C. L. LAWSON, Editor

An Algorithm for Finding a Fundamental Set of Cycles of a Graph

KEITH PATON
Medical Research Council, London, England*

A fast method is presented for finding a fundamental set of cycles for an undirected finite graph. A spanning tree is grown and the vertices examined in turn, unexamined vertices being stored in a pushdown list to await examination. One stage in the process is to take the top element v of the pushdown list and examine it, i.e. inspect all those edges (v, z) of the graph for which z has not yet been examined. If z is already in the tree, a fundamental cycle is added; if not, the edge (v, z) is placed in the tree. There is exactly one such stage for each of the n vertices of the graph. For large n, the store required increases as n^2 and the time as n^γ where γ depends on the type of graph involved. γ is bounded below by 2 and above by 3, and it is shown that both bounds are attained.

In terms of storage our algorithm is similar to that of Gotlieb and Corneil and superior to that of Welch; in terms of speed it is similar to that of Welch and superior to that of Gotlieb and Corneil. Tests show our algorithm to be remarkably efficient ($\gamma = 2$) on random graphs.

KEY WORDS AND KEY PHRASES: fundamental cycle set, graph, algorithm, cycle, spanning tree
CR CATEGORIES: 5.32

1. Introduction

Let T be a spanning tree of a connected, finite, undirected graph G. The fundamental *cycle-set* of G corresponding to T is the set of cycles $(f, g, \cdots, f)$ of G, each consisting of one edge (f, g) of $G - T$ together with the unique path $(g, \cdots, f)$ in T. It is known [1] that any cycle C of G may be expressed as a sum modulo-2 of the form $C = \sum \gamma_j S_j$, where the γ_j are 0 or 1 and the S_j constitute a fundamental cycle-set for the graph G. Before the set of all cycles can be enumerated, it is convenient to determine a fundamental cycle-set, and Welch [2], and Gotlieb and Corneil [3] have recently published algorithms for this purpose. That of Welch takes less machine time, whereas that of Gotlieb and Corneil requires less machine storage. In this paper we present an algorithm which is as economical in storage as that of Gotlieb and Corneil and as fast as that of Welch.

2. The Algorithm

The algorithm may be concisely expressed as follows. Let E be the set of edges and V the set of vertices of the given graph. Let T be the set of vertices already in the spanning tree, and let X be the set of vertices not yet "examined" as defined below. Initially $T = \varnothing$, $X = V$. Take any vertex v from X as the root of the tree. Then $T = \{v\}$, $X = V$.

Let z be any vertex in $T \cap X$. If there are none, the process is complete. Now *examine* the vertex z; i.e. consider each edge $z - w$ in E and act as follows. If $w \in T$ find the

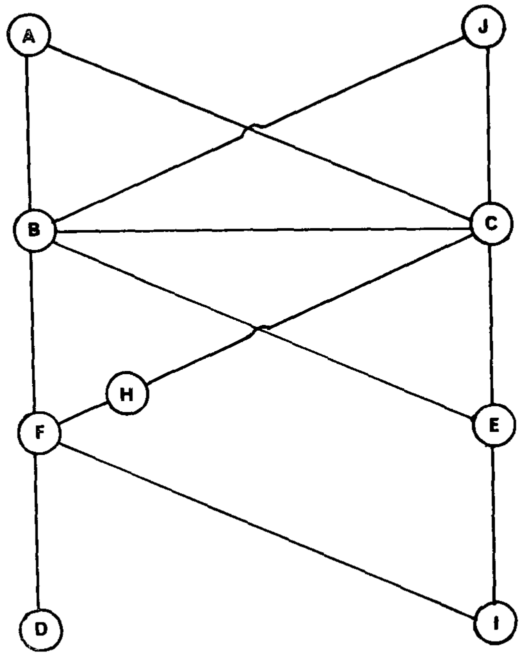

FIG. 1. A graph

* Computer Unit

fundamental cycle consisting of $z - w$ together with the unique path in the tree from w to z. If $w \notin T$ add edge $z - w$ to the tree and therefore w to T. In each case delete edge $w - z$ from E. When all edges $z - w$ have been considered, remove this current z from X and repeat the process with a new z.

Computationally, the selection of "any" vertex from the set $T \cap X$ must take place according to some rule. It is convenient to list the elements of this set in their order of appearance in the tree; two possible rules are to select the last element of the set or the first element of the set. If we always select the last element then the construction of the fundamental cycles is especially simple.

To illustrate the point let us consider the graph of Figure 1. The vertices are inspected in alphabetical order. We define the *trunk* of a vertex v as the path in the tree from v to the first vertex examined, in this case, A.

If we use the last-element method we find the tree of Figure 2, and when we examine vertex F we find that the edge $F - B$ yields a fundamental cycle made up of paths F-H-C-A, B-A, and F-B. We observe that only one of these paths consists of more than a single edge. This is always the case in the last-element method and follows from the fact that every vertex already in the tree but not yet examined lies at distance 1 from the trunk of the vertex now under examination. Because two of the paths always consist of a single edge, the last-element method can use a comparatively straightforward process in tracing through the fundamental cycle.

If we use the first-element method we obtain the tree shown in Figure 3. When we examine vertex F we find that the edge F-H yields a fundamental cycle made up of paths F-B-A, H-C-A, and F-H. Here two of the paths consist of more than a single edge. In order to take account of this possibility, the first-element method must use a much more complicated process in tracing through the fundamental cycle.

The last-element method should therefore be the faster of the two, and this is confirmed by experiments on random graphs of various sizes and densities in which it is usually about 35 percent faster than the first-element method. The last-element method is therefore adopted as the algorithm.

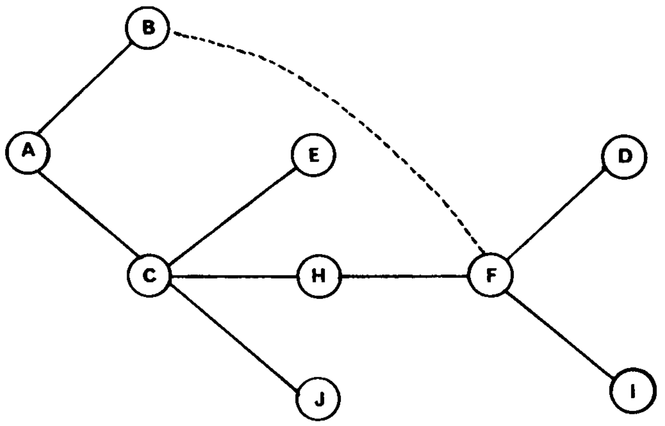

FIG. 2. In the last-element method the vertices join the tree in the order ABCEHJFDI.

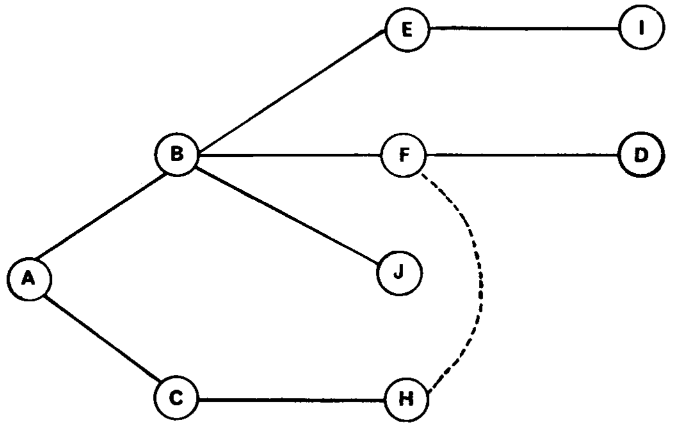

FIG. 3. In the first-element method the vertices join the tree in the order ABCEFJHID.

3. Store

We now assess the store required by our method. If w is the word length of the computer then each row of the adjacency matrix may be written as a sequence of n bits in $\lceil n/w \rceil$ machine words. ($\lceil x \rceil$ denotes the smallest integer not less than x.) Since the adjacency matrix is destroyed in the process of finding the cycles, space must be provided for two copies, taking $2n\lceil n/w \rceil$ words in all.[1] A pushdown list is used to store those vertices already in the spanning tree which have not yet been examined; this can grow to length $n - 2$ at most.[2] Two vectors of length n are em-

[1] One of the referees has suggested that the computation be reorganized so as to use only the lower triangle of the adjacency matrix. At the end of the process the adjacency matrix may then be reconstituted from the unchanged upper triangle. This strategy cuts storage requirements by nearly 50 percent and the increase in program complexity is very slight.

[2] Dr. Corneil has pointed out that this list may be replaced by one of $\lceil n/w \rceil$ words, at a slight cost in increased time.

ployed, LEVEL (i) being the distance of vertex i from the root of the spanning tree, and ANC(i) the vertex v for which $v - i$ is an edge of the tree with v nearer the root. The store required is therefore $2n\lceil n/w\rceil + 3n - 2$ machine words.

To compare our method with the two others we show in Table I the store required for a graph of n vertices and m edges on a machine of word length w. The comparison is complicated by the fact that Gotlieb and Corneil use the vertex-incidence matrix of n^2 elements (as we do), whereas Welch uses the edge-incidence matrix of mn elements. In general, the latter is a less economical representation of a graph than the former, and this discrepancy[3] increases with m.

If we exclude the case $m = n - 1$, then it may be shown from Table I that our method is more economical of store than Welch's in all cases and more economical than Gotlieb's provided $\lceil n/w\rceil \geq 2$. For very large n, the storage requirements are dominated by $2n\lceil m/w\rceil$, $3n\lceil n/w\rceil$, and $2n\lceil n/w\rceil$; it is clear that unless the number of edges is small, Welch's method requires much more store than the other two, and that our method requires about two-thirds as much store as Gotlieb's.

TABLE I

Method	*Store required in machine words*
Welch algorithm (modified)	$2n\left\lceil \frac{m}{w}\right\rceil + 2m + n$
Gotlieb and Corneil algorithm	$3n\left\lceil \frac{n}{w}\right\rceil + 2\left\lceil \frac{n}{w}\right\rceil + n$
Paton algorithm	$2n\left\lceil \frac{n}{w}\right\rceil + 3n - 2$

In common with Gotlieb we allot no space in our algorithm for the storage of the fundamental cycles, since we assume that these can be written to auxiliary store as soon as they are formed. Various schemes may be devised for the compact storage of the fundamental cycle set. For example, the edges of the spanning tree may be numbered $1, 2, \cdots, n - 1$ in the order in which they are added to the tree. Each fundamental cycle consists of one edge not in the tree and a certain set of edges from the tree. Considering a set of $\lceil (n - 1)/w\rceil$ consecutive words as a sequence of $n - 1$ bits, we record the presence of the kth edge in the cycle by a 1 in the kth bit; to record the end points of the edge not in the tree requires two machine words. Thus a fundamental cycle set may be expressed in $(m - n + 1)(2 + \lceil (n - 1)/w\rceil)$ machine words. For certain purposes this representation is convenient; it lends itself naturally to such operations as the logical union and intersection of the cycles of a fundamental set.

[3] For certain types of graphs this discrepancy is not important; for example, if every vertex has degree less than some small constant, say d, then we have $m \leq dn/2$. This is the case in [4], where the graphs represent chemical structures; here the vertices denote atoms and the edges valency bonds; since no atom has valency more than 6, no vertex can have degree more than 6, and the number of edges is not greater than thrice the number of vertices. Welch's method is at no great disadvantage here.

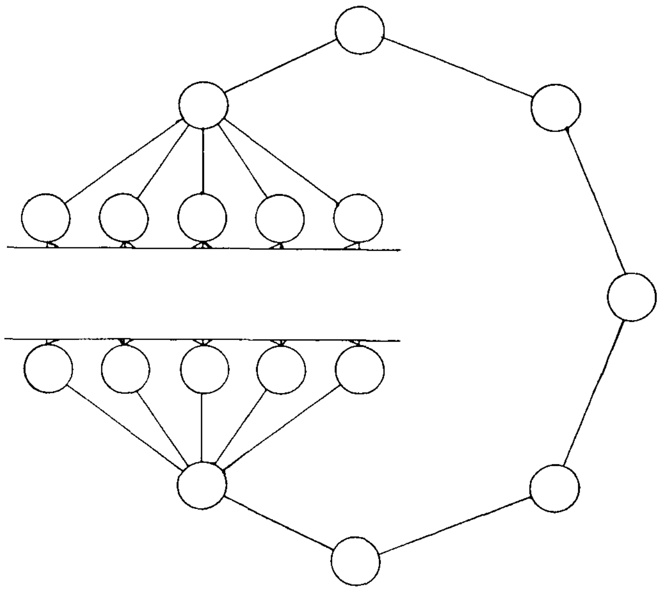
FIG. 4. A graph. The 25 connections across the horizontal lines have been omitted.

4. Speed

We now assess the speed of the algorithm by considering, as in [3], the dependence of the processing time on the number of vertices, n, and the number of edges, m. To do this we consider the separate steps of the method, estimating the number of times each step is performed and the number of orders[4] required on each occasion. Each of $\binom{n}{2}$ possible pairs (i, j) must be tested, at x_1 orders per test, to find whether $i - j$ is an edge in E; $n - 1$ edges are placed in the tree at x_2 orders per entry. Each of the remaining $m - n + 1$ edges yields a fundamental cycle, whose edges are to be enumerated; the time taken for this step may be considered as x_3 orders per cycle and x_4 orders per edge. The total time N may therefore be expressed as $N \approx n(n - 1)x_1/2 + (n - 1)x_2 + (m - n + 1)x_3 + Tx_4$, where T is the total number of edges in the fundamental cycle set. It is sufficient for our purpose to note that the x_i are independent of n, m, T and are all of the same order of magnitude.

Writing m as $\rho n(n - 1)/2$, where ρ is the density, and k for the mean length of the cycles in the fundamental set,

[4] For reasons of euphony we use "orders" as a synonym for "machine cycles."

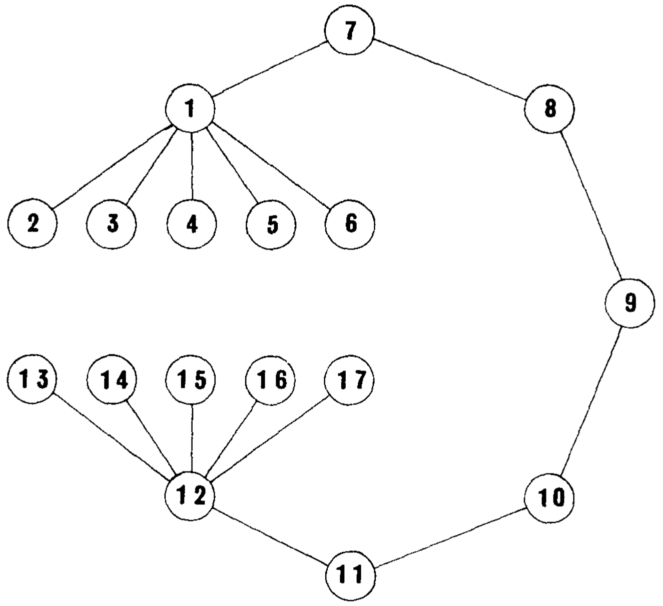

FIG. 5. The graph labeled to yield a fundamental cycle-set of maximum total length

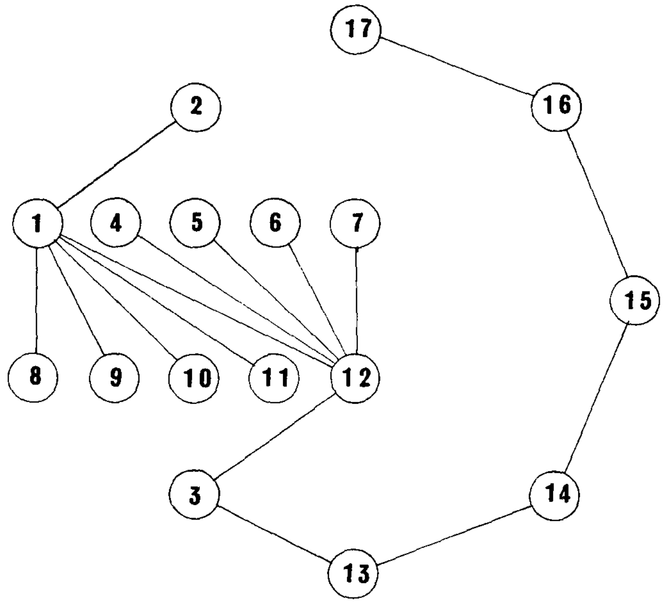

FIG. 6. The graph labeled to yield a fundamental cycle-set of minimum total length

we have $N \approx n(n-1)x_1/2 + (n-1)x_2 + (n-1)\cdot(\rho n - 2)(x_3 + kx_4)/2$. The speed of the algorithm on a given class of graphs is defined in terms of the positive number for which $N = O(n^\gamma)$. By noting that, for all graphs, $3 \leq k \leq n$, we obtain the bounds $2 \leq \gamma \leq 3$, and we will show that each of these bounds is attained.

For sparse graphs—such as the single cycle—whose density tends to zero with increasing n in such a way that $\rho n - 2$ is bounded we have $\gamma = 2$. For the complete graph we have $\rho = 1$, but the spanning tree consists of $n - 1$ edges with a common end point at the root; whence $k = 3$ and again $\gamma = 2$.

We now describe a class of graphs for which γ is 3. The vertex set consists of three subsets A, B, C with p, p, q vertices, respectively; the vertices of C are joined in a string, one end of the string being joined to each vertex of A and the other to each vertex of B; each vertex of A is joined to each vertex of B. The case $p = 5$, $q = 7$ is shown in Figure 4, although the edges joining members of A and B have been omitted for clarity. The total length of the fundamental cycle set found by our algorithm is critically dependent on the ordering of the vertices; thus the ordering of Figure 5 yields 25 cycles of length 9, whereas that of Figure 6 yields 24 cycles of length 4 and one of length 9.

In general, the maximum total length for a graph of this type is given by $T = p^2(q + 2)$; to find the graph with the largest T for a given n we note that $n = q + 2p$ and deduce that T has a maximum of approximately $(n + 2)^3/27$ at $p \cong ((n + 2)/3)$. Thus on a graph of this type, with its vertices ordered in the most unfavorable fashion, the algorithm is of order n^3. For this graph the dependence of T on the ordering is very striking for the minimum total length is given approximately by $4(((n + 2)/3)^2 - 1) + ((n + 2)/3) + 2$, which is of order n^2.

To test the method we record the average time taken by a FORTRAN IV version of the algorithm to find a fundamental cycle-set for several random graphs of given size and density. As expected, we find that the time, t minutes, and the number of vertices, n, are related by $\log(t) = a + b \log(n)$, where the constant b is the γ which defines the speed of the algorithm. The constant a is a property not of the algorithm itself but of the algorithm as written in FORTRAN IV and run on the IBM 7094-II. The values of a, b are those underlined in Table II. For comparison we also show the values for three other possible techniques. We conclude that where space permits the storing of the

TABLE II

Technique used	*Density of graphs*					
	.2		.5		.8	
	a	*b*	*a*	*b*	*a*	*b*
Last element unpacked matrix	−14.9	2.04	−14.2	1.95	−14.0	1.96
Last element packed matrix	−13.9	2.05	−13.5	2.04	−13.3	2.03
First element unpacked matrix	−14.0	1.84	−13.8	1.96	−13.7	2.01
First element packed matrix	−14.0	2.08	−13.2	2.02	−12.9	2.00

The values of a, b are those for which $\log(t) \approx a + b \log(n)$, t being the time in minutes to deal with a graph of n vertices.

TABLE III

Method	Number of vertices 10	20	30	40	50	60
Welch	3.54	4.06	3.76	4.03	4.09	4.27
Gotlieb and Corneil	4.80	6.06	7.93	10.59	11.18	11.44
Paton algorithm	3.57	4.32	4.14	4.46	4.12	4.33

Values of k, the mean length of a fundamental cycle in random graphs of density 0.5. (Data supplied by Dr. Corneil.)

adjacency matrix one entry per word (rather than in binary form) a saving in time of some 35 percent may be made by doing so.

It is interesting to note that a and b are independent of the density of the random graph and very encouraging to find that b is approximately 2. From this we deduce that k must be more or less independent of n and inspection shows that for random graphs of various densities from .1 to .9 the value of k is bounded by 6. Welch's method [2] shares this advantage, whereas in that of Gotlieb and Corneil [3] the value of k increases with n. Table III illustrates this point by showing the values of k for all three methods on the same set of random graphs of density 0.5.

From Table II we see that the order of our algorithm is a constant, independent of the density of the random graph involved, and Gotlieb and Corneil [3] have found that the same property holds for both their own method and that of Welch. Results on random graphs of one density are therefore typical, and it is possible to make a direct comparison of our method with these two by running all three methods[5] on random graphs of one given density, say 0.5. The times obtained are shown in Figure 7, where the slope of the line gives the order of the method; it will be seen that our method (slope 2) is slightly superior to that of Welch (slope 2.6) and considerably superior to that of Gotlieb and Corneil (slope 3.6).

5. Conclusion

Of the two algorithms considered in [3], Gotlieb and Corneil point out that their own method is the more economical of store whereas that of Welch is the faster. As our method is as economical of store as that of Gotlieb and Corneil and as fast as that of Welch, the algorithm presented here is superior to both. Our method is fast because the cycles are found as the tree is grown, and economical because the graph is represented by its vertex-incidence matrix rather than its edge-incidence matrix. For very sparse graphs these two matrices do not differ significantly in size, and for these graphs the superiority of our method to that of Welch is less strongly marked. It is interesting to note that for random graphs of various densities the time required by our algorithm increases only as n^2, as compared with $n^{2.6}$ for that of Welch and $n^{3.6}$ for that of Gotlieb and Corneil.

[5] I am much indebted to Dr. Corneil, who programmed my algorithm in MAP in order to carry out this comparison.

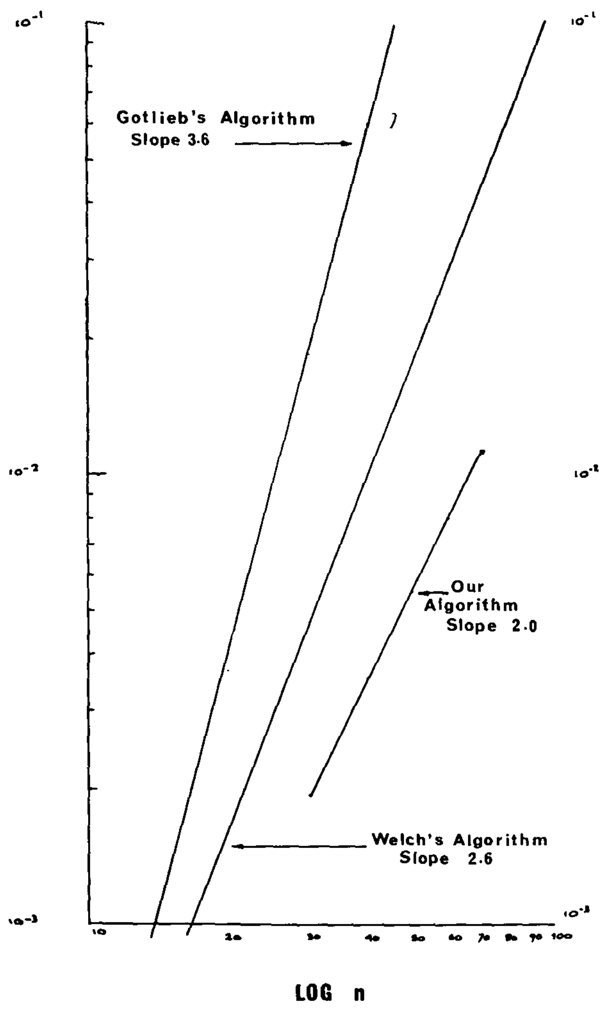

FIG. 7. The times refer to MAP programs processing random graphs of density .5 on the IBM 7094-II (Data supplied by Dr. Corneil.)

Acknowledgments. In refereeing this paper, Dr. D. Corneil coded my algorithm in MAP and ran all three methods on the IBM 7094-II at the University of Toronto. He has generously allowed me to use the results of his comparison, which appear in Table III and Figure 7. I should also like to thank Miss P. A. Thruston for illustrations and Mr. F. Paice for photography.

RECEIVED JUNE, 1968; REVISED MAY, 1969

REFERENCES

1. GOULD, R. The application of graph theory to the synthesis of contact networks. Proc. International Symp. on the Theory of Switching, Pt. I, Apr. 2–5, 1957. In *The Annals of the Computation Laboratory of Harvard University, Annals No. 29*, Harvard U. Press, Cambridge, Mass., 1959, pp. 244–292.
2. WELCH, J. T., JR. A mechanical analysis of the cyclic structure of undirected linear graphs. *J. ACM 13*, 2 (Apr. 1966), 205–210.
3. GOTLIEB, C. C., AND CORNEIL, D. G. Algorithms for finding a fundamental set of cycles for an undirected linear graph. *Comm. ACM 10*, 12 (Dec. 1967), 780–783.
4. SUSSENGUTH, E., JR. A graph theoretical algorithm for matching chemical structures. *J. Chem. Doc. 5*, 1 (Feb. 1965), 36–43.