GRAPHER

AN INTERACTIVE ENVIRONMENT FOR THE STUDY OF GRAPH THEORY

Stephen Hoover

Stuart C. Shapiro

Computer Science Department

Indiana University

Bloomington, Indiana 47401

TECHNICAL REPORT NO. 28

GRAPHER
AN INTERACTIVE ENVIRONMENT
FOR THE STUDY OF GRAPH THEORY

STEPHEN HOOVER

STUART C. SHAPIRO

MAY, 1975

GRAPHER

An Interactive Environment
For the Study of Graph Theory

Stephen Hoover
Stuart C. Shapiro

Computer Science Department
Indiana University
Bloomington, Indiana 47401

I. PURPOSE

GRAPHER is a lesson in basic graph theory designed for use in conjunction with data structures lessons on the Plato IV system [1]. It provides an interactive environment in which a student can, by entering sentences of the GRAPHER language, create and effect changes in a graph structure displayed on his screen. Different areas within the lesson allow the student to examine the abilities and limitations of the GRAPHER language on his own or to apply the language toward the solution of specified problems, the solution being monitored by the lesson.

II. ENVIRONMENT

GRAPHER uses a graph representation commonly found in texts. Nodes are represented by circles, and arcs by lines. Each node is labeled with a unique letter of the Roman alphabet which is used by the student in referencing that node with the GRAPHER language. Arcs are labeled with a unique number to be used in referencing them. In parts of the lesson where it is desirable to differentiate

arcs defined by the student and arcs defined by the lesson, the arcs given by the lesson are drawn as thin lines, student-drawn arcs as thick lines.

III. THE GRAPHER LANGUAGE

The following are sentences of the language to be used by a student in this lesson. "L" is any Roman letter designating a node included within the student's graph, "n" is any integer serving as an arc designator, and "k" is any unsigned integer between 0 and 9.

1. avail ⇐ L
2. link ⇐ L ⇐ L
3. move ⇐ L
4. data(L) ⇐ k
5. return ⇐ n
6. pressing -NEXT- key (puts a new node on the screen)
7. pressing -DATA- key (erase and redraw screen)
8. pressing -LAB- key (remove all work done, reinitialize the graph)

Other commands, allowing branching within the lesson or graph manipulations unique to a single area of the lesson are available at several places within the lesson. Within the area for monitored tasks only a subset of the language is allowed. At any time, however, the commands available are displayed in their proper syntax on the screen. Notice that the GRAPHER language is similar to the

NODER [1, p. 13-15] language, allowing some carry-over between the two lessons.

Inappropriate responses by the student are not allowed. If a student types a command using proper syntax, but refers to items

not occuring within his graph, an appropriate error message appears. As the parser for the language utilizes the "pause" command rather than "arrow", syntactically incorrect commands are impossible to enter. The lesson will, at any time, allow only keypresses which are valid at that point. All others fail to appear on the screen. This has the effect of funnelling a student unsure of what he should type toward entering syntactically correct responses, of which only appropriate responses affect the graph.

IV. THE WORK AREAS

From the title page (shown in Figure 1) the student may enter either the undirected work area or the monitored task area. At any point within the lesson the student may return to the title page and enter either area. In either area the student may work as long as he wishes.

The undirected work area consists of several pages of text explaining the command language, and the work area. The student may wish to read the text a page at a time or, if during his work he is unsure of a command or its effect upon his graph, he may access each page separately as a help sequence without affecting any work he has done. Figure 2 shows the undirected work area after some items have been added. The labelling scheme for arcs and nodes may be seen, and a reflexive arc appears. At the bottom of the page the available language appears, as do all the branching options. Student responses are entered at the arrow. This subset of the language yields the ability to create and erase nodes and arcs, associate data with nodes, move nodes (and associated arcs) about on the screen, replot the screen if it becomes "dirty" or partially

erased, and reinitialize the existing graph to start over.

The monitored task area allows limited use of the GRAPHER language to create several types of graph structures. This section of the lesson consists of four work areas. Each area consists of the following:

i. Resource material which presents the terms and concepts the student will encounter in that area. Graph structures he will be asked to construct are defined.

ii. A "problem" given to the student in the form of a partially-complete graph and the subset of the GRAPHER language he may use to add to the structure of information contained in his graph.

iii. Grading routines to determine the correctness of the student's work. At any time during his manipulation of the graph he may request that his work be graded. If he is in an area equipped to present more than one graph structure, he may choose from several routines. The routines inform the student that his graph is correct if it is of the type he has indicated and is free of errors. If his work is incomplete or contains an error an appropriate error message appears. After grading, the student may either enter another area or continue work on the same graph. A student may have his work graded several times before it is correct and, once correct, he may continue work in the same area to explore the variety of graphs of the indicated type.

iv. Diagnostics are available in many grading routines in the event a graph has been found incorrect. After grading, a student may request help and receive either the statement

of a rule which the routine has found to have been improperly applied, or specific information as to the location on the graph and type of his error.

Each work area may be occupied until the student is satisfied with his understanding of the material. Before beginning monitored tasks the student whould be familiar with the GRAPHER language presented in the undirected work area.

V. RELATIONS, FUNCTIONS, AND 1:1 FUNCTIONS

Figure 3 shows the first monitored task work area after several arcs have been added. Upon entry to this area the student is presented with ten nodes divided into a domain and a range. Using "link" and "return" (commands which create and remove arcs respectively) the student may define relations between the two sets. Three grading routines are available to check the graph for the occurrence of specific types of relations between the sets.

The student may wish to have his graph graded as a relation between the two sets which maps the domain onto the range. In this case the existence of any arc connecting two nodes of the same set constitutes an error.

A student may also request that his graph be graded as a function between the sets. All graphs of this type must also fulfill the requirements of a relation, as stated above. A request for function grading causes the relation test to first be applied to the graph. If this routine is successful the function test is then applied. A function need not be complete in order to be graded correct.

A third routine provides testing for the existence of a 1:1 function. A request for this routine causes the graph to be tested for the occurrence of a proper relation and a proper function before the conditions for the existence of a 1:1 function are examined. Thus, grading a graph for the existence of a 1:1 function may fail at each level, causing the appropriate error message to appear.

VI. TREES AND CYCLES

Figure 4 shows the work area for trees and cycles. Several arcs have been added. From a graph containing nine nodes and no arcs, the student may add and remove arcs to construct a tree or arbitrary structures which involve cycles.

A graph is graded correct by the tree-grading routine if it contains arcs which connect all nodes by some path without creating a cycle. The detection of any cycle, or node(s) not connected to an origin (chosen arbitrarily by the routine) by some path results in the display of a message announcing the type of error found.

When a student issues a request to have his graph examined for the occurrence of a cycle he must specify a node within the path of the (a) cycle. If a cyclic path is found whose path includes the designated node, the student is informed that he is correct and the cycle found by the routine is traced on the screen. If no such cycle is found an error is indicated.

Cycles and trees are mutually exclusive structures. The grading routines of this area may be used to compliment each other. If a graph fails as a tree because the presence of a cycle has been detected, the cycle routine may be used to further verify its existence.

VII. SPANNING TREES, CYCLES AND PATHS

A work area involving spanning trees, cycles and paths is available. The student is given a graph containing nine nodes and twelve arcs which have been defined by the lesson and cannot be removed. The student is to designate a set of these arcs, defining a subset of the given graph. The subset he designates may be tested by routines discussed previously for the occurrence of a spanning tree of the given graph, or a cycle within the graph.

An additional routine is provided to find the occurrence of a set of arcs which have been designated by the student and constitute a path between two nodes. In this area the given graph may not be altered, and the student may deal only with subsets of the graph.

VIII. TRANSITIVE CLOSURE

Figure 5 shows the work area on transitive closure upon entry. The graph presented to the student consists of ten nodes and a number of randomly-generated directed arcs. The student may not remove these arcs which appear as thin, single lines in the figure. The student is to add arcs to this graph (he may erase his own arcs)

which close the given graph under transitivity. In this area only, arcs are directed.

In Figure 6, several arcs have been added by the student. These arcs are distinguished from the problem set of arcs in that they are thick, multiple lines. To be graded correct, a graph must be closed under transitivity using the smallest closure set.

IX. IMPLEMENTATION

The graph diagram which is the object of the student's manipulation is a schematic representation of an internal data structure. A node is represented internally by a set of array elements which contain the x- and y-screen coordinates of the node, data associated with the node by the GRAPHER "data" command, and the number of arcs incident with the node. An arc's internal description consists of array elements which contain the status of the arc (student or lesson defined) and identifiers of the nodes terminating the arc. Arrays which contain node descriptive elements are indexed using the ordinal value of the letters identifying the node. Arc descriptive arrays are indexed using the integers identifying the arcs on the screen display.

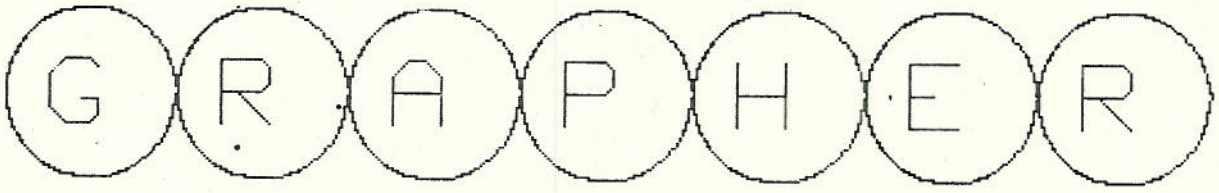
REFERENCES

1. Shapiro, Stuart C. PLATO lessons for a data structures course. Technical Report No. 15, Computer Science Department, Indiana University (August, 1974).

Typed by Christopher Charles

Welcome

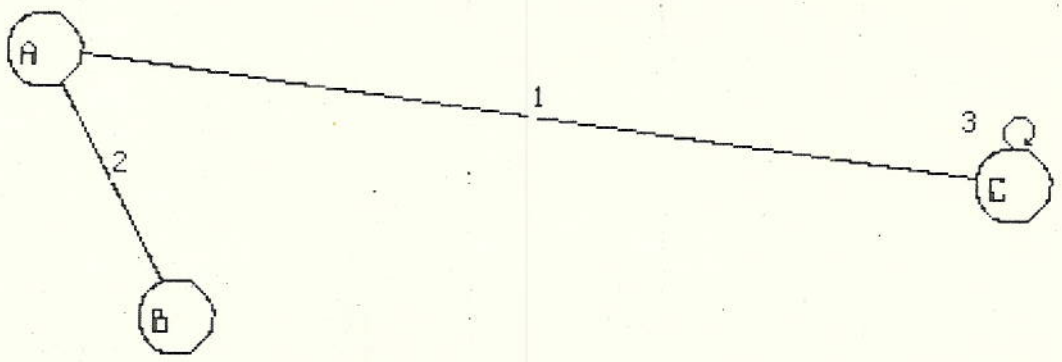
to the many worlds of...



press [NEXT] to make graphs
press [DATA] for instructions
[LAB] for monitored tasks

comment, etc. to hoover of iucs

Figure 1: Title Page

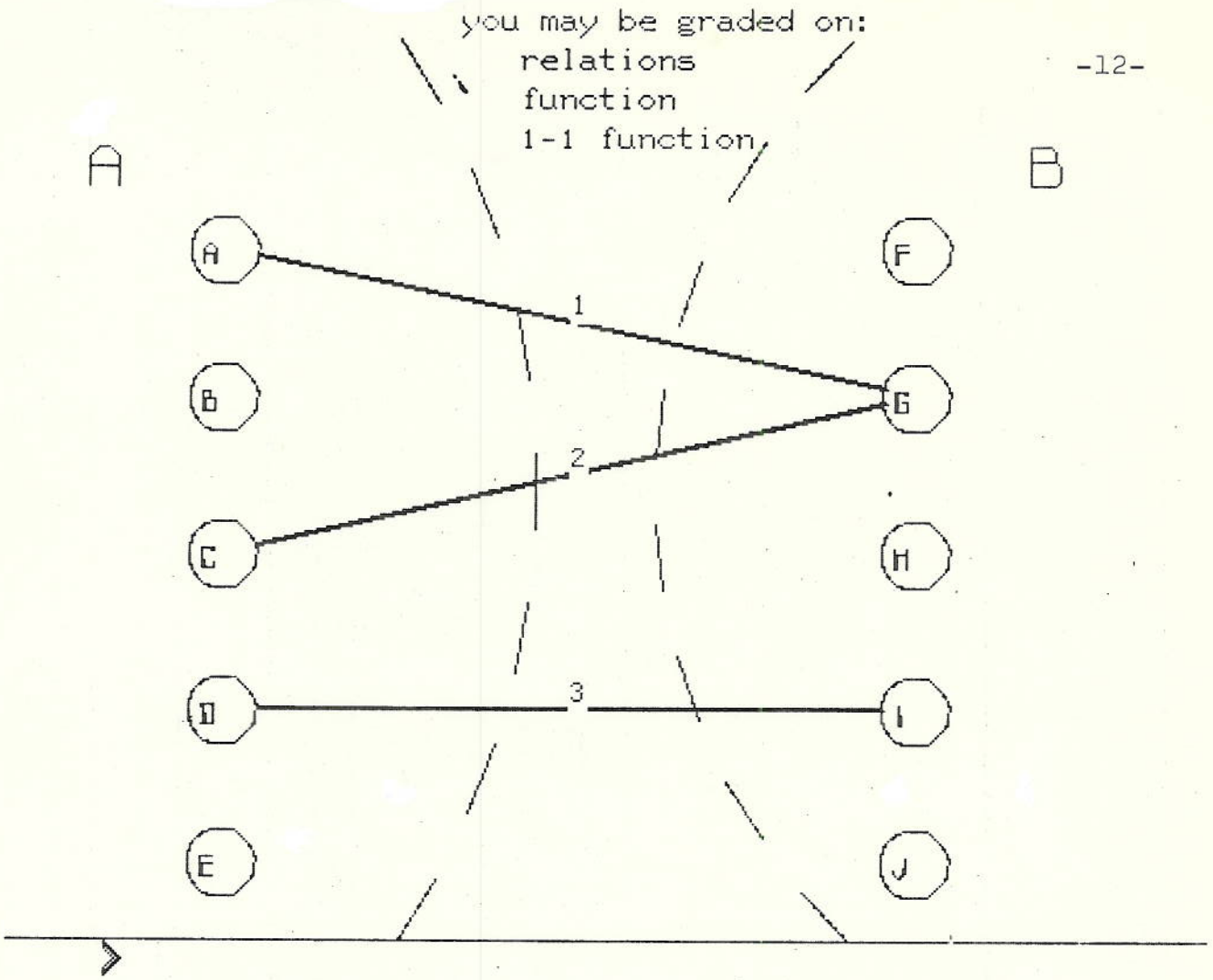


>

press (NEXT) to obtain
a node from AVAIL
avail⇐MODE letter
move⇐MODE letter
returns

link⇐M1⇐M2
(LAB) will let you start over.
data(MODE)⇐n
press (DATA) to replot screen
(HELP) for aid

Figure 2: Undirected Work Area



link and return are allowed
press [LAB] when you are done
for grading
[DATA] to redraw
[CLEAR] for choices

Figure 3: Functions and Relations Work Area

With 3 Added Arcs

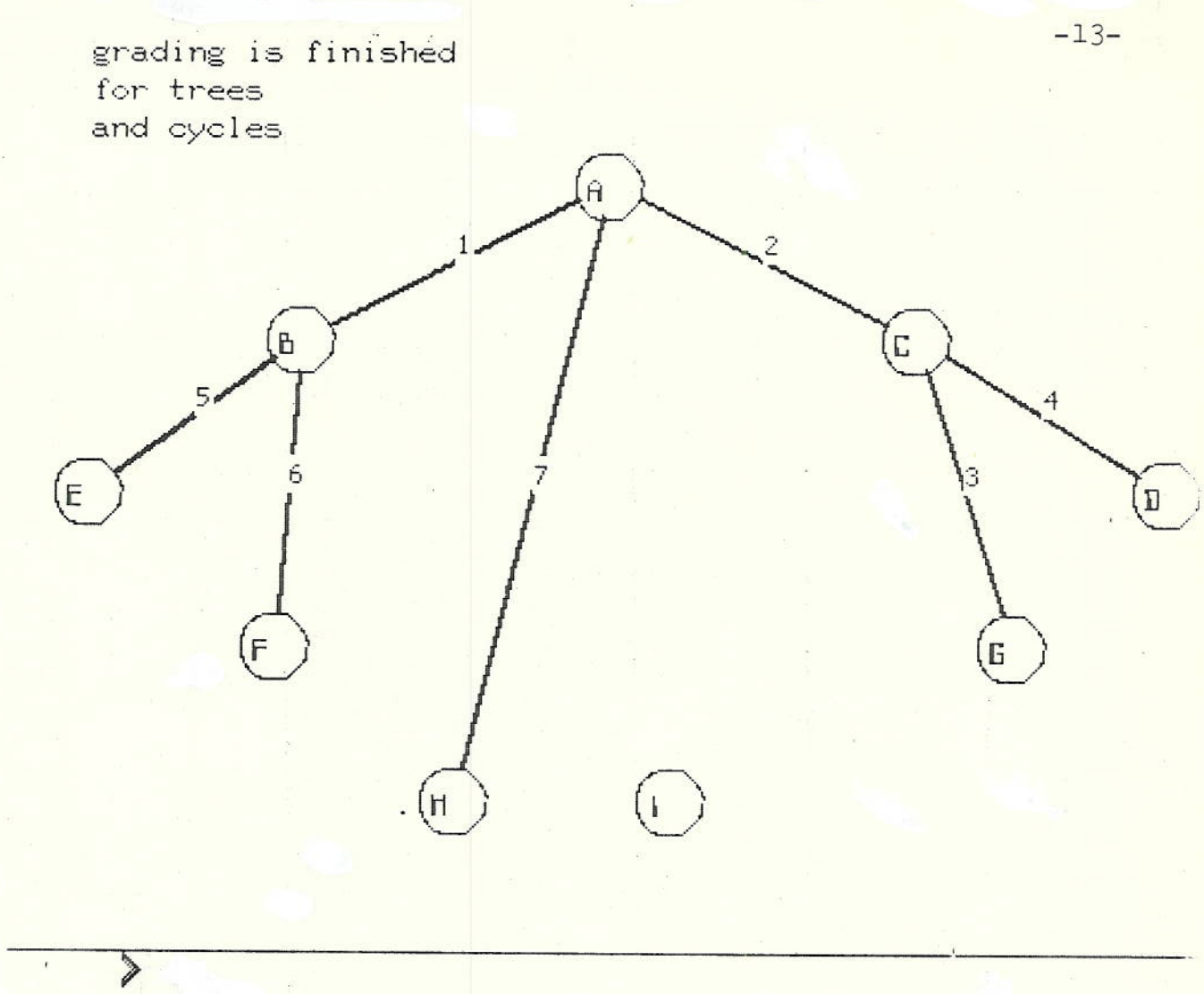
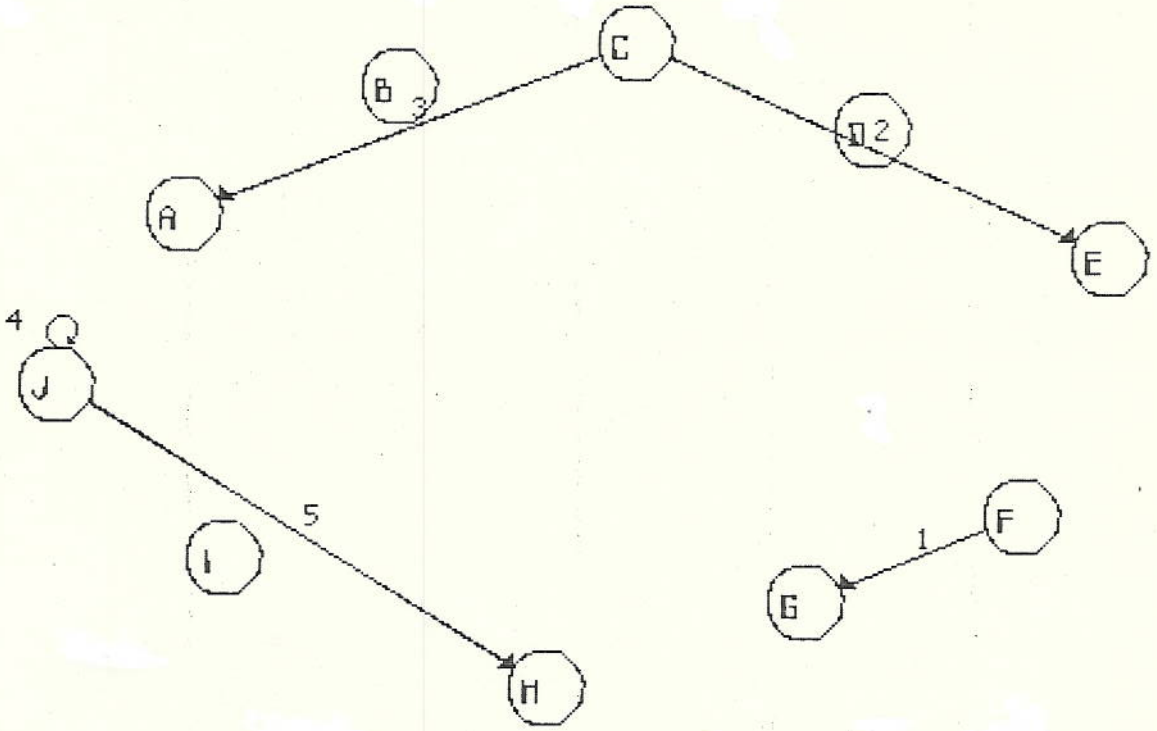


Figure 4: Tree and Cycle Work Area Showing Nearly-Completed Tree

give the transitive closure
of this relation.

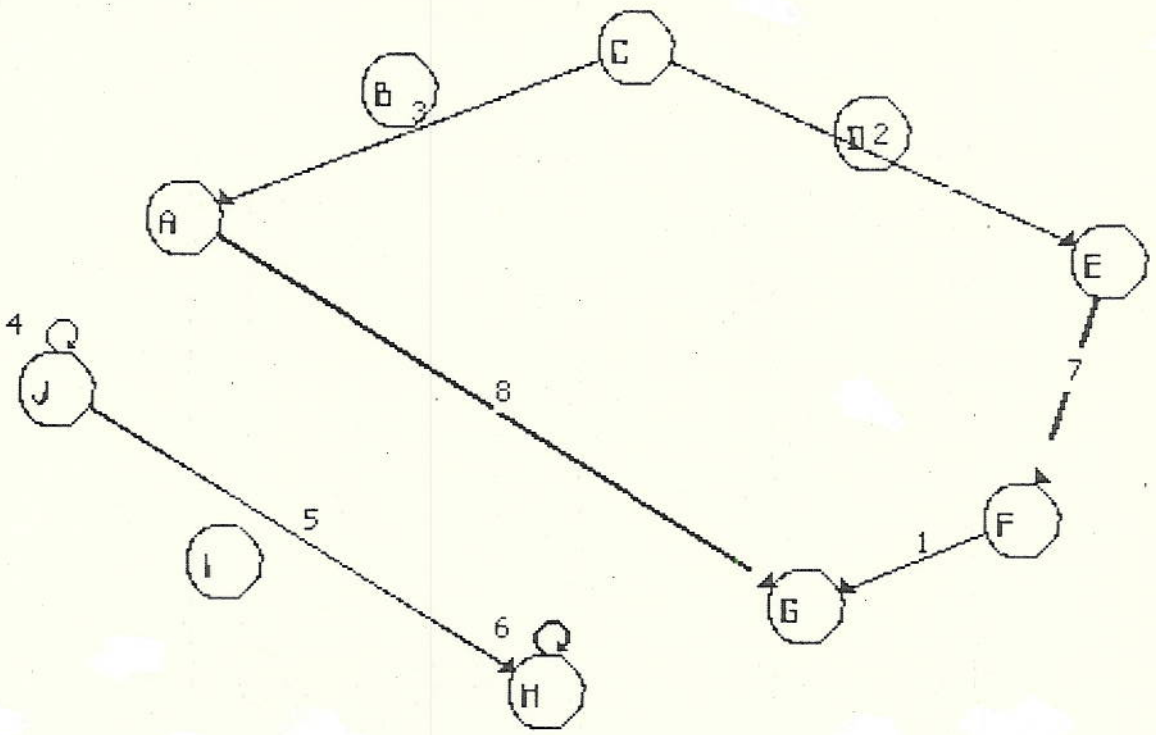


>

link is available
return works for
your arcs only!
LAB for grading

Figure 5: Transitive Closure Work Area Showing Sample Problem

give the transitive closure
of this relation.



>

link is available
return works for
your arcs only!
LAB for grading
[illegible]

Figure 6: Transitive Closure Work Area With 3 Arcs Added by the Student