Presenting a Simple Method of Finding Cycles in Graphs

Hossein Jafari [1], Kiyana Salehi [2], Farzane Hosseinieh Farahani[3]

1. Young Researchers and Elite Club,Arak Branch,Islamic Azad Univercity,Arak, Iran.
2. Department of Industrial Engineering Malayer Branch, Islamic Azad University, Malayer, Iran.
3. Department of Industrial Engineering,Arak Branch,Islamic Azad University,Arak,Iran.

Corresponding Author: Hossein Jafari

Abstract: The issue of finding cycles in graphs has been discussed for a long time and different solutions have been suggested for this problem; but, some of those solutions are not easy to be used in computers. In the paper, using Matrix Algebra, a technique is proposed that is easy to use on computers. The technique could be very useful in problems regarding the optimization of trees in networks. In the end a numerical problem gets solved using the proposed method.
Keywords: Graph, Cycle, Walk, Tree, Kruskal Algorithm.

Introduction and Problem Definition

In many optimization problems, industrial engineers try to find the tree that, in addition to covering the entire network, also has the least cost. The cost may be determined by distance, traffic amount, et cetera. Among the most famous methods to solve the aforementioned problem (finding the minimum spanning tree) in discrete mathematics is the Kruskal algorithm [1]. In case the problem has vast dimensions, the problem turns into an NP-Hard problem, in which case there would be no other way but to use a computer [2]. This is despite the fact that whenever Kruskal algorithm is applied to find cycles we can use visual analysis, or algorithms such as Depth-First Search (DFS) or Breadth-First Search (BFS). Applying any of the three mentioned methods on a computer would be a complex and difficult task. Because of this, in the paper a simple approach in the theory of graph will be proved. The said approach will improve the Kruskal algorithm; also, it will contain non-visual logic based on Matrix Algebra. Because of these, the Kruskal algorithm would be applied to computers more easily.
Background research

In 2006, Reimann and Laumanns used the Ant Colony Optimization (ACO) algorithm to introduce a metaheuristic method to find the minimum spanning tree in a network. In the algorithm, the simulated ants after a few repetitions would learn to use the optimal edges in the tree design [3]. In 2015, Cong and Zhao in a paper in a case study to design an urban water network used the minimum spanning tree (MST) method to reduce both the time needed to build the network and the expenses needed for its piping [4]. In 2016, Sarkar and his colleagues, in a paper, used the minimum spanning tree in the Kruskal algorithm for the optimization of the voltage in RK networks so that a steady a steady voltage would be achieved [5].

Graph

A graph consists of two sets: a nonempty set of nodes or vertices, and a set of edges that connect the vertices [6].

Weighted Graph

A weighted graph is a graph whose edges have been assigned weights. Weight can represent cost, distance, time, or any other feature of edges [6].

Walk

A walk of $G = (V, E)$ is a finite sequence of the vertices and edges of G such as $W = v_0 e_1 v_1 e_2 \dots e_k v_k$ which in case$1 \leq i \leq k$, the vertices $v_{i-1}$and$v_i$are the two ends of the e_i edge. In such case we say W is a walk from v_0 to v_k, or in other words, (v_0, v_k) is a walk [1].

Cycle

A cycle is a simple path that its starting node is the same as its ending node [1].

Connection

An undirected graph is called connected when a walk exist between each two vertices of it. That means that both its vertices are connected. In a directed graph the connection is more complicated since the direction needs to be considered. It is possible that node a is connected to node b but there is no walk from node b to node a [6].

Tree

A connected simple graph with no cycles is called a tree. A tree is a graph where only one path exists between any two vertices of it. A tree with n vertices has $n - 1$ edges [6].

Adjacency Matrix of a Graph

There are many ways to show graphs on computers. The two basic data structures used to demonstrate graphs are the adjacency matrix and the adjacency list. The adjacency matrix of graph G with n vertices (the edges of which are named from $v_1$to v_n) is a bit matrix of $n\times n$ with the name A in which: $a_{ij}$is equal to 1 if there is an edge from v_i to v_j; a_{ij} is equal to 0 if there is no edge from v_i to v_j [6].

Kruskal Algorithm

In this algorithm the edges of the graph are sorted in ascending order. We start with the first (smallest) edge and add each edge, provided that a cycle would not be created, to the graph. We continue this process until the minimum spanning tree is created [1].

Theorem (1)

Suppose that $G = (V, E)$ is a multiple edge graph. $V = \{v_1, \dots, v_n\}$ and $A = \left(a_{ij}\right)_{n*n}$ are its adjacency matrix. In that case the $(i, j)^{th}$entry raised to the power k of matrix of A equals walk-v_0, v_k with the longitude of k in G [6].

Theorem (2)

Suppose that $G = (V, E)$ is a multiple edge graph. $V = \{v_1, \dots, v_n\}$and$A = \left(a_{ij}\right)_{n*n}$ are its adjacency matrix. If the matrix of S is defined as $S = \sum_{j=0}^{n} A^j$, then:

$$\begin{cases} s_{ij} = 0 \Leftrightarrow & \text{There is no walks from node } i \text{ to node j} \\ s_{ij} > 0 \Leftrightarrow & \text{There is at least one walk from node } i \text{ to node j} \end{cases} \qquad (1)$$

Proof

Considering that in adjacency matrix the connections are shown by the numbers one and zero, all the entries raised to the power of k of matrix of A (in other words, A^k) for $k = 0,1,..,n$ are nonnegative (that is, they are either zero or positive number). Therefore, it is evident that all the entries of the matrix of S are also nonnegative.

Proof of the First Case

Based on theorem (1), the $(i, j)^{th}$entry raised to the power of k of matrix of A equals walk-(v_i, v_j) with the longitude of k in G. So if there are no walks from node i to node j that means the $(i, j)^{th}$entry raised to the power k of matrix of A equals zero so it can be easily observed that:

$$s_{ij} = 0 \qquad (2)$$

Proof of the Second Case

If $s_{ij} = 0$ then A raised to all the powers of k in the $(i, j)^{th}$entry include the amount zero. On the other hand based on theorem (1), the $(i, j)^{th}$entry raised to the power of k of matrix of A equals walk-(v_i, v_j) with the longitude of k in G, so it can be said: there is no walk from node i to node j.

Proof of the Third Case

Based on theorem (1), the $(i, j)^{th}$entry raised to the power of k of matrix of A equals walk-(v_i, v_j) with the longitude of k in G. So if there is at least one walk from node i to node j, then the $(i, j)^{th}$entry raised to one of the powers of matrix of A equals a number larger than zero. Thus it can be easily concluded that:

$$s_{ij} > 0 \qquad (3)$$

Proof of the Fourth Case

If $s_{ij} > 0$ then one of the powers of matrix of A to which the $(i, j)^{th}$entry has been raised to equals a number larger than zero. On the other hand based on theorem (1), the $(i, j)^{th}$entry raised to the power of k of matrix of A equals walk-(v_i, v_j) with the longitude of k in G, therefore we can say: there is at least one walk from node i to node j.

The proof is over and the rule stands.

Theorem (3)

Suppose $A = \left(a_{ij}\right)_{n*n}$ is a matrix with real entries and matrix of S is defined as $S = \sum_{j=0}^{n} A^j$, in that case if the matrix of $(A - I)$ is invertible it could be said that:

$$S = \frac{A^{n+1} - I_n}{A - I_n} \qquad (4)$$

Proof:

We define the matrix of S as the following:

$$S = I_n + A + A^2 + A^3 + \cdots + A^n \qquad (5)$$

We multiply the relation (5) by the matrix of A from the right, the following is obtained:

$$S * A = A + A^2 + A^3 + \cdots + A^n + A^{n+1} \qquad (6)$$

Now we add and subtract the identity matrix of I_n to and from the right side of the relation (6):

$$S * A = I_n + A + A^2 + A^3 + \cdots + A^n + A^{n+1} - I_n \qquad (7)$$

According to relations (5) and (7) we can say:

$$S * A = S + A^{n+1} - I_n \qquad (8)$$

We simplify relation (8) and thus have:

$$S * (A - I_n) = A^{n+1} - I_n \qquad (9)$$

Based on the assumption the matrix of $(A - I_n)$ is invertible, therefore we multiply the two sides of the relation (9) by $(A - I_n)^{-1}$:

$$S = \frac{A^{n+1} - I_n}{A - I_n} \qquad (10)$$

The proof is over and the rule stands.

Numerical Example

Consider a graph G with an adjacency matrix of A. Use the new method to fin out if the third node is connected to the fifth node or not?

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

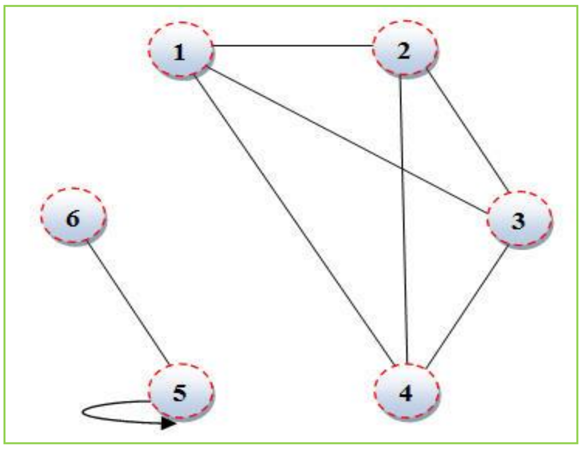

Figure.1 graph G

The determinant of the square matrix $(A - I)$ is $16(|A - I| \neq 0)$ Therefore We calculate the matrix of S:

$$S = \begin{bmatrix} 274 & 273 & 273 & 273 & 0 & 0 \\ 273 & 274 & 273 & 273 & 0 & 0 \\ 273 & 273 & 274 & 273 & 0 & 0 \\ 273 & 273 & 273 & 274 & 0 & 0 \\ 0 & 0 & 0 & 0 & 33 & 20 \\ 0 & 0 & 0 & 0 & 20 & 13 \end{bmatrix}$$

Considering that $s_{5,3} = 0 = s_{3,5}$, therefore the two vertices are apart and there is no connection between them.

CONCLUSION

The Kruskal algorithm is among the most used methods in finding minimum spanning trees in graphs. Applying said algorithm in problems with mass input is impossible to do by human beings; on the other hand, using programming and computers in applying the method would require a computer with high processing capabilities, something most college students do not have access to. In the paper a very simple method was introduced, and eventually proved, to find cycles and prevent their formation in the Kruskal algorithm.

REFERENCES

Bondy, J.A., and Murty, U.S.R.(2008).Graph Theory.Graduate Texts in Mathematics.

Cong, F., Zhao, Y.(2015). The Application of Minimum Spanning Tree Algorithm in the Water Supply Network. International Industrial Informatics and Computer Engineering Conference (IIICEC 2015), 52-55.

Grimaldi, R.P.(2005).Discrete And Combinatorial Mathematics:an applied introduction. rose-hulman institue of technology.

Reimann, M., Laumanns, M. (2006). Savings based ant colony optimization for the capacitated minimum spanning tree problem . Computers & Operations Research , 33,1794 – 1822.

Sarkar, D., De, A., Kumar Chanda , C., Goswami.(2016). Kruskal's Maximal Spanning Tree Algo rithm fo r Optimizing Distribution Netwo rk To pology to Improve Vo ltage Stability. Electric Power C omponents and Systems, 43(17),1921–1930, 2015.

Sharifov, F., Kutucu, H.(2010). Minimum Cost ≤ k Edges Connected Subgraph Problems. Electronic Notes in Discrete Mathematics ,36,25–32.