5. Graph Matrices & Applications

→ Graphs were abstraction of software structure. whenever a graph is used as a model, we trace paths through it to find a set of covering paths, a set of values used for the logic function that controls the flow, processing time of the routine, the equations that define the domain (or) whether a state is reachable (or) not.

→ path is not easy, you can miss a link here & there (or) cover some links twice.

→ one solution to this problem is to represent(a) graph as a matrix & to use matrix operations equivalent to path tracing.

→ The basic algorithm 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 loop free graphs (or) equivalent classes

→ A collapsing process which gets the path expression from any node to any other node.

Matrix of a Graph

→ A graph matrix is a square array with one row & one column for every node in the graph.

→ Each row-column combination corresponds to a relation between the node corresponding to the row & 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.

→ The size of the matrix equals the number of nodes.

→ There is a place to put every possible direct connection or link between any & any other node.

→ The entry at a row & column intersection is the link weight of the link that connects two nodes in that direction.

→ A connection from node i to j does not ~~do~~ imply a connection from node j to node i.

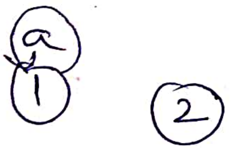
→ if there are several links between two nodes, then the entry is sum, the '+' sign denotes parallel links.

Some graphs & their matrices

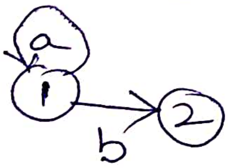
0

5

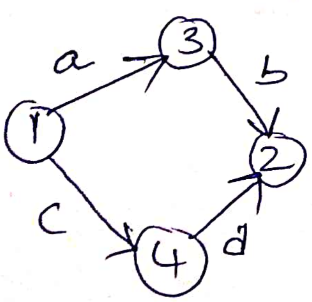
A



a	



a	b



	1	2	3	4
1			a	c
2				
3		b		
4		d		

Simple weight

→ A simplest weight we can use is to note that there is a
isn't a connection.

Let '1' means there is a connection & '0' means no connectio

→ The arithmetic rules are

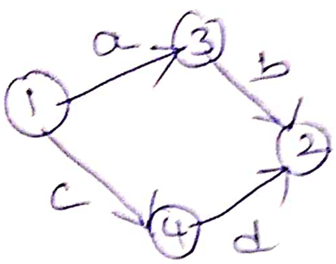
$1+1=1 \qquad 1*1=1$

$1+0=1 \qquad 1*0=0$

$0+0=0 \qquad 0*0=0$

→ A matrix is a connection matrix

→ Connection matrix is obtained by replacing each entry with '1' if there is a link and '0' if there is no link.



	1	2	3	4
1			a	c
2				
3		b		
4		d		

		1	1
	1		
	1		

→ 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 non zero entry in its row

→ A junction is node with more than one non zero entry in its column

→ A self loop is an entry along the diagonal.

Cyclomatic complexity

Is obtained by subtracting 1 from the total number of entries in each row & ignoring rows with no entries, we obtain equivalent number of decisions for each row. Adding these values & 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: Is a property that exits between 2 objects (or) aRb where

Ex:- Node 'a' connected to node 'b'

R means "is connected to".

a >= b or aRb where R means greater than or equal

→ A graph consists of set of objects called nodes & a relation R between the nodes.

→ if aRb means 'a' has the relation R to 'b', it is denoted by a link from a to b.

1) Transitive Relation
2) Reflexive Relation
3) Symmetric Relation
4) Antisymmetric Relation
5) Equivalence Relation
6) partial ordering Relation

Transitive Relation: A relation is transitive if aRb and bRc implies aRc

→ most relations used in testing are transitive.

→ Ex.. of transitive relation 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,

→ Ex.. of intransitive relations include; is a friend of, is a neighbor of, is tied to, has a chain between.

Reflexive Relations

A relation R is reflexive, if for every a, aRa

→ reflexive relation is equivalent to a self loop at every node.

Ex.. equals, is a relative of

Ex.. of irreflexive relation include: not equals, is under, is a friend of,

Symmetric Relation

→ A relation R is symmetric if for every a and b, aRb implies bRa.

→ 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 directed graph. A graph over a symmetric relation is called 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.

→ Ex.. is greater than or equal to, is a subset of.

Ex.. of nonantisymmetric relations : is connected to, can be reached from, is greater than, is a friend of, is a relative of.

Equivalence Relations : Is a relation that

→ satisfies the reflexive, transitive & symmetric properties. equality is the most familiar example of an equivalence relation

→ If 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 equivalence class with respect to the relation equivalent to any other member of that class.

Partial ordering Relations : satisfies reflexive, transitive, and antisymmetric properties

→ these partial ordering graphs have several important properties, they are loop free, there is atleast one maximum element & there is atleast 1 minimum element.

Power of a matrix: Each entry in the graph's matrix expresses a relation between the pair of nodes that corresponds to that entry

→ graph 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 two links long, the third power represents all path segments 3 links long.

→ Given a matrix whose entries are a_{ij}, the square of that matrix is obtained by replacing every entry with 1) n

2) $a_{ij} = \sum a_{ik} a_{kj}$

3) $k=1$

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} 1) n 2) $k=1$

$$c_{ij} = \sum a_{ik} b_{kj}$$

Partitioning Algorithm:

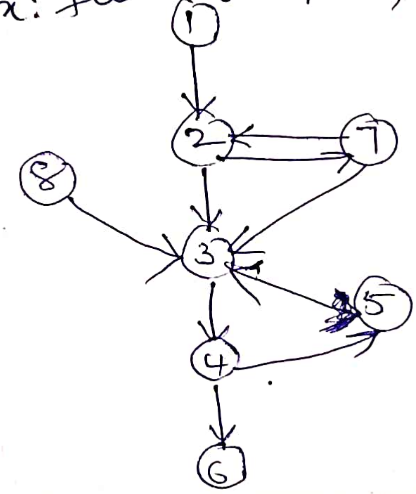
→ consider any graph over a transitive relation. The graph may have loops. We have to partition the graph by grouping nodes in such a way that every loop is contained within one group (or) another. Such graph is partially ordered.

step ... from node 3 to node 6 ...

→ There are many used for an algorithm includes 5-8 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.

Ex: flow graphs



Graph matrix ↓

Relation Matrix

1	1						
	1	1				1	
		1	1				
			1	1	1		
		1		1			
					1		
	1	1				1	
		1					1

Diagonal entries are made to represent self loop.. if diagonal values are all '1' means it is transitive.

→ The transitive closure matrix $(A+I)^n$ can be obtained by using the following steps

Step 1: Mark all diagonal values by 1

Step 2: The flow from node 1 to node 6 is

1-2-7-2-3-4-5-3-4-6

So mark nodes 1,2,3,4,5,6,7 by 1 in the first row

Step 3: The flow from node 2 to node 6 is

2-7-2-3-4-5-3-4-6

So mark nodes 2,3,4,5,6,7 by 1 in the second row.

Step 4: The flow from node 3 to node 6 is 3-4-5-3-4-6. So mark nodes 3,4,5,6 by 1 in the ~~fourth~~ third row.

Step 5: The flow from node 4 to node 6 is 4-5-3-4-6. So mark nodes 3,4,5,6 by 1 in fourth row.

Step 6: The flow from node 5 to node 6 is 5-3-4-6. So mark nodes 3,4,5,6 by 1 in the fifth row.

Step 7: The flow from node 6 is only 6. So mark node 6 by 1 in the sixth row.

Step 8: The flow from node 7 to node 6 is 7-2-3-4-5-3-4-6. So mark nodes 2,3,4,5,6,7 by 1 in seventh row.

Step 9: The flow from node 8 to node 6 is 8-3-4-5-3-4-6. So mark nodes 3,4,5,6,8 by 1 in the 8th row.

Therefore the transitive closure matrix $(A+I)^n$ is

	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	
2		1	1	1	1	1	1	
3			1	1	1	1		
4			1	1	1	1		
5			1	1	1	1		
6						1		
7		1	1	1	1	1	1	
8			1	1	1	1		1

The transpose of $(A+I)^{nT}$

1							
1	1					1	
1	1	1	1	1		1	1
1	1	1	1	1		1	1
1	1	1	1	1		1	1
1	1	1	1	1	1	1	1
1	1					1	
							1

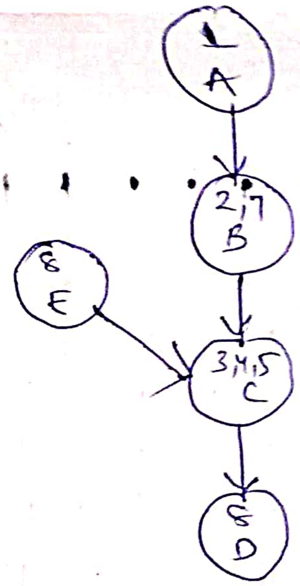
→ The intersection of transitive closure matrix $(A+I)^n$ & transpose matrix $(A+I)^{nT}$ is given by identifying similar rows & columns entries from $(A+I)^n$ and $(A+I)^{nT}$.

	1	2	3	4	5	6	7	8
1	1							
2		1					1	
3			1	1	1			
4			1	1	1			
5			1	1	1			
6						1		
7		1					1	
8								1

→ From the above matrix by comparing a row/column with other rows/columns, the equivalent nodes to be grouped.

→ After grouping A = [1], B = [2,7], C = [3,4,5], D = [6], E = [8]

→ The graph & graph matrix representation to the above values is given by



flow Graph

	A	B	C	D	E
A	1	1			
B		1	1		
C			1	1	
D				1	
E			1		1

Graph Matrix

Building Tools

→ The out-degree of a node is the number of outlinks it has. The average degree (mean) of a node is between the two nodes

→ The in-degree of a node is the no. of in links it has

→ The degree of a node is the sum of in-degree and Out-degree.