A STUDY ON TREES AND PETRINETS

Paper Recceived 8th Aug 2017
Paper Accepted 20th Aug 2017
Paper Published 30th Aug 2017
Reviewers Dr. S. Loghambal
Dr.V.Biju

Mrs.USHASREE DONAPATI
M.Phil. Research Scholar
PRIST University, Mahabhalipuram
Tamilnadu, India

Mr.KARTHIKEYAN
Research Supervisor, Department of Mathematics
PRIST University, Mahabhalipuram
Tamilnadu, India.

Abstract

A study on trees and Petri nets" is prepared after a deep study on certain concepts of Graph Theory. Due to the importance of trees a brief discussion on trees is done. A Study on Tree and Petri nets deals with explanation of Properties of trees, Enumeration and Spanning Trees. Binary Trees, Dynamic Graph Algorithms and Petri nets are explained with definitions and examples. The Tree structure in other Branches includes the Characterization of a Tree and includes few problems on Trees.

Keywords— Construction, Women, Female Employment, Tree, Dynamic Graph.

INTRODUCTION

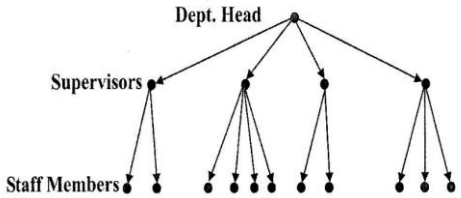
Configurations of vertices and connection occur in a great diversity of applications. They may represent physical networks, such as electrical circuits, roadways or organic molecules. Any they are used in representing less tangible interactions as might occur in ecosystems, sociological relationships, databases or in the flow of control in a computer program.

Formally, such configurations are modeled by combinational structures called graphs, consisting of two sets called vertices and edges and an incidence relation between them. The vertices and edges may have additional attributes, such as color or weight, or anything else useful to a model: Graph models tend to fall into handful of categories. For instance, the network of one-way streets of a city requires a model in which each edge is assigned a direction; two atoms in an organic molecule may have more than one bond between them; and a computer program is likely to have loop structures. These examples require graphs with directions on their edges, with multiple connections between vertices, or with connections from a vertex to itself.

In the past, these different types of graphs were often regarded as separate entities, each with its own set of definitions and properties. We have adopted a unified approach by introducing these various graphs at once. This allows us to establish properties that are shared by classes of graphs without having to repeat arguments. Moreover, this broader perspective has a bonus: it inspires computer representations that lead to design of fully reusable software for applications.

Example

The following are the objectives of the study area are to find out the major problems of women construction workers in the study area, to study the reasons for choosing the construction work in the study area. To identify overall satisfaction about construction work in Tiruchengode Taluk.



Enumeration and Spanning Trees

Theorems on spanning Trees – A spanning sub graph with vertex set v (G). A Spanning tree is a spanning sub graph that is a tree.

Theorem Every connected graph contains a spanning tree.

Proof Let G be a connected graph. Delete edge from G that are not bridges until we get a connected sub graph in which each edge is a bridge. Then H is a spanning tree. On the other hand, if there is a spanning tree in G, there is a path between every pair of vertices in G, Thus G is connected.

Corollary If G is connected, then m>n-1.

Proof: Let G be a connected graph.

By the theorem above "every connected graph contains spanning tree" G contains a spanning tree T.

i.e G>T
M (G) >m (T)
Since T is a spanning tree,
M (T) = n (T) – 1
V (G) = v (T) => n (G) = n (T)
M (G) > m (T) = n (T) – 1
=n(G) – 1
Therefore m(G) >n(G) – 1
=>m>n-1.

Spanning Trees A spanning tree of a connected graph G is a sub graph which is a tree and which includes every vertex of G.

Minimal Spanning Tree A minimal spanning Tree of a connected weighted graph is a spanning tree of least weight, that is, a spanning tree for which the sum of the weights of all its edges is least among all spanning trees. The concept of spanning tree exists only for a connected graph G and if G has n vertices, any spanning tree must necessarily contain n-1 edges.

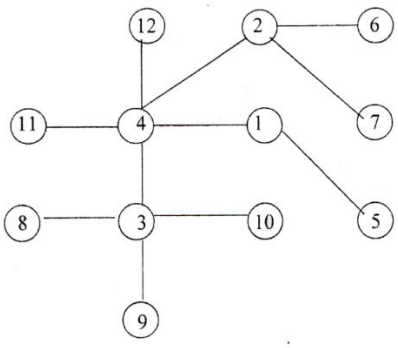
Terminal vertex, diametral path and labeled Trees

A vertex of degree 1 in a graph is called a terminal vertex (or pendant vertex or end - vertex). Show that every tree of order two or more has at least two terminal vertices.

Solution

Suppose the degrees of the n vertices of a tree are di, where I = 1, 2 ……n. Then d1 + d2…+ dn = 2n-2. If each degree is more than 1, the sum of n degrees is at least 2n. So there is at least one vertex (say vertex 1) with degree 1. Then d2 + d3 + ….2n-1. At least one of these (n-1) positive numbers is necessarily 1. So there is one more vertex of degree 1. Thus at least two of the degrees must be 1.

Find the Unique Vector Corresponding to the Labeled Tree Shown In Below



Solution:

Since there are 12 vertices in T, we are looking for a vector s with 10 components, each component being an integer between 1 and 12. Here W = {5, 6, 7, 8, 9, 10, 11, 12} is the set of all terminal vertices in T, with labels arranged in increasing order. The vertex adjacent to 5 is 1, so s1 = 5. Deleting vertex 5 from T, we get the sub tree (the current set, again denoted by T) in which the set of terminal vertices (the current set, again denoted by W) is W= {1, 5, 6, 7, 8, 9, 10, 11, 12}. The vertex adjacent to 1 is 4, so s2 =4.

Deleting 1 from the current tree, we get W= {6, 7, 8, 9, 10, 11, 12} and s2=2. In the next interation, W= {7, 8, 9, 10, 11, 12} and s4 = 2. In the next interation, W= {2, 8, 9, 10, 11, 12} s5=4. In the next interation, W= {8, 9, 10, 11, 12} and s6 = 3. In the next interation W= {9, 10, 11, 12} and s7=3. In the next interation W= {10, 11, 12} and s8=3. In the interation w= {11, 12} and s9=4. At the final stage, we have the tree consisting of two vertices 4 and 1, so s10=4. Thus s= {1, 4, 2, 2, 4, 3, 3, 34, 4} is the unique vector defined by the given labeled tree.

Introduction to Binary Trees

A binary tree is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller "subtrees" on either side. A null pointer represents a binary tree with no elements -- the empty tree. The formal recursive definition is: a binary tree is either empty or is made of a single node, where the left and right pointers each point to a binary tree.

Definition

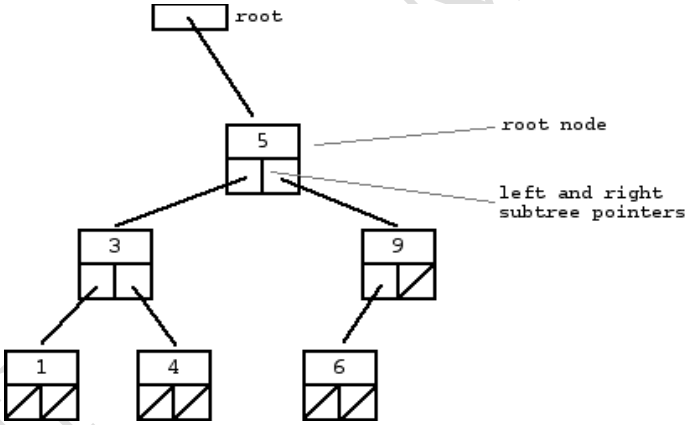
A binary tree is a connected acyclic graph such that the degree of each vertex is no more than three. It can be shown that in any binary tree of two or more nodes, there are exactly two more nodes of degree one than there are of degree three, but there can be any number of nodes of degree two. A rooted binary tree is such a graph that has one of its vertices of degree no more than two singled out as the root.

With the root thus chosen, each vertex will have a uniquely defined parent, and up to two children; however, so far there is insufficient information to distinguish a left or right child. If we drop the connectedness requirement, allowing multiple connected components in the graph, we call such a structure a forest.

Another way of defining binary trees is a recursive definition on directed graphs. A binary tree is either:

- A single vertex.
- A graph formed by taking two binary trees, adding a vertex, and adding an edge directed from the new vertex to the root of each binary tree.

Binary Trees



Petri nets - Introduction and Definition

A Petri net is a kind of bipartite directed graphs populated by three types of objects. These objects are *places*, *transitions*, and *directed arcs*. Directed arcs connect places to transitions or transitions to places. In its simplest form, a Petri net can be represented by a transition together with an input place and an output place. This elementary net may be used to represent various aspects of the modeled systems. For example, a transition and its input place and output place can be used to represent a data processing event, its input data and output data, respectively, in a data processing system. In order to study the dynamic behavior of a Petri net modeled system in terms of its states and state changes, each place may potentially hold either none or a positive number of *tokens*. Tokens are a primitive concept for Petri nets in addition to places and transitions. The presence or absence of a token in a place can indicate whether a

condition associated with this place is true or false, for instance.

A Petri net is formally defined as a 5-tuple $N = (P, T, I, O, M_0)$, where

(1) $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;

(2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$;

(3) I: $P \times T \rightarrow N$ is an *input function* that defines directed arcs from places to transitions, where N is a set of nonnegative integers;

(4) O: $T \times P \rightarrow N$ is an *output function* that defines directed arcs from transitions to places; and

(5) M_0: $P \rightarrow N$ is the *initial marking*.

A *marking* in a Petri net is an assignment of tokens to the places of a Petri net. Tokens reside in the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net.

Most theoretical work on Petri nets is based on the formal definition of Petri net structures. However, a graphical representation of a Petri net structure is much more useful for illustrating the concepts of Petri net theory. A Petri net graph is a Petri net structure as a bipartite directed multi graph. Corresponding to the definition of Petri nets, a Petri net graph has two types of nodes. A *circle* represents a place; a *bar* or a *box* represents a transition. Directed arcs (arrows) connect places and transitions, with some arcs directed from places to transitions and other arcs directed from transitions to places. An arc directed from a place p_j to a transition t_i defines p_j to be an *input place* of t_i, denoted by $I(t_i, p_j) = 1$. An arc directed from a transition t_i to a place p_j defines p_j to be an *output place* of t_i, denoted by $O(t_i, p_j) = 1$. If $I(t_i, p_j) = k$ (or $O(t_i, p_j) = k$), then there exist k directed (parallel) arcs connecting place p_j to transition t_i (or connecting transition t_i to place p_j). Usually, in the graphical representation, parallel arcs connecting a place (transition) to a transition (place) are represented by a single directed arc labeled with its multiplicity, or weight k. A circle contains a *dot* represents a place contains a token (Peterson 1981).

Example: A simple Petri net.

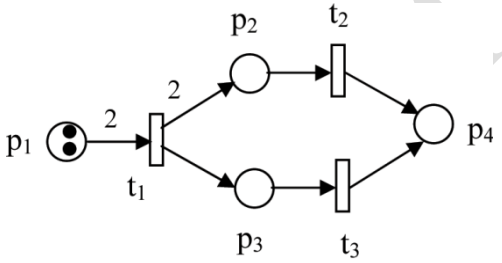


Figure 1 shows a simple Petri net. In this Petri net, we have

P = {p1, p2, p3, p4};
T = {t1, t2, t3};
I (t1, p1) = 2, I (t1, pi) = 0 for i= 2, 3, 4;
I (t2, p2) = 1, I (t2, pi) = 0 for i= 1, 3, 4;
I (t3, p3) = 1, I (t3, pi) = 0 for i= 1, 2, 4;
O (t1, p2) = 2, O (t1, p3) = 1, O(t1, pi) = 0 for i= 1, 4;
O (t2, p4) = 1, O (t2, pi) = 0 for i= 1, 2, 3;
O (t3, p4) = 1, O(t3, pi) = 0 for i= 1, 2, 3;
M0 = (2 0 0 0)T.

Trees in Coding Theory

[x] Denote the smallest positive greater than or equal to x. define an alphabet A to be a finite set of symbols. A code is a set C of sequence of elements A. The elements of C will be called code words. For simplicity we assume A= {0, 1} then our code word is a binary sequence. One example of such a code is {0, 00, 01, 01, 111, 001}.

Since computers are inherently binary it is very natural to use codes consisting of binary sequences to store information in a computer. Now the problem is to represent the message using a code Support we wish to represent the message ROAD IS GOOD

using a code. I.e. we wish to encode the message. We can see that we need word to represent R, one for 0, one for A, one for D, one for I, one for S, one for G and one for blank space.

We would like to have encoded message as short as possible, so we might select the code C={0,1,00,10,01,11,100,00} to represent the symbols {R,O,A,D,I,S,G, blank space} respectively. When encode our message.

CONCLUSION

"A study on trees and Petri nets" is prepared after a deep study on certain concepts of Graph Theory. Due to the importance of trees a brief discussion on trees is done.

The first study covers basic definition, properties of trees, some results on tree, some problems and examples.

The second study covers the spanning trees and enumeration of trees which or very important concepts due to the application.

The third study deals with binary trees, combinatory and binary search tree Niche.

A few more concepts on dynamic graph algorithm, the general techniques for undirected graphs their definition and examples.

Petri nets, properties of Petri nets and analysis of Petri nets are studied in the fifth study.

Trees in other branches like chemistry, coding theory and computer science is studied in the Sixth study.

Reference

[1] B. Berthomieu, M. Menashe, A State Enumeration approach for analyzing time Petri Nets, Proceedings of 3rd European Workshop on Applications and Theory of Petri Nets, Varenna, Italy, Sept. 1982

[2] R. E. Bloomfield, J. H. Cheng, J. Górski, Towards A Common Safety Description Model, Proceedings of Safecomp'91, Pergamon Press, 1991

[3] J. Górski, Extending Safety Analysis Techniques With Formal Semantics, In Technology and Assessment of Safety Critical Systems, (F.J. Redmill and T. Anderson, Eds.), Springer-Verlag, 1994

[4] J. Górski, A. Wardziński, Formalizing Fault Trees, Proceedings of SCSS'95, Brighton, UK, (F.J. Redmill and T. Anderson, Eds.), Springer-Verlag, 1995