

UNIT-5

GRAPH MATRICES AND APPLICATIONS

Problem with Pictorial Graphs

- Graphs were introduced as an abstraction of software structure.
- Whenever a graph is used as a model, sooner or later we trace paths through it- to find a set of covering paths, a set of values that will sensitize paths, the logic function that controls the flow, the processing time of the routine, the equations that define the domain, or whether a state is reachable or not.
- Path is not easy, and it's subject to error. You can miss a link here and there or cover some links twice.
- One solution to this problem is to represent the graph as a matrix and to use matrix operations equivalent to path tracing. These methods are more methodical and mechanical and don't depend on your ability to see a path they are more reliable.

Tool Building

- ☐ If you build test tools or want to know how they work, sooner or later you will be implementing or investigating analysis routines based on these methods.
- ☐ It is hard to build algorithms over visual graphs so the properties of graph matrices are fundamental to tool building.

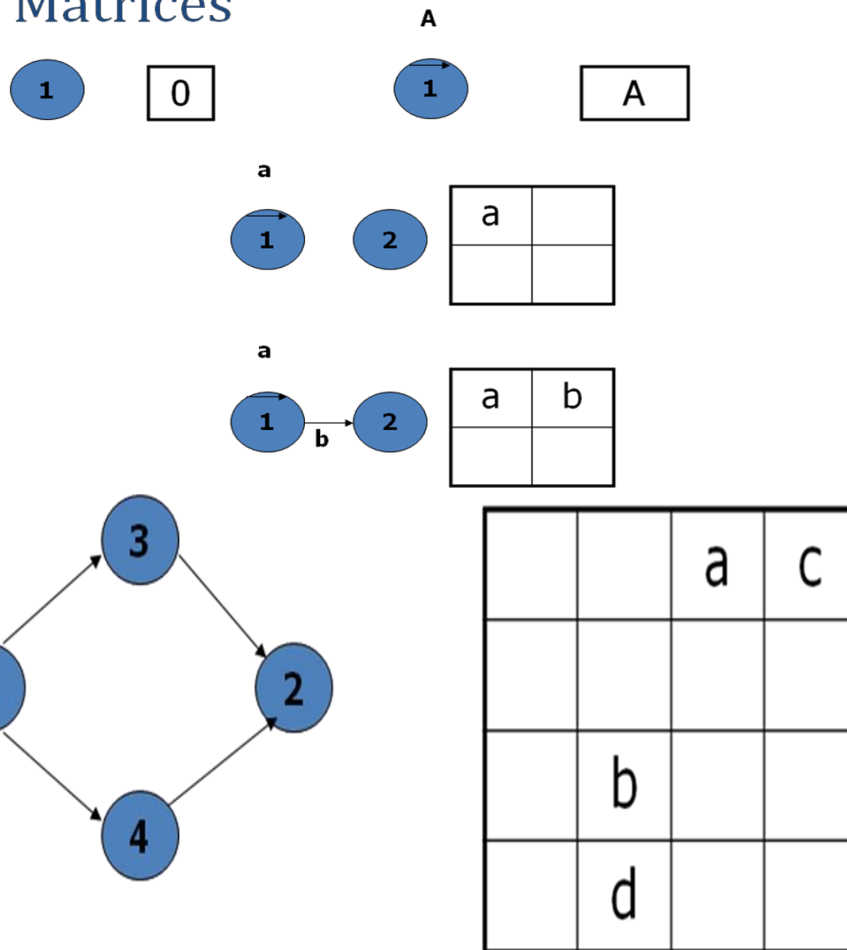
The Basic Algorithms

- ☐ The basic tool kit consists of:
- ☐ Matrix multiplication, which is used to get the path expression from every node to every other node.
- ☐ A partitioning algorithm for converting graphs with loops into loop free graphs or equivalence classes.
- ☐ A collapsing process which gets the path expression from any node to any other node.

The Matrix of a Graph

- A graph matrix is a square array with one row and one column for every node in the graph.
- Each row-column combination corresponds to a relation between the node corresponding to the row and the node corresponding to the column.
- The relation for example, could be as simple as the link name, if there is a link between the nodes.
- Some of the things to be observed:
- The size of the matrix equals the number of nodes.
- There is a place to put every possible direct connection or link between any and any other node.
- The entry at a row and column intersection is the link weight of the link that connects the two nodes in that direction.
- ☐ A connection from node i to j does not imply a connection from node j to node i.
- If there are several links between two nodes, then the entry is a sum; the “+” sign denotes parallel links as usual.

Some Graphs and their Matrices



A simple weight

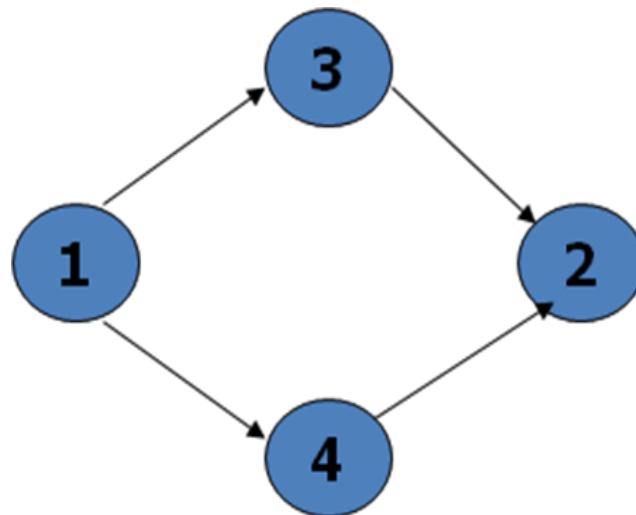
- ✓ A simplest weight we can use is to note that there is or isn't a connection. Let "1" mean that there is a connection and "0" mean that there isn't.
- ✓ The arithmetic rules are:
 - ✓ $1+1=1$ $1*1=1$
 - ✓ $1+0=1$ $1*0=0$
 - ✓ $0+0=0$ $0*0=0$
- ✓ A matrix defined like this is called connection matrix.

Connection matrix

- ✓ The connection matrix is obtained by replacing each entry with 1 if there is a link and 0 if there isn't.
- ✓ As usual we don't write down 0 entries to reduce the clutter.

		a	c
	b		
	d		

		1	1
	1		
	1		



Connection Matrix-continued

- ☐ Each row of a matrix denotes the out links of the node corresponding to that row.
- ☐ Each column denotes the in links corresponding to that node.
- ☐ A branch is a node with more than one nonzero entry in its row.
- ☐ A junction is node with more than one nonzero entry in its column.
- ☐ A self loop is an entry along the diagonal.

Cyclomatic Complexity

- The cyclomatic complexity obtained by subtracting 1 from the total number of entries in each row and ignoring rows with no entries, we obtain the equivalent number of decisions for each row. Adding these values and then adding 1 to the sum yields the graph's cyclomatic complexity.

		1	1	$2-1=1$
	1			$1-1=0$
	1			$1-1=0$

$1+1=2$ (cyclomatic complexity)

Relations

- ☐ A relation is a property that exists between two objects of interest.
- ☐ For example,
 - “Node a is connected to node b” or aRb where “R” means “is connected to”.
 - “ $a \geq b$ ” or aRb where “R” means greater than or equal”.
- ☐ A graph consists of set of abstract objects called nodes and a relation R between the nodes.
- ☐ If aRb , which is to say that a has the relation R to b, it is denoted by a link from a to b.
- ☐ For some relations we can associate properties called as link weights.

Transitive Relations

- ☐ A relation is transitive if aRb and bRc implies aRc .
- ☐ Most relations used in testing are transitive.
- ☐ Examples of transitive relations include: is connected to, is greater than or equal to, is less than or equal to, is a relative of, is faster than, is slower than, takes more time than, is a subset of, includes, shadows, is the boss of.
- ☐ Examples of intransitive relations include: is acquainted with, is a friend of, is a neighbor of, is lied to, has a du chain between.

Reflexive Relations

- ☐ A relation R is reflexive if, for every a, aRa .
- ☐ A reflexive relation is equivalent to a self loop at every node.
- ☐ Examples of reflexive relations include: equals, is acquainted with, is a relative of.
- ☐ Examples of irreflexive relations include: not equals, is a friend of, is on top of, is under.

Symmetric Relations

- ☐ A relation R is symmetric if for every a and b, aRb implies bRa .
- ☐ A symmetric relation mean that if there is a link from a to b then there is also a link from b to a.
- ☐ A graph whose relations are not symmetric are called directedgraph.

- ☐ A graph over a symmetric relation is called an undirected graph.
- ☐ The matrix of an undirected graph is symmetric ($a_{ij}=a_{ji}$) for all i,j

Antisymmetric Relations

- ☐ A relation R is antisymmetric if for every a and b , if aRb and bRa , then $a=b$, or they are the same elements.
- ☐ Examples of antisymmetric relations: is greater than or equal to, is a subset of, time.
- ☐ Examples of nonantisymmetric relations: is connected to, can be reached from, is greater than, is a relative of, is a friend of

equivalence Relations

- ☐ An equivalence relation is a relation that satisfies the reflexive, transitive, and symmetric properties.
- ☐ Equality is the most familiar example of an equivalence relation.
- ☐ If a set of objects satisfy an equivalence relation, we say that they form an equivalence class over that relation.
- ☐ The importance of equivalence classes and relations is that any member of the equivalence class is, with respect to the relation, equivalent to any other member of that class.
- ☐ The idea behind partition testing strategies such as domain testing and path testing, is that we can partition the input space into equivalence classes.
- ☐ Testing any member of the equivalence class is as effective as testing them all.

Partial Ordering Relations

- ☐ A partial ordering relation satisfies the reflexive, transitive, and antisymmetric properties.
- ☐ Partial ordered graphs have several important properties: they are loop free, there is at least one maximum element, and there is at least one minimum element.

The Powers of a Matrix

- Each entry in the graph's matrix expresses a relation between the pair of nodes that corresponds to that entry.
- ☐ Squaring the matrix yields a new matrix that expresses the relation between each pair of nodes via one intermediate node under the assumption that the relation is transitive.
- ☐ The square of the matrix represents all path segments twolinks long.
- ☐ The third power represents all path segments three links long.

Matrix Powers and Products

- Given a matrix whose entries are a_{ij} , the square of that matrix is obtained by replacing every entry with
 - n
 - $a_{ij} = \sum_{k=1}^n a_{ik} a_{kj}$
 - $k=1$
- more generally, given two matrices A and B with entries a_{ik} and b_{kj} , respectively, their product is a new matrix C, whose entries are c_{ij} , where:
 - n
 - $C_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$
 - $k=1$

Partitioning Algorithm

- Consider any graph over a transitive relation. The graph may have loops.
- We would like to partition the graph by grouping nodes in such a way that every loop is contained within one group or another.
- Such a graph is partially ordered.
- There are many used for an algorithm that does that:
- We might want to embed the loops within a subroutine so as to have a resulting graph which is loop free at the top level.
- Many graphs with loops are easy to analyze if you know where to break the loops.
- While you and I can recognize loops, it's much harder to program a tool to do it unless you have a solid algorithm on which to base the tool.

Node Reduction Algorithm (General)

- The matrix powers usually tell us more than we want to know about most graphs.
 - In the context of testing, we usually interested in establishing a relation between two nodes- typically the entry and exit nodes.
 - In a debugging context it is unlikely that we would want to know the path expression between every node and every other node.
 - The advantage of matrix reduction method is that it is more methodical than the graphical method called as node by node removal algorithm.
1. Select a node for removal; replace the node by equivalent links that bypass that node and add those links to the links they parallel.
 2. Combine the parallel terms and simplify as you can.
 3. Observe loop terms and adjust the out links of every node that had a self loop to account for the effect of the loop.
 4. The result is a matrix whose size has been reduced by 1. Continue until only the two nodes of interest exist.