Fourth International Derive TI-89/92 Conference
Liverpool John Moores University, July 12 – 15, 2000

Using DERIVE to Interpret an Algorithmic Method for Finding Hamiltonian Circuits (and Rooted Paths) in Network Graphs

Peter Schofield
Trinity and All Saints (University of Leeds), UK
Email: p.schofield@tasc.ac.uk

Introduction

The problem of finding Hamiltonian circuits in a network (directed) graph is one of the well-known problems of graph theory. There are no known simple necessary and sufficient conditions for a general digraph or graph to have a Hamiltonian circuit. However, there are many algorithms for finding and calculating Hamiltonian circuits of a digraph or graph. This paper deals with such an algorithm. The algorithmic method used is particularly straightforward, suitable for interpretation with DERIVE and suitable for introducing the problem to mathematics students.

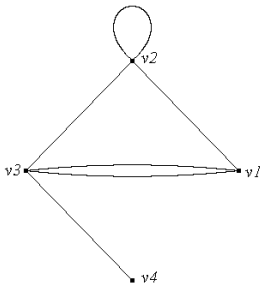
Network Graphs and Digraphs

Some Basic Definitions

A **network (di)graph** G is a pair (V , E) where V is a non-empty set of **vertices** and E is a set of **edges**. Each edge $e \in E$ corresponds to a pair 'v1,v2' of vertices. If 'v1,v2' is an unordered pair then we say that e is an **undirected** edge, else if 'v1,v2' is an ordered pair then e is a **directed** edge v1 → v2. We say that e **joins** v1 and v2, v1 and v2 are **adjacent** and v1 and v2 are **incident with** e.

If all of the edges of G are undirected then we simply refer to G as a **graph**, otherwise G will be called a **digraph**. However, each undirected edge can be represented by a pair of directed edges (one in each direction), and so a graph can also be thought of as a special type of digraph.

The **adjacency matrix** $A = [a_{i,j}]$ for a graph G with $V = \{v1, \ldots\ldots vn\}$ is an $n \times n$ matrix such that $a_{i,j}$ = number of edges joining vi to vj $(1 \le i, j \le n)$**.** For example

$G =$ 

$\Rightarrow$

	v1	v2	v3	v4
v1	0	1	2	0
v2	1	2	1	0
v3	2	1	0	1
v4	0	0	1	0

$\Rightarrow A =$

$$\begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Fourth International Derive TI-89/92 Conference

Remarks

(i) In this example, since *G* is graph, *A* is a symmetric matrix.

(ii) v2 has an undirected loop, and so v2 is technically joined to itself by a pair of directed edges. Hence $a_{2,2} = 2$.

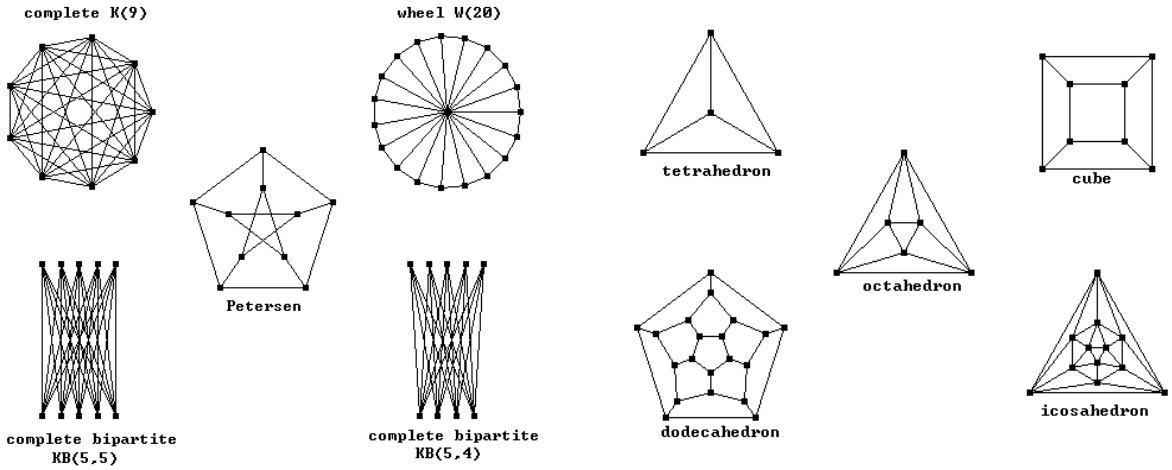
(iii) We shall often adopt the convention of labelling the vertices of a (di)graph *G* with n vertices as 1, 2, ..., n. This sets up a direct correlation between the vertices of the graph drawing and rows/columns of the adjacency matrix.

In a digraph, each incident edge directed towards a vertex v is called an **in-edge** of v and each incident edge directed away from v is called an **out-edge**. The **in/out degree** of a vertex v is the number of in/out-edges incident with v. In a graph, the **degree** of a vertex v is simply the number of edges incident with v.

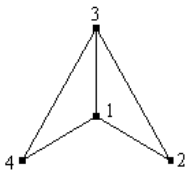
A **walk** is a sequence e1 , ... , ek of edges such that, there exists a sequence of vertices v1, ..., v(k+1) , where (for i = 1, ..., k) ei joins vi and v(i+1). We say the v1 and v(k+1) are **connected**. A digraph G is called **strongly connected** if each pair of distinct vertices of G is connected. A **path** is a walk in which each vertex is distinct, and a **trail** is a walk in which each edge is distinct. A trail is called a **circuit** (or a **cycle**) if it begins and ends with the same vertex.

Some Examples of Graphs

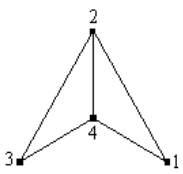
Here is a selection of drawings of some of the featured graphs:



Note: the appearance of the adjacency matrix of a graph will depend upon the way its vertices are labelled. For example,



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



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

Fourth International Derive TI-89/92 Conference

Hamiltonian Circuits and Paths

A **Hamiltonian circuit** in a (di)graph *G* is a circuit that visits each vertex of G. Necessarily, *G* must be strongly connected to have a Hamiltonian circuit. A Hamiltonian circuit can be directed or undirected. An undirected Hamiltonian circuit can be represented as pair of directed Hamiltonian circuits.

A **Hamiltonian path** in a (di)graph *G* will be defined as a path in *G* which visits each vertex of *G*. Such a path will start from a **root vertex**, u say, and visit every other vertex of G but, from the end vertex of the path, there will be no edge link back to u.

Remarks

(i) The problem of finding Hamiltonian circuits and rooted paths in graphs and digraphs is one of the well-known problems of graph theory. In general, there are no known simple necessary and sufficient conditions for a graph to be Hamiltonian (have a Hamiltonian circuit). The Open University Course Unit MT356 (1995) states "The search for necessary or sufficient conditions for a graph to be Hamiltonian is a major area of study in graph theory today". Wilson, R. J. (1995) states "the finding of such a characterisation (for a graph to be Hamiltonian) is one of the major unsolved problems of graph theory".

(ii) There are many algorithmic methods for finding and calculating Hamiltonian circuits and rooted paths in graphs and digraphs. A comprehensive introduction to some of these, and to the more advanced topic of NP completeness, can be found in Jungnickel, D. (1998). The straightforward algorithms in this paper make principal use of the adjacency matrix of a digraph G for finding Hamiltonian circuits and rooted paths. The methods employed are particularly suitable for processing and interpreting with an algebraic manipulator like DERIVE.

Contracting on an Edge of a Digraph

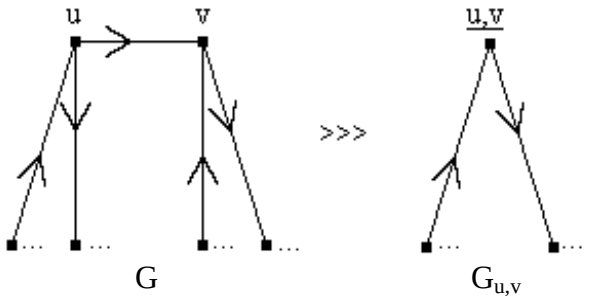
The reduction step we shall use in algorithms for Hamiltonian circuits and paths involves contracting on the edge of a directed graph.

The Contraction Step

Suppose *G* is a digraph with an edge u→v joining distinct vertices. Form $G_{u,v}$ by:

(i) deleting all the out-edges from u and all of the in-edges to v;

(ii) combining u and v to form a single vertex with label u,v.



Fourth International Derive TI-89/92 Conference

Remark
If G also has an edge v→u, then $G_{u,v}$ will contain a loop u,v→u,v. In most cases it will be simpler to remove this loop as well.

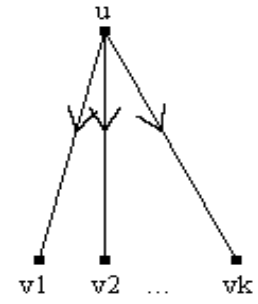
LEMMA 1
Given a contraction $G \Rightarrow G_{u,v}$ on a diedge u→v.
For each Hamiltonian circuit H of G, containing u→v, there exists a unique Hamiltonian circuit H* of $G_{u,v}$,with the same edge sequence as H, except that, when H follows the path t→u→v→w, H* follows the path t→u,v→w.

Proof
Follows directly from the construction of the contraction step.

Remark
A similar result to LEMMA 1 will hold for each Hamiltonian path rooted at u.

Repeating the Contraction on all of the Out-edges of a Root Vertex

The next stage in the reduction is to select a vertex u of G as a 'root' vertex and carry out the contraction step on all of the out-edges of u joining distinct vertices.



$G \Rightarrow G_{u,v1}, G_{u,v2}, ..., G_{u,vk}$

LEMMA 2

Given contractions $G \Rightarrow G_{u,v1}, G_{u,v2}, ..., G_{u,vk}$ on all of the out-edges of a root vertex u of a digraph G. For each Hamiltonian circuit H of G there corresponds a unique Hamiltonian circuit H* belonging to exactly one of $G_{u,v1}, G_{u,v2}, ..., G_{u,vk}$.

Proof
H contains exactly one of the out-edges from u. Suppose it is u→vj (for some fixed j, 1≤ j ≤k) then, by LEMMA 1, $G_{u,vj}$ contains a unique Hamiltonian circuit H* corresponding to H. None of the other $G_{u,vi}$ ($i \neq j$) will have contracted on the edge u→vj, and so they do not have Hamiltonian circuits corresponding to H. (Note: some of the v1, v2, ..., vk may be labels representing the same vertex of G.)

Remark
A similar result to LEMMA 2 will hold for each Hamiltonian path rooted at u.

LEMMA 3
Given contractions $G \Rightarrow G_{u,v1}, G_{u,v2}, ..., G_{u,vk}$ on all of the out-edges of a root vertex u of a digraph G.
Let H(G) = the number of Hamiltonian circuits of G and $HP_u(G)$ = the number of Hamiltonian paths (rooted at u) of G, then

(i) $H(G) = \sum_{j=1}^{k} H(G_{u,vj})$

(ii) $HP_u(G) = \sum_{j=1}^{k} HP_{u,vj}(G_{u,vj})$

Proof
Follows as a direct consequence from LEMMA 2 and its Remark.

Illustrating the Algorithm by means of Graph Drawings

An Algorithm for finding Hamiltonian Circuits (and Rooted Hamiltonian Paths) in a Digraph *G*:

(i) select a root vertex u of *G*;

(ii) contract on all of the out-edges of u joining distinct vertices to form $G_{u,v1}$, $G_{u,v2}$, ..., $G_{u,vk}$;

(iii) for each j = 1, ..., k take u,vj to be the root vertex of $G_{u,vj}$ and repeat step (ii);

(iv) continue until all of the Hamiltonian circuits (rooted paths) in the reduced graphs can be easily identified.

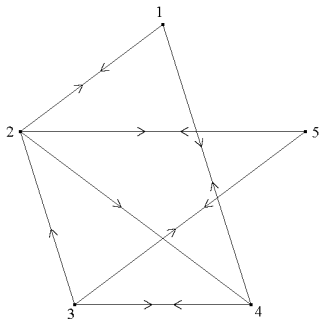
Remarks

(i) Obviously, for a general digraph there is no way this algorithm can be completed in 'polynomial time'. For example, in the case of the complete graph K_n we have (n-1)! distinct directed Hamiltonian circuits. However, this algorithm is still worth some detailed study.

(ii) For simple graphs it is instructive to carry out the algorithm using digraph drawings.

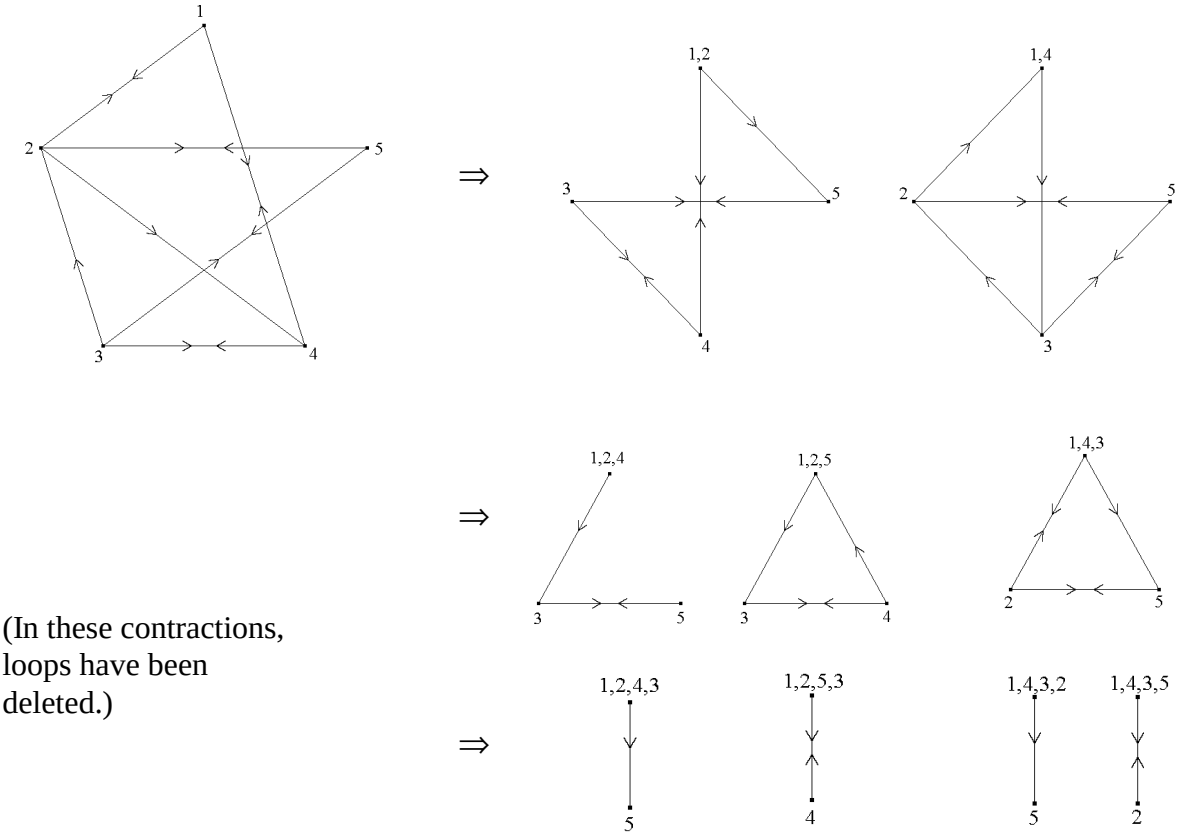
Example 1
Find all of the Hamiltonian circuits and paths (rooted at 1) of the following digraph:

G =



(Note: undirected edges have been represented as bi-directional diedges)

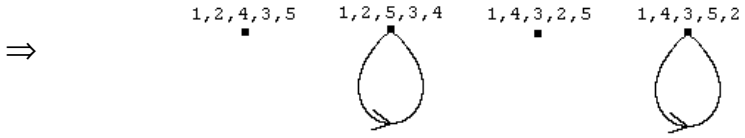
Solution



(In these contractions, loops have been deleted.)

By this stage it is obvious that *G* has two directed Hamiltonian circuits: 1,2,5,3,4,1 and 1,4,3,5,2,1 (these are the same undirected circuit transcribed in opposite directions) and two Hamiltonian paths (rooted at 1): 1,2,4,3,5 and 1,4,3,2,5.

However, continuing the reduction process a further final step, and this time leaving in any resultant loops, we obtain:



The Hamiltonian circuits or paths of *G* can now be identified by the presence or absence of loops.

Counting Directed Hamiltonian Circuits and Rooted Paths

Applying the Contraction to the Adjacency Matrix of a Digraph *G*

Label the vertices of *G* using 1, 2, ..., n and designate 1 as the root vertex. Let *A* be the n×n adjacency matrix of *G*. That is, $A = \left[a_{i,j} \right]$ for $1 \le i, j \le n$.

What happens to A when we carry out the contraction step on an edge 1→j (for some fixed j, $1 \le j \le n$)?
Deleting all out-edges from 1 ⇒ deleting first row of A;
deleting all in-edges to j ⇒ deleting jth column of A.
That is, we form the minor[1] $M_{1,j}$. However, to continue the algorithm, 1,j has to become the root vertex of $G_{1,j}$, and so we transfer the (j−1)th row of $M_{1,j}$ to the first row to form the **minog** MG_j.

For example,

$$\text{if } A = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \end{bmatrix}, \text{ then } MG_2 = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \text{ and } MG_3 = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}.$$

LEMMA 4
Let G be a digraph with an n×n adjacency matrix A.
Let H(A) = the number of directed Hamiltonian circuits of G.
Let HP(A) = the number of rooted Hamiltonian paths of G.

Then

(i) $$H(A) = \sum_{j=2}^{n} a_{1,j}.H(MG_j)$$

(ii) $$HP(A) = \sum_{j=2}^{n} a_{1,j}.HP(MG_j)$$

Proof
$a_{1,1}$ registers G's loops at vertex 1. Loops cannot be part of Hamiltonian circuits (rooted paths), consequently the summation starts at j = 2 .
$a_{1,j}$ $(2 \le j \le n)$ registers the number of edges of G of form 1→j. A contraction can be carried out on each one of these edges to form $G_{1,j}$ with adjacency matrix MG_j. As a consequence of LEMMA 2 the number of Hamiltonian circuits (rooted paths) in each $G_{1,j}$ is equal to the number of Hamiltonian circuits of G using that particular edge of form 1→j. There are $a_{1,j}$ such edges. Whence result.

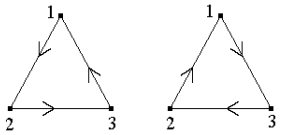
Remarks

(i) For teaching purposes, parallels can be drawn between the results of LEMMA 4 and the calculation of the standard determinant of matrix A.

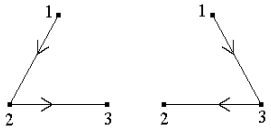
(ii) For each j $(2 \le j \le n)$ MG_j will be an (n−1)×(n−1) matrix, and so LEMMA 4 can be used as a recursion reduction step.

(iii) It remains to establish an initial case. We can do this for n = 3.

In this case, there are two possible directed Hamiltonian circuits:
Hence, $H(A)=a_{1,2}.a_{2,3}.a_{3,1} + a_{1,3}.a_{3,2}.a_{2,1}$.



There are also two possible Hamiltonian paths (rooted at 1):



[1] Here, I am using the term 'minor' to represent the submatrix rather than its determinant.

$$HP(A)=a_{1,2}.a_{2,3}.\overline{sg}(a_{3,1})+a_{1,3}.a_{3,2}.\overline{sg}(a_{2,1})$$

Hence, where $\overline{sg}(x) = \begin{cases}1 & \text{if } x=0\\ 0 & \text{if } x>0\end{cases}$.

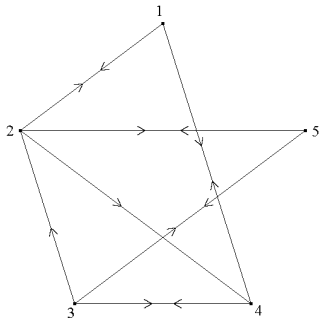
Using DERIVE to Calculate the Number of Directed Hamiltonian Circuits and Rooted Paths

Using DERIVE with the above results it is straightforward to set up recursive definitions for H(*A*) and HP(*A*) (see DERIVE file HAM.MTH in the Appendix).

Example 2

Load HAM.MTH as a DERIVE Utility File and assign to A the adjacency matrix of the graph *G* of Example 1 by Authoring:

A := [[0,1,0,1,0],[1,0,0,1,1],[0,1,0,1,1],[1,0,1,0,0],[0,1,1,0,0]]



Simplifying H(A) yields H(*A*) = 2.
Simplifying HP(A) yields HP(*A*) = 2.

Remark
In order to make the best use HAM.MTH, students will need an efficient method of generating a wide selection of network graphs together with their adjacency matrices. For this purpose, I have written a DERIVE Utility File NGRAPHS.MTH (together with NGRAPHS.DMO and NGRAPHS.DOC). As well as providing facilities for drawing a wide selection of network graphs (see the graph diagrams in this paper) NGRAPHS.MTH provides instructions for generating the adjacency matrices and adjacency lists of these graphs. Albert Rich (co-author of DERIVE) has kindly agreed to include these three files in the Users Folder of DERIVE 5. These files (including HAM.MTH) can also be obtained by email from: p.schofield@tasc.ac.uk.

The following table has been constructed using adjacency matrices generated by NGRAPHS.MTH and HAM.MTH:

TABLE 1 Counting Hamiltonian Circuits and Rooted Paths

GRAPH	Number of directed Hamiltonian circuits	Number of rooted Hamiltonian Paths	Approx Completion Time (seconds)
Wheel Graph W_{20}	38	0	13
Complete Graph K_9	40320	0	47
Complete Bipartite Graph $K_{5,5}$	2880	0	11
Complete Bipartite Graph $K_{5,4}$	0	576	2.6
Complete Bipartite Graph $K_{4,5}$	0	0	2.5
Tetrahedron	6	0	.09
Cube	12	6	.6

Octahedron	32	8	.4
Dodecahedron	60	102	41
Icosahedron	2560	3760	57
Petersen	0	24	1.25

Remarks

(i) These results agree with known results for these graphs. The times are for a laptop with 64 MB of RAM running at 400 Mhz.

(ii) Since these are all examples of undirected graphs, halve the values in Column 1 for the number of undirected Hamiltonian circuits in each graph.

(iii) The different values for number of Hamiltonian paths in $K_{5,4}$ and $K_{4,5}$ result from, in the first case, the root vertex is in the '5-subset' and, in the second, it is in the '4-subset'.

(iv) The Petersen graph is a well-known example of a graph with Hamiltonian paths but no Hamiltonian circuits.

Generating Hamiltonian Circuits and Rooted Paths of a Digraph

Lists of Hamiltonian Circuits and Rooted Paths

A **simple digraph** is a digraph with at most one directed edge linking each ordered pair of vertices (and no loops). Throughout this Section we shall restrict our examples and arguments to simple digraphs (although HAM.MTH can also process digraphs with multiple edges and loops).

In Example 1, using the algorithmic process, we were able to find all of the Hamiltonian circuits and Hamiltonian paths (rooted at 1) of G. It is possible to construct these using DERIVE.

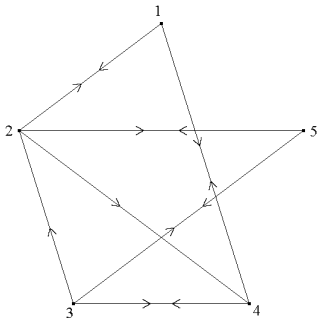
HAM.MTH has been designed to do this using ITERATION, with a seed (input) of form:
[A, [[1],[2, ..., n]]], where A is the n×n adjacency matrix of a digraph. On each application of the contraction step, A is reduced to a corresponding minog and, in the second element of the seed, the label of the corresponding vertex is transferred from the second list to the first.

This is not the place to analyse the structure of HAM.MTH in detail, but it is instructive to explain how it can be used with examples.

Example 3

Load HAM.MTH as a DERIVE Utility File and assign to A the adjacency matrix of the graph *G* of Example 1 by Authoring:

A := [[0,1,0,1,0],[1,0,0,1,1],[0,1,0,1,1],[1,0,1,0,0],[0,1,1,0,0]]



Fourth International Derive TI-89/92 Conference

Simplifying ROUTES(A) yields the following sequence, illustrating each stage of the algorithm:

$$\left[\left[\begin{bmatrix}1&0&1&1\\0&0&1&1\\1&1&0&0\\0&1&0&0\end{bmatrix}\ [[1,\ 2],\ [3,\ 4,\ 5]]\right],\left[\begin{bmatrix}1&0&1&0\\1&0&0&1\\0&1&0&1\\0&1&1&0\end{bmatrix}\ [[1,\ 4],\ [2,\ 3,\ 5]]\right]\right],\left[\left[\begin{bmatrix}1&1&0\\0&0&1\\0&1&0\end{bmatrix}\ [[1,\ 2,\ 4],\ [3,\ 5]]\right],\left[\begin{bmatrix}0&1&0\\0&0&1\\1&1&0\end{bmatrix}\ [[1,\ 2,\ 5],\ [3,\ 4]]\right],\left[\begin{bmatrix}0&1&1\\1&0&1\\0&1&0\end{bmatrix}\ [[1,\ 4,\ 3],\ [2,\ 5]]\right]\right],\left[\left[\begin{bmatrix}0&1\\0&0\end{bmatrix}\ [[1,\ 2,\ 4,\ 3],\ [5]]\right],\left[\begin{bmatrix}0&1\\1&0\end{bmatrix}\ [[1,\ 2,\ 5,\ 3],\ [4]]\right],\left[\begin{bmatrix}1&1\\0&0\end{bmatrix}\ [[1,\ 4,\ 3,\ 2],\ [5]]\right],\left[\begin{bmatrix}0&1\\1&0\end{bmatrix}\ [[1,\ 4,\ 3,\ 5],\ [2]]\right]\right]$$

```
[[0], [1, 2, 4, 3, 5], [], [1], [1, 2, 5, 3, 4], [], [0], [1, 4, 3, 2, 5], [], [1], [1, 4, 3, 5, 2], []]
```

(Note how these correspond to the graph drawings in the solution of Example 1.)

In the final stage, Hamiltonian circuits, paths are identified by 1's, 0's, respectively, in the 1st, 4th, 7th, etc. matrix (these are the iterated minogs reduced to single element matrices).

Simplifying HCS(A) and HPS(A) will display the following lists of Hamiltonian circuits and Hamiltonian paths for *G*:

$\begin{bmatrix}1&2&5&3&4&1\\1&4&3&5&2&1\end{bmatrix}$ - Hamiltonian directed circuits;

$\begin{bmatrix}1&2&4&3&5\\1&4&3&2&5\end{bmatrix}$ - Hamiltonian rooted paths.

Remark
A nice feature of HAM.MTH is the way it can be used to illustrate the process as well as generating the final lists of results. This can also be carried out for the graphs in TABLE 1 above.

Canopy Diagrams

We shall now introduce a form of graphical interpretation.

Definitions
Let *G* be a digraph with vertices 1, 2, ..., n.
The **Hamiltonian circuit canopy diagram** plots each Hamiltonian circuit [1, h1, h2, ..., h(n-1),1] of *G* as a path of points [(0,1), (1,h1), ..., (n−1,h(n−1)), (n,1)].
The **Hamiltonian path canopy diagram** plots each Hamiltonian path [1, h1, h2, ..., h(n-1)] of *G* as a path of points [(0,1), (1,h1), ..., (n−1,h(n−1))] (in this case there is no edge link back to vertex 1).

HAM.MTH can construct canopy diagrams. The 2D-plot facilities of DERIVE are particularly suitable for this purpose. In the 2D-plot Window it is possible to zoom out until the entire canopy diagram is displayed, then select a suitable frame to fit the diagram into a maximized 2D-plot Window display.

For best results use the following 2D-plot Window Option settings: Options> Change Plot Colors (Off) ; Options>Display>Points- Connected, Large.

Fourth International Derive TI-89/92 Conference

To operate, assign the adjacency matrix of a graph to A and Simplify HCAN(A) and HPCAN(A) to generate sets of points for plotting the canopy diagrams of Hamiltonian circuits and rooted paths, respectively. Plot these points in DERIVE's 2D-plot Window to obtain the diagrams (see TABLE 2). In DERIVE 5, with Options>Simplify Before Plotting (On), canopy diagrams can be plotted directly into the 2D-plot Window, without first having to Simplify the expressions in the Algebra Window. Be patient! For the larger canopy diagrams, it will take DERIVE several minutes to calculate all of the required points.

Remarks

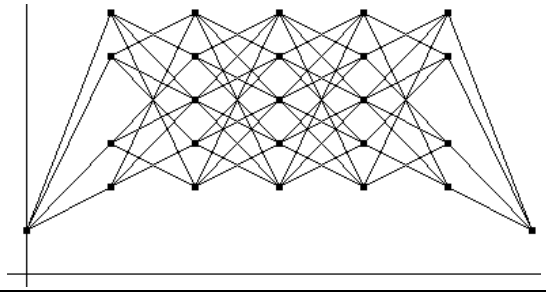
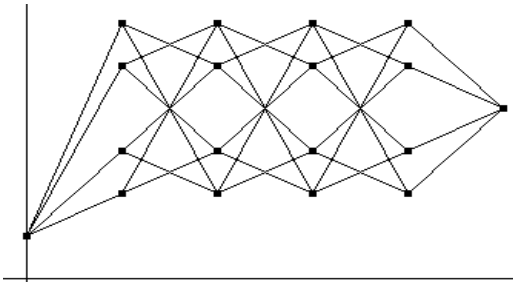
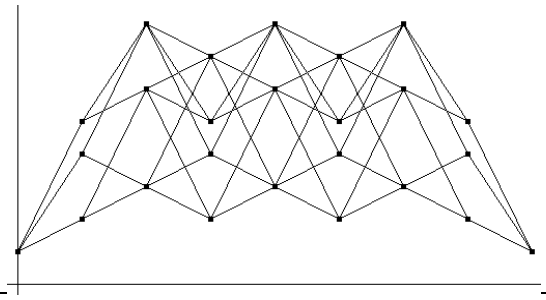
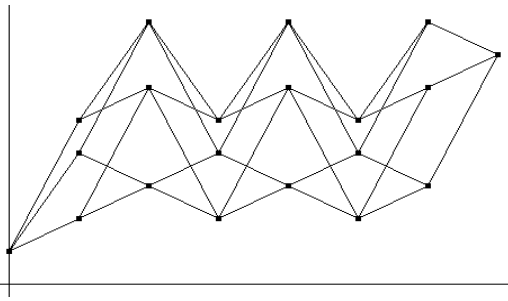
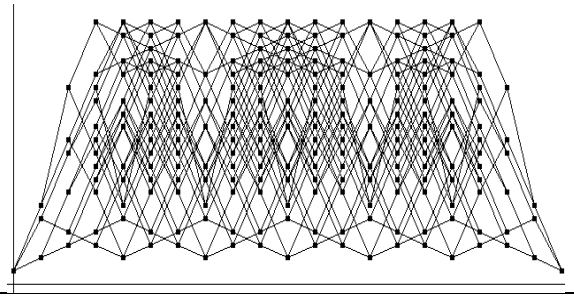
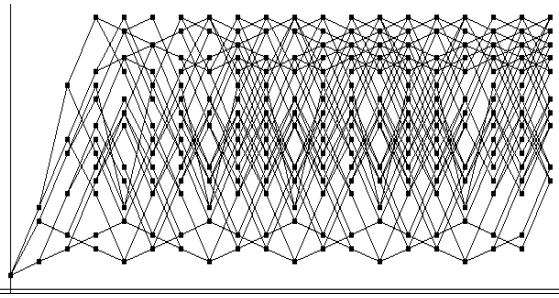
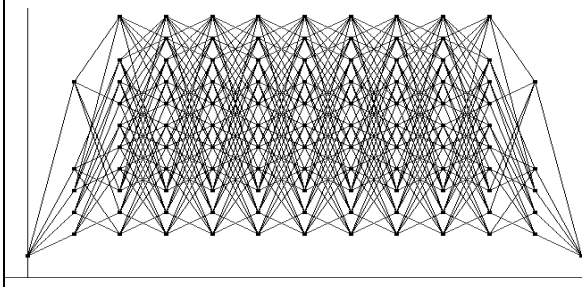
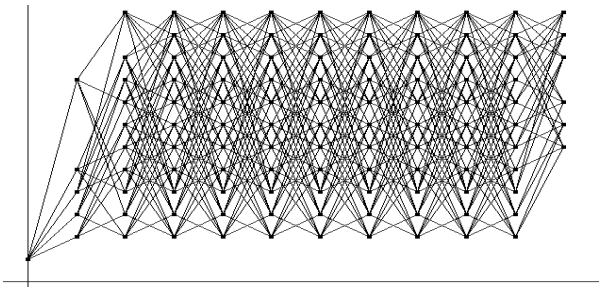
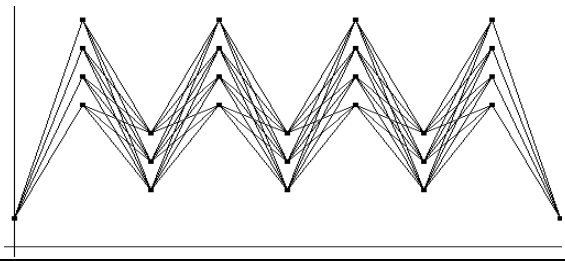
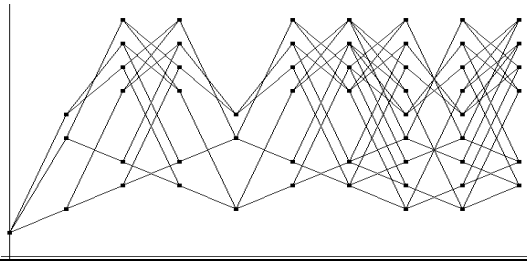
(i) Since all of the examples in TABLE 2 are graphs, their canopy diagrams for directed Hamiltonian circuits are necessarily symmetric about the line $x = n/2$, where n is the number of vertices.

(ii) Note that, the problem of finding a Hamiltonian circuit in a network digraph *G* corresponds to the problem of finding a special type of path in the Hamiltonian circuit canopy diagram of *G*.

(iii) The appearance of a canopy diagram will also be affected by the way the vertices of a graph are labelled.

Fourth International Derive TI-89/92 Conference

TABLE 2. Canopy Diagrams

Graph	Hamiltonian Directed Circuits	Hamiltonian Rooted Paths
Octa-hedron		
Cube		
Dodeca-hedron		
Icosa-hedron		
Complete Bipatite Graph $K_{4,4}$		
Petersen Graph		

Fourth International Derive TI-89/92 Conference

The Travelling Salesperson Problem

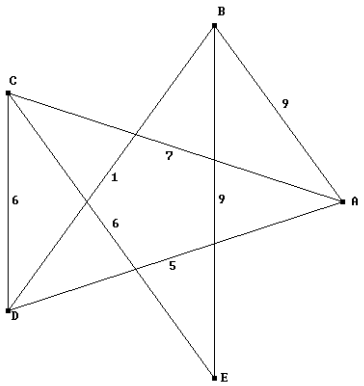
The Problem

This well-known problem involves a travelling salesperson, wishing to visit several locations and returning to his/her starting point, covering the least possible distance. The problem can be reformulated in terms of finding a minimum weight Hamiltonian circuit in a weighted graph.

Definition
In a weighted graph (digraph) each edge (diedge) is given a weight (a value > 0). For the travelling salesperson problem the weight of each edge represents the distance between incident vertices (locations).

Example 4
Find a minimum weight Hamiltonian circuit for the following weighted graph:

G =



A Solution

The weighted adjacency matrix of G is:

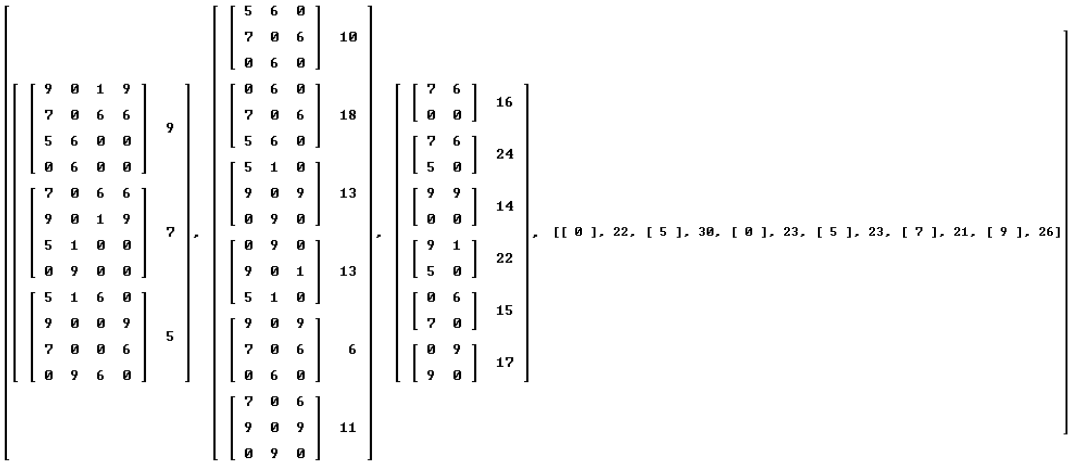
$$\begin{bmatrix} 0 & 9 & 7 & 5 & 0 \\ 9 & 0 & 0 & 1 & 9 \\ 7 & 0 & 0 & 6 & 6 \\ 5 & 1 & 6 & 0 & 0 \\ 0 & 9 & 6 & 0 & 0 \end{bmatrix}$$

HAM.MTH applies a variation of the reduction algorithm to calculate the weights of all of the Hamiltonian circuits of G. As the successive minogs are generated, the weights of the corresponding matrix entries are added together to form the combined weights of the paths. The final step calculates the weights of all of the Hamiltonian circuits. It is then a simple matter to ask DERIVE to find the minimum (or maximum) of this set of values.

To operate, assign the above adjacency matrix to A. Simplifying WROUTES(A) will reveal the iterative process stage-by-stage:

Fourth International Derive TI-89/92 Conference

Fourth International Derive TI-89/92 Conference



In the final step, if a matrix entry is zero, then the result is a Hamiltonian path (not a circuit), otherwise the matrix entry can be added to the sum of the existing path to calculate the total weight of the resulting Hamiltonian circuit.

A vector containing the weights of all the Hamiltonian circuits can be generated by Simplifying WHCS(A). The minimum, maximum weights of all of the Hamiltonian circuits can be found by Simplifying MINWHCS(A), MAXWHCS(A), respectively.

For graph *G* with matrix A we have: WHCS(A) = [35, 28, 28, 35]; MINWHCS(A) = 28; MAXWHCS(A) = 35.

To find an example of a Hamilton circuit of *G* with the minimum weight Author and Simplify: INDEX(WHCS(A),28) - to obtain the value 2. Next, Author and Simplify: HCS(A)↓2.
This will display the Hamiltonian circuit [1, 3, 5, 2, 4, 1]. It is a simple matter to check on the drawing of *G* that the weight of this circuit is indeed 28. (Note: **1, 2, 3, 4, 5** correspond to **A, B, C, D, E**, respectively, in the drawing of *G*.)

Remarks

(i) Of course, this very simple example can easily be done by hand. It has been included at this stage to illustrate the manner in which HAM.MTH works.

(ii) For non-trivial examples of the travelling salesperson problem, students are often asked to use an algorithm for calculating a lower bound for the Hamiltonian weighted circuits. A typical example of this kind would be:

Example 5

A weighted complete bipartite graph $K_{5,5}$ has vertex sets {A, B, C, D, E} and {V, W, X, Y, Z}. The following table gives the corresponding edge weights:

	V	**W**	**X**	**Y**	**Z**
A	28	15	42	46	55
B	32	35	37	32	62
C	12	49	44	53	68
D	19	67	38	18	26
E	40	72	60	21	24

Calculate a lower bound for the weighted Hamiltonian circuits of this graph.

Exact solution to Example 5, using DERIVE
There are 1440 distinct undirected Hamiltonian circuits (2880 directed circuits) for this graph.

Assign the above matrix table to a variable, B say, by authoring:
B := [[28,15,42,46,55],[32,35,37,32,62],[12,49,44,53,68],[19,67,38,18,26],[40,72,60,21,24]]

Next, construct the 5×5 zero matrix by: Z := vector(vector(0,i,1,5),j,1,5).

An adjacency matrix for the bipartite graph can now be define as:
A := APPEND(APPEND(Z,B`)`,APPEND(B,Z)`)

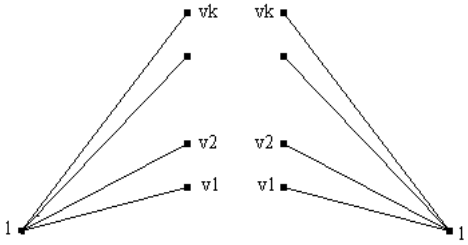
Using this matrix with HAM.MTH yields the following exact results:
minimum weight of Hamiltonian circuits = 270 (index 683 , 22.6 sec.);
Hamiltonian circuit with this weight = [1, 7, 2, 9, 5, 10, 4, 6, 3, 8, 1] (27.8 sec.)
(Note that, for this example**, 1,2,3,4,5,6,7,8,9,10** correspond to **A,B,C,D,E,V,W,X,Y,Z,** respectively.)

Remark
Adding more vertices to this graph would put the size of the calculation beyond the scope of HAM.MTH on a PC.

Refining the Algorithm for Graphs

The Basic Refinement

For a graph it is possible to refine the basic algorithm so that each undirected Hamiltonian circuit is counted only once, instead of twice. The argument can be illustrated using the canopy diagram for the directed Hamiltonian circuits of a graph. From the diagrams above it is clear that the Hamiltonian circuit canopy diagram for a graph with n vertices is symmetric about the line $x = n/2$.
Consider the first and last edges in the Hamiltonian canopy diagram of a graph *G*, where v1, ..., vk is the set of all vertices adjacent to the root vertex 1:



Suppose we have calculated all of the directed Hamiltonian circuits with first edge 1→v1. These will have terminated in a selection of last edges from v2→1, ..., vk→1. When we subsequently calculate all of the directed Hamiltonian circuits with first edge 1→v2 we will have already calculated any (undirected) circuits ending in v1→1. To eliminate these repeated circuits from the calculation we "cross out" the last edge v1→1 when calculating the circuits starting with 1→v2. Similarly, when calculating the circuits starting with 1→v3, we will have already calculated any (undirected) circuits ending in v1→1 and v2→1. Therefore, to eliminate these repeated circuits from the calculation we "cross out" both of the last edges v1→1 and v2→1 when calculating the circuits starting with 1→v3.

Continuing in this manner (up to vk) each undirected Hamiltonian circuit will be counted exactly once by this refinement of the algorithm.
(Note that, for vk itself, all of the last links will have been "crossed out" and so there will be no remaining (undirected) Hamiltonian circuits to calculate at this final stage.)

Applying the Refinement to the Adjacency Matrix of a Graph

Let *A* be the adjacency matrix of a graph *G* with vertices labelled 1, 2, ..., n. Then *A* is a symmetric matrix, and the first and last edges of Hamiltonian circuits (rooted at 1) of *G* will be represented by non-zero entries in the first row and first column of *A*, respectively. Suppose, these non-zero entries are $a_{1,vi}$ (= $a_{vi,1}$), for i = 1, ..., k. Let ZA_0= *A*, ZA_j = matrix *A* with elements $a_{v1,1}$, ..., $a_{vj,1}$ replaced by 0 ($1 \le j < k$) and let MGZ_j be the minog MG_j of $ZA_{(j-1)}$. To calculate the number of undirected Hamiltonian circuits of *G* we can refine the first reduction step of the basic algorithm to:

$$H(A) = \sum_{j=1}^{k-1} a_{1,vj}.H(MGZ_j))$$

Then continue with the basic reduction for digraphs on the resulting minogs.

In HAM.MTH Simplifying H1(A) (where the adjacency matrix of a graph *G* has been assigned to A) will calculate the number of undirected Hamiltonian circuits in *G*.

The following table was constructed using HAM.MTH in combination with NGRAPHS.MTH:

TABLE 3 Counting undirected Hamiltonian circuits

GRAPH	Number of undirected Hamiltionian circuits	Approx Completion Time (seconds)
Wheel Graph W_{20}	19	12.6
Complete Graph K_9	20160	40
Complete Bipartite Graph $K_{5,5}$	1140	8.5
Tetrahedron	3	.09
Cube	6	.6
Octahedron	16	.35
Dodecahedron	30	41
Icosahedron	1280	56.2
Petersen	0	1.26

Remarks

(i) Although the 'size' of the calculation has been reduced, in the sense that the algorithm is only calculating half as many Hamiltonian circuits, there is not a corresponding saving in time.

(ii) Similar refinements for graphs have been incorporated into HAM.MTH to apply to the other algorithms. For example: ROUTES1(A); HCS1(A); WROUTES1(A); WHCS1(A).

(iii) Suggestions for further refinements of the algorithms for graphs can generate discussion points. For example:

- always select a vertex of least degree as the root vertex; label the root vertex 1 and label the adjacent vertices to the root vertex 2, ..., k, then the summation in HAM.MTH can be over j = 2 to k–1;
- extra necessary conditions could be included at each stage of the basic algorithm to filter out minogs that do not lead to Hamiltonian circuits.

Conclusions

The adjacency matrix of a graph is normally the first method that a student encounters for numerically storing the vertex-edge information. The algorithms of this paper make principal use of the adjacency matrix to achieve their results. I feel that this treatment is suitable for introducing the topic to a cohort of mathematics students (as opposed to students following courses with a mainly computing background). Techniques based upon the adjacency matrix of the graph, together with constructions allied to minors and row expansions of matrices, should follow on naturally from a mathematics student's experience.

The algorithmic methods used are particularly suitable for interpretation with DERIVE (or a similar algebraic manipulation package). A nice feature of DERIVE is the application of the ITERATES instruction to illustrate the step-by-step reductions at each stage of the algorithms. With the advent of more powerful algebraic manipulation programs, in future, more uses may be made of adjacency matrices in algorithms involving network graphs.

The powerful 2D-graphics facilities of DERIVE have also played an important role in understanding the topic. The ability to display canopy diagrams in a flexible and informative manner has already lead to an argument in this paper for refining the basic algorithm to calculate undirected Hamiltonian circuits using the symmetric adjacency matrix of a graph.

References

Open University (1995). *MT365 Graphs, Networks and Designs (Graphs 1: Graphs)*, OU Press, Milton Keynes, England.

Wilson, R. J. (1996). *Introduction to Graph Theory*, Fourth Edition, Longman.

Jungnickel, D. (1999). *Graphs, Networks and Algorithms*, English Edition, Springer.

BIOGRAPHICAL NOTE

Peter Schofield is Head of Mathematics at Trinity and All Saints, University of Leeds, where he has taught mathematics for over 25 years. College students follow combined degrees in education with mathematics or mathematics pathways in a single honours management degree course. Over the past nine years Dr. Schofield has made extensive use of DERIVE in working with these students.

Fourth International Derive TI-89/92 Conference

Appendix (Listing of HAM.MTH)

HAM.MTH will run in both DERIVE 4 and DERIVE 5. However, in DERIVE 4 you will need to Simplify expressions in the Algebra Window before 2D-plotting. In DERIVE 5, with the 2D-plot option (On), expressions can be plotted directly into the 2D-plot Window.

```
#1:   HAMILTONIAN CIRCUITS IN DIGRAPHS AND GRAPHS
#2:   Number of Hamiltonian c's and rooted paths from the adjacency matrix of a digraph
#3:   MINO(a, j) := DELETE_ELEMENT(DELETE_ELEMENT(a, 1)`, j)`
#4:   [R1(x, j) := APPEND([x_(j - 1)], x), R2(x, j) := [APPEND(x_1, [x_(2, j - 1)]),
        DELETE_ELEMENT(x_2, j - 1)]]
#5:   [H3(a) := a_(1,2)·a_(2,3)·a_(3,1) + a_(1,3)·a_(3,2)·a_(2,1), HP3(a) := a_(1,2)·a_(2,3)·IF(a_(3,1) = 0, 1, 0) +
        a_(1,3)·a_(3,2)·IF(a_(2,1) = 0, 1, 0)]
      H(a) :=
        If DIMENSION(a) > 3
#6:        Σ(IF(a↓1↓j = 0, 0, a↓1↓j · H(DELETE_ELEMENT(R1(MINO(a, j), j), j))), j, 2,
           H3(a)

        DIMENSION(a))

      HP(a) :=
        If DIMENSION(a) > 3
#7:        Σ(IF(a↓1↓j = 0, 0, a↓1↓j · HP(DELETE_ELEMENT(R1(MINO(a, j), j), j))), j, 2,
           HP3(a)

        DIMENSION(a))

#8:

#9:   Hamiltonian c's and rooted paths from the adjacency matrix of a digraph
#10:  CS(x) := APPEND(VECTOR(IF(x_(1,1,j) = 0, [], [[DELETE_ELEMENT(R1(MINO(x_1, j), j), j),
        R2(x_2, j)]]), j, 2, DIMENSION(x_1)))
#11:  CHILDREN(x) := APPEND(VECTOR(CS(x_i), i, 1, DIMENSION(x)))
#12:  ROUTES(a) := ITERATES(CHILDREN(x), x, CS([a, [[1], VECTOR(i, i, 2, DIMENSION(a))]]),
        DIMENSION(a) - 2)
#13:  ROUTE(a) := ITERATE(CHILDREN(x), x, CS([a, [[1], VECTOR(i, i, 2, DIMENSION(a))]]),
        DIMENSION(a) - 2)
#14:  H_CS(x) := APPEND(VECTOR(IF(x_(3·i - 2,1) ≠ 0, [APPEND(x_(3·i - 1), [1])], []), i, 1,
        DIMENSION(x)/3))
#15:  H_PS(x) := APPEND(VECTOR(IF(x_(3·i - 2,1) = 0, [x_(3·i - 1)], []), i, 1, DIMENSION(x)/3))
#16:  [HCS(a) := H_CS(ROUTE(a)), HPS(a) := H_PS(ROUTE(a))]
#17:
```

Fourth International Derive TI-89/92 Conference

#18: Constructing canopy diagrams

#19: CANOPY(x) := VECTOR(VECTOR($[i - 1, x_{j,i}]$, i, 1, DIMENSION(x_j)), j, DIMENSION(x))

#20: [HCAN(a) := CANOPY(H_CS(ROUTE(a))), HPCAN(a) := CANOPY(H_PS(ROUTE(a)))]

#21:

#22: Weights of Hamiltonian c's from the weighted adjacency matrix of a digraph

#23: WCS(x) := APPEND(VECTOR(IF($x_{1,1,j}$ = 0, [], [[DELETE_ELEMENT(R1(MINO(x_1, j), j), j), x_2 + $x_{1,1,j}$]]), j, 2, DIMENSION(x_1)))

#24: WCHILDREN(x) := APPEND(VECTOR(WCS(x_i), i, 1, DIMENSION(x)))

#25: WROUTES(a) := ITERATES(WCHILDREN(x), x, WCS([a, 0]), DIMENSION(a) - 2)

#26: WH_CS(x) := APPEND(VECTOR(IF($x_{2 \cdot i - 1}$ = 0, [], [$x_{2 \cdot i}$ + $x_{2 \cdot i - 1,1,1}$]), i, 1, $\frac{\text{DIMENSION}(x)}{2}$))

#27: WHCS(a) := WH_CS(ITERATE(WCHILDREN(x), x, WCS([a, 0]), DIMENSION(a) - 2))

#28: INDEX(v, n) := ITERATE(IF(n = v_i, i, i + 1), i, 1)

#29:

#30: Number of undirected Hamiltonian c's from the adjacency matrix of a graph

#31: ZZ(a, j) := VECTOR(IF(k ≤ j, REPLACE_ELEMENT(0, a_k, 1), a_k), k, 1, DIMENSION(a))

#32: H1(a) := $\sum_{j=2}^{\text{DIMENSION}(a)-1}$ IF($a_{1,j}$ = 0, 0, $a_{1,j}$·H(DELETE_ELEMENT(R1(MINO(ZZ(a, j), j), j), j)))

#33:

#34: Hamiltonian c's from the adjacency matrix of a graph

#35: CS1(x) := APPEND(VECTOR(IF($x_{1,1,j}$ = 0, [], [[DELETE_ELEMENT(R1(MINO(ZZ(x_1, j), j), j), j), R2(x_2, j)]]), j, 2, DIMENSION(x_1) - 1))

#36: ROUTES1(a) := ITERATES(CHILDREN(x), x, CS1([a, [[1], VECTOR(i, i, 2, DIMENSION(a))]]), DIMENSION(a) - 2)

#37: HCS1(a) := H_CS(ITERATE(CHILDREN(x), x, CS1([a, [[1], VECTOR(i, i, 2, DIMENSION(a))]]), DIMENSION(a) - 2))

#38:

#39: Weights of Hamiltonian c's from a weighted adjacency matrix of a graph

#40: WCS1(x) := APPEND(VECTOR(IF($x_{1,1,j}$ = 0, [], [[DELETE_ELEMENT(R1(MINO(ZZ(x_1, j), j), j), j), x_2 + $x_{1,1,j}$]]), j, 2, DIMENSION(x_1)))

#41: WROUTES1(a) := ITERATES(WCHILDREN(x), x, WCS1([a, 0]), DIMENSION(a) - 2)

#42: WHCS1(a) := WH_CS(ITERATE(WCHILDREN(x), x, WCS1([a, 0]), DIMENSION(a) - 2))