in G if $G - V$ contains more components than G does; in particular if G is connected, then a cut-point is a vertex V such that $(G - V)$ is disconnected. A bridge (or cut-edge) is an edge whose deletion increases the number of components. A minimal collection of edges whose deletion disconnects G is called a cut-set in G . A cut-set partitions the vertex-set $V(G)$ into two nonempty components, say A and B , such that the edges joining vertices in A to vertices in B are precisely the edges of the cut-set. For example, the figure 3.1 below shows the cut-point and cut-edge.

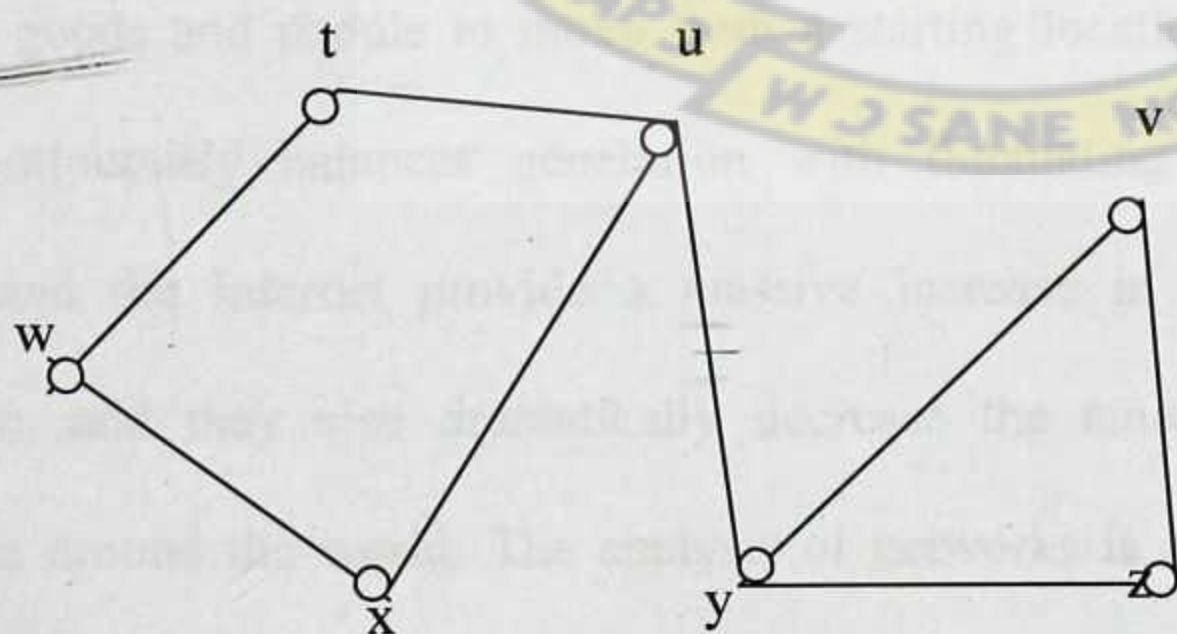


Figure 3.1: A connected graph

From Figure 3.1, removal of u or y disconnects the graph; also removal of $u y$ disconnects the graph. Hence u and y are the cut-points and $u y$ is the cut-edge.

Graph theory is applied in almost every day life from communication network, water supply system, road network system etc.

3.1.2 NETWORKS

A network is a specific type of graph, where associated with each arc or node is additional information, such as the cost or capacity of the arc or the demand at a node. Networks are integral to a variety of systems that we rely upon each day. Our transportation system is made up of a variety of networks including road, rail and airline networks. Our electrical system is a network of wires that ensures power reaches homes and businesses. Communications systems, including the Internet, are expanding beyond the typical hard wired lines to include wireless networks.

Each of these networks plays an important role in society. A transportation network provides a means for goods and people to move from a starting location to a destination. The electrical system continuously balances generation with fluctuating user demand. Communication networks and the Internet provide a massive increase in the amount of easily obtainable information, and they also dramatically decrease the amount of time required to transfer information around the world. The analysis of networks is even helping to fight terrorism by identifying terrorist networks so that we can determine where it is most effective to disrupt them.

The study of networks is actually centuries old. Graph theory dates back to Leonard Euler in 1736 (Biggs, Lloyd and Wilson, 1998), when he proved there was no feasible solution to the Königsberg Bridge Problem. The development of random graph theory in the 1940s and 1950s generated great interest in the characteristics of graphs and networks (Watts et al, 2006). Most recently, the advent of “network science” during the last decade has witnessed renewed interest in the large-scale properties of graphs (National Research Council, 2005).

Erdős and Rényi (1959) pioneered the exploration of random graphs models, which generated interest in graph and network theory. More recently, the study of network science has focused attention on “small-world networks” and “scale-free networks.”

Small-world networks (Watts et al., 1998) are networks that have high local clustering and have path lengths between arbitrarily chosen nodes that are still relatively short. Another area of increasing importance is the use of Hastily Formed Networks (HFNs) in response to humanitarian aid and disaster relief operations, such as a Hurricane Katrina scenario (Denning, 2006). These types of networks require rapid coordination and information between a variety of agencies.

3.1.3 SPANNING TREE

A spanning tree of a graph is just a sub-graph that contains all the vertices and is a tree. Suppose you have a connected undirected graph (Connected: every node is reachable from every other node, Undirected: edges do not have an associated direction) then a spanning tree of the graph is a connected sub-graph in which there are no cycles. A graph may have many spanning trees, for example;

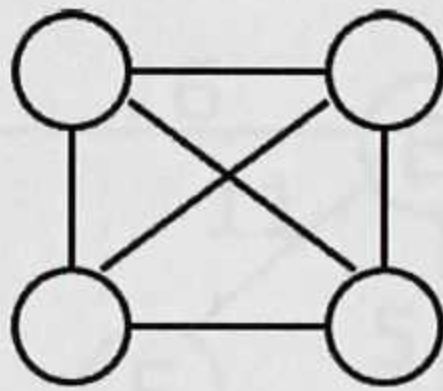


Figure 3.2 A connected, undirected graph

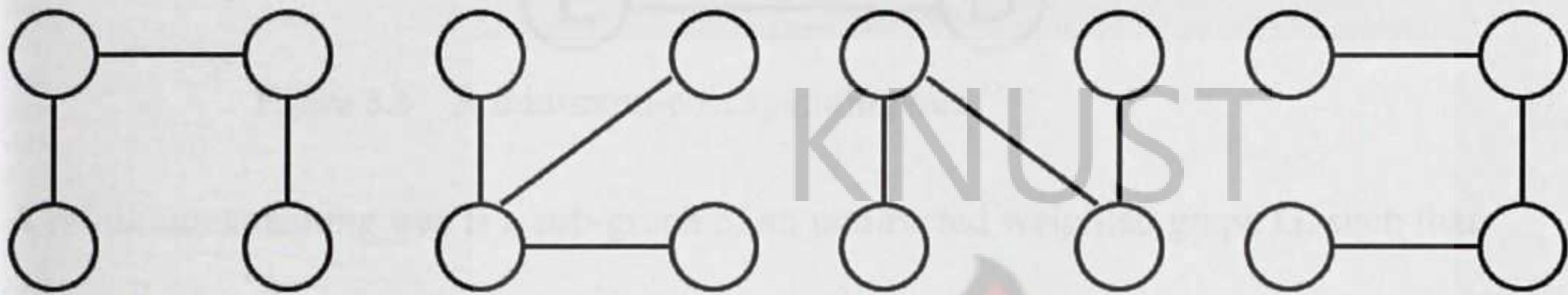


Figure 3.3 Four of the spanning trees of the graph

3.1.4 MINIMUM SPANNING TREE

Suppose you have a connected undirected graph with a weight (or cost) associated with each edge, the cost of a spanning tree would be the sum of the costs of its edges. A minimum-cost spanning tree is a spanning tree that has the lowest cost. The weight of a sub-graph is the sum of the weights of its edges. A minimum spanning tree for a weighted graph is a spanning tree with minimum weight.

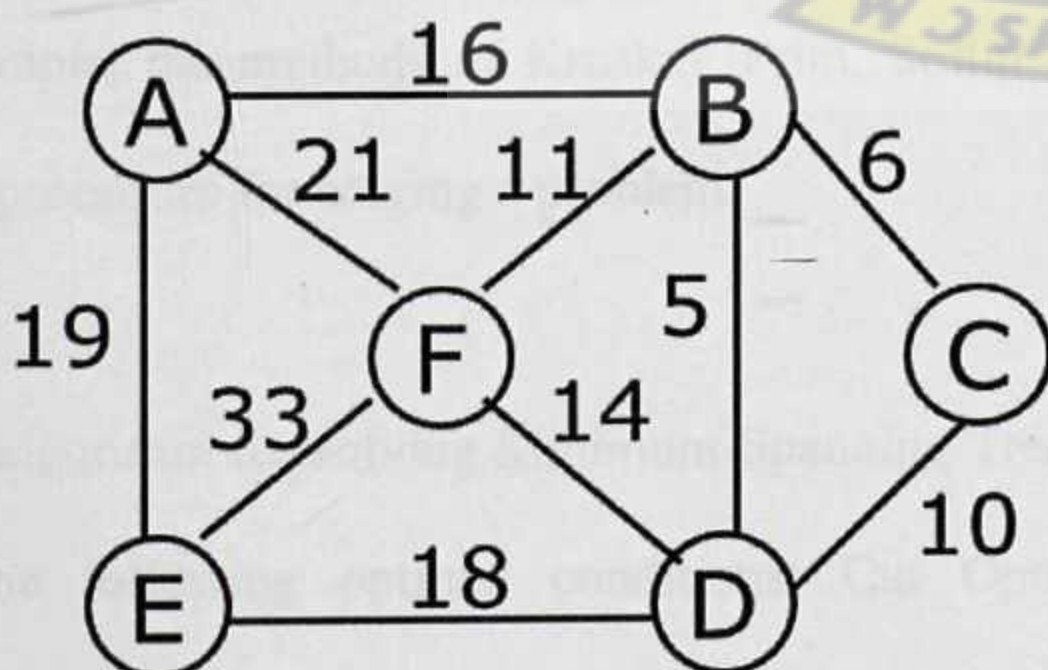


Figure 3.4 A connected, undirected graph

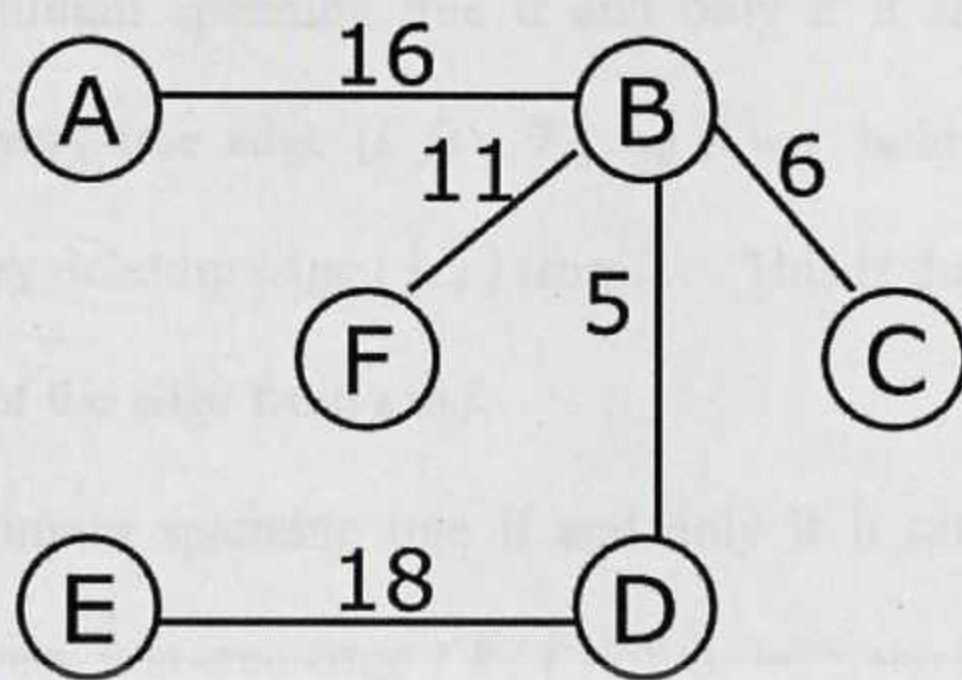


Figure 3.5 A minimum-cost spanning tree

A minimum spanning tree is a sub-graph of an undirected weighted graph G , such that:

- it is a tree (i.e., it is acyclic)
- it covers all the vertices V
 - contains $|V| - 1$ edges
- the total cost associated with tree edges is the minimum among all possible spanning trees
- not necessarily unique

3.1.5 ALGORITHMS FOR SOLVING MST PROBLEM

To find a minimum spanning tree for a given input graph there are several algorithms available, for example, the methods of Kruskal, Prim, Sollin or Borůvka. An algorithm is a systematic logical procedure for solving a problem.

Prim's algorithm for solving Minimum Spanning Tree problem will be considered. The methods have the following optimal conditions: Cut Optimality Conditions and Path Optimality Conditions.

A spanning tree T is a minimum spanning tree if and only if it satisfies the following cut optimality conditions: For every tree edge $(i, j) \in T$, $w_{ij} \leq w_{kl}$ holds for every edge (k, l) contained in the cut formed by deleting edge (i, j) from T . This is the fundament of the Prim's algorithm. w_{ij} is the weight of the edge from i to j .

A spanning T tree is a minimum spanning tree if and only if it satisfies the following path optimality conditions: For every non-tree edge (k, l) of G , $w_{ij} \leq w_{kl}$ holds for every edge (i, j) contained in the path T connecting nodes k and l .

3.1.5.1 PRIM'S ALGORITHM

Prim's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it will only find a minimum spanning tree for one of the connected components.

Prim's algorithm is one of the best algorithms for solving minimum spanning tree problems. The algorithm represents an n - node network as a square matrix with n rows and n columns. Entry (i, j) of the matrix gives the distance or edge d_{ij} or e_{ij} from node i to node j , which is finite if i is directly linked to j and infinite otherwise.

The algorithm is a greedy algorithm; it starts by selecting an arbitrary vertex as the root of the tree. It then grows the tree by adding a vertex that is closest (has the shortest edge to) the current tree, and adding the shortest edge from any vertex already in the tree to the new vertex. The

algorithm terminates once all vertices have been added to the tree. The sum of all added edges is the cost of the minimum spanning tree (MST). The serial computational complexity of the algorithm is $\Theta(N^2)$. Prim's algorithm has the property that the edges in the set A always form a single tree.

If for any undirected graph, $G = (V, E)$ where V is the set of vertices and E is the set of edges. For each $v \in V$ the $\text{cost}[v]$ denotes the minimum weight among all edges connecting v to the vertex in the tree T , and the $\text{parent}[v]$ denotes parent of v in T . During the algorithm's execution vertices v that are not in T are organized in the minimum-priority queue Q , partition according to $\text{cost}[v]$. Lines 1 to 3 set each $\text{cost}[v]$ to infinity usually written as ∞ . The parent of each vertex is set to $NULL$ because the construction of the minimum spanning tree is yet to begin. Lines 4 to 6 choose an arbitrary vertex r from V as the root of the tree (starting vertex). The minimum priority queue is set to be all vertices from V . Since r is the starting vertex, $\text{cost}[r]$ is set to zero.

During the execution of the while loop from lines 7 to 12, r is the first vertex to be extracted from Q and processed. Line 8 extract a vertex u from Q based on key cost, thus moving u to the vertex set of T . Line 9 considers all vertices adjacent to u . The while loop updates the cost and the parent fields of each vertex v adjacent to u that is not in T . If $\text{parent}[v] \neq NULL$ then $\text{cost}[v] < \infty$ and $\text{cost}[v]$ is the weight of the edge v to some vertex already in T . Lines 13 and 14 construct the edge set of the minimum spanning tree and return this edge set.

3.1.5.2 PSEUDOCODE OF PRIM'S ALGORITHM

Given a connected weighted graph $G = (V, E)$ with a weight function w and a minimum

spanning tree T can be derived from the code below.

1. for any $v \in V$ do
2. $\text{cost}[v] \leftarrow \infty$
3. $\text{parent}[v] \leftarrow \text{NULL}$
4. $r \leftarrow$ arbitrary vertex of V
5. $\text{Cost}[r] \leftarrow 0$
6. $Q \leftarrow V$
7. While $Q \neq \{ \}$
8. $u \leftarrow \text{extract Min}(Q)$
9. for each $v \in \text{adja}(u)$ do
10. if $v \in Q$ and $w(u, v) < \text{cost}[v]$ then
11. $\text{parent}[v] \leftarrow u$
12. $\text{cost}[v] \leftarrow w(u, v)$
13. $T \leftarrow \{(v, \text{parent}[v]) \mid v \in V - \{r\}\}$
14. Return T

3.1.5.3 HOW THE ALGORITHM WORKS

Prim's algorithm works from a starting point and builds up the spanning tree step by step, connecting edges into the existing solution. The algorithm can be stated as follows:

Prim's MST algorithm (from a network)

Step 0: Choose any element r ; and set $S = \{r\}$ and $A = \{ \}$

(r is the root of the spanning tree)

Step 1: Find the lightest edge such that one endpoint is in $@$ and the other is in $\setminus @$. Add this edge to $@$ and its (other) endpoint to $@$.

Step 2: If $V \setminus S = \{ \}$ then stop and output the minimum spanning tree (S, A)

Otherwise go to step 1.

Prim's MST algorithm (from a distance matrix)

Step 0: With the matrix representing the network, choose a starting vertex. Delete the row corresponding to that vertex.

Step 1: Label with '1' the column corresponding to the start vertex and ring the smallest undeleted entry in that column.

Step 2: Delete the row corresponding to the ringed entry.

Step 3: Label (with the next number) the column corresponding to the deleted row.

Step 4: Ring the lowest undeleted entry in all labeled columns.

Step 5: Repeat the last three steps until all rows are deleted. The ringed entries represent the edges in the minimum connector.

When there is a tie in the smallest values, it is broken arbitrarily.

EXAMPLE 1

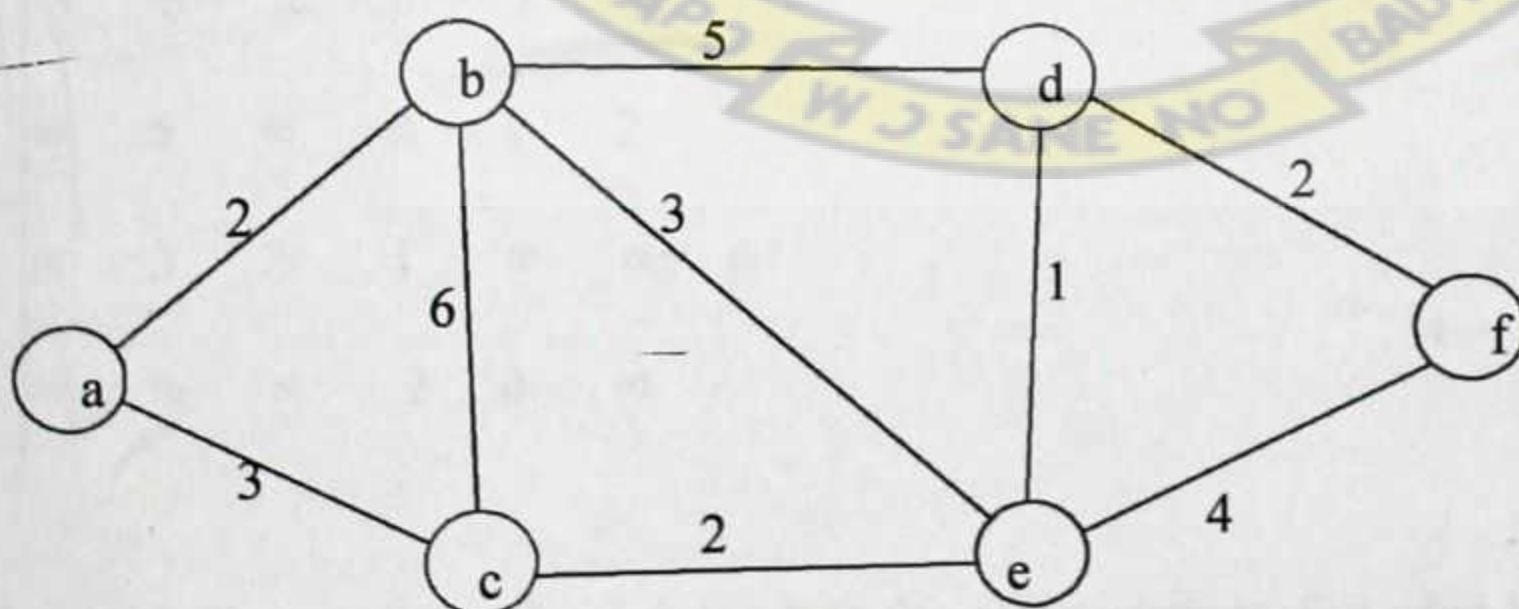


Figure 3.6: Hypothetical network

For example, Figure 3.5 a hypothetical network can be put in distance matrix form and solved by

Prim's algorithm as follows:

3.1.5.5 SOLUTION BY MATRIX METHOD

Table 3.1 Node distance matrix 1

	a	b	c	d	e	f
a	∞	2	3	∞	∞	∞
b	2	∞	6	5	3	∞
c	3	6	∞	∞	2	∞
d	∞	5	∞	∞	1	2
e	∞	3	2	1	∞	∞
f	∞	∞	∞	2	4	∞

Choose a starting vertex say **b**, delete row **b**, and look for the smallest entry in column **b**.

Table 3.2 solution matrix 1

	a	b	c	d	e	f
a	∞	2	3	∞	∞	∞
b	2	∞	6	5	3	∞
c	3	6	∞	∞	2	∞
d	∞	5	∞	∞	1	2
e	∞	3	2	1	∞	∞
f	∞	∞	∞	2	4	∞

The edge **ba** is the smallest edge joining **b** to the other vertices. Put edge **ba** into the solution.

Delete row **a** and look for the smallest entry columns **b** and **a**.

Table 3.3 solution matrix 2

	<div>2 ↓ a</div>	<div>1 ↓ b</div>	c	d	e	f
a	∞	2	3	∞	∞	∞
c	3	6	∞	∞	2	∞
d	∞	5	∞	∞	1	2
e	∞	3	2	1	∞	∞
f	∞	∞	∞	2	4	∞

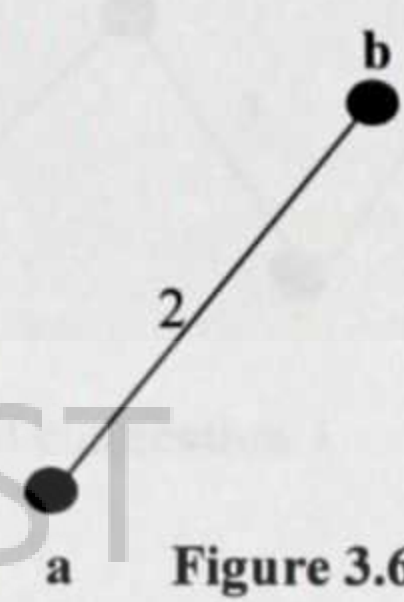


Figure 3.6:1 partial

be is the smallest edge joining **b** and **a** to the other vertices. Put **be** into the solution and delete row **e**. Look for **smallest** entry in columns **b**, **a** and **e**.

Table 3.4 solution matrix 3

	<div>2 ↓ a</div>	<div>1 ↓ b</div>	c	d	e	<div>3 ↓ f</div>
c	3	6	∞	∞	2	∞
d	∞	5	∞	∞	1	2
e	∞	3	2	1	∞	∞
f	∞	∞	∞	2	4	∞

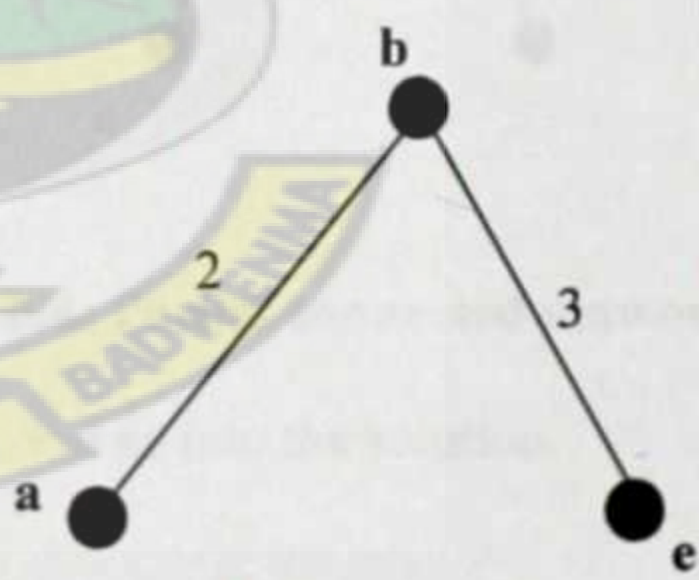


Figure 3.6.2: Partial connection

ed is the smallest edge joining **b**, **a**, and **e** to the other vertices. Put the edge **ed** into the solution and delete row **d**. Look for the smallest entry in columns **b**, **a**, **e** and **d**

Table 3.5 solution matrix 4

	<div>2 ↓ a</div>	<div>1 ↓ b</div>	c	<div>4 ↓ d</div>	<div>3 ↓ e</div>	f
c	3	6	∞	∞	2	∞
d	∞	5	∞	∞	1	2
f	∞	∞	∞	2	4	∞

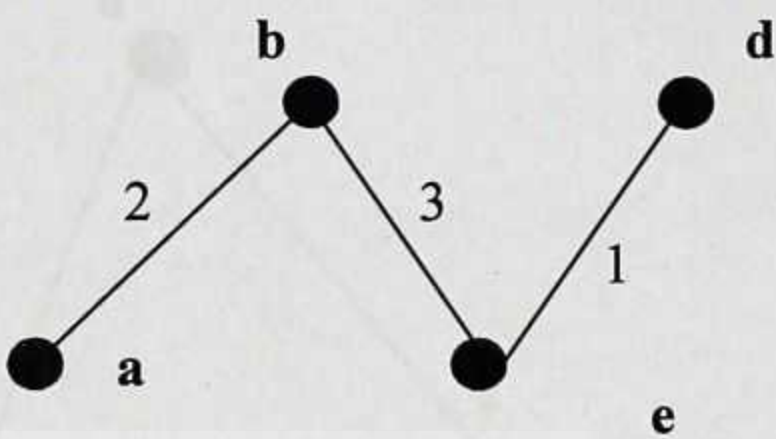


Figure3.6.3:Partial connection 1

df is the smallest edge joining **b, a, e, and d** to the other vertices. Put **df** into the solution and delete row **f**. Look for the smallest entry in columns **b, a, e, d** and **f**.

Table 3.6 solution matrix 5

	<div>2 ↓ a</div>	<div>1 ↓ b</div>	c	<div>4 ↓ d</div>	<div>3 ↓ e</div>	<div>5 ↓ f</div>
c	3	6	∞	∞	2	∞
f	∞	∞	∞	2	4	∞

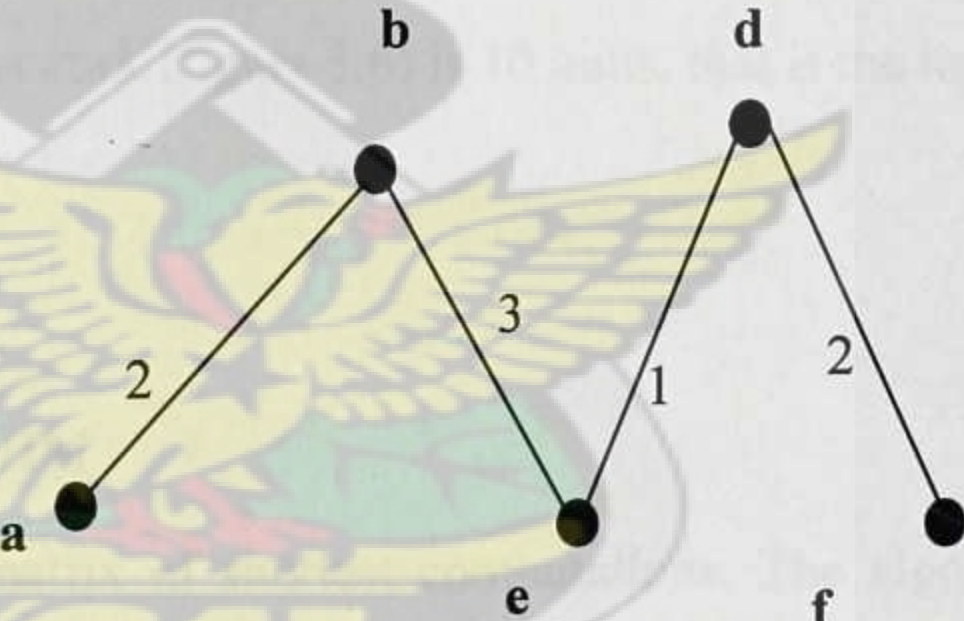


Figure 3.6.4: Partial connected network

ec is the smallest edge joining **b,a,e,d, and f** to the other vertices. Put **ec** into the solution.

Table 3.7 solution matrix 6

	<div>2 ↓ a</div>	<div>1 ↓ b</div>	<div>6 ↓ c</div>	<div>4 ↓ d</div>	<div>3 ↓ e</div>	<div>5 ↓ f</div>
c	3	6	∞	∞	2	∞

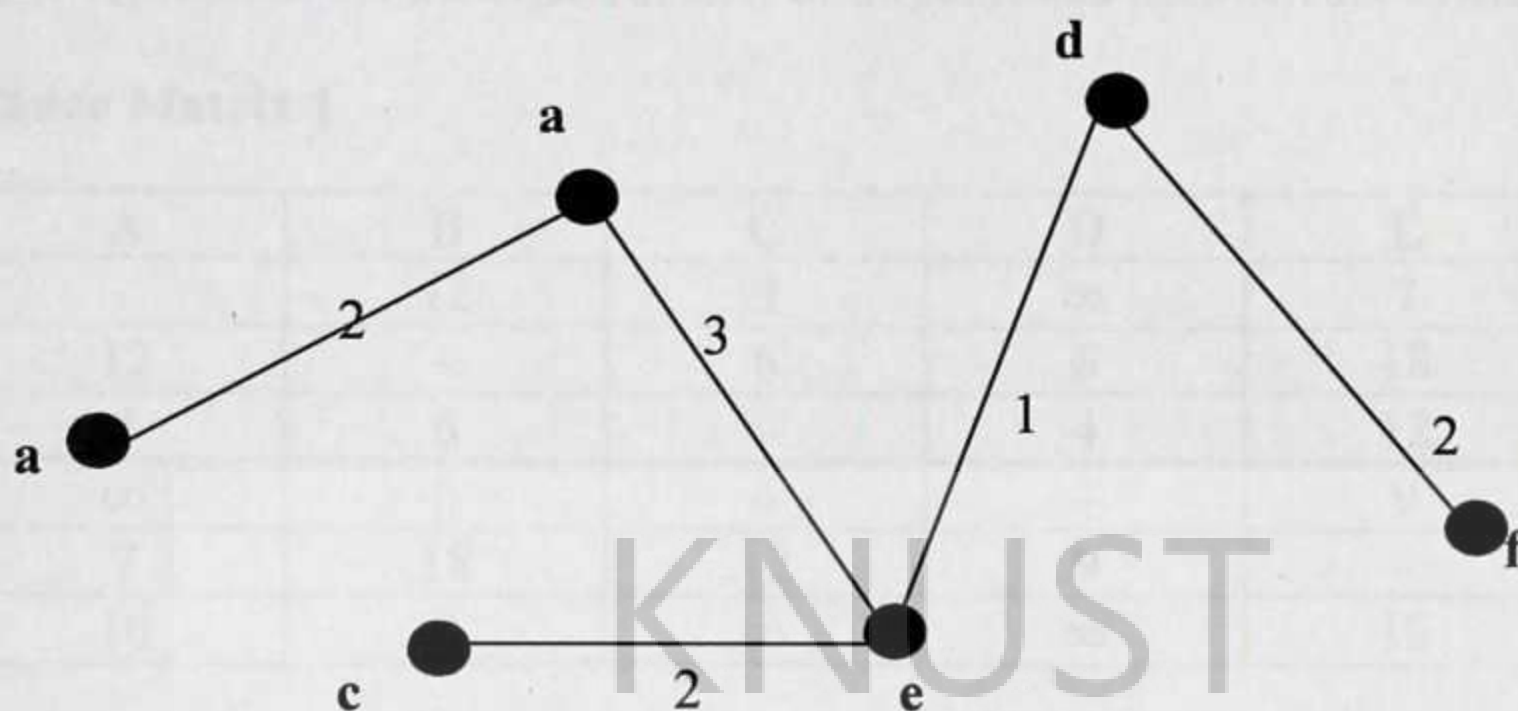


Figure 3.6.5: Minimum spanning tree (MST)

Figure 3.6.5 is the minimum spanning tree for Figure 3.6.

The minimum length (weight or cost) of the network (figure 3.6) is 10 units, that is the total sum of the edge values ($2+3+2+1+2=10$).

3.1.6 FLOYD – WARSHALL'S ALGORITHM

The Floyd – Warshall algorithm obtains a matrix of shortest computations. The algorithm is based on inductive arguments and dynamic programming technique.

Let $d^k(i,j)$ represents the length of the shortest path from node i to node j subject to the condition that this path uses the nodes $1,2,\dots,k-1$ as internal nodes clearly $d^{n+1}(i,j)$ represents the actual shortest path distance from i to j . The algorithm first computes $d^1(i,j)$ for all node pairs i and j . Using $d^1(i,j)$, it then computes $d^2(i,j)$ for all pairs of nodes i and j . It repeats the process until it obtains $d^{n+1}(i,j)$ for all node pairs i and j when it terminates. Given $d^k(i,j)$, the algorithm computes $d^{k+1}(i,j) = \min\{d^k(i,k), d^k(k,j)\}$. The Floyd-Warshall algorithm remains of interest because it handles negative weight edges correctly.

Example

The matrix below represents the distance (in km) of direct roads between six towns.

Table 3.8 Distance Matrix 1

	A	B	C	D	E	F
A	-	12	3	∞	7	10
B	12	-	6	6	18	8
C	3	6	-	4	12	∞
D	∞	6	4	-	9	∞
E	7	18	12	9	-	12
F	10	8	∞	∞	12	-

Use Floyd’s algorithm to find the shortest distances between all pairs of nodes

Note: The symbol ∞ means that there is no direct connection between these vertices. For example

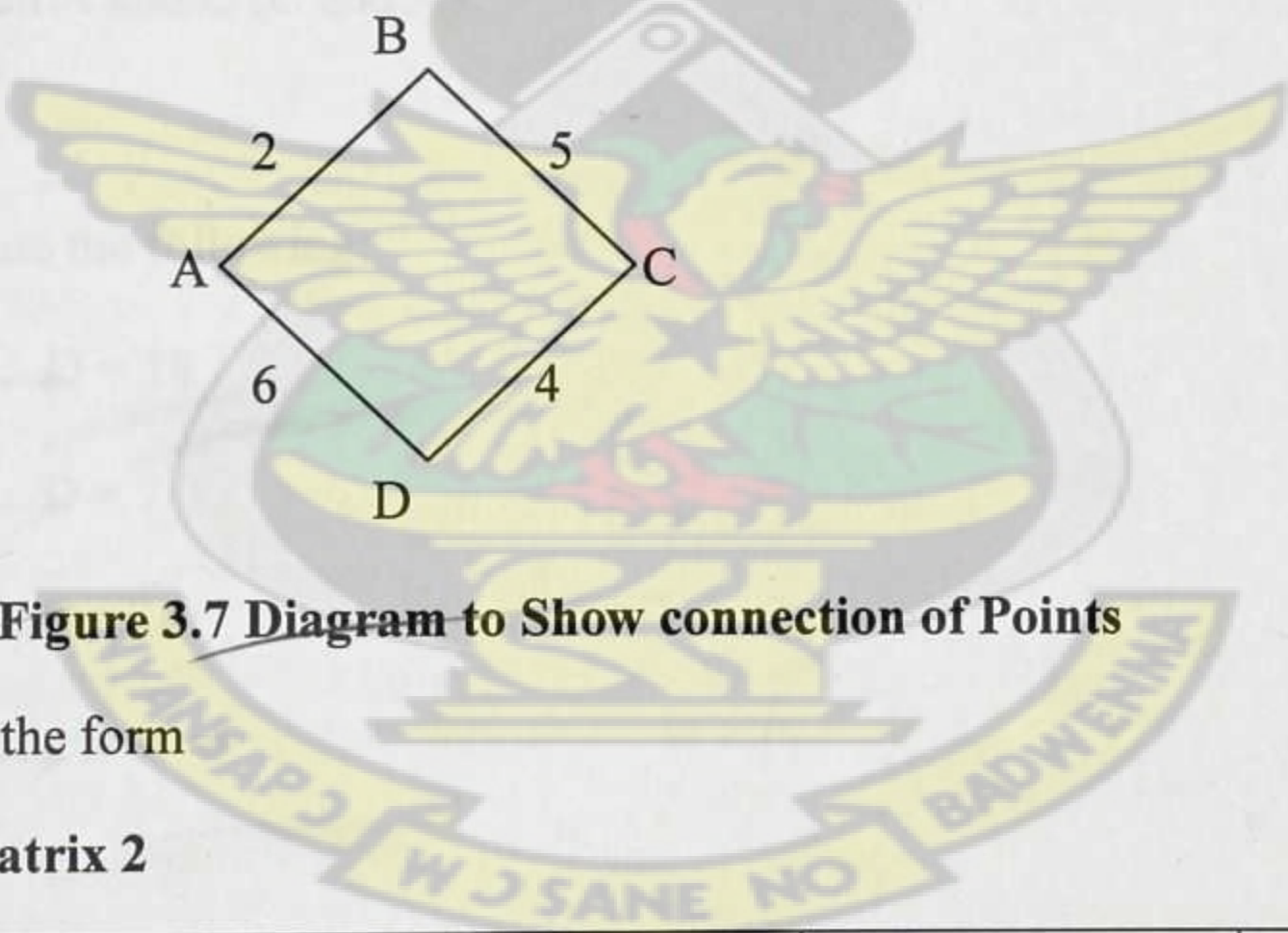


Figure 3.7 Diagram to Show connection of Points

We obtain the table of the form

Table 3.9 Distance Matrix 2

	A	B	C	D
A	-	2	∞	6
B	2	-	5	∞
C	∞	5	-	4
D	6	∞	4	-

Solution

- (i) From A to B: (the direct distance from A to B is 12)

$$A \xrightarrow{3} C \xrightarrow{6} B = 9$$

$$A \xrightarrow{7} E \xrightarrow{18} B = 25$$

$$A \xrightarrow{10} F \xrightarrow{8} B = 18$$

Since the values of the distances between A and C, A and E and F are all less than 12, we compute the following distances for the detours;

We can see that $\min \{9, 25, 18\} = 9$ is less than 12 the direct distance between A and B and so we replace 12 by 9 in the table and likewise the distance between B and A in the table.

(ii) From A to C: the direct connections between A and C is 3 and no other direct connection between A and any other node is less than 3, so we retain the value 3 as the minimum distance between A and C (C and A).

(iii) A to D:

Here we compute the following;

$$A \xrightarrow{12} B \xrightarrow{6} D = 18$$

$$A \xrightarrow{3} C \xrightarrow{4} D = 7$$

$$A \xrightarrow{7} E \xrightarrow{8} D = 16$$

$$A \xrightarrow{10} F \xrightarrow{\infty} D = \infty$$

Now $\min \{18, 7, 16, \infty\} = 7$ so we replace ∞ from A to D and D to A by 7.

(iv) A to E: (the direct connection between A and E is 7) and here only 3, the direct distance between A and C is less than 7, so we compute;

$$A \xrightarrow{3} C \xrightarrow{12} E = 15, \text{ which is not less than 7 so we retain the 7 in the cell AE.}$$

(v) A to F: (the direct connection between A and F is 10) but the distances between A and C, A and D, and A and E are less than 10 so we compute the distances for the detours,

$$A \xrightarrow{3} C \xrightarrow{\infty} F = \infty$$

$$A \xrightarrow{7} D \xrightarrow{\infty} F = \infty$$

$$A \xrightarrow{7} E \xrightarrow{12} F = 19$$

$\text{Min } \{\infty, \infty, 19\} = 19 > 10$, so we retain the value of 10 in the cell AF.

Now from A to the other nodes, the table is

Table 3.10 Distance Matrix 3

	A	B	C	D	E	F
A	-	9	3	7	7	10
B	9	-	6	6	18	8
C	3	6	-	4	12	∞
D	7	6	4	-	9	∞
E	7	18	12	9	-	12
F	10	8	∞	∞	12	-

Now from the node B

- (i) From B to A, now the values in the cells BC, BD and BE are all less than that of BA.

We then compute the distances of the detours.

$$B \xrightarrow{6} C \xrightarrow{6} A = 12$$

$$B \xrightarrow{6} D \xrightarrow{7} A = 13$$

$$B \xrightarrow{8} E \xrightarrow{7} A = 15$$

Now $\text{min } \{12, 13, 15\} = 12$, which is not less than 9 so we retain the value of 9 in the cell BA and AB

- (ii) For the cells BC and BD each with value 6, no other value in the cells beginning with B is less than 6, so we retain the value 6 in the cells BC and BD (CB and DB)

- (iii) B to E: The direct connection between B and E is 18, which is greater than all the values in the cells beginning with B, so we compute the distance of the following detours;

$$B \xrightarrow{9} A \xrightarrow{7} E = 16$$

$$B \xrightarrow{6} D \xrightarrow{12} E = 18$$

$$B \xrightarrow{6} D \xrightarrow{9} E = 15$$

$$B \xrightarrow{8} F \xrightarrow{12} E = 20$$

Now $\min \{16, 18, 15, 20\} = 15 < 18$, and so we replace the values of the cells BE and EB with 15

- (iv) B to F: (now the shortest distance between B and F is 8) but the values in the cells BC and BD are all less than that of BF. We thus compute the distances of the detour;

$$B \xrightarrow{6} C \xrightarrow{\infty} F = \infty$$

$$B \xrightarrow{6} D \xrightarrow{\infty} A = \infty$$

$\min \{\infty, \infty\} = \infty$ which is not less than 8, so we retain the value in the cell BF and FB.

Now from node B to the other nodes, we obtain the table;

Table 3.11 Distance Matrix 4

	A	B	C	D	E	F
A	-	9	3	7	7	10
B	9	-	6	6	15	8
C	3	6	-	4	12	∞
D	7	6	4	-	9	∞
E	7	15	12	9	-	12
F	10	8	∞	∞	12	-

Now from node c

(i) The value in the cell $CA = 3$ is less than all the other values in the cells beginning with C, so we retain the value 3 in the cell CA and AC

(ii) From $C \rightarrow B$: the value in the cell $CB = 6$ but those in the cells CA and CD are less than 6 so we compute the distances of the detours;

$$C \xrightarrow{3} A \xrightarrow{9} B = 12$$

$$C \xrightarrow{4} D \xrightarrow{6} B = 10$$

$\text{Min} \{12, 10\} = 10 > 6$, so we shall retain the values in cells CB and BC.

(iii) $C \rightarrow D$: The value in the cell $CD = 4$ and only the cell CA is less than 4, so we evaluate the distance of the detour $D \xrightarrow{3} A \xrightarrow{7} D = 10 > 4$, so we retain 4 in the cells cd and DC

(iv) From $C \rightarrow E$: The value in the cell $CE = 12$, but those in the cells CA, CB, CD are all less than 12 so we evaluate the distance of the detours.

$$C \xrightarrow{3} A \xrightarrow{7} E = 10$$

$$C \xrightarrow{6} B \xrightarrow{8} E = 14$$

$$C \xrightarrow{4} D \xrightarrow{9} E = 13$$

$\text{Min} \{10, 14, 13\} = 10 < 12$, so we replace 12 by 10 in the cells CE and EC. C to

F: For C to F, we compute the distances of the following detours;

$$C \xrightarrow{3} A \xrightarrow{10} F = 13$$

$$C \xrightarrow{6} B \xrightarrow{8} F = 14$$

$$C \xrightarrow{4} D \xrightarrow{\infty} F = \infty$$

$$C \xrightarrow{12} E \xrightarrow{12} F = 24$$

$\text{Min } \{13, 14, \infty, 24\} = 13 < \infty$, so we replace the value in the cells CF and FC by 13

After node C, we have the following table;

Table 3.12 Distance Matrix 5

	A	B	C	D	E	F
A	-	9	3	7	7	10
B	9	-	6	6	15	8
C	3	6	-	4	10	13
D	7	6	4	-	9	∞
E	7	15	10	9	-	12
F	10	8	13	∞	12	-

From node D

- (i) D to A: Let us consider the value in the cell DA = 7, here the values in the cells DB and DC are all less than 7. We compute the values of the distance of the detours

$$D \xrightarrow{6} B \xrightarrow{9} A = 15$$

$$D \xrightarrow{4} C \xrightarrow{3} A = 7$$

$\text{Min } \{15, 7\} = 7$ which is not less than 7 so we retain the values in the cells DA and AD.

- (ii) D to B: Here, we shall only compute the distance of the detour $D \xrightarrow{4} C \xrightarrow{6} B = 10 > 6$, we retain the value 6 in the cells DB and BD.
- (iii) D to C: The value in the cell DC is less than all the other values in the beginning with D so we retain it.

- (iv) D to E: For DE, we compute the distances for the following detours;

$$D \xrightarrow{7} A \xrightarrow{7} E = 14$$

$$D \xrightarrow{6} B \xrightarrow{15} E = 21$$

$$D \xrightarrow{4} C \xrightarrow{10} E = 24$$

Min {14, 21, 24} = 14 which is not less than 9, and so we retain the value 9 in the cells DE and ED.

- (v) D to F: For DF, we compute the distances of the following detours;

$$D \xrightarrow{7} A \xrightarrow{10} F = 17$$

$$D \xrightarrow{6} B \xrightarrow{8} F = 14$$

$$D \xrightarrow{4} C \xrightarrow{13} F = 17$$

$$D \xrightarrow{9} E \xrightarrow{12} F = 21$$

Min {17, 14, 17, 21} = 14 < ∞, so we replace ∞, in the cells DF and FD by 14.

Here, we obtain the following table,

Table 3.13 Distance Matrix 6

	A	B	C	D	E	F
A	-	9	3	7	7	10
B	9	-	6	6	15	8
C	3	6	-	4	10	13
D	7	6	4	-	9	14
E	7	15	10	9	-	12
F	10	8	13	14	12	-

From node E

- (i) E to A: The value in the cell EA = 7 is less than the values in all the cells beginning with E, and so we retain it.

- (ii) E to B: We compute the distances for the following detours;

$$E \xrightarrow{7} A \xrightarrow{9} B = 16$$

$$E \xrightarrow{10} C \xrightarrow{6} B = 16$$

$$E \xrightarrow{9} D \xrightarrow{6} B = 15$$

$$E \xrightarrow{12} F \xrightarrow{8} B = 20$$

Min {16, 16, 15, 20} = 15 which is not less than 15 so we retain it.

(iii) E to C: We compute the distances for the following detours;

$$E \xrightarrow{7} A \xrightarrow{3} C = 10$$

$$E \xrightarrow{9} D \xrightarrow{4} C = 13$$

Min {10, 13} = 10 which is not less than 10 and so we retain it.

(iv) E to D: We compute the distance of the detour $E \xrightarrow{7} A \xrightarrow{7} D = 14$ which is not less than 9 so we retain the value 9 in the cells ED and DE.

(v) E to D: We compute the distance for the following detours;

$$E \xrightarrow{7} A \xrightarrow{10} F = 17$$

$$E \xrightarrow{10} C \xrightarrow{13} F = 23$$

$$E \xrightarrow{9} D \xrightarrow{14} F = 23$$

Min {17, 23, 23} = 17 which is not less than 12 and so we retain the value 12 in the cells EF and FE.

For the node E, there were no changes so we shall retain the previous table.

Since the values in the tables are symmetric, after computing the value in EF we can stop. Hence the shortest distances between all pairs of nodes is as shown in the table below:

Table 3.14 Distance Matrix 7

	A	B	C	D	E	F
A	-	9	3	7	7	10
B	9	-	6	6	15	8
C	3	6	-	4	10	13
D	7	6	4	-	9	14
E	7	15	10	9	-	12
F	10	8	13	14	12	-

3.3 SUMMARY

In this Chapter we treated the mathematical model and the theories used in solving the Minimum Spanning Tree problem and the Shortest Path problem. In the next chapter we shall put forward the data collection and the analysis of the data for the study.

KNUST



CHAPTER 4

DATA ANALYSIS AND RESULTS

4.0 INTRODUCTION

This chapter analyses the data used to determine the minimum spanning tree route for the National Interconnected Transmission System, and the shortest path between the various nodes in the network. The data analysis was done by using a developed MATLAB programme. Prim's algorithm was used to determine the minimum spanning tree, and Floyd Warshall's Algorithm was used to determine the shortest path route for our study.

4.1 DATA COLLECTION

The data used for the study were collected from the Ghana Grid Company Ltd (GRIDCo) and the Centre for Remote Sensing and Geographic Information Services (CERSGIS). Data was collected on all 161KV Substations across the country, totaling forty seven (47). A GPS device was used to pick the geographic location of each substation. The data collected was analyzed using the ArcGIS software to determine the distance from one substation to the other. All distances are in kilometers (km). The name and type of substation was provided by GRIDCo. The data together with the distances calculated were put together to form a distance node matrix in Table 4.1 (in Appendix II) in an Excel Spreadsheet. Where there is a direct link between two substations a real value is assigned. The distance node matrix was used to construct a network in Figure 4.1.

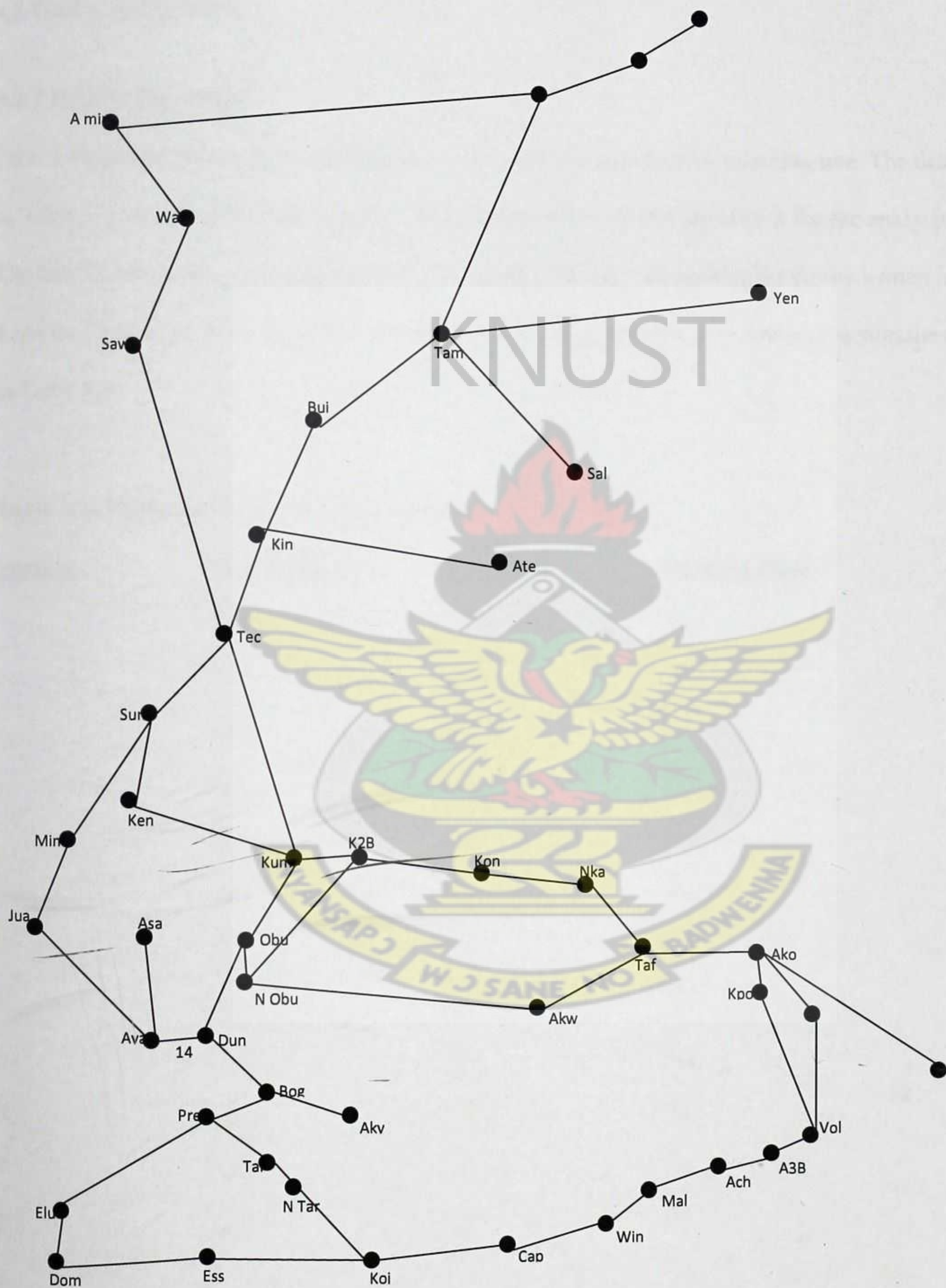


Figure 4.1 Network of the National Interconnected Transmission System

4.2 DATA ANALYSIS

4.2.1 Prim’s Algorithm

Prim’s algorithm was used for the analysis of the data to find minimum spanning tree. The data in Table 4.1 was used as input into the MATLAB program (PrimsAlgorithm) for the analysis. The MATLAB program is in appendix I. The output from the code starting from was written in a matrix form in an excel spreadsheet in Table 4.2 (in Appendix II), it is however summarized in Table 4.3.

Table 4.3: Minimum Spanning Tree Table

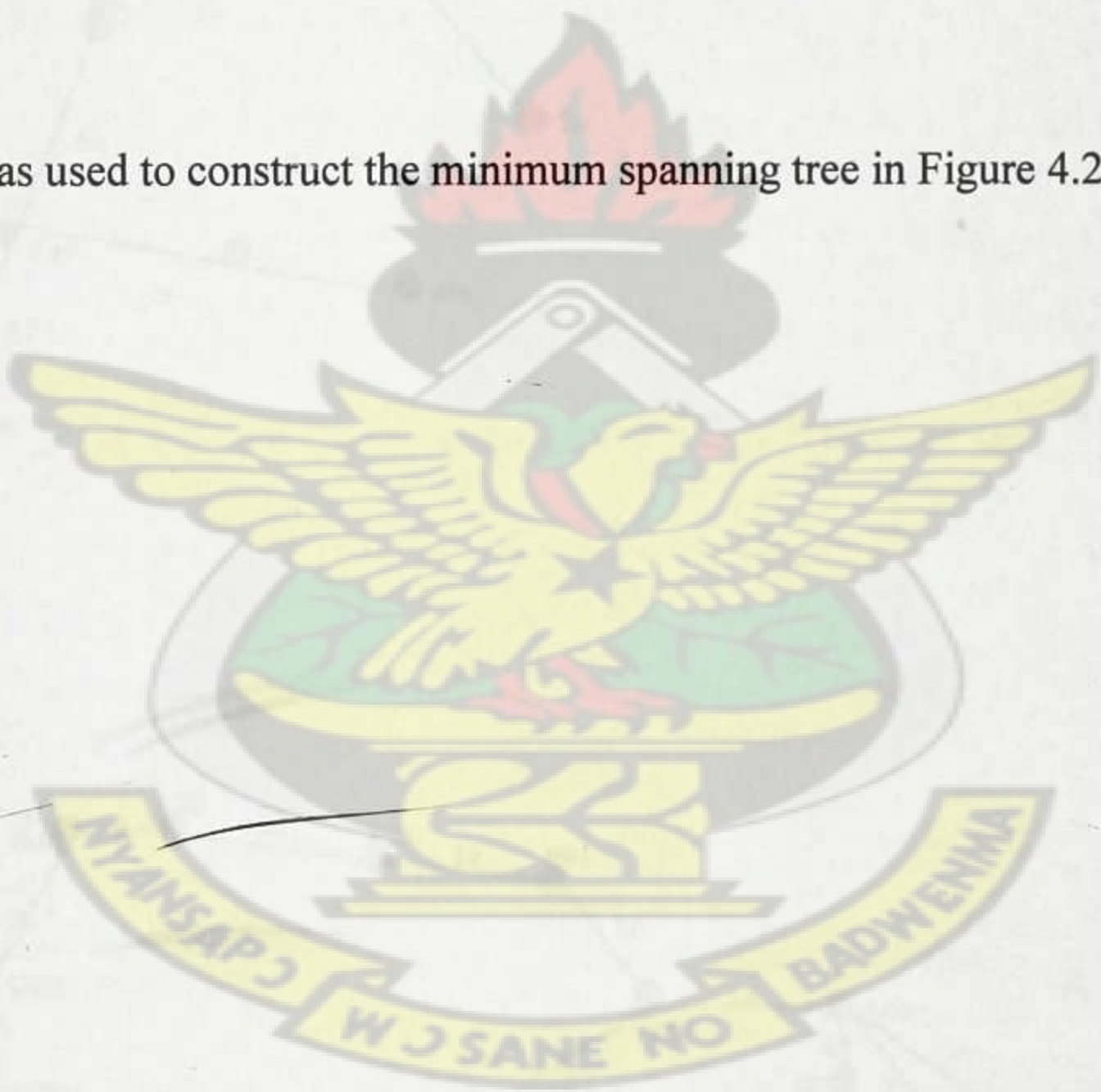
<u>Iteration</u>	<u>Starting Node</u>	<u>End Node</u>	<u>Distance (km)</u>
1	Vol	KGS	47
2	Win	Mal	43
3	Kpo	Ako	13
4	Tar	Pre	21
5	Pre	Bog	20
6	Mal	Ach	10
7	Ach	A3B	6
8	Cap	Koj	55
9	Bog	Dun	51
10	Koj	N Tar	40
11	N Tar	Tar	7
12	Taf	Ako	48
13	N Obu	Obu	5
14	Obu	Kum	55

15	Dun	Obu	28
16	Saw	Tec	194
17	Bol	Tam	154
18	Sun	Ken	43
19	Tec	Sun	51
20	Ken	Kum	91
21	Ess	Dom	46
22	Elu	Pre	73
23	Nka	Taf	56
24	Kon	Nka	52
25	Baw	Zeb	35
26	Tam	Bui	97
27	Kum	K2B	8
28	KGS	Ako	21
29	Akw	Taf	52
30	Kin	Tec	57
31	Bui	Kin	83
32	Aya	Jua	110
33	Asa	Aya	48
34	Ate	Kin	89
35	Min	Sun	55
36	ZeB	Bol	35
37	Aky	Bog	33
38	Afl	Ako	125
39	Yen	Tam	95

40	Jua	Mim	70
41	Dom	Elu	24
42	K2B	Kon	36
43	Sal	Tam	101
44	A Min	Wa	51
45	Wa	Saw	88
46	A3B	Vol	20

KNUST

The output generated was used to construct the minimum spanning tree in Figure 4.2.



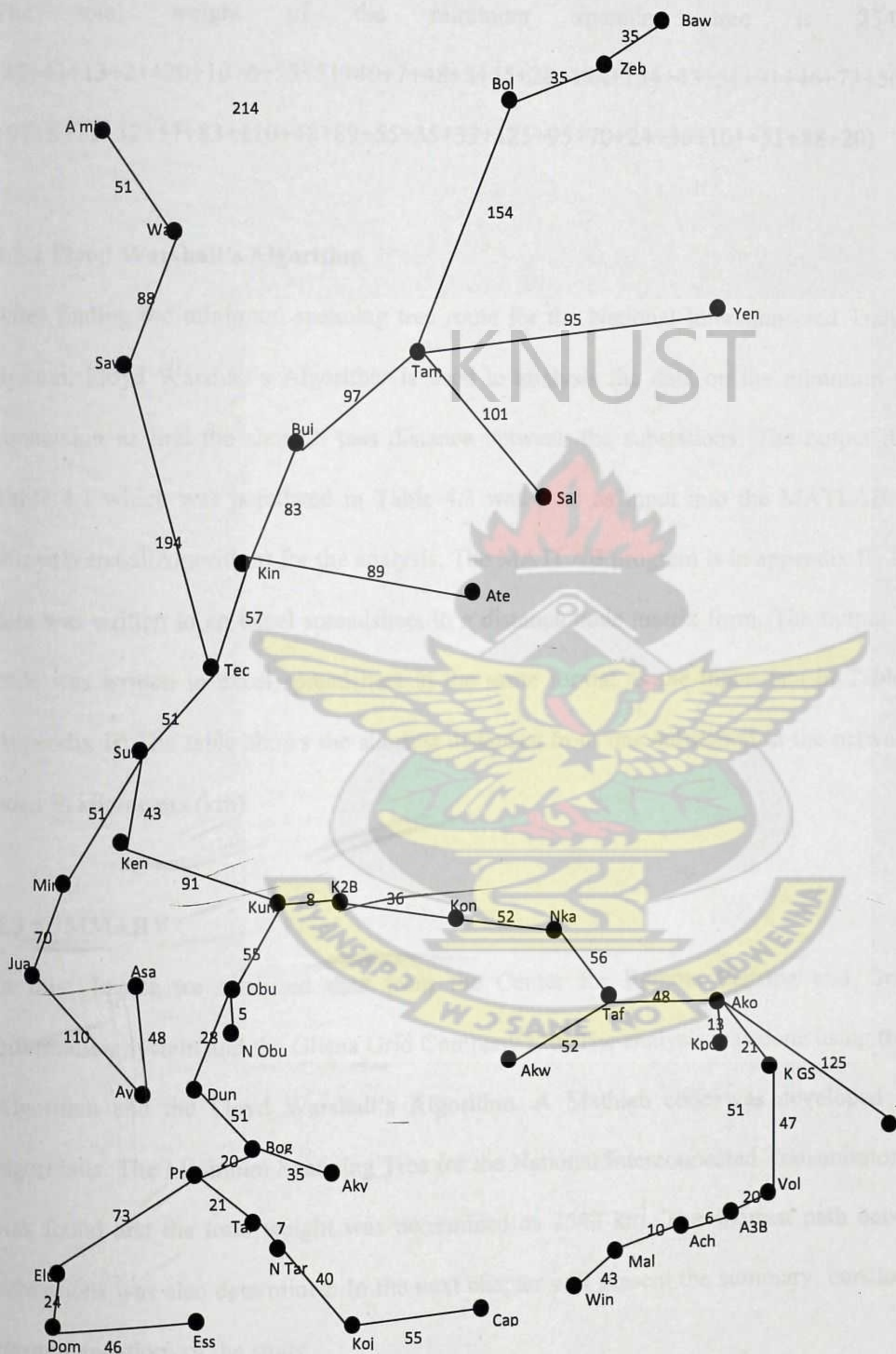


Figure 4.2: Minimum Spanning Tree for the National Interconnected Transmission System

The total weight of the minimum spanning tree is 2542 km
(47+43+13+21+20+10+6+55+51+40+7+48+5+55+28+194+154+43+51+91+46+73+56+52+35
+97+8+21+52+57+83+110+48+89+55+35+33+125+95+70+24+36+101+51+88+20)

4.2.2 Flyod Warshall's Algorithm

After finding the minimum spanning tree route for the National Interconnected Transmission System, Floyd Warshall's Algorithm is used to analysis the data on the minimum spanning connection to find the shortest part distance between the substations. The output data from Table 4.1 which was populated in Table 4.3 was used as input into the MATLAB program (FloydWarshallAlgorithm) for the analysis. The MATLAB program is in appendix II. The input data was written in an excel spreadsheet in a distance node matrix form. The output from the code was written in excel spreadsheet in the same format as the input data in Table 4.4 (in Appendix II).The table shows the shortest distances from one substation in the network to the other in kilometers (km).

4.3 SUMMARY

In this chapter we analyzed data from the Center for Remote Sensing and Geographic Information System and the Ghana Grid Company Ltd. The analysis was done using the Prim's Algorithm and the Floyd Warshall's Algorithm. A Mathlab code was developed for both algorithms. The Minimum Spanning Tree for the National Interconnected Transmission System was found and the total weight was determined as 2542 km. The shortest path between the substations was also determined. In the next chapter will present the summary, conclusion and recommendations of the study.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

The study sought to find the Minimum Spanning Tree (MST) for the National Interconnected Transmission System and the shortest path that connect the nodes between them. The first objective was to find shortest route, Minimum Spanning Tree that connects the substations in the National Interconnected Transmission System, covering the substations in the network, affording minimum cost and thus connecting all substations under consideration. The second objective was to find the shortest path that connects one substation to the other in the network.

A total of forty-seven (47) 161Kv substations across the country were used for the study. The data used for the study was gathered from the Centre for Remote Sensing and Geographic Information services (CERGIS) and the Ghana Grid Company Ltd (GRIDCo).

Prim's Algorithm was used as the mathematical tool to solve the minimum spanning tree problem. A Matlab code for Prim's Algorithm was developed to compute the minimum spanning tree. The analysis showed that a total of two thousand five hundred and forty-two kilometers (2542 km) distance long will be covered by connecting the substations used in the study. Floyd Warshall's Algorithm was used as the mathematical tool to solve the shortest path problem. A Matlab code for Floyd Warshall's Algorithm was developed to compute the shortest path. The shortest paths between the substations were computed.

5.2 CONCLUSIONS

The study presented forty-seven (47) 161kv substations on the National Interconnected Transmission System. Prim's Algorithm was applied to the network to construct the minimum spanning tree. The study shows that the minimum total distance connecting the substation on the National Interconnected Transmission System is two thousand five hundred and forty-two kilometers (2542 km). The study also used Floyd Warshall's Algorithm to further determine the shortest path between the substations on the minimum spanning tree.

5.3 RECOMMENDATIONS

In order to reduce cost in terms of materials used in constructing grid lines and increase efficiency in the transmission of electric power in our energy sector, it is recommended that the Ghana Grid Company Ltd (GRIDCo), who operates the transmission sector in our Country, adopts the Prim's Algorithm model problem in the construction of lines to connect their substations.

It is also recommended that GRIDCo adopts the Floyd Warshall's Algorithm model Problem to determine the shortest path connecting their substations to enhance response to emergency situation.

Finally the approach from the research can be applied to problems that applies to Prim's and Floyd Warshall's Algorithms.

REFERENCES

1. Ahuja, R. K., Mehlhorn, K., Orlin, J. B., and Tarjan, R. E. (2011). Faster algorithms for the shortest path problem.
2. Ahuja, R. K., Orlin, J. B., and Sharma, D. (2001). Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91(1), 71-97.
3. Amponsah S. K. and Darkwah F. K. (2007). Operations Research. Kwame Nkrumah University of Science and Technology.
4. Barr, R. S., Helgaon, R. V., and Kennington, J. L. (1989). Minimal spanning trees: An empirical investigation of parallel algorithms. *Parallel Computing*, 12(1), 45-52
5. Beasley, J. E., and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19(4), 379-394.
6. Bentley, J. L. (1980). A parallel algorithm for constructing minimum spanning trees. *Journal of algorithms*, 1(1), 51-59.
7. Bigg, N. L., Lloyd, E. K., and Wilson, R. J. (1976). *Graph Theory 1736-1936*. Oxford University Press on Demand.
8. Borgwardt, K. M., and Kriegel, H. P. (2005, November). Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on* (pp. 8-pp). IEEE.
9. Chabini, I. (1998). Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record: Journal of the Transportation Research Board*, 1645(1), 170-175.

10. Denning, S. (2006). Effective storytelling: strategic business narrative techniques. *Strategy and Leadership*, 34(1), 42-48.
11. Dunn, J. C. (1974). A graph theoretic analysis of pattern classification via Tamura's fuzzy relation. *Systems, Man and Cybernetics, IEEE Transactions on*, (3), 310-313.
12. Dusser, C., Rasigni, M., Palmari, J., Rasigni, G., Llebaria, A., and Marty, F. (1987). Minimal spanning tree analysis of biological structures. *Journal of theoretical biology*, 125(3), 317-323.
13. Dussert, C., Rasigni, G., Rasigni, M., Palmari, J., and Llebaria, A. (1986) Minimal spanning tree: A new approach for studying order and disorder. *Physical Review B* 34, no. 5
14. Erdős, P., and Rényi, A. (1959). On random graphs. *Publicationes Mathematicae Debrecen*, 6, 290-297.
15. Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3), 216-229.
16. Gonina, E. and Kalé, L. V. (2007). Parallel Prim's algorithm on dense graphs with a novel extension.
17. Gower, J. C., and Ross, G. J. S. "Minimum spanning trees and single linkage cluster analysis." *Applied statistics* (1969): 54-64
18. Graham, R. L., and Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1), 43-57.

19. Han, S. C., Franchetti, F., and Püschel, M. (2006, September). Program generation for the all-pairs shortest path problem. In *Proceedings of the 15th international conference on Parallel architectures and compilation techniques* (pp. 222-232). ACM.
20. Handler, G. Y., and Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks*, 10(4), 293-309.
21. Harish, P., and Narayanan, P. (2007). Accelerating large graph algorithms on the GPU using CUDA. *High Performance Computing-HiPC 2007*, 197-208.
22. Hassin, R. (1992). Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research* 17, no. 1: 36-42.
23. Ioachim, I., Gelinas, S., Soumis, F., and Desrosiers, J. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3), 193-204.
24. Howland, J. and Murrow, D., ENE, Petraglia, L. and Comings, T. (October 2009) Economic development Research Group Inc.
25. Katz, G. J., and Kider Jr, J. T. (2008, June). All-pairs shortest-paths for large graphs on the GPU. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (pp. 47-55). Eurographics Association
26. Kershenbaum, A., and Van Slyke, R. (1972, August). Computing minimum spanning trees efficiently. In *Proceedings of the ACM annual conference-Volume 1* (pp. 518-527). ACM
27. Kossinets, G., and Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757), 88-90.

28. Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1), 48-50.
29. Lawler, E. L. (1972). A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7), 401-405.
30. Martins, E. Q. V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2), 236-245.
31. Michailidis, G. (2005). Minimum Spanning Tree. *Encyclopedia of Statistics in Behavioral Science*
32. Moret, B. and Shapiro, H. (1991). An empirical analysis of algorithms for constructing a minimum spanning tree." *Algorithms and Data Structures*: 400-411.
33. Nešetřil, J., Milková, E., and Nešetřilová, H. (2001). Otakar Borůvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1), 3-36.
34. Okada, S., and Soper, T. (2000). A shortest path problem on a network with fuzzy arc lengths. *Fuzzy Sets and Systems*, 109(1), 129-140.
35. Palottino, S. (2006). Shortest-path methods: Complexity, interrelations and new propositions. *Networks*, 14(2), 257-267.
36. Pettie, S., and Ramachandran, V. (2002, January). Computing shortest paths with comparisons and additions. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 267-276). Society for Industrial and Applied Mathematics.

37. Stoer, M., and Wagner, F. (1994). A simple min cut algorithm. *Algorithms—ESA'94*, 141-147.
38. Vandervalk, B. P., McCarthy, E. L., and Wilkinson, M. D. (2009, August). Optimization of Distributed SPARQL Queries Using Edmonds' Algorithm and Prim's Algorithm. In *Computational Science and Engineering, 2009. CSE'09. International Conference on* (Vol. 1, pp. 330-337). IEEE.
39. Wang, W., Yongzhong H., and Shaozhong G. (2011). Design and Implementation of GPU-Based Prim's Algorithm. *International Journal of Modern Education and Computer Science (IJMECS)* 3, no. 4: 55.
40. Watts, D. J., and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *nature*, 393(6684), 440-442.
41. Zhou, G., and Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114(1), 141-152.

APPENDIX I

```
function PrimsAlgorithm()
% THIS FUNCTION RECIEVES INPUT DATA FROM AN EXCEL WORK SHEET AND APPLIES
% PRIM'S ALGORITHM TO THE DATA AFTER WHICH THE RESULT IS WRITTEN TO ANOTHER
% EXCEL WORK SHEET.
clear all
clc

disp('*****');
disp('*****');
disp('** EXECUTION OF PRIMS ALGORITHM');
disp('** MSc. INDUSTRIAL MATHEMATICS');
disp('*****');
disp('*****');

disp(' ');
disp('READING OF INPUT DATA FROM EXCEL WORK SHEET BEGINS');
disp(' ');
disp('WAIT FOR READING TO COMPLETE.....');
tic; %Timer started
[numeric_data, text_data] = xlsread('Test.xlsx','Input'); % Reading of input worksheet
time_used = toc; %Timer ends
disp(' ');
fprintf('INPUT DATA WAS READ FROM EXCEL WORK SHEET IN :%10s SECONDS\n',num2str(time_used));

numeric_content_of_matrix = numeric_data; %Numeric content of the excel work sheet is read into this variable.
text_content_of_matrix = text_data; %Text content of the excel work sheet is read into this variable.

size_of_matrix = size(numeric_content_of_matrix); %The number of rows and columns of the numeric content of the excel work sheet is read as a row vector.

if size_of_matrix(1) == size_of_matrix(2) %Check is done to ensure that the number of rows and columns are the same
disp(' ');
fprintf('SELECT THE NODE TO START WITHIN THE RANGE 1,2,... %5s\n',num2str(size_of_matrix(1)));

while_loop_control_1 = 0;
while_loop_stop_value_1 = 1;

tic; %Timer started
while while_loop_control_1 < while_loop_stop_value_1 %Looping will continue until a value within range is entered
```



```

starting_node = input('ENTER STARTING NODE NUMBER HERE >> ');
starting = round(starting_node);
if starting <= size_of_matrix(1) && starting >= 1
cancelled_rows = zeros(1,size_of_matrix(1)); %All rows that have been canceled will be stored in this row vector.
cancelled_rows_column_pointer = 0; %Points at the highest column number in "cancelled_rows" that has been used

%The starting row is cancelled at this point
for col = 1:(size_of_matrix(1))
    %if isnan(numeric_content_of_matrix(starting, col)) ~=
    %1 % 1 means is not a number
    %numeric_content_of_matrix(starting, col) = NaN;
    %end
    numeric_content_of_matrix(starting, col) = NaN;
end

cancelled_rows_column_pointer = cancelled_rows_column_pointer + 1;
cancelled_rows(cancelled_rows_column_pointer) = starting; %Contains the row numbers that has been canceled
minimum_value = inf;
row_of_minimum_value = 0;
col_of_minimum_value = 0;
while_loop_control_2 = 0;
while_loop_stop_value_2 = 1;
cancelled_rows_checker = 0;

while while_loop_control_2 < while_loop_stop_value_2
for col_to_be_checked = 1:cancelled_rows_column_pointer
for row_to_be_checked = 1:size_of_matrix(1)
for rows_already_cancelled = 1:cancelled_rows_column_pointer
if row_to_be_checked == cancelled_rows(rows_already_cancelled)
cancelled_rows_checker = cancelled_rows_checker + 1;
end
end
if cancelled_rows_checker == 0
if isnan(numeric_content_of_matrix(row_to_be_checked, cancelled_rows(col_to_be_checked))) ~= 1 % 1 means is not a number
if minimum_value > numeric_content_of_matrix(row_to_be_checked, cancelled_rows(col_to_be_checked))
minimum_value = numeric_content_of_matrix(row_to_be_checked, cancelled_rows(col_to_be_checked));
row_of_minimum_value = row_to_be_checked;
col_of_minimum_value = cancelled_rows(col_to_be_checked);
end
end
end

```



```

end
else
    cancelled_rows_checker = 0;
end
end
end

if col_of_minimum_value == 1
    for col = 2:size_of_matrix(1)
        numeric_content_of_matrix(row_of_minimum_value,col) = NaN;
    end
elseif col_of_minimum_value == size_of_matrix(1)
    for col = 1:(size_of_matrix(1) - 1)
        numeric_content_of_matrix(row_of_minimum_value,col) = NaN;
    end
else
    for col = 1:size_of_matrix(1)
        if col ~= col_of_minimum_value
            numeric_content_of_matrix(row_of_minimum_value,col) = NaN;
        end
    end
end

cancelled_rows_column_pointer = cancelled_rows_column_pointer + 1;
cancelled_rows(cancelled_rows_column_pointer) = row_of_minimum_value; %Contains the row numbers that has been canceled

if cancelled_rows_column_pointer == size_of_matrix(1)
    while_loop_control_2 = 1;
end
minimum_value = inf; %Reset this value before the next looping.
end

time_used = toc;
disp(' ');
fprintf('PRIMS ALGORITHM EXECUTED IN :%10s SECONDS\n',num2str(time_used));

while_loop_control_3 = 0;
while_loop_stop_value_3 = 1;

%Specify whether data will be written in decimal places of

```



```

%integers
while while_loop_control_3 < while_loop_stop_value_3
    disp(' ');
    data_format = input('ENTER THE NUMBER 1 FOR INTEGER OUTPUT OR 2 FOR OUTPUT WITH THE SAME FORMAT AS INPUT DATA\n >>');
    if data_format == 1
        for col = 1:size_of_matrix(2)
            for row = 1:size_of_matrix(1)
                numeric_content_of_matrix(row, col) = round(numeric_content_of_matrix(row, col));
            end
        end
    end
    while_loop_control_3 = 1;
elseif data_format == 2
    while_loop_control_3 = 1;
else
    while_loop_control_3 = 0;
end
end

disp(' ');
disp('OUTPUT DATA IS BEING WRITTEN TO EXCEL WORK SHEET');
disp(' ');
disp('WAIT FOR WRITTING TO COMPLETE.....');

tic; %Timer started
xlswrite('Test.xlsx',text_content_of_matrix,'Output'); % Writting to output worksheet (Text Portion)
xlswrite('Test.xlsx',numeric_content_of_matrix,'Output','B2'); % Writting to output worksheet (Numeric Portion)
time_used = toc; %Timer ends
disp(' ');
fprintf('WRITING COMPLETED SUCCESSFULLY IN :%10s SECONDS\n',num2str(time_used));
while_loop_control_1 = 1; %Causes the while loop to be exited
disp(' ');
disp('*****');
disp('*****');
disp('*****');
disp('** END OF EXECUTION OF PRIMS ALGORITHM **');
disp('*****');
disp('*****');
disp('*****');
else
    disp('')
    disp('KINDLY ENTER AN INTERGER NUMBER WITHIN THE SPECIFIED RANGE. ');
end

```



```
end
else
    fprintf('THE MATRIX IS NOT A SQUARE MATRIX. THE ROWS AND COLUMNS ARE :%10s\n',num2str(size_of_matrix));
end
end
```



APPENDIX II

```
function FloydWarshallAlgorithm()  
% THIS FUNCTION RECIEVES INPUT DATA FROM AN EXCEL WORK SHEET AND APPLIES  
% FLOYD WARSHALL'S ALGORITHM TO THE DATA AFTER WHICH THE RESULT IS WRITTEN TO ANOTHER  
% EXCEL WORK SHEET.
```

```
% VERY IMPORTANT NOTE  
% THE INPUT EXCEL SHEET MUST HAVE EMPTY CELLS FOR DISTANCES  
% THAT DO NOT EXIST
```

```
clear all  
clc
```

```
disp('*****');  
disp('*****');  
disp('** EXECUTION OF FLOYD WARSHALL ALGORITHM  
** MSc. INDUSTRIAL MATHEMATICS  
*****');  
disp('*****');
```

```
disp(' ');  
disp('READING OF INPUT DATA FROM EXCEL WORK SHEET BEGINS');  
disp(' ');  
disp('WAIT FOR READING TO COMPLETE.....');  
tic; %Timer started  
[numeric_data, text_data] = xlsread('Test.xlsx','Input'); % Reading of input worksheet  
time_used = toc; %Timer ends  
disp(' ');  
fprintf('INPUT DATA WAS READ FROM EXCEL WORK SHEET IN :%10s SECONDS\n',num2str(time_used));
```



```
numeric_content_of_matrix = numeric_data; %Numeric content of the excel work sheet is read into this variable.  
text_content_of_matrix = text_data; %Text content of the excel work sheet is read into this variable.
```

```
size_of_matrix = size(numeric_content_of_matrix); %The number of rows and columns of the numeric content of the excel work sheet is read  
as a row vector.
```

```
if size_of_matrix(1) == size_of_matrix(2) %Check is done to ensure that the number of rows and columns are the same
```

```
tic; %Timer started
```

```
numeric_content_of_matrix_final = zeros(size_of_matrix(1),size_of_matrix(2));
```

```
for row = 1:size_of_matrix(1)
```

```
for col = 1:size_of_matrix(2)
```

```
if row == col
```

```
numeric_content_of_matrix(row, col) = 0;
```

```
numeric_content_of_matrix_final(col, row) = 0;
```

```
else
```

```
if isnan(numeric_content_of_matrix(row, col)) == 1 % 1 means is not a number
```

```
numeric_content_of_matrix(row, col) = inf;
```

```
end
```

```
numeric_content_of_matrix_final(row, col) = inf;
```

```
end
```

```
end
```

```
end
```

```
initial_table_value = 0;
```

```
final_table_value = 0;
```

```
temp_value = inf;
```

```
temp_value_previous = inf;
```



```

current_value = 0;
val_to_be_written = inf; % MINIMUM DISTANCE THAT WILL BE COMPUTED

for row = 1:(size_of_matrix(1) - 1)
    % 1
    for col = row:size_of_matrix(2)
        % 2
        if row ~= col
            % 3
            current_value = numeric_content_of_matrix(row, col);
            if row == 1 && col == 2
                % 4
                val_to_be_written = current_value;
            end
        end
    end
end

for col1 = 3:size_of_matrix(2)
    initial_table_value = numeric_content_of_matrix(row, col1);

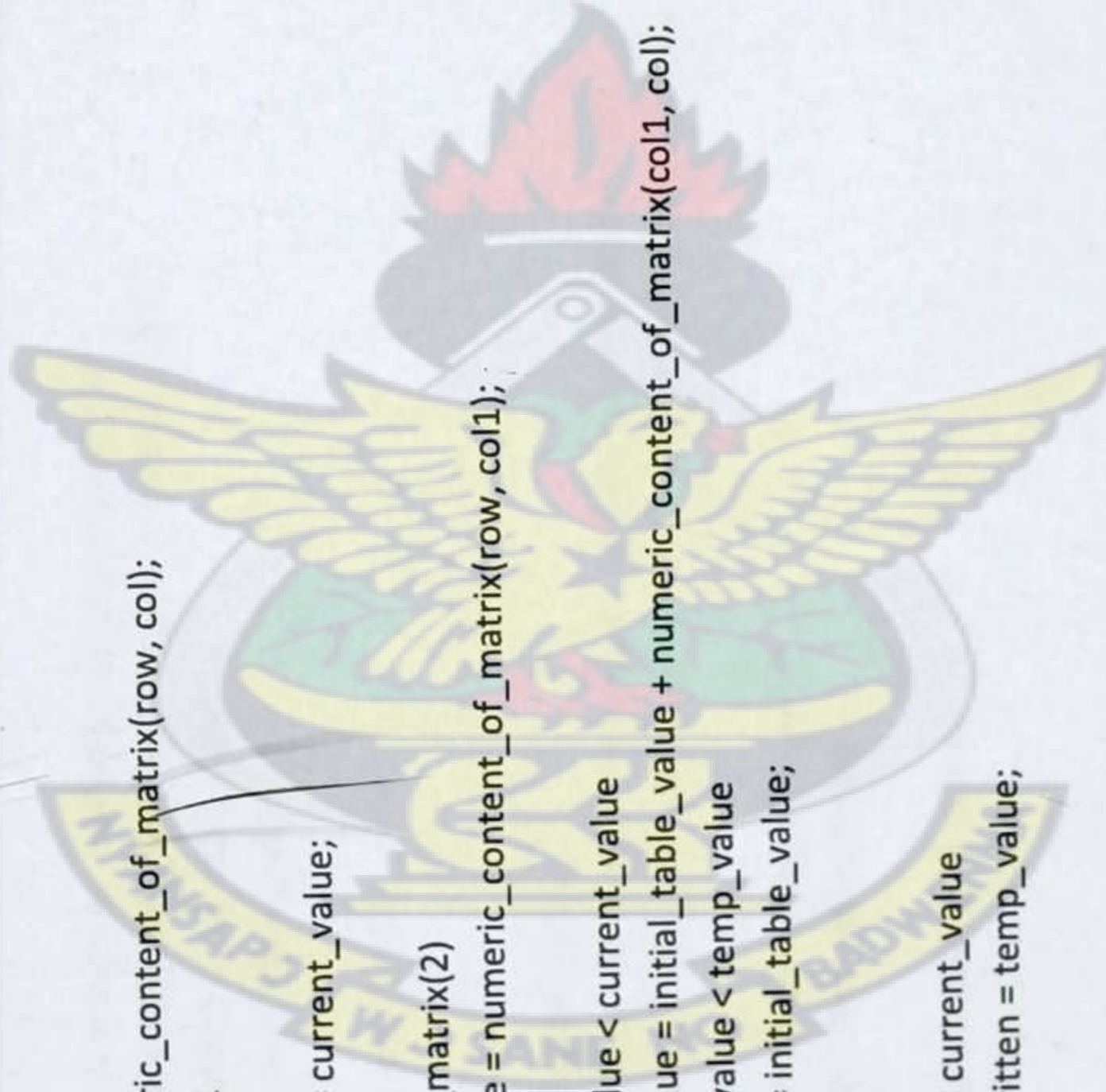
    if initial_table_value < current_value
        initial_table_value = initial_table_value + numeric_content_of_matrix(col1, col);
        if initial_table_value < temp_value
            temp_value = initial_table_value;
        end
    end

    if temp_value < current_value
        val_to_be_written = temp_value;
    end
end

numeric_content_of_matrix_final(row, col) = val_to_be_written;

```

KNUST




```
numeric_content_of_matrix_final(col, row) = val_to_be_written;
% 4
elseif col == size_of_matrix(2)
% 5
val_to_be_written = current_value;

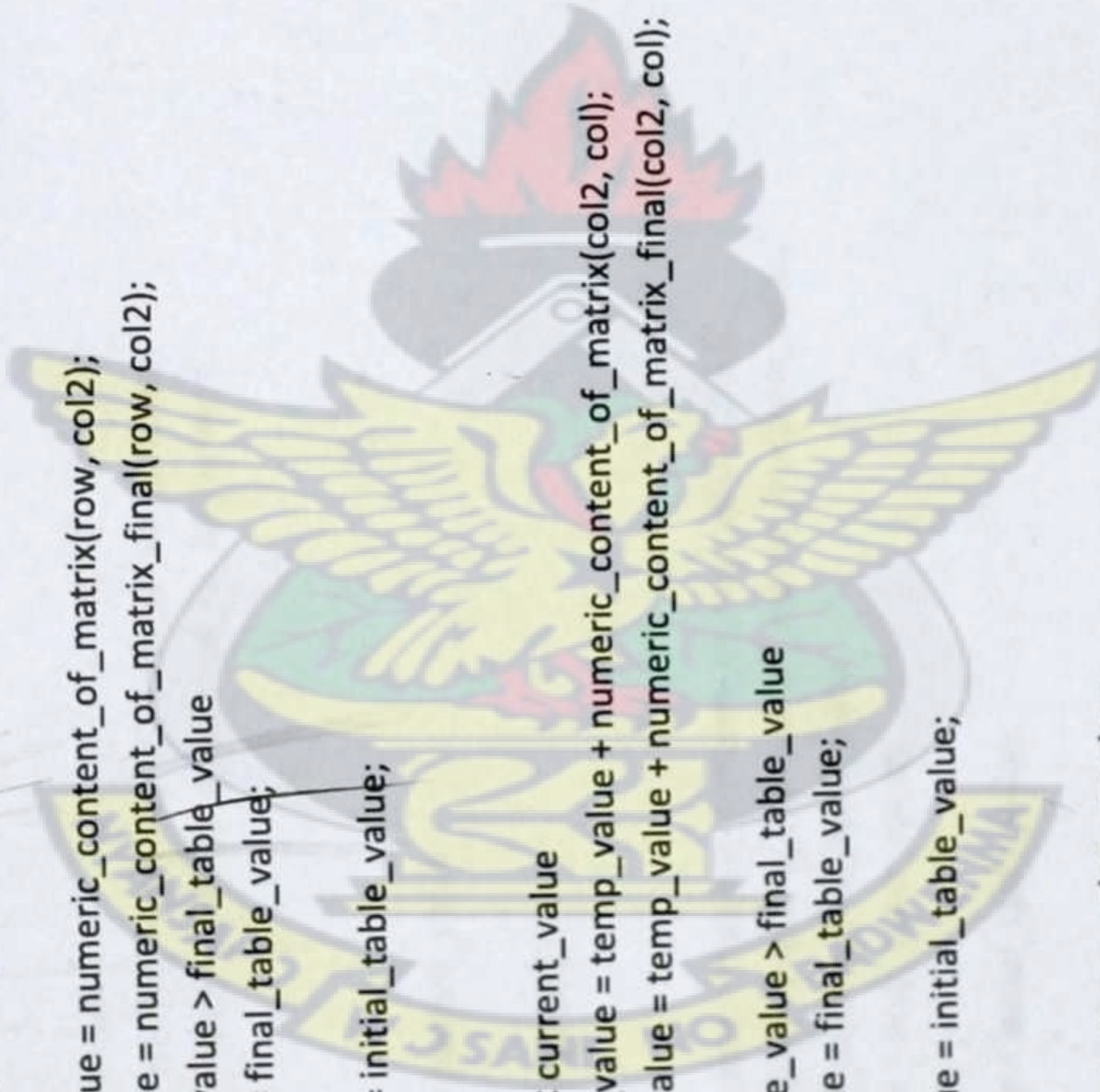
for col2 = 1:(size_of_matrix(2) - 1)
    if row ~= col2
        initial_table_value = numeric_content_of_matrix(row, col2);
        final_table_value = numeric_content_of_matrix_final(row, col2);
        if initial_table_value > final_table_value
            temp_value = final_table_value;
        else
            temp_value = initial_table_value;
        end

        if temp_value < current_value
            initial_table_value = temp_value + numeric_content_of_matrix(col2, col);
            final_table_value = temp_value + numeric_content_of_matrix_final(col2, col);

            if initial_table_value > final_table_value
                temp_value = final_table_value;
            else
                temp_value = initial_table_value;
            end

            if temp_value > temp_value_previous
                temp_value = temp_value_previous;
            else
                temp_value_previous = temp_value;
```

KNUST




```

end
if temp_value < current_value
    val_to_be_written = temp_value;
end
end
end
temp_value_previous = inf;
numeric_content_of_matrix_final(row, col) = val_to_be_written;
numeric_content_of_matrix_final(col, row) = val_to_be_written;
% 5
else
    % 6
    val_to_be_written = current_value;

for col3 = 1:size_of_matrix(2)
    if row ~= col3
        if col3 ~= col
            initial_table_value = numeric_content_of_matrix(row, col3);
            final_table_value = numeric_content_of_matrix_final(row, col3);
            if initial_table_value > final_table_value
                temp_value = final_table_value;
            else
                temp_value = initial_table_value;
            end
        end
    end

if temp_value < current_value
    initial_table_value = temp_value + numeric_content_of_matrix(col3, col);
    final_table_value = temp_value + numeric_content_of_matrix_final(col3, col);

```



```

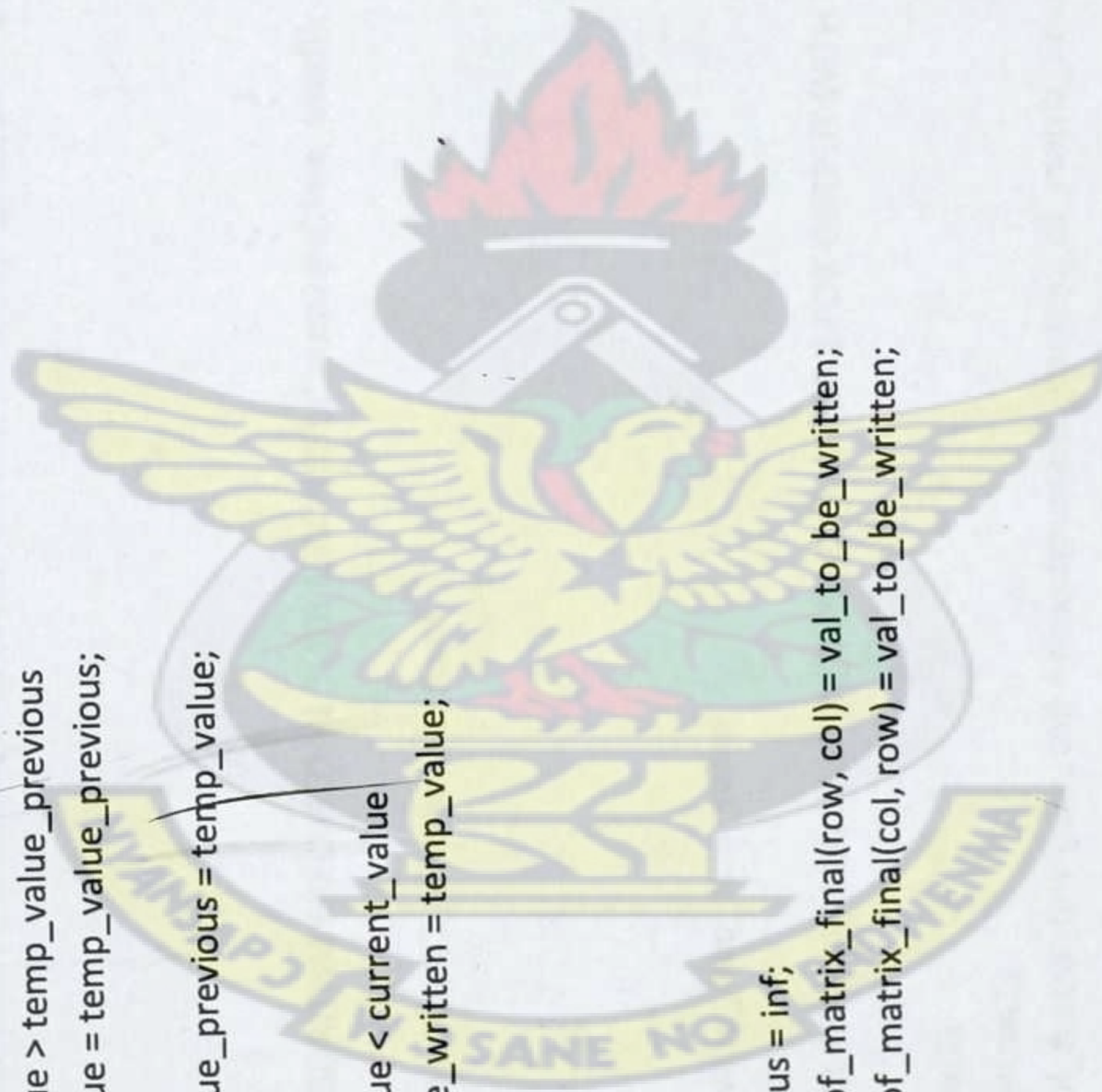
if initial_table_value > final_table_value
    temp_value = final_table_value;
else
    temp_value = initial_table_value;
end

if temp_value > temp_value_previous
    temp_value = temp_value_previous;
else
    temp_value_previous = temp_value;
end

if temp_value < current_value
    val_to_be_written = temp_value;
end
end
end
end
temp_value_previous = inf;
numeric_content_of_matrix_final(row, col) = val_to_be_written;
numeric_content_of_matrix_final(col, row) = val_to_be_written;
% 6
end
% 3
end
% 2
end
% 1

```

KNUST




```

end

for row = 1:size_of_matrix(1)
    for col = 1:size_of_matrix(2)
        if numeric_content_of_matrix_final(col, row) == inf
            numeric_content_of_matrix_final(col, row) = NaN;
        end
    end
end

time_used = toc;
disp(' ');
fprintf('FLOYD WARSHALL ALGORITHM EXECUTED IN :%10s SECONDS\n', num2str(time_used));

%Specify whether data will be written in decimal places of
%integers
while_loop_control = 0;
while_loop_stop_value = 1;

while while_loop_control < while_loop_stop_value
    disp(' ');
    data_format = input('ENTER THE NUMBER 1 FOR INTEGER OUTPUT OR 2 FOR OUTPUT WITH THE SAME FORMAT AS INPUT DATA\n >>');
    if data_format == 1
        for col = 1:size_of_matrix(2)
            for row = 1:size_of_matrix(1)
                numeric_content_of_matrix_final(row, col) = round(numeric_content_of_matrix_final(row, col));
            end
        end
    end
    while_loop_control = 1;
elseif data_format == 2

```



```

while_loop_control = 1;
else
while_loop_control = 0;
end
end

disp(' ');
disp('OUTPUT DATA IS BEING WRITTEN TO EXCEL WORK SHEET');
disp(' ');
disp('WAIT FOR WRITTING TO COMPLETE.....');

tic; %Timer started
xlswrite('Test.xlsx',text_content_of_matrix,'Output'); % Writting to output worksheet (Text Portion)
xlswrite('Test.xlsx',numeric_content_of_matrix_final,'Output','B2'); % Writting to output worksheet (Numeric Portion)
time_used = toc; %Timer ends
disp(' ');
fprintf('WRITING COMPLETED SUCCESSFULLY IN :%10s SECONDS\n',num2str(time_used));

disp(' ');
disp('*****');
disp('*****');
disp('** END OF EXECUTION OF FLOYD WARSHALL ALGORITHM **');
disp('*****');
disp('*****');
else
fprintf('THE MATRIX IS NOT A SQUARE MATRIX. THE ROWS AND COLUMNS ARE :%10s\n',num2str(size_of_matrix));
end
end

```


Distance Matrix of Substations

[illegible]

