1)

TARJAN'S Algorithm

It is used to find articulation point in a graph.

Steps

1) Find DFS of G and assign discovery number to every vertex.

2) Find low() value for every vertex.

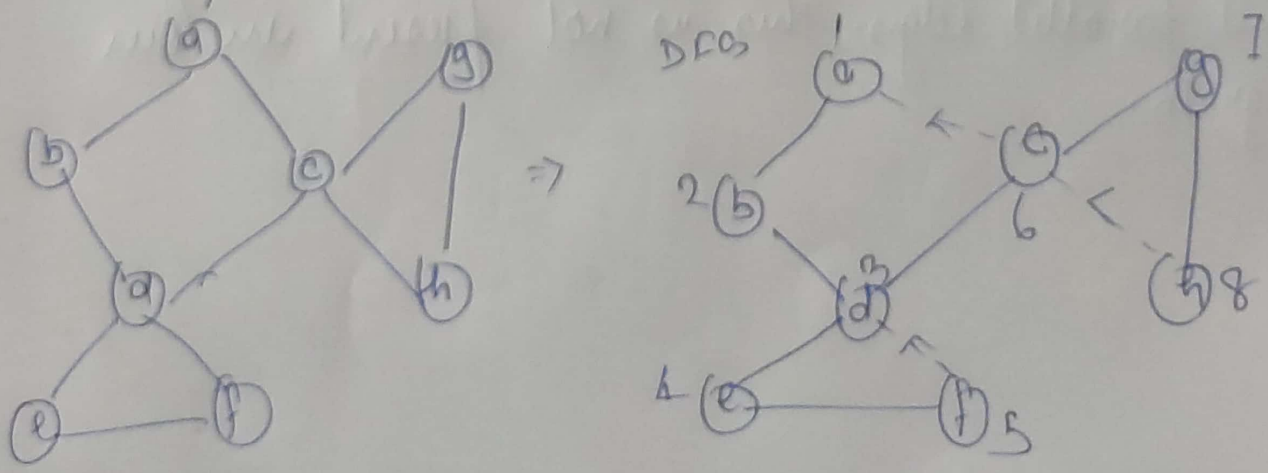
low() is the lowest discovery that can be reached from a vertex by taking one back edge.

If DisN(u) ≤ low(v) u → parent, v → child

Some vertices can be exempted from the above check if they satisfy any of the following condition.

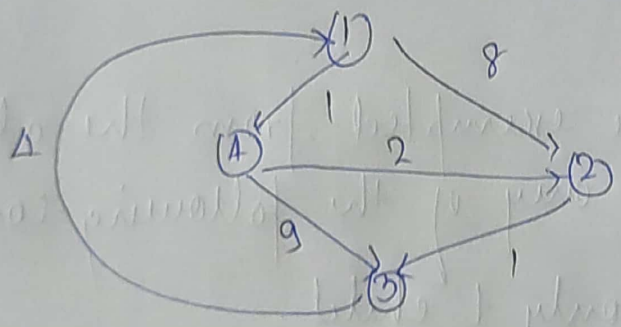
1) If the root has only 1 child.

2) Leaf vertices can be avoided.



	h	g	c	f	e	d	b	a
DISN ⇒	8	7	6	5	4	3	2	1
low() ⇒	6	6	1	3	3	1	1	1

2)
```
for(k=1; k<n; k++)
{
  for(i=1; i<n; i++)
  {
    for(j=1; j<n; j++)
    {
      A[i,j] = min[A[i,j], A[i,k]+A[k,j]]
    }
  }
}
```



Step-1 First of all we will remove all the self loops and parallel edges. This is not present in our graph.

Step-2 initial distance matrix

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix} \end{array}$$

$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{array}$$

$$A^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 6 \end{bmatrix} \end{array}$$

3) (i) Dynamic programming :- This method guarantees to find the best answer for TSP. However, its

3) Strategies for solving travelling salesmen problem :

i) Naive solution

1) Consider city 1 as the starting and ending point.
2) Generate all (n-1) permutation of cities.
3) Calculate cost of every permutation and keep track of minimum cost permutation.
4) Return the permutation with minimum cost.

ii) Dynamic programming

S : Subset of the graph which is not yet traversed
dist (i,1) : distance from i to 1.

If size of S is 2, then S must be {1,i},
c(S,i) = dist (1,i)

Else if size of S is greater than 2
c(S,i) = min { c(S-{i},i) + dis (i,i)} where
i belongs to S, j != i
and j != 1.

iii) Backtracking :

Consider city 1 (let say 0th node) as the starting and ending point. Since route is cyclic, we can consider any point as starting point.

Start traversing from the source to its adjacent nodes in dfs manner.

Calculate cost of every traversal and keep track of minimum cost and keep on updating values of minimum cost stored value.

Return the permutation with minimum cost.