International Journal in Management and Social Science
Volume 09 Issue 11, November 2021 ISSN: 2321-1784 Impact Factor: 7.088
Journal Homepage: http://ijmr.net.in, Email: irjmss@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal

GRAPH-THEORETIC ALGORITHMS AND COMPUTER PROGRAMS

SHANKAR S
Research Scholar, Sunrise University, Alwar, Rajasthan
DR MANJEET SINGH
Research Supervisor, Sunrise University, Alwar, Rajasthan

ABSTRACT

The study of algorithms for graphs is quickly developing and has reached the status of a traditional field of study. Those interested in applications of graph theory should prioritize learning how to utilize a digital computer in addressing graph-theoretic issues. Most real-world situations necessitating graph theory need massive, almost incomputable graphs. In fact, the introduction of the high-speed electronic computer is partly responsible for the current uptick in interest in graph theory. The first algorithm we develop will analyze a given network and identify its edges and nodes. The spanning-tree algorithms are some of the best-known and most-used algorithms in graph theory. To put it simply, a spanning-tree method will always produce exactly one spanning tree for each given linked network. Finding a collection of essential circuits in a given graph is a common task.

KEYWORDS: Digital, Algorithms, Graph, heory

INTRODUCTION

The information used to begin processing by an algorithm is called "inputs" (just as a recipe for a dish calls for raw ingredients). One or more graphs will be the obvious input for our algorithms here (or digraphs). Most graphs given to and saved in a digital computer will have one of the five main types. There are benefits and drawbacks to both. The graph, issue, programming language, computing platform, and dynamic changes to the graph during computation all have a role in the final decision.

Adjacency Matrix: Typically, an adjacency matrix is the most common representation of a graph or digraph that is supplied into a computer. The adjacency matrix is one tool defined and used by algorithms.Occasionally, a graph's data may also be stored and manipulated as a matrix called an incidence matrix. Incidence matrices like A are used by the procedure (G).

Alternatively, after arbitrarily numbering the n vertices, one may list all of the edges of the graph as vertex pairs, which is known as (c) Edge Listing. The following ordered pairings make up the digraph: (1, 2), (2, 1), (2, 4), (3, 2), (3, 3), (3, 4), (4, 1), (4, 1). (5,2). If the graph was undirected, we might disregard the ordering between each vertex pair. This form of a graph or digraph naturally accommodates parallel edges and self-loops. Each vertex has to be labeled with a value from 1 to n, and this takes b bits.

$2^{b-1} < n \leq 2^b$.

Since two such integers must be stored for each of the e edges, the total storage needed is

$$2e \cdot b \text{ bits.}$$

When compared to n^2, it is clear that this representation is more cost-effective than the adjacency matrix.

$$2e \cdot b < n^2$$

Thus, edge listing is the preferred technique of graph storage when the adjacency matrix of the graph is sparse. While edge listing is a simple way to enter a graph into a computer, it presents significant challenges when it comes to storing, retrieving, and manipulating the graph itself. Detecting if a graph is linked, for instance, would need more sophisticated search strategies.The final product of any algorithm is the finished meal. While every issue starts with a graph, the solution will be unique. Adjacency matrices may be printed by the software if the result is a collection of subgraphs. However, we may ask the computer to simply display YES or NO if the result is a yes or no to a query, such as whether or not a certain graph is planar.Also, we may request the graph's thickness or its planar representation depending on the response we get. Oftentimes, all we need from a shortest-path method is the ability to display the shortest distance (or route length) between any two given vertices (x, y). A shortest route between two points x and y might be reported as a list of edges (or vertices). ...and so on. Different algorithms provide a wide range of results.

The issue of the Koinsber bridge, which was first posed in 1735, is considered the seed from which graph theory sprang. The notion of a Eulerian Graph emerged as a solution to this issue. After researching the Koinsberg bridge issue, Euler developed a structure he dubbed a Eulerian graph to address it. Kuratowski used game problems to demonstrate the planarity of complete graphs and bipartite graphs, both of which had been proposed by A.F. Mobius in 1840. Gustav Kirchhoff used graph theoretical notions to the computation of currents in electrical networks or circuits in 1845, when he introduced the concept of tree (a linked graph without cycles). Thomas Gutherie discovered the renowned four-color issue in 1852. Later that year, in 1856, Thomas. P. Kirkman and William R.Hamilton investigated cycles on polyhydra and developed the idea of the Hamiltonian graph by analyzing itineraries that made precisely one stop at each of a set of specified locations. There was a puzzling issue that H.Dudeney brought up in 1913. The four-color dilemma was conceived of by an inventor, but it took Kenneth Appel and Wolfgang Haken a century to find a solution to it. The field of Graph Theory may trace its origins back to this era.

Caley analyzed the trees using certain specialized techniques of analysis from differential calculus. Many theoretical chemistry topics were affected by this. Because of this, enumerative graph theory was developed. However, Sylvester coined the name "Graph" in 1878, making an analogy between "Quantic invariants" and the co-variants of algebra and molecular diagrams. The subfield of graph theory known as "extremel graph theory" was founded in 1941 thanks to Ramsey's work on colorations. As early as 1969, Heinrich used computers to crack the four-color puzzle. Asymptotic graph connection is what sparked the development of random graph theory.

LITERATURE AND REVIEW

Pranav Patel et al (2013) The prevalence of computers in modern society is undeniable. Everything is moving toward a more digital focus. While this innovation is at its pinnacle the most of the highly utilized applications one way or the other employ graph theory, such search engines are mostly dependent on graphs. Graph theory has grown into a sizable area of study in mathematics as a result of decades of dedicated study. Real-world success requires a thorough familiarity of graph theory and the many types of graphs it contains. In this article, we show a variety of graphs, discussing their characteristics, defining them, and outlining their practical uses. We've done the legwork to figure out which graphs are crucial to the most consequential real-world applications, and we've explained how they work theoretically.

Dr.Dankan V Gowda et al (2021) Mathematics is an essential part of many disciplines. Graph theory is a significant branch of mathematics that is used to structural models. There have been recent innovations and shifts in various fields thanks to the underlying structural similarities between seemingly unlike things and technology. It was the Koinsberg Bridge issue in 1735 that marked the beginning of the area of graph theory. Information theory, electrical engineering, linguistics, physics and chemistry, computer network science, biotechnology, and graphical theoretical applications are only few of the fields covered in this study. Several works on graph theory relating to scheduling ideas, engineering tech applications, and a framework have been read.

VinuthaMs (2017) Graph theory has important applications outside mathematics. Graph theory is widely used for the modeling of a wide variety of relationships and processes in several contexts, including the physical, biological, social, and digital ones. Network, database management system, artificial intelligence, software architecture, design of algorithms, multiprocessing, data structure, image processing, etc., are only few of the many fields of computer science study that make use of graph theory principles. In this work, we focus on those uses of the idea of graph coloring and graph labeling that are most relevant to the field at hand.

SOME BASIC ALGORITHMS

Algorithm 1: Connectedness and Components

Whether you have a graph's adjacency matrix X, you may use permutations of the rows and columns to see if the matrix takes on a block-diagonal shape, which indicates that the graph is linked.

However, this approach might entail n! permutations, making it inefficient. Corollary B, which involves checking the matrix for zeros, is a more time-efficient alternative.

$$Y = X + X^2 + \ldots + X^{n-1}$$

As it requires a significant number of matrix multiplications, it is not particularly efficient either. An effective algorithm is as follows.

Adjoint fusion in the adjacency matrixj^{th} vertex to the i^{th}in which the vertex is created by logically combining the j^{th} row to the i^{th} row as well as the j^{th} column to the i^{th} column. (Remember that in logical adding 1 + 0 = 0 + 1 = 1 + 1 = 1 and 0 + 0 = 0.) Then the j^{th} row and the j^{th}columns from the matrix are thrown away. (If discarding the required rows and columns is too laborious or time-consuming, you may keep them in the matrix; just make sure they aren't taken into account in any fusions.) An OR logical operation immediately replaces parallel edges with a single edge, such that a self-loop arising from a fusion appears as a 1 in the main diagonal. They have no bearing on whether or not a graph is linked. Assuming n is the total number of vertices, the most fusions that may occur in this procedure is n-1. Additionally, the maximum execution time is proportional to n(n-1) as at most n logical additions are performed in each fusion.

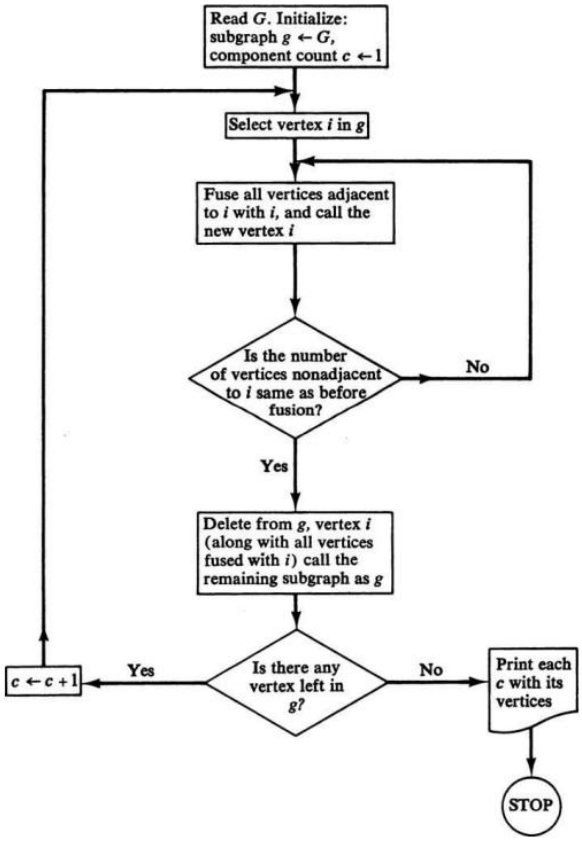

Fig.1 Algorithm 1: Components of G.

Selecting the beginning vertex (to which surrounding vertices are fused) in each component correctly may increase efficiency, but only if the cost of doing so is reasonable.

A data flow diagram representing the "Connectedness and Components Algorithm"

At the conclusion of the chapter, you'll find a fully-executable computer program written in APL360, together with a legend that describes the program's variables. In each subcomponent, the software chooses a vertex with the highest degree as the starting vertex.

The following examples demonstrate an input and the corresponding output from the program: An adjacency matrix of size 20 by 20 depicting a network with 20 vertices is read in as input, and as output, a list of components (COMP) and the names of vertices (VERT) contained in each component is given. Each vertex in X represents a certain row and column.

INPUT

N = 20

X =

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

OUTPUT

```
COMP 1; VERT:  6  7   8   10  16  19  20
COMP 2; VERT:  1  2   3   5   11
COMP 3; VERT:  9  12  13  17
COMP 4; VERT:  14 15
COMP 5; VERT:  4
COMP 6; VERT:  18
```

Algorithm 2: A Spanning Tree

The spanning-tree algorithms are some of the best-known and most-used algorithms in graph theory. To put it simply, a spanning-tree method will always produce exactly one spanning tree for each given linked network. A spanning forest with n + p edges, where p > 1 is the number of components in the unconnected graph, should be generated by the method if the network is disconnected. A natural by-product of such an algorithm would be to determine whether or not the network is linked, and if it isn't, to identify its individual nodes. A spanning-tree approach is sometimes used to check the connectivity of a network. On the other hand, if each edge in the supplied graph is assigned a weight or distance (a weighted graph), we may want to locate a spanning tree with the minimum feasible weight.

A Breakdown of the Algorithm: Allow the inherent, aimless cycle to run its course (if the graph has any self-loops, they may be discarded) The number of vertices in graph G is n, and it has e

edges. Let the edges be 1, 2,..., n, and the graph be given by two linear arrays F and H such that $f_1, \in F$ and $h_1 \in H$are the terminal points of a i[th] edge in G

Each new edge is checked to verify whether its end vertices already exist in a tree at the current stage of the process. † The kth iteration of edge inspection, 1< k< e, (f_k, h_k) five different conditions may arise:

1. If neither vertex f_k nor h_k if the k-th edge of every tree built in G so far contains the given edge, then the tree is labeled and its endpoints identified$f_k, h_k$are assigned the component c after c has been incremented by 1.

2. If vertex fk is in some tree T_i (i = 1, 2, . . . , c) and hk in tree T_j (j = 1, 2, . . . , c, and i ≠ j), These two trees are connected through the kth edgeTjpresently has a component number of T_iAs a result, c is decreased by 1.

3. An edge has a tree structure if both nodes are in the same tree(f_k, h_k) serves as a basic circuit and is ignored.

4. If vertex f_k is in a tree T_i and $h_k$is not located on any tree's (f_k, h_k) is added to $T_i$component numbering by T_t to $h_k$also.

5. If vertex f_k is in no tree and h_k is in a tree T_j the edge (f_k, h_k) is added to T_j component numbering byT_j to f_k also.

In the shown flowchart of the algorithm, the five situations indicated by the encircled numbers are.

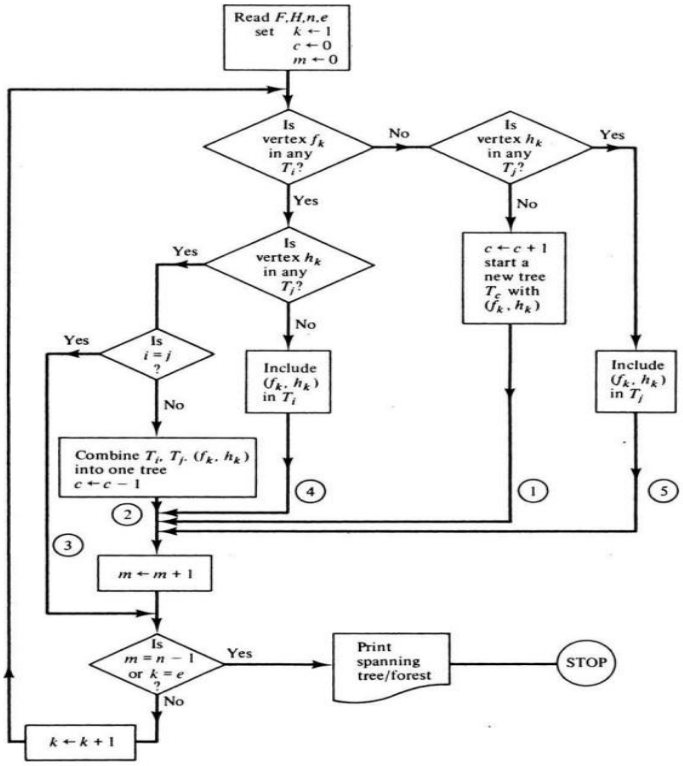

Fig. 2 Algorithm 2: Spanning tree/forest.

Algorithm 3: A Set of Fundamental Circuits

In this case, too, each edge is checked to determine whether it completes a circuit with the tree built thus far, but the edges are not taken in any particular sequence. Once a vertex z is chosen, we follow it around the graph until we've looked at every edge that's ever hit it. (As we'll see in a moment, vertex z is the most recent addition to the embryonic tree.) Let the edges of the graph X be 1, 2,..., n, and the vertices of the graph G = (V, E) be 1, 2,. Let T represent the currently explored collection of nodes in the developing tree, and let W represent the unexplored set of nodes (i.e., those vertices, in T as well as not in T, which have one or more unexamined edges incident on them). T =∅ and W = V, the whole set of vertices, at the outset.

To begin the process, T = 1, representing the first vertex, and W = V are chosen. The newly generated tree will have its origin at vertex 1. The following steps are taken when startup has been completed.

1. If $T \cap W = \emptyset$, If it happens, the algorithm stops.

2. If $T \cap W \neq \emptyset$, To choose a vertex z in $T \cap W$.

3. Analyze z by taking into account all of the edges that intersect it. In the event that there is no longer any such edge, z must be subtracted from W and the procedure returned to Step 1.

4. Check whether the pth vertex is in T if there is an edge of the form (z, p).

5. If p ∈ T, Figure out the bare-bones circuit that includes the edge (z, p) and the one and only way the edge may be followed from z to p in the tree (formed so far). Remove the vertices from the graph at (z, p) and proceed to the third phase.

6. If p ∈ T, Set T to include the node z and the vertex p added by the operation. To proceed to Step 3, remove the edge connecting Vertex z and Vertex p.

Step 5 involves looking at vertex z and finding an edge (z, p) such that vertex z is next to point p, p ∈ T. To complete the basic loop established by (z, p) in the tree, we look for the ancestors of each node in the tree, starting with PRED(z), then PRED(PRED(z), etc., until we reach PRED(p), the ancestor of p. Simply said, the underlying circuit that is produced is

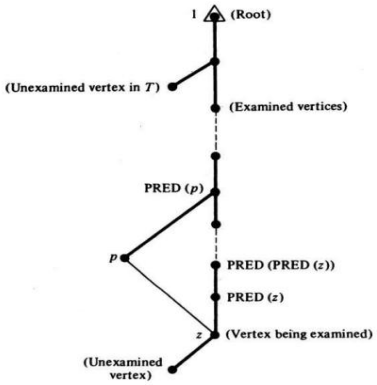

Fig. 3 Generation of a fundamental circuit.

z, PRED(z), PRED(PRED(z)), . . . , PRED(p), p, z.

CONCLUSION

an overview of the foundations of graph-theoretic methods in computing. One must be fluent in both computer programming and graph theory to be able to instruct a computer on the subject. One must also discover effective algorithms. The most crucial requirement for a practical graph theoretic algorithm is that its computer running time does not increase by a factor of n or even an exponential factor. Specifically, it creates and employs a language tailored to the needs of combinatorial applications.

REFERENCE:

1. Pranav Patel and Chirag Patel (2013) Various Graphs and Their Applications in Real World. International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 12,ISSN: 2278-0181, December – 2013

2. Gowda, Dr.Dankan&Shashidhara, K.S. & M., Ramesha & S B, & S B, Manoj Kumar. (2021). RECENT ADVANCES IN GRAPH THEORY AND ITS APPLICATIONS. Advances in Mathematics: Scientific Journal. 10. 1407-1412. 10.37418/amsj.10.3.29.

3. Ms, Vinutha&Puranik, Arathi. (2017). Applications of Graph Coloring and Labeling in Computer Science. 3. 14-16.

4. KiranKaundal, Applications of Graph Theory in Everyday Life and Technology, Imperical Journal of Interdisciplinary Research, 3(3)(2017).

5. Bhagya Jyoti Nath, Applications of Graph Theory in Different Branches of Science, International Journal of Mathematics And its Applications, 5 (3-A)(2017),57-60.

6. A.Prathik, K.Uma and J.Anuradha, An Overview of Application of Graph Theory, Internatinal Journal of ChemTech Research, 9(No. 2)(2010),242-248.

7. DakineniDurgaprasad, AppalaSrinuvasuMuttipati, MesdisettiSnehadivya and Sanaka Kavitha, Application of Computer Science Based on Graph theory, Internatinal Journal of Engineering, Science & Mathematics, 6(8)(2017).

8. F.Harary , Graph Theory, Narosa Publishing House, (2013).

9. Pranav Patel, Chirag Patel, Various Graphs and Their Applications in Real world, International Journal of Engineering Research & Technology, 2(12)(2013).

10. S.G.Shrinivas, S.Vetrivel and N.M.Elango, Applications of graph theory in computer science an overview, International Journal of Engineering Science and Technology, 2(9)(2010), 4610-4621.

11. Sangkaran, Theyvaa& Abdullah, Azween&Zaman, Noor. (2020). Community Detection Based on Isomorphic Subgraph Analytics in Criminal Network. 20. 94.

12. K, Vijayalakshmi & N.B, Prajwala. (2019). An Efficient Stack Based Graph Traversal Method for Network Configuration.

13. MeenakshiWasadikar, PradnyaSurvase, Some properties of graphs derived from lattice,Bulletine of Calcutta Mathematical Society, Vol. 104, (2012), Page 125-138.