On Minimum Spanning Trees and Determinants

Perrin Wright
Florida State University
Tallahassee, FL 32306-4510

1. Introduction.

In the 1840's, Kirchhoff, while putting forth his defining work on circuit theory, was simultaneously pioneering the theory of graphs, a development which occupies a central place in circuit theory. Using algebraic methods, Kirchhoff [**3**] found a way to determine whether a set of edges in a connected graph was a *spanning tree*, that is, a tree containing all the vertices of the graph. In his setting, the spanning trees correspond to certain nonsingular submatrices of a matrix constructed from the incidence matrix of the graph.

Over time, graphs came to be treated as discrete structures as well, consisting of finite sets of vertices and edges, without the algebraic structure that Kirchhoff used. Later in the century, attention was given to weighted graphs, graphs whose edges have been assigned real values. A classical problem in weighted graphs is this: Given a connected weighted graph, find all the spanning trees that have the minimum edge weight sum. These trees are called *minimum spanning trees*, and an algorithm for finding one was given as early as 1926 by Boruvka, whose work is discussed in [**2**]. In 1956 Kruskal [**4**] developed another algorithm for finding a minimum spanning tree; it is perhaps the most famous of all such algorithms. Minimum spanning trees have many applications; some of the most obvious ones deal with the minimizing of cost, such as the cost of building pipelines to connect storage facilities.

We will begin by outlining the results of Kirchhoff and Kruskal. Although both approaches have to do with spanning trees in graphs, they appear to have little else in common. We show that there is indeed a more profound connection, which we develop here. In particular, we will develop an algorithm that uses Kirchhoff's determinants to generate the set of all minimum spanning trees in a weighted graph. In the process, the linear algebra and the discrete mathematics come together in a pleasing, intriguing, and accessible way, and it is hoped that students may be encouraged to follow this expository article with further exploration.

2. Kirchhoff's Theorem.

If a graph G has n vertices and b edges, the *incidence matrix* of G is the $n \times b$ matrix in which each column (edge) has exactly two non-zero entries, $+1$ and -1, indicating the

1991 Mathematics Subject Classification: Primary 05C05, 68R10.

two vertices for that edge. (The signs, which can be placed arbitrarily, effectively put a direction on each edge, but this direction is not relevant to what follows.) We assume that G has no loops.

For a connected graph G, the rank of the incidence matrix is $n-1$, and the deletion of any row leaves a matrix of rank $n-1$. We delete the last row to obtain an $(n-1)\times b$ matrix A called the *reduced incidence matrix*. The determinant of any square submatrix of A is either zero or ± 1. An $(n-1)\times(n-1)$ submatrix of A is non-singular if and only if its columns correspond to the edges of a spanning tree of G. A thorough development of these properties can be found in [**1**], but all are straightforward and make good exercises.

Kirchhoff's observation becomes our first theorem.

Theorem 1. *The spanning trees of the connected graph G are the nonsingular $(n-1)\times(n-1)$ submatrices of the reduced incidence matrix A, and the determinants of these submatrices are all ± 1.*

Kirchhoff's celebrated formula states that $\det(AA^T)$ is equal to the number of spanning trees in G. This formula, which follows from Theorem 1 and the Binet-Cauchy theorem on determinants, has gathered much attention but does not shed as much light as Theorem 1 itself.

3. Kruskal's Algorithm.

There are many algorithms for finding a minimum spanning tree in a weighted graph G. Kruskal's algorithm is perhaps the most widely known.

Kruskal's Algorithm. Choose any unchosen edge of lowest weight that does not create a cycle with the chosen edges, and continue until no more edges are available.

The graph in Figure 1a has four different minimum spanning trees, and Kruskal's algorithm can be executed in such a way as to yield any one of them.

In the next section we use Kruskal's algorithm to connect Kirchhoff's determinants and the set of minimum spanning trees in G. First, we need to generalize Kirchhoff's reduced incidence matrix to weighted graphs. Let w be the weight function on the edges of G. For the remainder of this article we assume that w has positive integer values.

We begin by altering the incidence matrix in such a way that the two entries in the column corresponding to the edge e are $\pm w(e)$ instead of ± 1. For the same reasons as before, a set H of $n-1$ edges of G will be a spanning tree if and only if $\det M(H) \neq 0$, where $M(H)$ is the $(n-1) \times (n-1)$ submatrix of the reduced incidence matrix A whose columns correspond to the edges of H. It is also clear that, for any spanning tree H, $\det M(H) = \pm \prod$, where $\prod$ is the product of the weights of the edges in the tree. Fortunately, these determinants can be used to identify the minimum spanning trees of G, as we will see later. For the time being, we will put the matrix A aside and consider geometric properties instead.

In the weighted graph G, suppose T is a minimum spanning tree and f is an edge of G not in T. Suppose further that e lies on the unique simple path (no repeated edges or vertices) in T joining the vertices of f, and that $w(e) = w(f)$. Then one can replace e by f to get another minimum spanning tree. We call this move an *equal edge replacement* in T; others call it an exchange of weight zero.

Theorem 2. *If T and S are minimum spanning trees in the weighted graph G, then there is a finite sequence of equal edge replacements that start with T and end with S.*

Theorem 2 should perhaps be the first result that students encounter. Yet we have seen it stated only once in a text [**5**], and there only implicitly, as an exercise.

Every weighted graph has an *edge weight spectrum*, the set of edge weights listed with multiplicities. The following result is an immediate corollary to Theorem 2:

Theorem 3. *Any two minimum spanning trees in G have the same edge weight spectrum.*

The constancy of the edge weight spectrum can be used to great advantage in studying minimum spanning trees, so it is puzzling that so basic a fact has not made its way into textbooks on discrete mathematics. When the author first began to teach this subject to undergraduates, he naively imagined that one might find two trees that achieved the minimum edge sum using different summands. But it cannot be done.

Proof of Theorem 2: Let k be the smallest integer such that T and S have different sets of edges of weight k. Thus, if an edge of G has weight less than k, then it belongs to T if and only if it belongs to S. Without loss of generality, assume that e is an edge of weight k belonging to T but not to S. In $S \cup \{e\}$ a cycle is formed by e and the unique simple path P joining the vertices of e in S. Every edge on this cycle has weight less than or equal to k, for otherwise, we could replace some edge of greater weight on P by e to obtain a spanning tree with smaller edge sum than S, contrary to hypothesis. Some edge on P must not belong to T, for otherwise, $P \cup \{e\}$ would be a cycle in T. Such an edge must have weight k, for otherwise its weight is less than k and it would belong to T by the choice of k. Choose such an edge f; that is, f is on P, f does not belong to T and $w(f) = k$. Replace f by e in S to get another tree $S' = (S - f) \cup \{e\}$. Now S' is a minimum spanning tree with one more edge in common with T than S. Continue this process (or use induction) to complete the proof. ∎

Certain elementary results about minimum spanning trees follow effortlessly from Theorem 3. Here are two:

(a) *If all the edges of the connected graph G have different weights, then G has exactly one minimum spanning tree.*

(b) *Every minimum spanning tree in G can be obtained as the output of Kruskal's algorithm.*

4. Determinants.

Theorem 4. *A spanning tree in the weighted graph G has the minimum edge* sum *if and only if it has the minimum edge* product.

To help the reader generate the appropriate surprise at this result, we observe that it is easy to find a weighted graph with two spanning trees T and S, such that T has the smaller edge sum and S has the smaller edge product. The weighted graph in Figure 1b has two such trees, and the reader may enjoy finding them. But if a tree has the *minimum* edge sum, then it must have the minimum edge product, and conversely.

A short proof of Theorem 4 can be had by replacing each edge weight by its logarithm. Doing so leaves the order of weights unchanged. Since Kruskal's algorithm respects only the ordering of the edges by weight, a spanning tree in G with the old weights can be selected by Kruskal's algorithm if and only if that same tree can be selected with the new weights. But Kruskal's algorithm generates *all* the minimum spanning trees in G, so the set of minimum spanning trees with the new weights is the same as with the old. However, minimizing the edge sum of a tree using the new weights is equivalent to minimizing its edge product using the old weights. Thus, under the old weights, a spanning tree has the minimum edge sum if and only if it has the minimum edge product.

Another proof, less elegant but perhaps more revealing, uses Kruskal's algorithm and the constancy of the edge weight spectrum. Suppose T has the minimum edge product but not the minimum edge sum. Let S be a tree with the minimum edge sum. Then the edge weight spectra of T and S disagree. Let k be the smallest weight at which they disagree.

T cannot have more edges of weight k than S. If T did have more, we could execute Kruskal's algorithm as follows:

Choose the edges of T up through weight $k-1$. (The spectrum of these edges agree with the spectrum of S, and so they can be chosen consistently with Kruskal's algorithm.)

Next, choose the edges of T of weight k. (They don't create a cycle with the previously chosen edges of T.)

Finally, finish Kruskal's algorithm in any acceptable way. The result is a minimum spanning tree, but it differs from S in the number of edges of weight k. This contradicts Theorem 3.

So T must have fewer edges of weight k than S. Let $T(k)$ be the set of edges of T with weight less than or equal to k. Then $T(k)$ can be chosen at the front end of Kruskal's algorithm, and since the spectrum has not yet been used up at weight k, there remains at least one edge e of G of weight k that does not make a cycle with $T(k)$. The edge e does, however, make a cycle with T, and so there is some edge on that cycle with greater weight than e. Replace it with e to get a tree with smaller edge product, contrary to hypothesis. Thus, if T has the minimum edge product, it has the minimum edge sum.

Conversely, let T have the minimum edge sum. Let S be a tree with the minimum edge product. By the preceding paragraph, S has the minimum edge sum as well, and therefore has the same edge weight spectrum as T. Thus T has the minimum edge product. ∎

Returning now to Kirchhoff's determinants, we can quickly see that these algebraic objects lead us to the minimum spanning trees of G.

Theorem 5. *Let H be a set of $n-1$ edges of the weighted graph G, and let $M(H)$ be the associated $(n-1)\times(n-1)$ submatrix of A. Then H is a minimum spanning tree if and only if $|\det M(H)|$ is minimal among all nonsingular $(n-1)\times(n-1)$ submatrices of A.*

To establish this theorem, recall that if H is not a tree, then $\det M(H) = 0$, and if H is a tree, then $|\det M(H)|$ is the product of the edge weights of H. Thus, H has the minimum nonzero $|\det M(H)|$ if and only if H is a spanning tree with the minimum edge product, and therefore, by Theorem 4, with the minimum edge sum also. ∎

5. An Algorithm for Finding All Minimum Spanning Trees.

To find all the minimum spanning trees of G, we could search the matrix A for those submatrices with the minimum determinant. This would require evaluating $\binom{b}{n-1}$ determinants. Fortunately, there are more efficient ways to do this search, which we now develop. (The remaining results appear in [**6**].)

We begin by arbitrarily selecting a reference tree $\mathcal{T}$. Certain edges of G belong to some minimum spanning tree while others do not. We can determine which are which by referring to $\mathcal{T}$.

Theorem 6. *An edge e of G belongs to some minimum spanning tree if and only if the following condition holds:*

Either e belongs to $\mathcal{T}$, or there is an edge f on the unique path in $\mathcal{T}$ joining the vertices of e such that $w(f) = w(e)$.

Proof: The "if" part is trivial. The "only if" part is not; a proof appears in [**6**].

Edges that fail to meet this condition can be deleted from G, since their deletion does not disconnect G and they will not be chosen in any search for a minimum spanning tree. Therefore we assume without loss of generality that all edges of G belong to some minimum spanning tree. Next, we partition the edges of G into equivalence classes.

Before defining the equivalence relation, we observe that every equal edge replacement determines a 1–1 function from the edges of T onto the edges of S, where T is a minimum spanning tree and S is the tree that results by replacing some edge e of T by some suitable edge f of equal weight. This function is the identity on all the edges of T except e, and it maps e to f. If $S_0, S_1, \ldots, S_n$ is a sequence of minimum spanning trees resulting from a finite sequence of equal edge replacements, these maps can be composed to give a 1–1 correspondence from the edges of S_0 to the edges of S_n.

For edges e and f, define $e \sim f$ if either $e = f$ or there is a sequence $S_0, S_1, \ldots, S_n$ of equal edge replacements such that e is an edge of S_0, f is an edge of S_n, and the image of e under the composition map is f.

It is easy to see that $\sim$ is an equivalence relation, but it is harder to determine the equivalence classes. Later we show there is a more practical way of describing this relation that makes the computation of the equivalence classes easier.

The reflexive and symmetric properties of $\sim$ are clear, and the transitive property requires only a little attention to the proof of Theorem 2, which can be supplied by the reader.

The graph in Figure 1a provides a simple illustration of this equivalence relation. The two edges of weight three on the left are equivalent to each other but not to either of the edges of weight three on the right.

Since every equal edge replacement preserves equivalence classes, we obtain two refinements of Theorem 3:

Theorem 7. *All minimum spanning trees contain the same number of edges from any given equivalence class.*

Theorem 8. *An edge is in every minimum spanning tree of G if and only if it is related only to itself.*

We now give an elementary algorithm for finding the equivalence classes, using a reference tree. It is shown in [**6**] that the equivalence relation produced by this algorithm is the same as the one defined earlier.

Recall that the edges that do not belong to any minimum spanning tree have been deleted from G. For each remaining edge g not in the reference tree $\mathcal{T}$, let $X(g)$ be the set consisting of g and all edges on the path in $\mathcal{T}$ joining the vertices of g whose weight is equal to $w(g)$. This set is illustrated in Figure 2, where $X(g) = \{g, e, f\}$.

Combine each pair of sets $X(g)$ that have an edge in common. Each resulting combined set forms one equivalence class. Thus, each such equivalence class is a transitivity class of the sets $X(g)$. Edges of G that belong to none of the $X(g)$ form singleton equivalence classes.

In Figure 2, $X(g)$ and $X(h)$ have the edge e in common and are combined. Since the edge d, for example, belongs to none of the sets $X(c)$, $X(g)$, $X(h)$, we see that $[d] = \{d\}$ and d must belong to every minimum spanning tree.

Since every minimum spanning tree of G must contain the same number of edges from a given equivalence class $[e]$, it is reasonable to ask whether any subset of $[e]$ with that same number of edges can be found in some minimum spanning tree. The answer is negative, and a simple example can be found in K_4, the complete graph on four vertices, in which all six edges have been assigned the same weight. All the edges are equivalent to each other and every minimum spanning tree has three edges, but some 3-element subsets are clearly not trees.

In general, we say that a subset S of $[e]$ is a *choice* from $[e]$ if S is the set of edges in $[e]$ in some minimum spanning tree. The main result in [**6**], which is not obvious, is as follows:

Theorem 9. *If S_1 and S_2 are two choices from $[e]$ and T is any minimum spanning tree of G containing S_1, then $(T - S_1) \cup S_2$ is also a minimum spanning tree.*

This theorem guarantees that the choices from a given class are fully interchangeable, and that the set of all minimum spanning trees in G can be generated algorithmically by taking one choice from each equivalence class. In particular, the number of minimum spanning trees of G is the product of the numbers of choices from all the classes.

Figure 3 offers an opportunity to execute this algorithm. In Figure 3a, a weighted graph is given. Each edge has a letter (its name) and a number (its weight). In Figure 3b, a reference tree $\mathcal{T}$ (dark lines) has been chosen arbitrarily. Three edges have been rejected because they do not belong to any minimum spanning tree (Theorem 6), and three others are shown with a dashed line because they do not belong to the reference tree. For the three dashed edges c, h and l, find $X(c)$, $X(h)$ and $X(l)$ and observe that $X(h)$ and $X(l)$ can be combined.

The equivalence classes are $\{a, c, j, k\}$, $\{b, d, h, l\}$, and six other singleton classes. The reader can verify that $\{a, c, j, k\}$ has four choices, each of which has three edges, and that $\{b, d, h, l\}$ has five choices, each with two edges. Each singleton class has, of course, only one choice.

Since there are twenty ways to select one choice from each class, G has exactly twenty minimum spanning trees.

For small graphs, this process can be done visually. For larger graphs, determinants can be used to identify the choices. In any event, Theorem 9 greatly increases the efficiency of the search. Once the equivalence classes have been found, we can determine the choices from any class $[e]$ by simply replacing the choice from $[e]$ in the reference tree $\mathcal{T}$ by another set C of edges from $[e]$ with the same cardinality and computing the determinant of the new submatrix of A so obtained. If the determinant is zero, the set C is not a choice. If the determinant is non-zero, it will have $\pm \det M(\mathcal{T})$ as its value, and thus the set C is a choice. Note that by proceeding in this manner, we never waste time evaluating the determinant of a non-minimum spanning tree.

The number of determinants to be evaluated is thus reduced from $\binom{b}{n-1}$ to

$$\binom{t_1}{i_1} + \binom{t_2}{i_2} + \ldots + \binom{t_m}{i_m}.$$

where t_j is the number of edges in the j^{th} equivalence class E_j, and i_j is the number of edges from E_j that belong to the reference tree.

It is hard to make a general statement about the complexity of this algorithm, because it depends heavily on the number of equivalence classes and the distribution of the edges of G among them. Observe that the complexity decreases as the number of equivalence classes increases.

The graph in Figure 3 has 12 vertices and 14 usable edges. Thus $\binom{b}{n-1} = \binom{14}{11} = 364$ determinants would have to be computed to find the 20 minimum spanning trees. If the algorithm is used, only

$$16 = \binom{4}{2} + \binom{4}{3} + \binom{1}{1} + \binom{1}{1} + \binom{1}{1} + \binom{1}{1} + \binom{1}{1} + \binom{1}{1}$$

determinants need to be computed to determine the choices from all the equivalence classes. This is overstated by $k = 8$, since the reference tree has already identified one choice from each class, and so only 8 determinants need to be computed. This improvement must be weighed against the initial effort of finding a reference tree and computing the equivalence classes.

References

1. P.R. Bryant, Graph theory applied to electrical networks, Chapter 3 of *Graph Theory and Theoretical Physics*, ed. Frank Harary, Academic Press, New York, NY, 1967, 111–137.

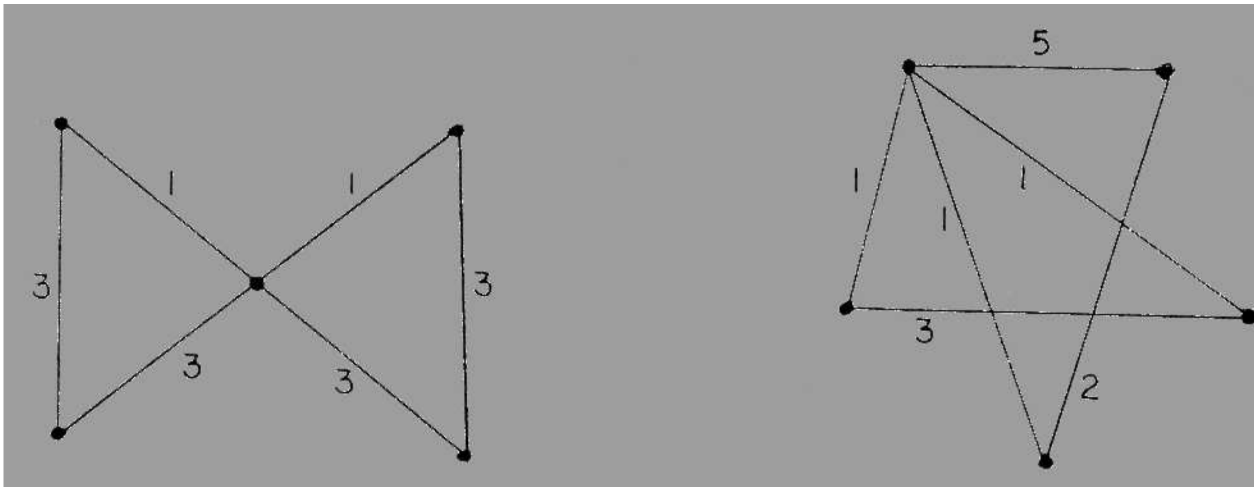
2. Ronald L. Graham and Pavol Hell, On the history of the minimum spanning tree problem, *Annals of the History of Computing* **1**(1985), 43–57.

3. G. Kirchhoff, Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung Galvanische Ströme geführt wird, *Annalen Physik Chemie*, **72** (1847), 497–508.

4. J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, **7**(1956), 48–50.

5. Alan Tucker, *Applied Combinatorics*, 2nd ed., John Wiley, New York, NY, 1984.

6. Perrin Wright, Counting and constructing minimal spanning trees, *Bulletin of the Institute of Combinatorics and its Applications*, **21** (1997), 65–76.



(a) (b)

Figure 1.

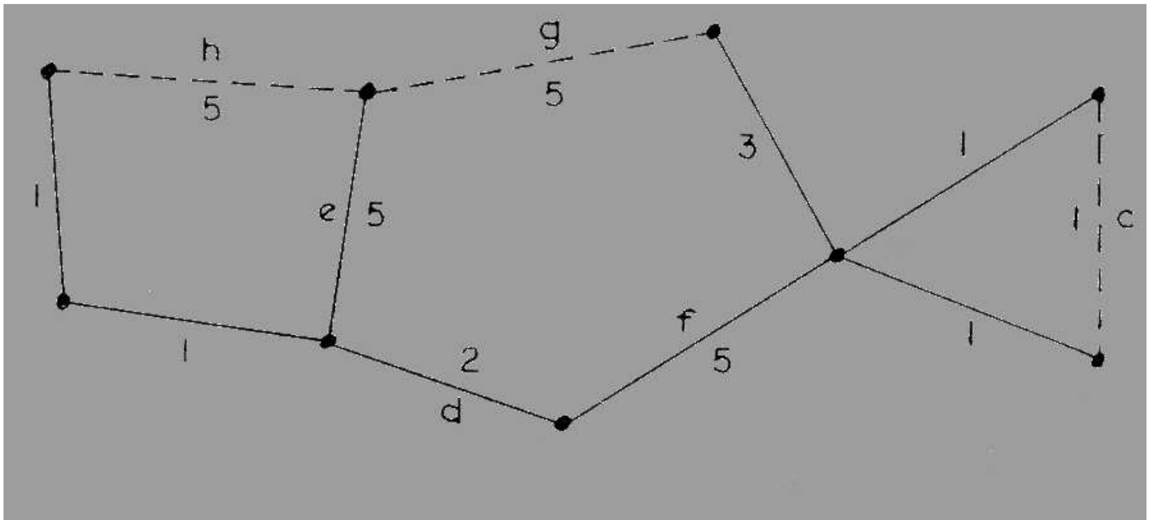
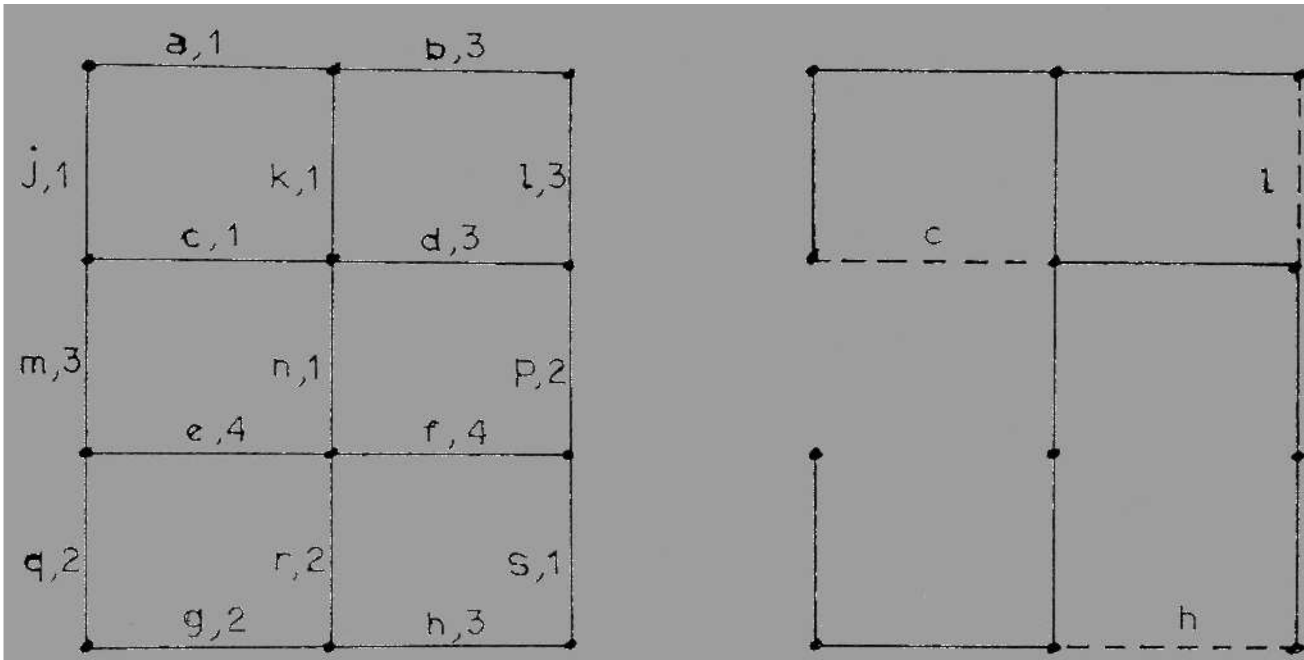


Figure 2.



(a) (b)

Figure 3.