Paths, Path Products and Regular Expressions.

1.1 Path Products and Path Expressions:-

1.1.1 Path Expressions:-

Path expression is defined as an expression which represents set of all the possible paths between an entry and an exit nodes.

Example: Consider the following flowgraph

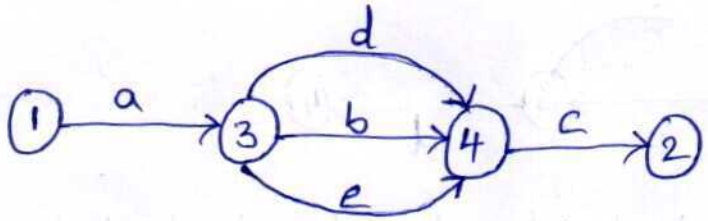
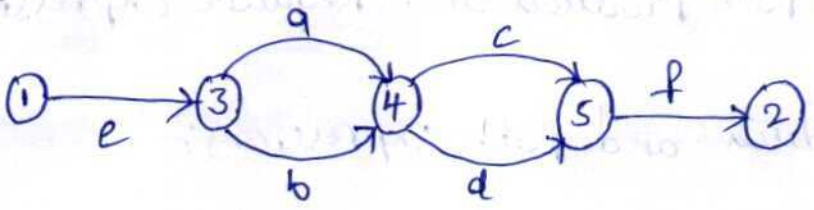


fig: Example of path

There are 3 available paths between node 1 and node 2: They are abc, adc, aec. Their path expression is a $a*(d+b+e)*c$.

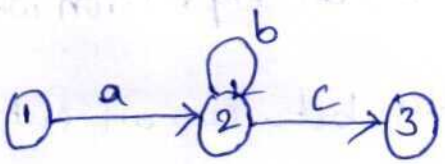
$a*(d+b+e)*c \Rightarrow adc+abc+aec$ which involves all the available paths.

Ex2:



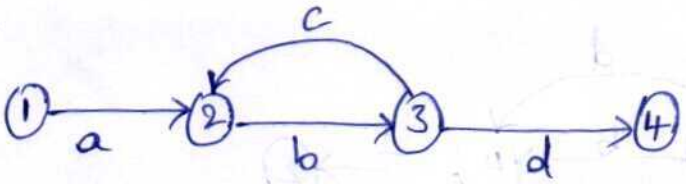
eacf, eadf, ebcf, ebdf

Ex3:



ac, abc, abbc, abbbc

Ex:4:



abd, ~~abcbd~~, ~~abcbcbd~~, abcbd, abcbcbd, abcbcbcbd.

1.1.2 Path Products :-

The concatenation of names of two ~~consecutive~~ consecutive path segments can be termed as path Product.

Ex: Let A and B be 2 path segments which has 'abcd' and 'efgh' paths respectively, then the path Product with A preceding B will be,

AB = abcd * efgh = abcdefgh

and Path Product with B Preceding A will be

BA = efgh * abcd = efghabcd.

If segments A and B has path expressions instead of individual paths, then their path product will be set of elements of A which are followed by elements of B in any order.

1.1.3 Path Sums:-

Path sum is the sum of all the parallel paths available between 2 nodes in which paths passing through intermediate nodes are also considered as parallel.

Path sum is denoted by '+' sign.

Example:- Consider the following example.

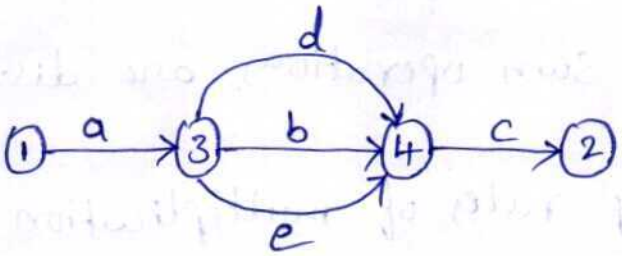


fig: Define path sums.

d, b, e are links between the nodes 3 and 4 and hence, they are referred as parallel paths and represented by d+b+e.

According to the definition, paths passing through intermediate nodes are also considered as parallel. So all paths between node 1 and node 2 are considered as parallel paths and can be denoted as adc + abc + aec.

Path sum is associative as well as commutative.

(i.e) adc + abc + aec = abc + aec + adc = aec + adc + abc

and also

adc + (abc + aec) = abc + (aec + adc) = aec + (adc + abc)

1.1.4 Distributive Laws :-

the product and sum operations are distributive and the ordinary rules of multiplication apply.

a(d + b + e)c = adc + abc + aec for the set of all paths from node 1 to node 2.

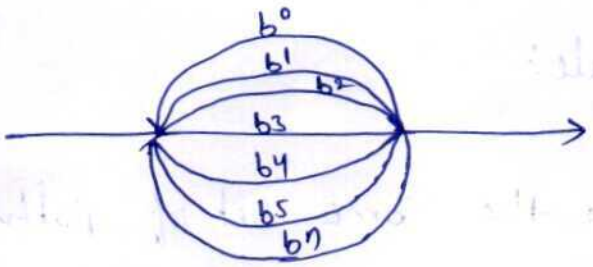
1.1.5 Absorption Rule:-

If X and Y denote the same set of paths, then the union of these sets is unchanged, consequently.

$$X + X = X \text{ (absorption rule)}$$

Loops:- Loops can be defined as a situation which occurs when a single link (or) path expression is being traversed indefinite number of times leading to infinite number of parallel paths.

Loops may result in never ending path expression for any flow graph. If a loop is a single link, it can be termed as self loop and indefinite number of repitition of that link is denoted by an asterik (*) placed on the link name. If the loop must be taken at least once, then it is denoted by 'a^+' or 'x^+'.



$b^0 + b^1 + b^2 + b^3 \ldots . b^n$

Ex:- $ab^*c = ac + abc + abbc + abbbc + \ldots .$

1.2 Reduction procedure:-

The main aim of node reduction procedure is to remove all the intermediate nodes between entry and exit nodes there by ascertain a relation between an entry and exit nodes.

There are certain rules and steps for removing the intermediate nodes. They are as follows,

1) Multiply the path expressions of the consecutive (successive) links to concatenate them.

2) Add the path expression of parallel links to produce a solo expression.

3) Eliminate all the self loops by replacing them with 0s their (self loops) path expression.

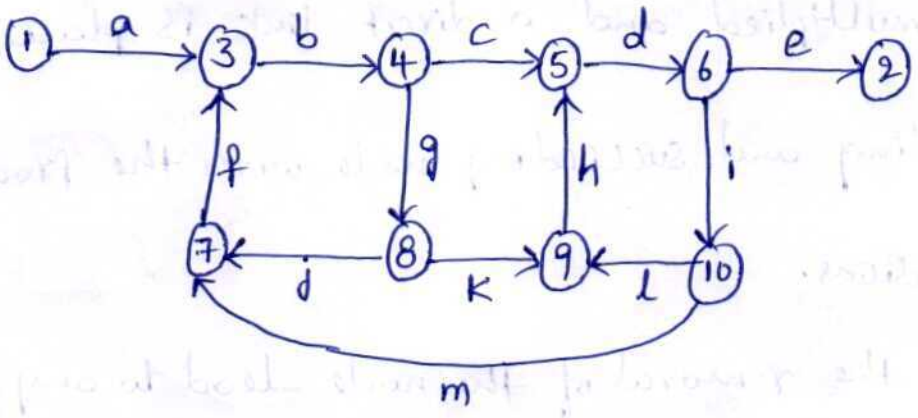
4) choose the node which is to be removed. The path expressions of the ~~links~~ inlinks and outlinks of this node are multiplied and a direct link is placed between Preceding and succeeding node with the product of path expressions.

5) If the removal of the node lead to any parallel paths follow step 2.

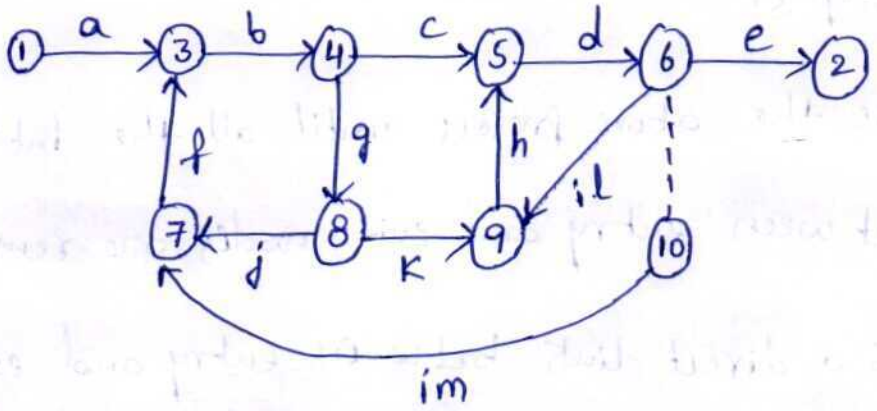
6) If the removal of the node lead to self loop formation follow step 3.

7) Continue the above process until all the intermediate nodes between entry and exit nodes are removed, and there is a direct link between entry and exit nodes with a path expression.

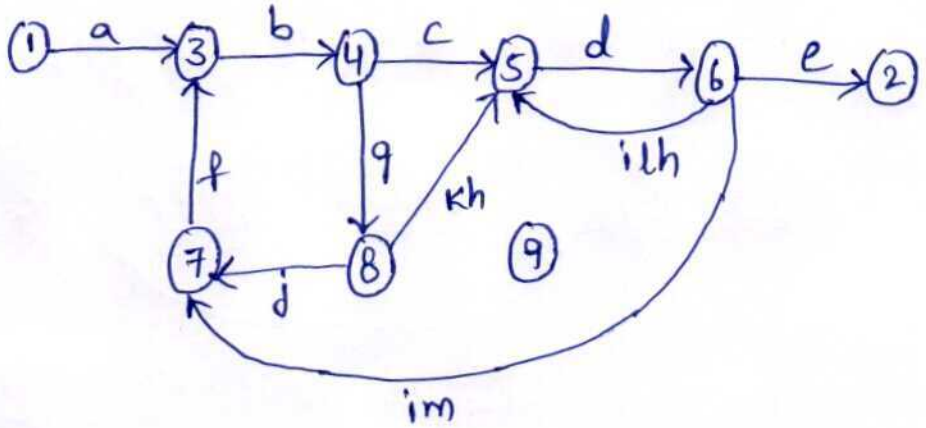
Applying node reduction procedure to the following graph.



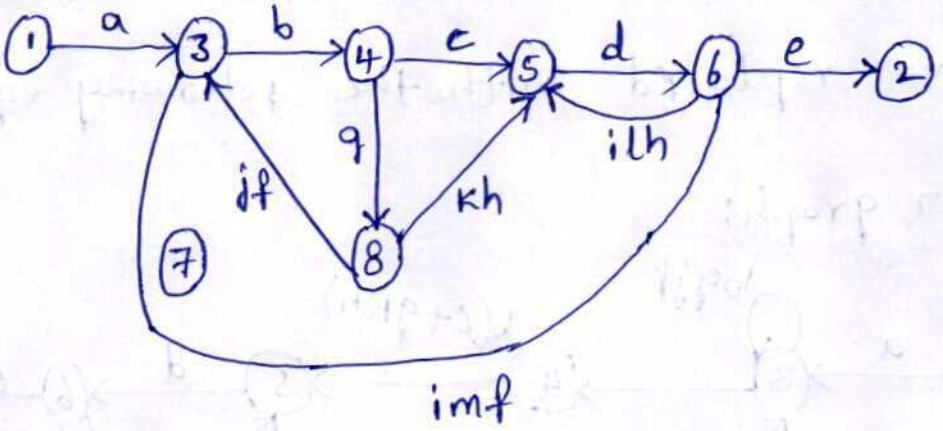
Remove node 10 by applying step 4 and combine by step 5 to yield.



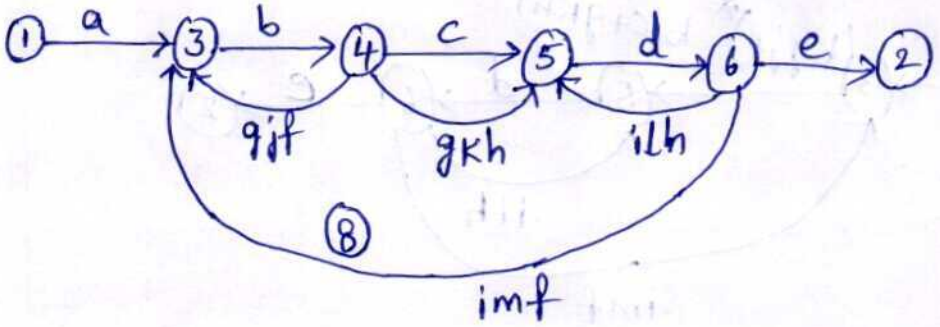
Remove node 9 by applying step 4 and 5 to yield.



Remove node 7 by step 4 and 5 as follows:

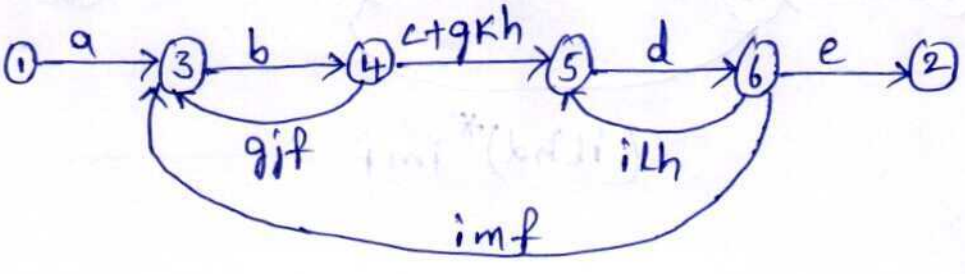


Remove node 8 by step 4 and 5, to obtain



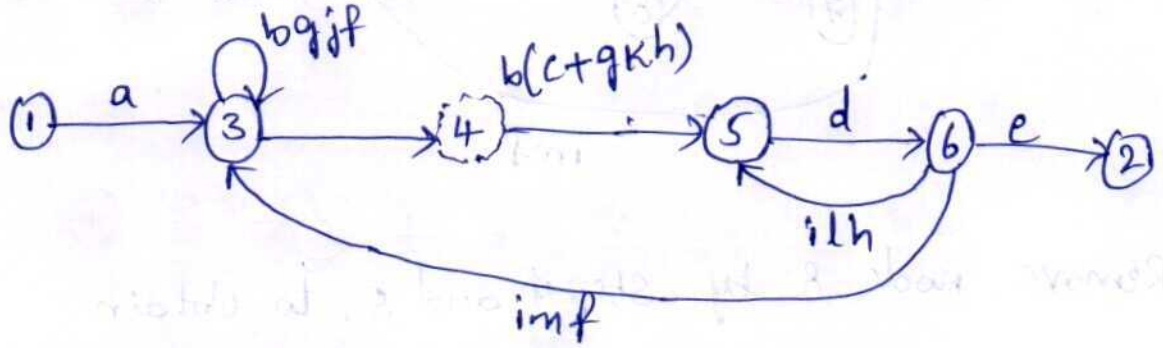
Parallel Term (Step 6)

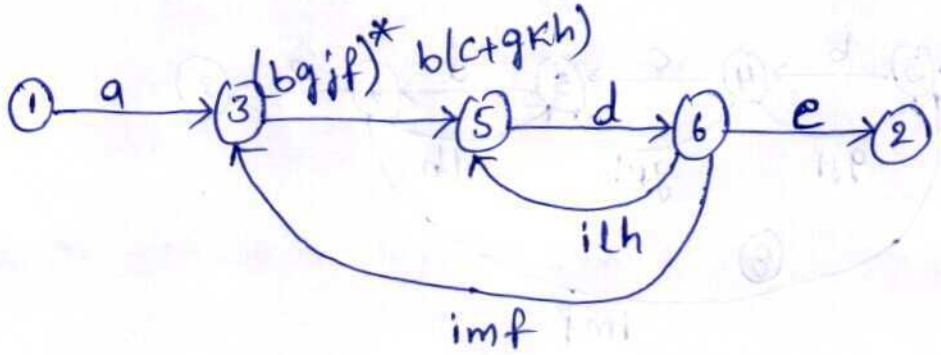
Removal of node 8 above led to a pair of parallel links between nodes 4 and 5. Combine them to create a path expression for an equivalent link whose path expression is "c+gkh"; that is



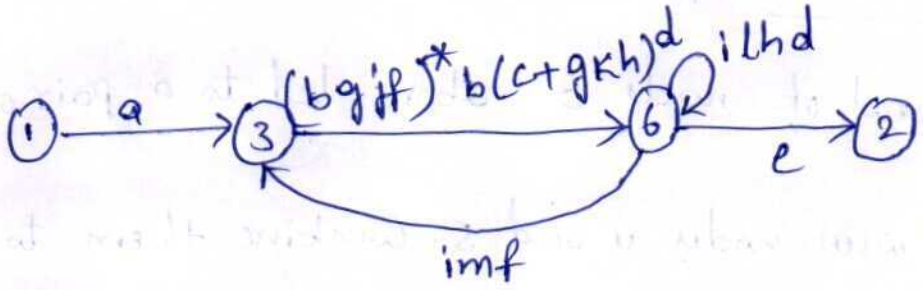
Loop Term (Step 7)

Removing node 4 leads to a loop term. The graph has now replaced with the following equivalent similar graph:

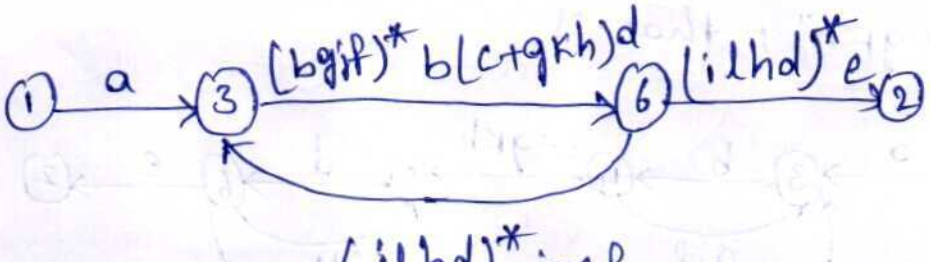




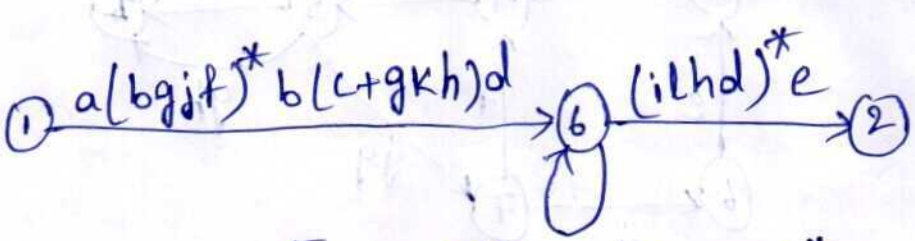
Removing node 5 procedures



Remove the loop at node 6 to yield.



Remove node 3 to yield:

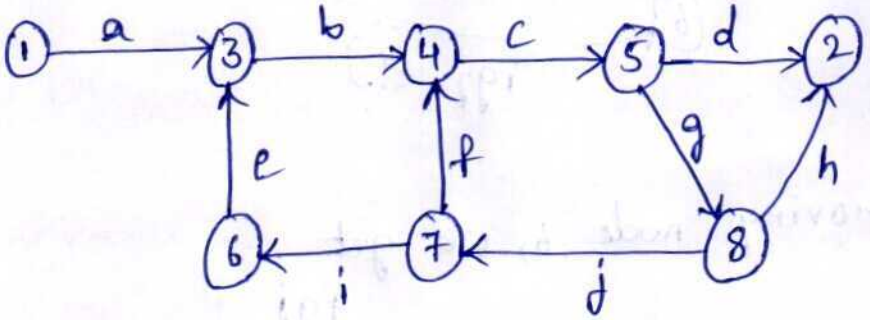


$$[(ilhd)^* imf \cdot (bgjf)^* b(c+gkh)d]$$

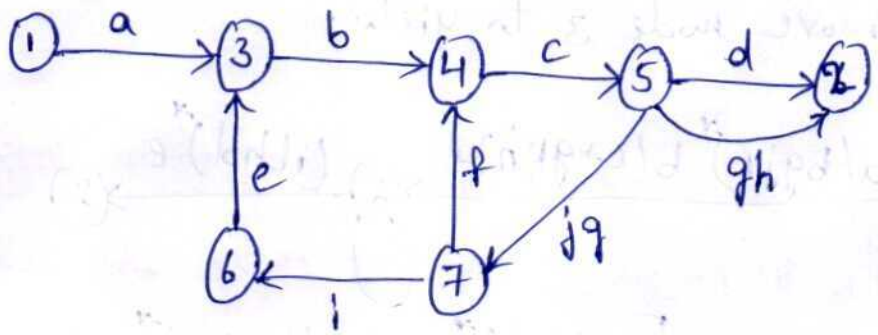
Removing the loop and then node 6 results in the following ~~eg~~ expression:

$$a(bgjf)^* b(c+gkh)d [(ilhd)^* imf \cdot (bgjf)^* b(c+gkh)d]^* (ilhd)^* e$$

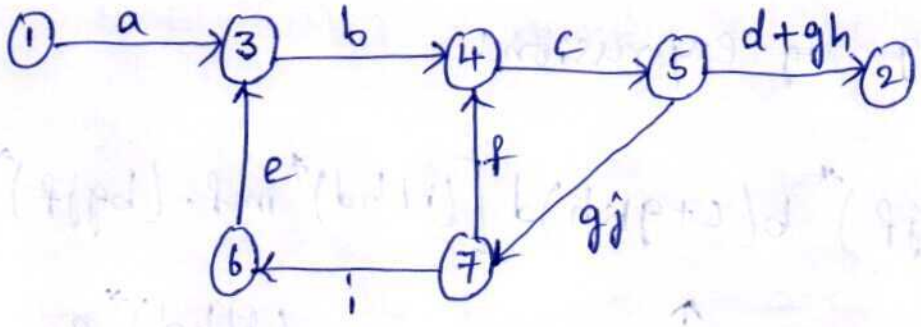
Ex2:- Applying node reduction Procedure to the following graph:-



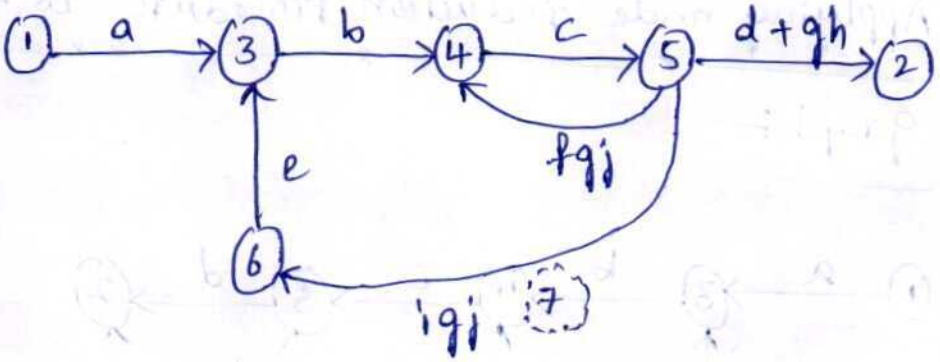
If node 8 is removed it will lead to a parallel path between node 5 and 2.



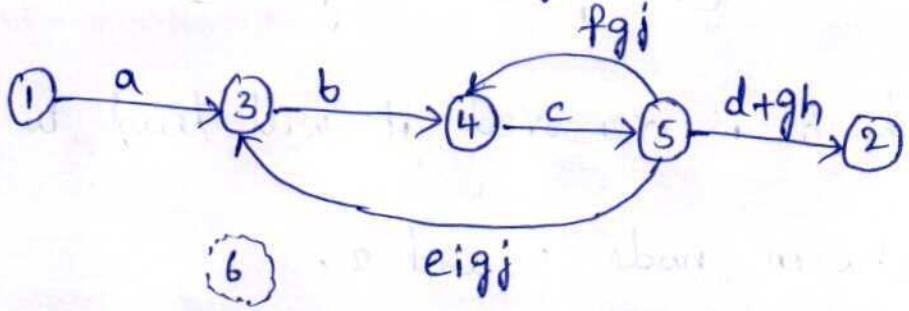
Now, Parallel paths are added to form a solo expression.



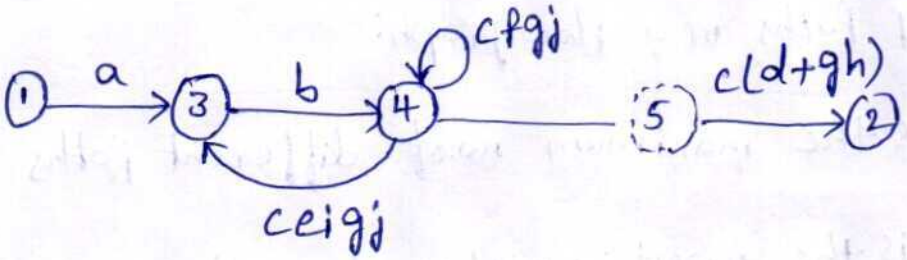
Selecting other node for removal node 7, we get.



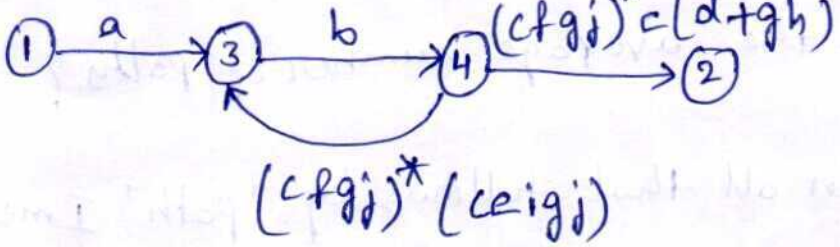
Now removing node 6, we get



Removing node 5 which leads to self loop on node 4

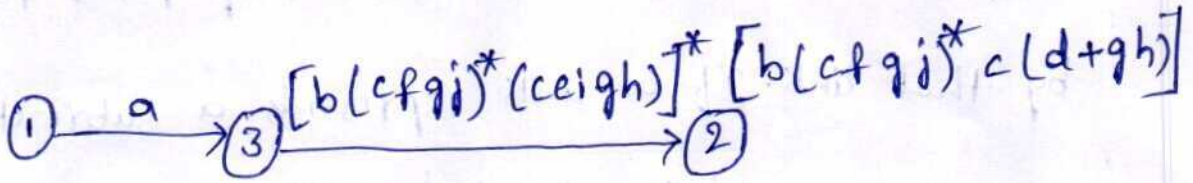


Remove self loop on node 4 using step 6

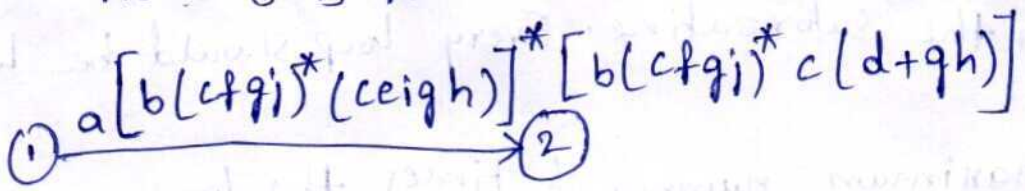


Selecting node 4 for removal, this node

Removing self loop on node 3, we get.



Node 3 is removed



Applications:-

How many Paths in a flowgraph:-

1) What is the maximum no. of different paths possible?
2) What is the fewest number of paths possible?
3) How many different paths are there really?
4) What is the average number of paths?

In all that follows by "path" I mean paths from the entrance to the exit of a single entry/ single exit routine.

i) Maximum path Count Arithmetic:-

Every link in a flowgraph is labelled with appropriate weights that specifies the number of paths represented by that link. If a link represents a subroutine call, then it should be labeled with the number of paths in the subroutine. Every loop should be labelled with maximum number of times the loop can iterate while

evaluating the expression. If the value is infinite, then maximum number of paths are also infinite.

There are 3 arithmetic cases that are considered while counting maximum number of paths. They are

1) Parallel rule
2) Series rule
3) loop rule

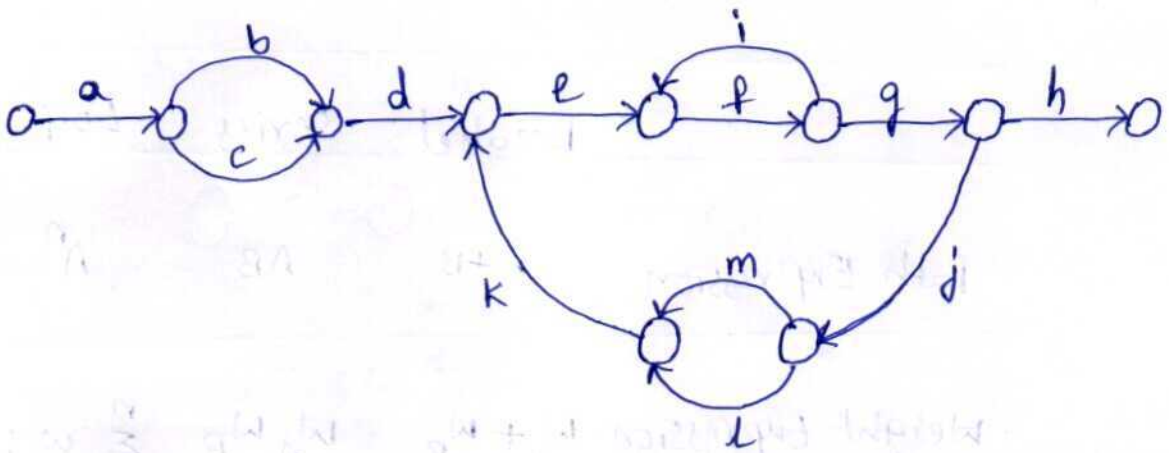
	Parallel	Series	Loop
Path Expression	$A+B$	AB	A^n
Weight Expression	$W_A + W_B$	$W_A W_B$	$\sum_{j=0}^{n} W_A^j$

Parallel Rule:- Each term in first path expression 'A' is summed up with each term in second path expression 'B'.

Series Rule:- Each term in path expression A is merged with each term in path expression B. i.e if there are W_A W_B paths in path expression A, B respectively then there

are $w_A \cdot w_B$ different combination of paths in the entire path expression.

Loop Rule:- It is the combination of parallel and series rule. It is evaluated by considering number of times, the path is iterated. If the minimum ~~no~~ number of times the loop is iterated is m, but not zero.



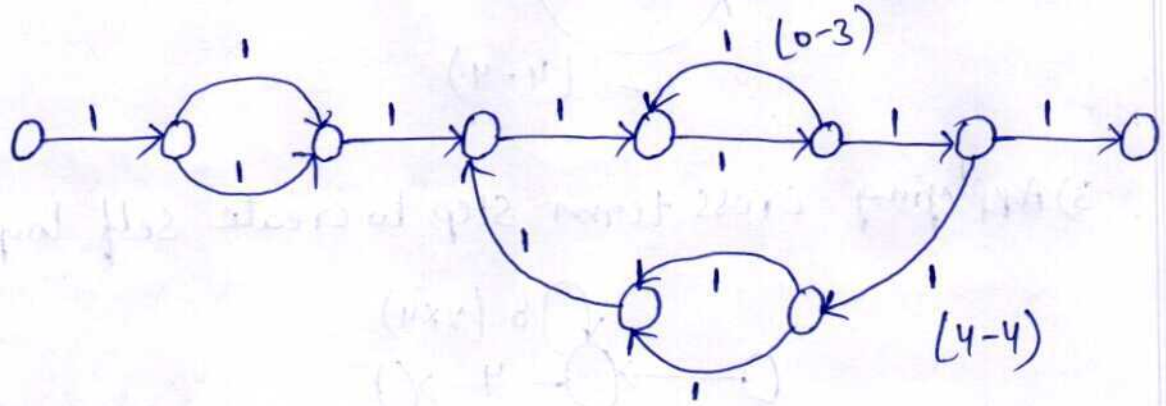
Path expression of the above fig is

$$a(b+c)d\{e(fi)^* fgj(m+l)k\}^* e(fi)^* fgh$$

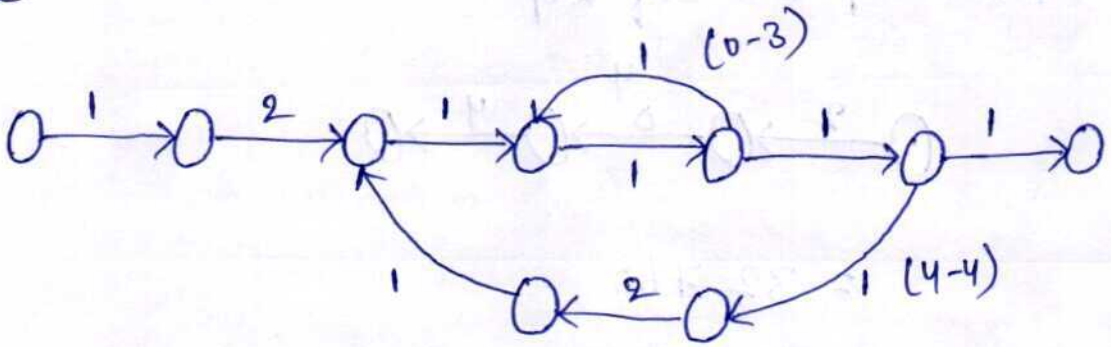
Each single link is subsequently given the weight of 'i' in the flowgraph to initiate. Assume that outer loop is considered "four times" and inner loop is iterated from "zero to three" times.

Ex:- for outer loop

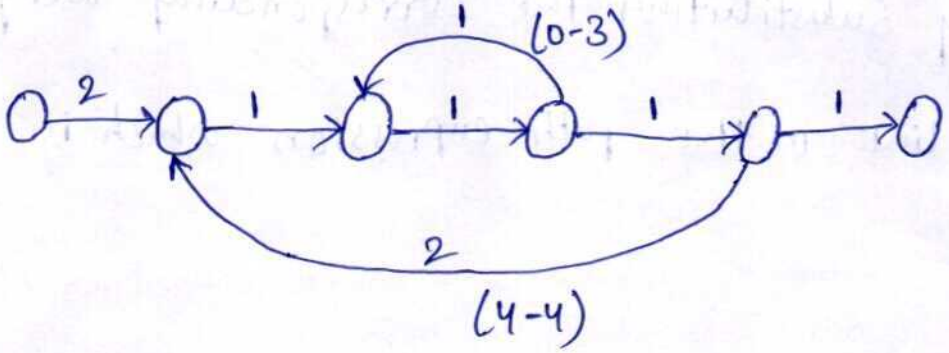
1) All single links are assigned weight of 1



2) All parallel loops are reduced by applying parallel rule.

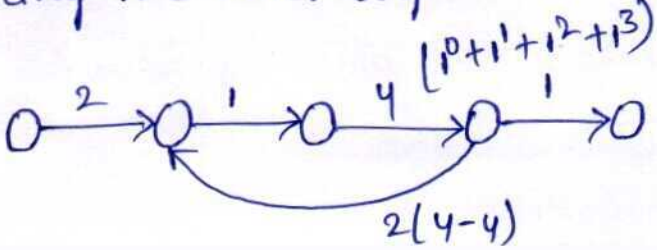


3) Serial links are combined by applying series rule.

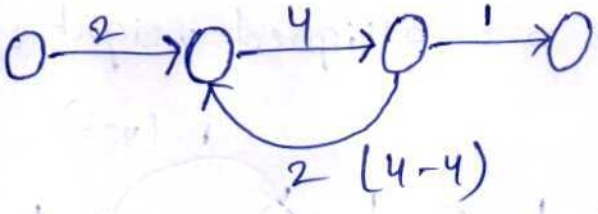


For Inner loop

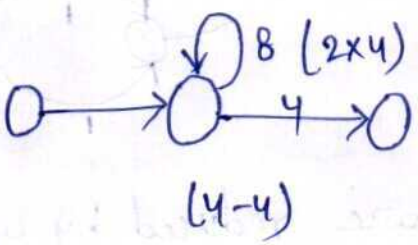
1) Reducing the inner loop



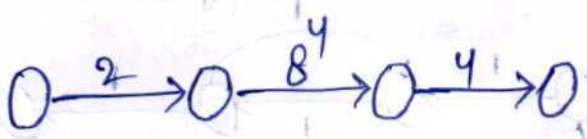
2) Combining Serial links



3) Applying cross term step to create self loop



4) Reducing the self loop



$= 32,768$

the alternate way to calculate maximum no of paths is by substituting the corresponding weight of each link in the path expression which is,

$$a(b+c)d\{e(fi)^* fg; (m+l)k\}^* e(fi)^* fgh$$

$$1(1+1)1\{1(1\times1)^3 1\times1\times1 (1+1)1\}^4 1(1\times1)^3 1\times1\times1$$

$$1(2)1\{1(1)^3 1(2)1\}^4 1(1)^3 1$$

$$2\{1(1)^3(2)1\}^4 1(1)^3 1 \qquad \because 1^3 = 1^0+1^1+1^2+1^3 = 4$$

$$2\{1(4)(2)\}^4 1(4)1$$

$$= 2\{4\times2\}^4 4$$

$$= 2\times8^4\times4$$

$$= 32,768$$

Following are the steps for calculating maximum number of path,

1) Assign each single link with their corresponding weights and also consider the maximum number of times a loop is iterated. The inner loop is iterated (0-3) times and outer loop is iterated exactly (4-4) times.

2) All possible parallel loops are combined that are present out side the loop and if-then-else construct.

3) All possible serial links are combined by applying series rule in order to discard nodes to eliminate the confusion.

4) Inner loops are then considered. In the above example 4 combination of values are possible that results in 4 different values. Each link is multiplied with the following link whose weight is 1.

5) Some nodes are discarded.

6) Cross term step is applied to create self loop.

Approximate Minimum Number of Paths:-

i) Structured code:-

Based on the path expression obtained by node-by-node reduction procedure we can determine whether the given graph is Structured (or) Unstructured.

ii) Lower path count Arithmetic:-

The lowest number of possible paths in a structured flowgraph can be approximately known, it may (or)

may not be accurate because there is every possibility of a path being ~~unachievable~~ unachievable which further lowers the number count.

The lowest number of paths identified by this method must be cross checked to the minimum number of paths in your test plan.

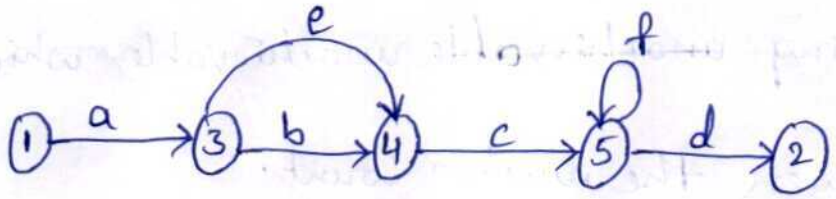
There are 3 cases on which, we have to concentrate they are parallel links, serial links and loops.

Arithmetic Rules

	Parallel	Serial	loop
Path Expression	A+B	AB	A_n
weight Expression	$w_A + w_B$	$MAX(w_A, w_B)$	$1, w_1$

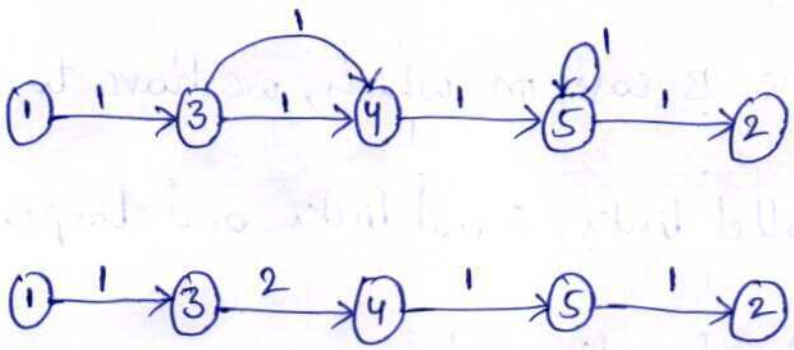
If a routine contains a loop, to know the minimum number of paths, it is better to assume that a routine will traverse a loop only once.

Ex:- Consider the graph below figure.

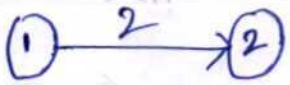


The path expression is $a(b+e)cf^*d$

The minimum number of paths can be determined by substituting the weights for each link.



Max(1, 2, 1, 1)



$a(b+e)cf^*d$

$= 1(1+1) \times 1 \times 1 \times 1 = 2$

The Mean processing Time of a Routine:-

Mean processing time of a routine can be calculated by the following two weights.

1) T - which is the execution time of every link in a flow graph.

2) Probability of each link.

The three arithmetic rules for calculating mean Processing time are as follows.

1) Parallel rule
2) Series rule
3) Loop rule

Arithmetic rules

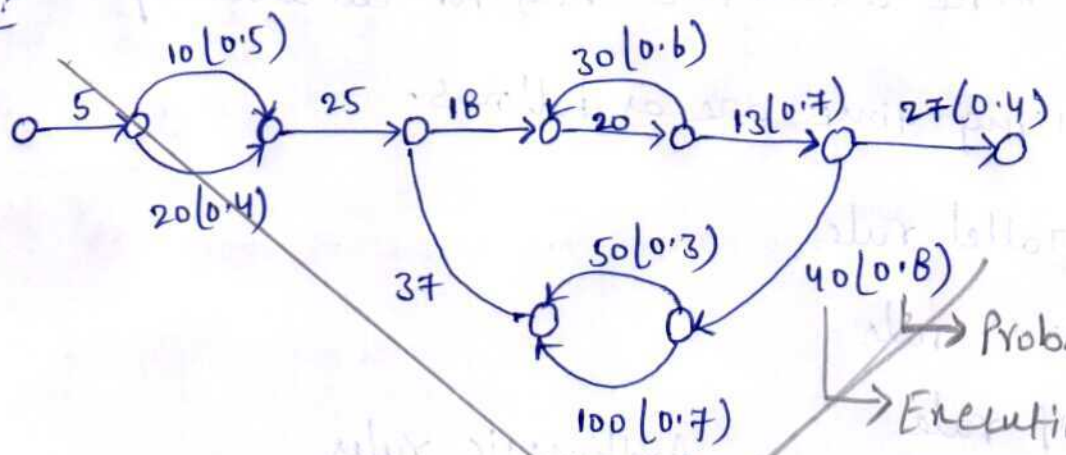
	Parallel	Series	Loop
Path Expression	$A+B$	AB	A^*
Weight Expression	$T_{A+B} = \frac{(P_A T_A + P_B T_B)}{P_A + P_B}$ $P_{A+B} = P_A + P_B$	$T_{AB} = T_A + T_B$ $P_{AB} = P_A \cdot P_B$	$T_A = \frac{T_L \cdot P_L}{(1-P_L)}$ $P_A = \frac{P_A}{(1-P_L)}$

Parallel Rule:- It is the arithmetic mean of all execution time over all parallel links in a flowgraph.

Series Rule:- It is the sum of two execution times.

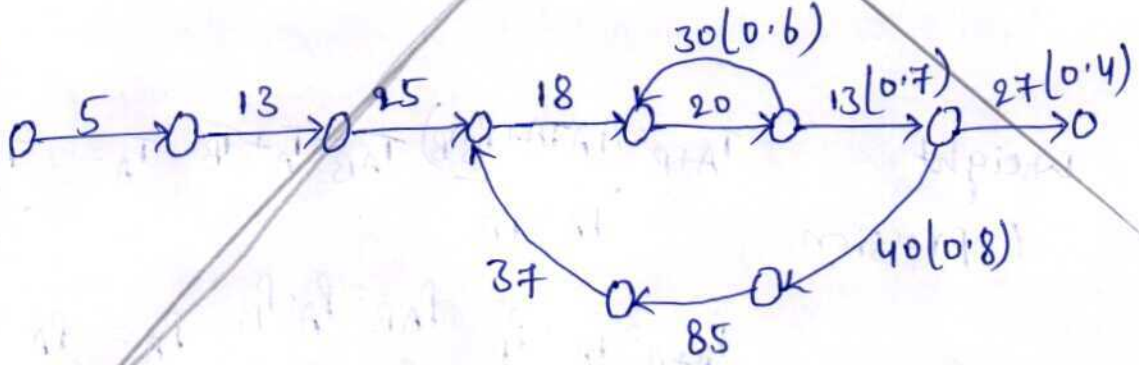
Loop Rule:- It is the combination of parallel and series rule. A loop is evaluated by considering number of times the path is iterated.

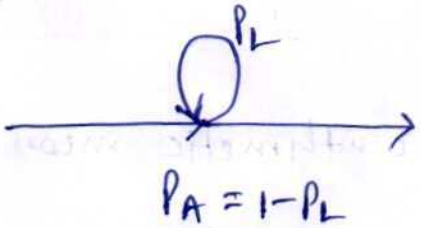
Ex:-



Each link is assigned with probabilities and execution time in microsecond.

1)

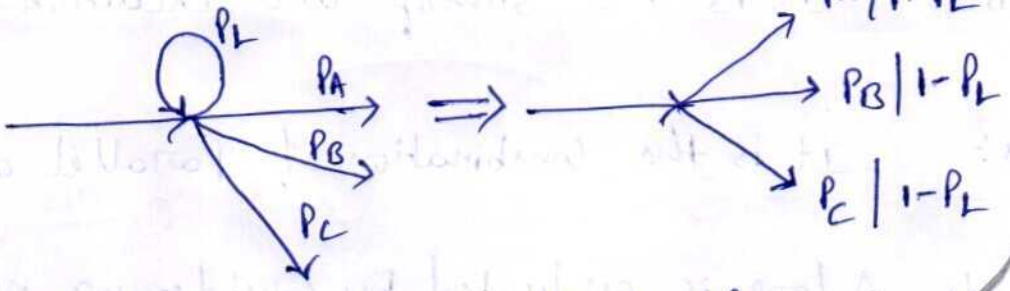




$P_A = 1 - P_L$

$$\frac{P_A}{1-P_L} + \frac{P_B}{1-P_L} + \frac{P_C}{1-P_L} = \frac{P_A + P_B + P_C}{1-P_L} = 1$$

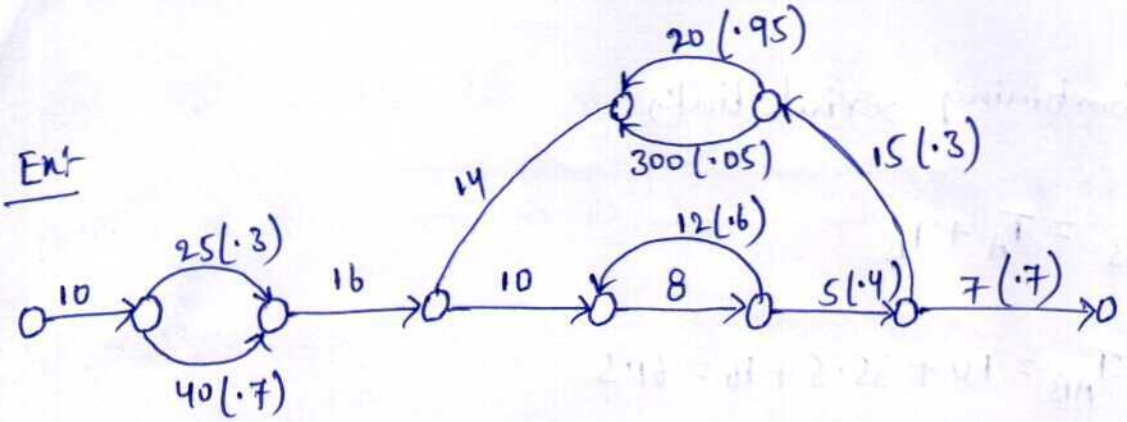
$$P_{NEW} = \frac{P_A}{1-P_L} = \frac{1-P_L}{1-P_L} = 1$$



$P_L + P_A + P_B + P_C = 1$, $1 - P_L = P_A + P_B + P_C$

~~$P_A/1-1$~~

Ex:

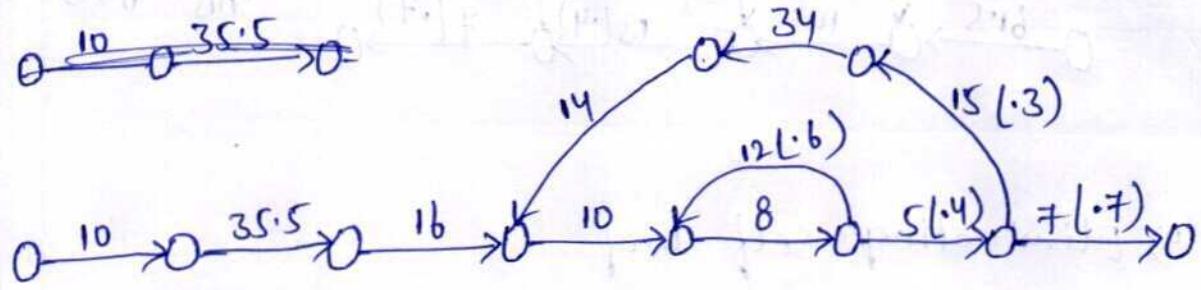


1) Combining Parallel Paths

$$T_{A+B} = (P_A T_A + P_B T_B) / (P_A + P_B)$$

a) $$T_{A+B} = \frac{25 \times .3 + 40 \times .7}{.3 + .7} = \frac{7.5 + 28.0}{1.0} = 35.5$$

b) $$T_{A+B} = \frac{20 \times .95 + 300 \times .05}{.95 + .05} = \frac{19.0 + 15.0}{.95 + 0.05} = \frac{34.0}{1.0} = 34.0$$



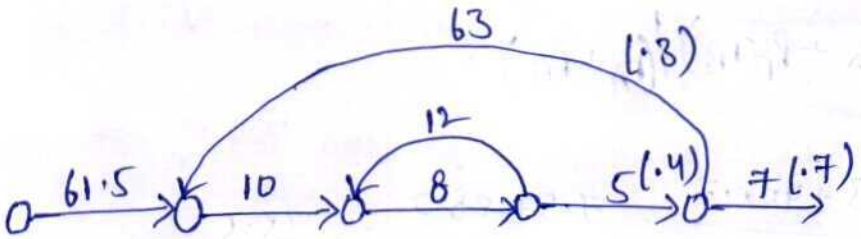
All possible parallel links are combined. The result that is obtained is the arithmetic mean of the execution times for the links.

2) Combining Serial links

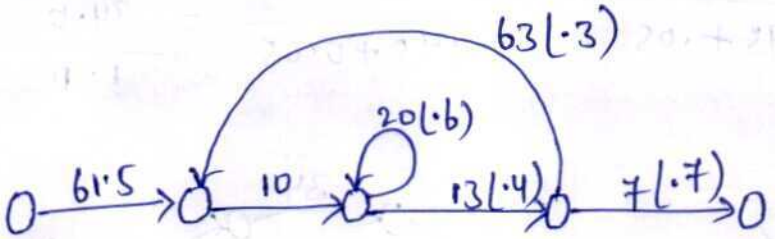
$T_{AB} = T_A + T_B$

a) $T_{AB} = 10 + 35.5 + 16 = 61.5$

b) $T_{AB} = 14 + 34 + 15 = 63$



3) Creating Inner self loop



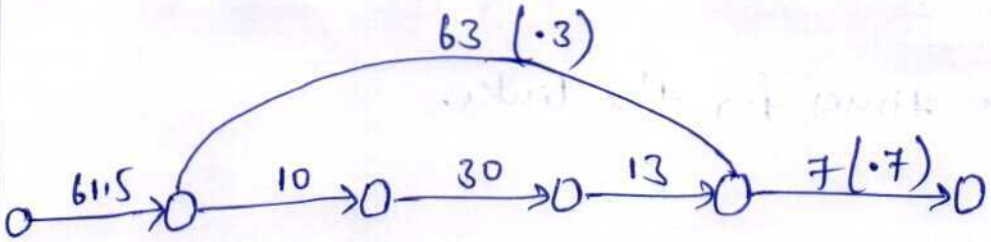
$T_{AB} = T_A + T_B$

$T_{AB} = 12 + 8 = 20$

$T_{AB} = 8 + 5 = 13$

4) Eliminating self loop

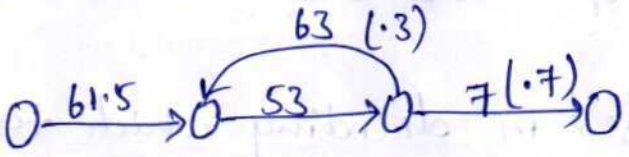
$$T_A = \frac{T_L P_L}{(1 - P_L)} = \frac{.6}{.4} \times 20 = 30$$



Combining Serial links

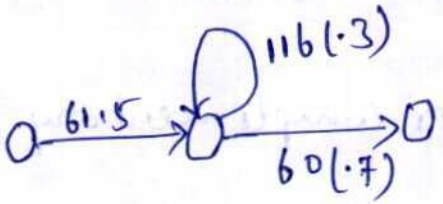
$T_{AB} = T_A + T_B$

$T_{AB} = 10 + 30 + 13 = 53$



5) creating self loop

$T_A = \frac{T_L P_L}{1 - P_L} = \frac{.3}{.7}$



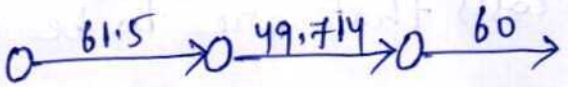
$T_{AB} = T_A + T_B$

$T_{AB} = 63 + 53 = 116$

$T_{AB} = 53 + 7 = 60$

6) Eliminating self loop

$T_A = \frac{T_L P_L}{(1 - P_L)} = \frac{.3 \times 116}{.7} = 49.714$



$= 171.214$

Combine serial links

$T_{AB} = T_A + T_B = 61.5 + 49.714 + 60 = 171.214$

Push/Pop, Get/Return :-

Push/Pop Arithmetic :- the solutions to the queries provided by push/pop model are used as a debugging tool and even used in deciding which test cases are to be designed. Push operation is performed to insert elements in the stack. Pop operation is performed to remove elements from the stack.

A part from push/pop, other complementary operations are

i) Get/Return

ii) Open/close

iii) Start/Stop

there are 3 arithmetic cases that are to be considered while performing push/pop operation.

	Parallel	Series	Loop
Path Expression	$A+B$	AB	A^*
weight Expression	$w_A + w_B$	$w_A \cdot w_B$	w_A^*

In all the three cases A, B represent the path expression, w_A and w_B represent the weight expression.

Push/Pop operations satisfies commutative, associative and distributive law of addition and multiplication.

In the arithmetic table for push/pop operation, P_1 represents the "push" push operation, P_2 represents "pop" operation and '1' denotes "no" operation.

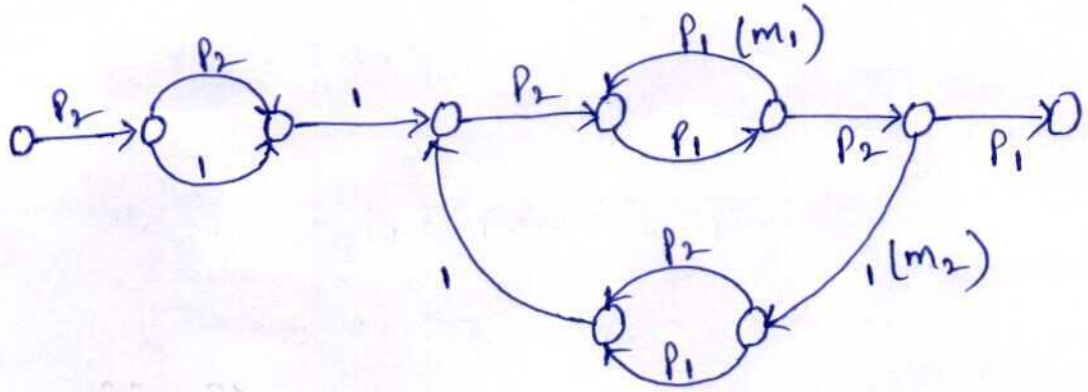
Push/Pop Multiplication Table

*	P_1	P_2	1
P_1	P_1^2	1	P_1
P_2	1	P_2^2	P_2
1	P_1	P_2	1

Push/Pop Addition Table

+	P_1	P_2	1
P_1	P_1	P_1+P_2	P_1+1
P_2	P_2+P_1	P_2	P_2+1
1	P_1+1	P_2+1	1

Consider the following flowgraph



Path expression for the flowgraph

$$P_2(P_2+1)\cdot\{P_2(P_1P_1)^{m_1}P_1P_2(P_1+P_2)\cdot\}^{m_2}P_2(P_1P_1)^{m_1}P_1P_2P_1$$

Simplifying the above expression.

$$(P_2^2+P_2)\cdot\{P_2(P_1^{2m_1})\cdot(P_1+P_2)\}^{m_2}P_2(P_1^{2m_1})P_1$$

$$(P_2^2+P_2)\{P_1^{2m_1}(P_2^2+P_1P_2)\}^{m_2}(P_1^{2m_1}) \quad [\because P_1P_2=1]$$

$$(P_2^2+P_2)\{P_1^{2m_1}(P_2^2+1)\}^{m_2}(P_1^{2m_1})$$

Let us Consider the following Cases where m_1, m_2 represents the looping terms.

m_1 represents the number of time inner loop is considered, m_2 represents the number of times outer loop is considered.

Case1: Consider the value of $m_1=0$, $m_2=0$

Substituting these values in expression below,

$$(P_2^2+P_2)\{P_1^{2m_1}(P_2^2+1)\}^{m_2}P_1^{2m_1}$$

$$(P_2^2+P_2)\{P_1^{2(0)}(P_2^2+1)\}^{0}P_1^{2(0)} \quad [\because P_1^0=1]$$

$$(P_2^2+P_2)\{1\cdot(P_2^2+1)\}^{0}P_1^0 = \boxed{P_2^2+P_2}$$

Case(ii) :- Consider the value of $m_1=0, m_2=1$

Substituting these values in expression below

$$(P_2^2+P_2)\{P_1^{2m_1}[P_2^2+1)\}^{m_2} P_1^{2m_1}$$

$$(P_2^2+P_2)\{P_1^{2(0)}\{P_2^2+1)\}^1 P_1^{2(0)}$$

$$(P_2^2+P_2)\{1[P_2^2+1)\}^1\times 1$$

$$=(P_2^2+P_2)(P_2^2+1)$$

$$=P_2^4+P_2^2+P_2^3+P_2$$

$$=\boxed{P_2+P_2^2+P_2^3+P_2^4}$$

Case(iii) :- $m_1=0, m_2=2$

$$(P_2^2+P_2)\{P_1^{2m_1}(P_2^2+1)\}^{m_2} P_1^{2m_1}$$

$$(P_2^2+P_2)\{P_1^{2(0)}(P_2^2+1)\}^2 P_1^{2(0)}$$

$$(P_2^2+P_2)\{1\times(P_2^2+1)\}^2\times 1$$

$$(P_2^2+P_2)\{P_2^4+2P_2^2+1\}$$

$$=P_2^6+2P_2^4+P_2^2+P_2^5+2P_2^3+P_2$$

$$=P_2^6+P_2^5+2P_2^4+2P_2^3+P_2^2+P_2$$

$$\Longrightarrow \sum_{i=1}^{6} P_2^i$$

Different combinations of m_1, m_2 values the following table is obtained.

m_1	0	0	0	0	1	1	1	2
m_2	0	1	2	3	0	1	2	0
Push/ Pop	$P_2 + P_2^2$	$P_2 + P_2^2 + P_2^3 + P_2^4$	$\sum_{i=1}^{6} P_2^i$	$\sum_{i=1}^{8} P_2^i$	$1 + P_1$	$\sum_{i=0}^{3} P_1^i$	$\sum_{i=0}^{5} P_1^i$	$P_1^2 + P_1$

2) Get/Return Model :-

The arithmetic tables for Get/Return

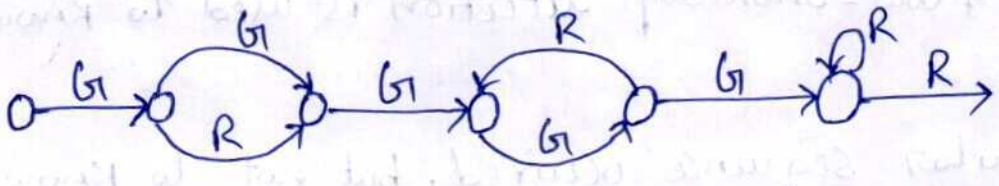
Multiplication Table:

X	Get G	Return R	NO 1
G	G^2	1	G
R	1	R^2	R
1	G	R	1

Addition Table:

+	G	R	1
G	G	G+R	G+1
R	G+R	R	R+1
1	G+1	R+1	1

Consider the following flowgraph



Example flowgraph for Get/Return

Path expression of the flowgraph

$G(G+R)\,G(GR)^*\,G\,GR^*\,R$

$G(G+R)\,G(1)^*\,G\times 1\times R^*$ $\quad [\because GR=1]$

$(G+R)\,G^3R^*$

$(G^4+G^3R)\,R^*$

$(G^4+G^2\cdot GR)R^*=(G^4+G^2)R^*$

Regular Expressions and flow Anomaly Detection:-

Regular expressions are used to examine the structural behaviour of flowgraph. The applications of regular expressions are as follows,

1) Counting maximum number of paths in the flowgraph.

2) Estimating the mean processing time of a routine.

3) To check if data flow anomaly occur(or) not.

Flow-anomaly detection is used to know if Particular sequence occured, but not to know the total

impact of the Procedure. It is used to detect the bug sequence in following situations.

1) The operations that are performed on a file are open (o), close (c), read (r), write (w). If operations read/write is performed on a file that is already closed the sequence cr/cw are said to be anomalous. In the same way, the sequence "or" is said to be anomalous if a file performs a read operation after it is opened, but before performing the write operation on that file.

2) The operations performed by tape-transport device are read (r), write (w), rewind (d), forward (f), skip (k), stop (p). There are certain rules that are to be followed while using a tape-transport device which rewind and forward operations can't be performed

one after the other without performing stop operation. Then the sequence anomaly df, fd are said to occur. The other sequence dr, dw, fr are also said to be anomalous.

3) With the help of generic flow anomaly detection, it is possible to detect the data flow bugs sequence such as dd, KK, dK, KU.

i) Huang theorem:-

Huang theorem is used to simplify the regular expression and to examine the specific operation sequence. Consider P, Q, R as non empty sets of character sequences that consist of at least one-character. Let 'M' be ~~the~~ two character long string. If 'M' is substring of "PQ^nR" it means that the string 'M' is present in the path "PQ^2R" Consider the following example

$P = aa$

$Q = bcc$

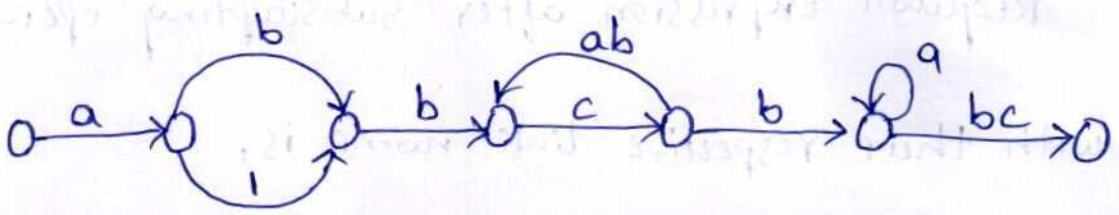
$R = ca$

$M = bb$

According to Huang's ~~theorm~~ theorem, bb will appear in sequence $aa(bcc)^n ca$, if it appears in sequence $aa(bcc)^2 ca$.

Example of Data flow Testing:-

By assigning appropriate operators on each link, the following flowgraph can be used to detect different data flow anomalies bugs.



the regular expression for the flowgraph

$$a(b+1)\,b(cab)^*\,cba^*bc$$

As the first loop is not considered while evaluating the regular expression.

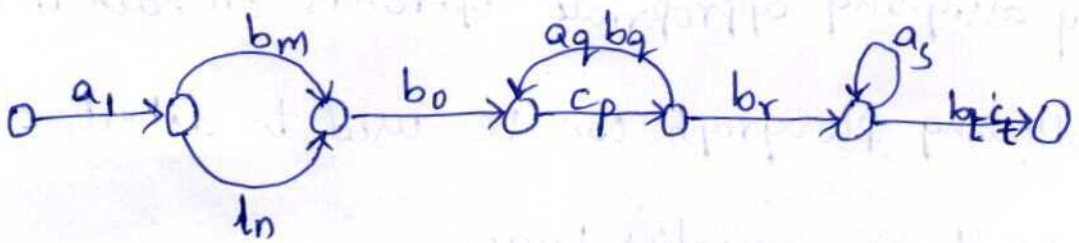
Applying Huang's theorem to detect two character sequence bugs. the above regular expression becomes,

$$a(b+1)b[1+(cab)^2]\,cb(1+a^2)bc$$

$$(ab+a)b[1+cabcab]\,(cb+cba^2)bc$$

$$(abb+ab)[1+cabcab]\,(cbbc+cba^2bc)$$

The above flowgraph is modified as



Regular expression after subscripting operators with their respective link name is,

$$a_1(b_m+l_n)b_o(c_p a_q b_q)^* c_p b_r a_s^* b_t c_t$$

Applying Huang's theorem to detect two character sequence anomalies,

$$a_1(b_m + L_n) b_0 (1 + (c_p a_q b_q)^2) c_p b_r (1 + a_s^2) b_t c_t$$

$$(a_1 b_m + a_1 L_n) b_0 (1 + (C_p a_q b_q)^2) (c_p b_r b_t c_t + c_p b_r a_s^2 b_t c_t)$$

$$(a_1 b_m + a_1 L_n)(b_0)(1 + c_p a_q b_q c_p a_q b_q)(c_p b_r b_t c_t + c_p b_r a_s^2 b_t c_t)$$

The looping Probability of a path Expression:-

Probabilities are calculated at every decision node including the loop node. Assign each outgoing link with a weight that is equal to the probability of moving the weights in the same direction.

The 3 arithmetic rules for calculating the looping Probability are as follows,

i) Parallel rule

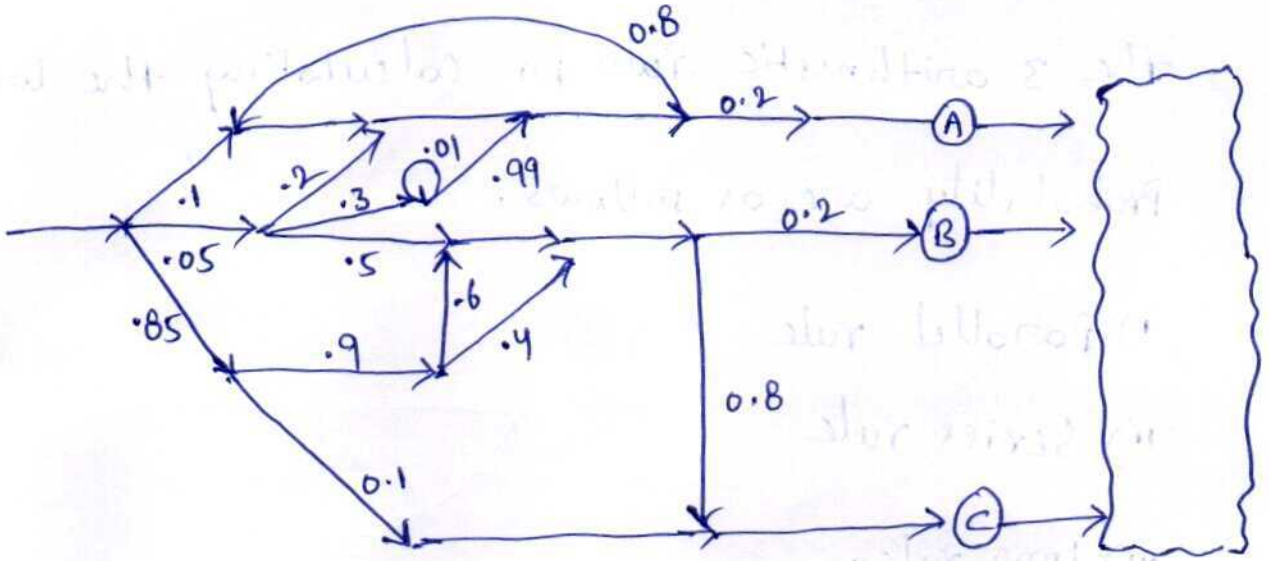
ii) Series rule

iii) Loop rule.

	Parallel	Series	loop
Path Expression	$A+B$	AB	A^*
weight Expression	P_A+P_B	$P_A \cdot P_B$	$P_A/(1-P_L)$

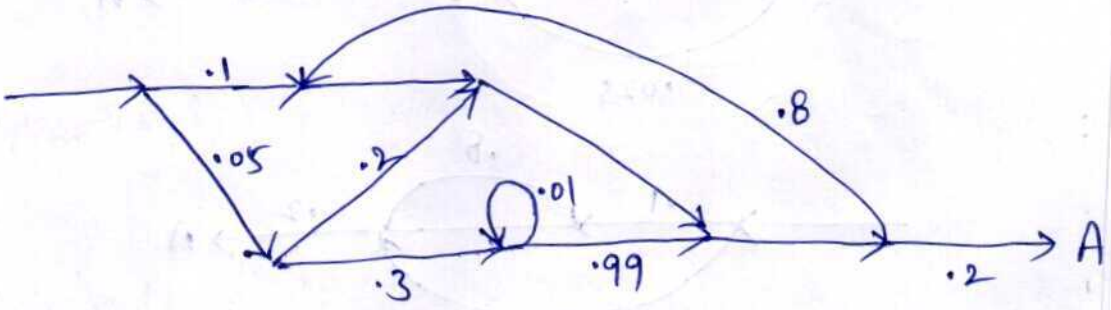
A and B represents path expression and P_A, P_B represents probabilities, where P_A is the probability of existing loop, P_L is the looping Probability.

Example:-



Note that the sum of the probabilities at each decision node is equal to 1.

Case A:-

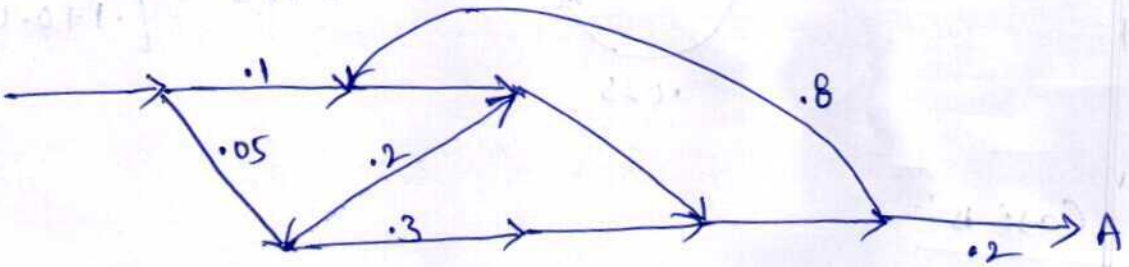


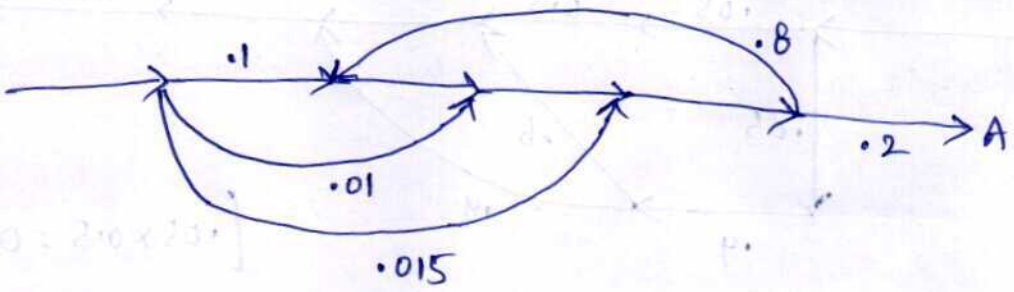
first eliminating self loop

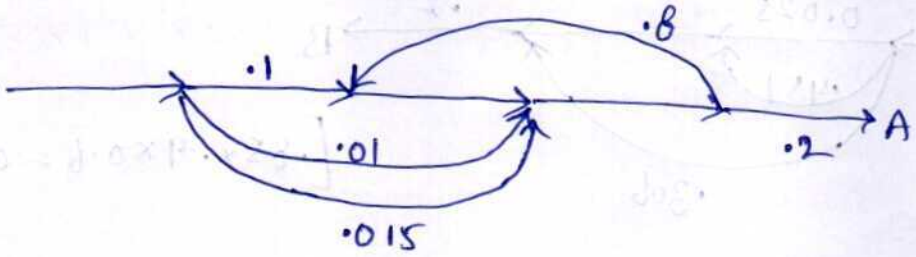
$$P_A = 1 - P_L$$

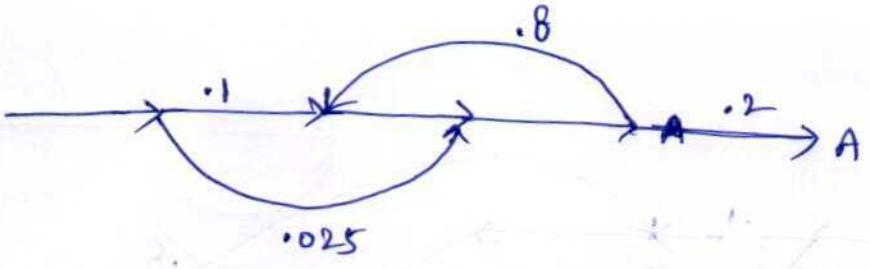
$$.99 = 1 - 0.01$$

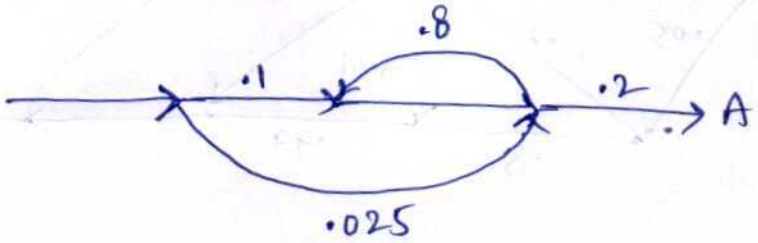
$$0.99 = 0.99$$

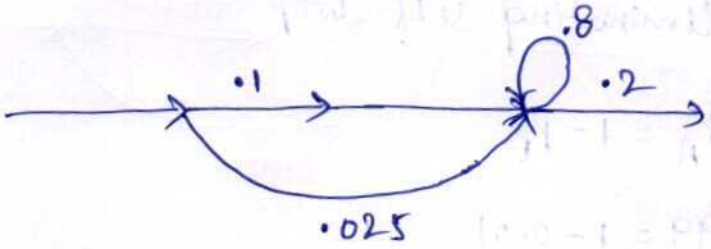




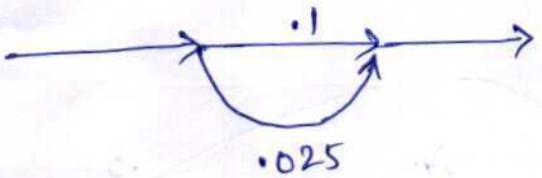








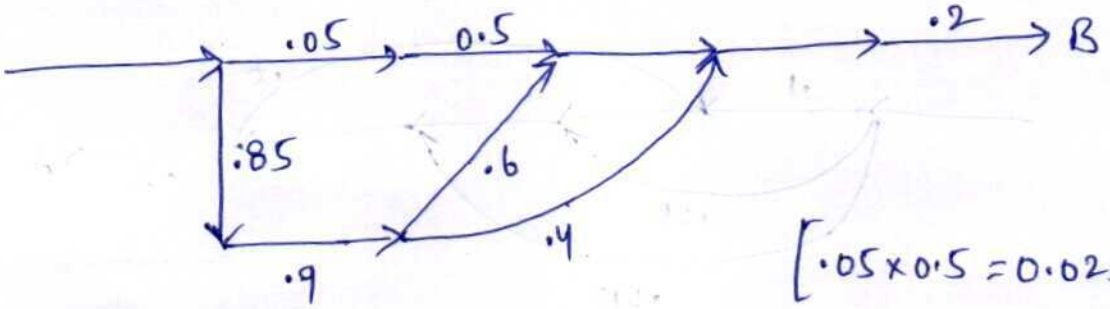
$[\because P_A = 1 - P_L$
$.2 = 1 - 0.8]$



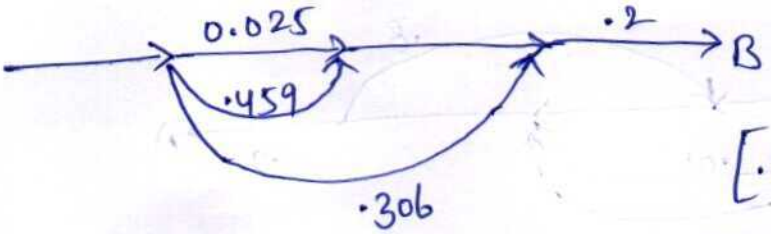
$= 0.125$

$[.1 + 0.025 = 0.125]$

Case B:-

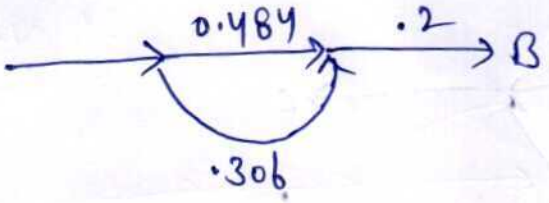


$[.05 \times 0.5 = 0.025]$



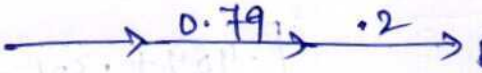
$[.85 \times .9 \times 0.6 = 0.4590]$

$[0.85 \times 0.9 \times 0.4 = 0.306]$



[.025 + 0.459 = 0.484]

[.484 + 0.306 = 0.79]



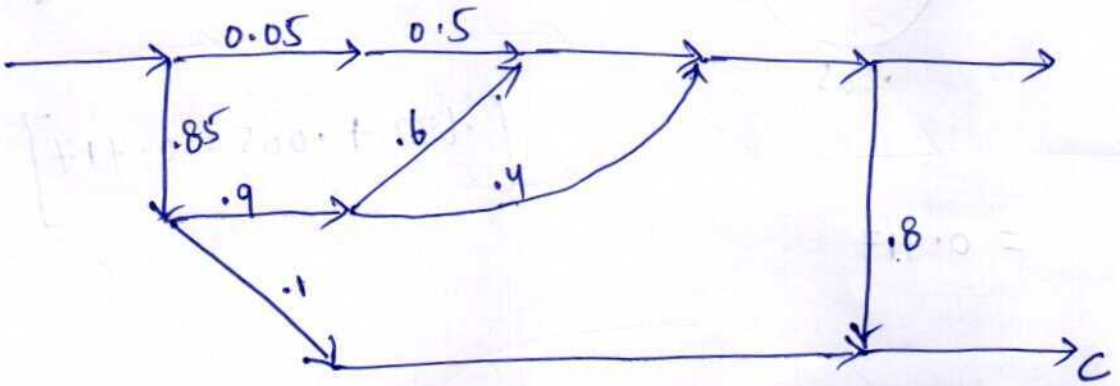
= 0.158

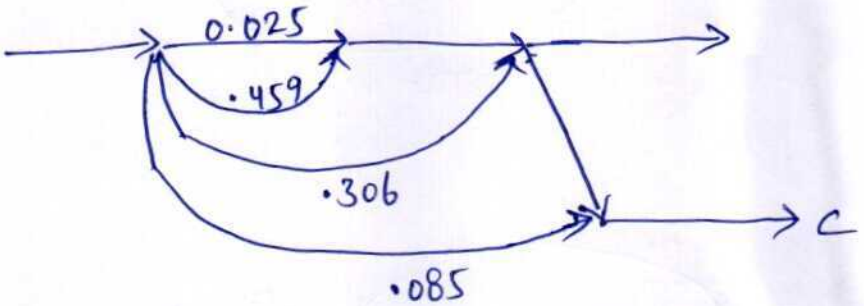
Case c:-

$1 - 0.125 - 0.158 = 0.717$

$P_A + P_B + P_C = 1$

$P_C = 1 - P_A - P_B$

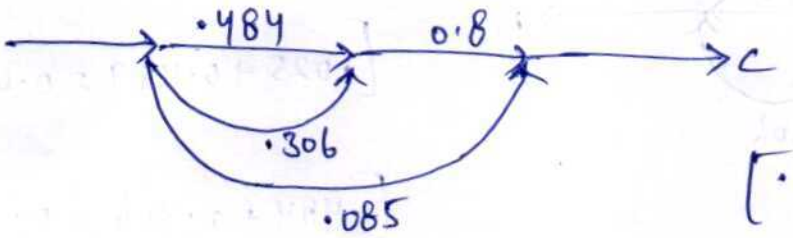




$.85 \times .9 \times .6 = .459$

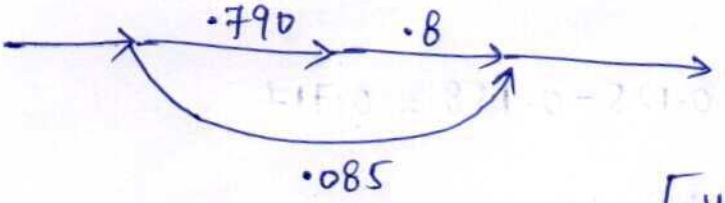
$.85 \times .9 \times .4 = .306$

$.85 \times .1 = 0.085$



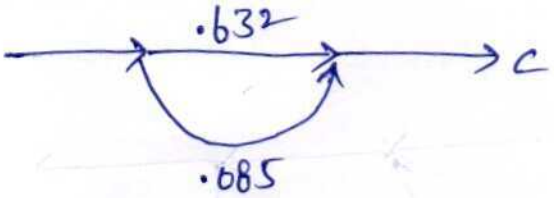
[·025 + ·459 = 0·484]

[·484 + ·306 = ·790]



[·484 + 0·306 = ·790]

[·790 × 0·8 = 0·632]



[·632 + ·085 = 0·717]

= 0·717