/* A global Array, to indicate the color of a node

/* A global Array, reserving a 'Pre'- Label for each of n nodes

/* A global Array, reserving an 'InA'- Label for each of n nodes

/* A global Array, reserving a 'Post'- Label for each of n nodes

/* A global double Array, reserving an 'Edge-Label' for each of m edges

raph, **node** *start*)

HITE //

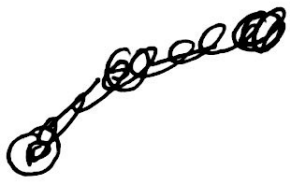
hanumerically **do**

t]

	Pre-Label	*In-Label*	*Post-Label*
S	1		14
A	5	4	12
B	6	5	11
C	2	1	3
D	4	1	13
E	7	6	10
F	8	7	9
G			
H			

(Please show the edge-counts on the graph itself)

DFS HW

② 

	color	Pre	In	Post
S	B	1		14
A	B	5		12
B	B	6		11
C	B	2		3
D	B	4		13
E	B	7		10
F	B	8		9
G	B			
H				

⑤

	Pre	In	Post	color
S	1		14	B
A	5		12	B
B	6		11	B
C	2		3	B
D	4		13	B
E	7		10	B
F	8		9	B
G				
H				

	Pre	In	PA
S	1	Ø X Z B 4 5 6	14
A	5	Ø X 2 B 4	12
B	6	Ø X 2 B 4	11
C	2	Ø X 2	3
D	4	Ø X 2 B 4 5 6	13
E	7	Ø X 2 3 4 5 6 7 8	10
F	8	Ø X 2	9

BFS SUGGESTED PROBLEMS

1. This is the first one in a number of exercises that will help in understanding the BFS algorithm. In all cases, you are asked to write the resulting values for the labels (the **Pre-**, **In-**, and **Post-**) for all the nodes of the graph[3] given, as well as those for the edges. There may be an algorithm that determines the BFS-Spanning-Tree in the correct order, another determines the visit-order for all edges, several others have no traditional or obvious practical applications, other than educational. Thus, you are asked to find any practical interpretations or relationship(s), if any, between the final values of the labels, and n (number of nodes) and m (number of edges). Note: the various labels are initially set to 0, and are overwritten zero or more times, depending on the number of neighboring nodes. Not all Labels are used in all exercises.

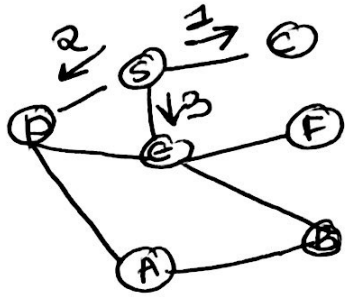
```
LINE 100: Algorithm void BFSNums (graph MyGraph, node start)
LINE 110:   int CFL ← 0,                          /* global variable to Count First & Last
LINE 120:   int Ced ← 0,                          /* global variable Count edges
LINE 130:   queue MyQueue.new;                    /* A Queue to impose an ordering
LINE 140:   int Color[n] ← {WHITE};               /* A global Array, to indicate the color of a node
LINE 150:   int Pre[n] ← {0};                     /* A global Array, reserving a 'Pre'- Label for each of n nodes
LINE 160:   int In[n] ← {0};                      /* A global Array, reserving an 'InA'- Label for each of n nodes
LINE 170:   int Dist[n] ← {∞};                    /* A global Array to record the 'Distance'- Label for each of n nodes
LINE 180:   int From[n] ← {NILL};                 /* A global Array to record the 'Predecessor'- Label for each of n nodes
LINE 190:   int Post[n] ← {0};                    /* A global Array, reserving a 'Post'- Label for each of n nodes
LINE 200:   int Edge[u, v] ← {0};                 /* A global double Array, reserving an 'Edge-Label' for each of m edges

LINE 300: Color[start] ← GRAY
LINE 310: Pre[start] ← ++CFL
LINE 320: Dist[start] ← 0
LINE 330: MyQueue.Insert(start)
LINE 400: while NOT MyQueue.IsEmpty do
LINE 410:     v ← MyQueue.Delete
LINE 420:     In[v] ← ++CFL
LINE 430:     for all w adjacent to v and selected alphanumerically do
LINE 440:         Edge[v, w] ← ++Ced
LINE 450:         if Color[w] == WHITE then
LINE 460:             Color[w] ← GRAY
LINE 470:             Pre[w] ← ++CFL
LINE 480:             Dist[w] ← Dist-Label[v] + 1
LINE 490:             From[w] ← v
LINE 500:             MyQueue.Insert(w)
LINE 510:         end-if Color[w]
LINE 520:     end-for-all
LINE 530:     Color[v] ← BLACK
LINE 540:     Post[v] ← ++CFL
LINE 550: end-while not MyQueue.IsEmpty

LINE 560: end-algorithm
```

	Color	*Pre*	*In*	*Dist*	*From*	*Post*
S	B	1	12	0	Nill.	6
A	B	10	16	2	D	17
B	B	13	18	2	E	19
C	B	3	7	1	S	8
D	B	4	9	1	S	11
E	B	5	12	1	S	15
F	B	14	20	2	E	21
G						
H						

BFS Homework

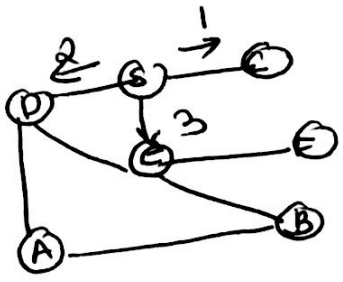
③

	Color	Pre	In	Dist	From	Post
S	~~W~~ B	1	2	0		3
A						
B						
C						
D						
E						
F						
G						
H						



④

	Color	Pre	In	Dist	From	Post
S	~~W~~ B	1	2	0		3
A						
B						
C						
D						
E						
F						
G						
H						



(6)

	Color	Pre	In	Dis	From	Post
S	G B	1				
A						
B						
C	G	3		1	5	
D	G	4		1	5	
E	G	5		1	5	
F						
G						
H						

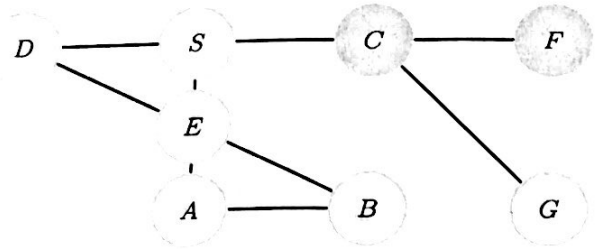
Topological sort

(b,f) = 5 , (8,K) = 3 and (K,O) = 4

	EC	LC
	0	0
a	0	0
b	0	0
c	7	7
d	5	6
e	5	5
f	5	6
g	4	10
h	10	9
i	8	8
J	8	9
K	1	6
l	6	14
m	14	12
n	12	10.
o	10	
p		

Homework: Understanding the Articulation Node detection

1. You are asked to write the resulting values for the labels as indicated for the graph as given, or for the graph that you yourself should create. Make sure it has a cycles as well as some 'dangling' legs.



	Children	*CameFrom*	*Num*	*Low*	*ArtPoint*
S	2	-	1	1	T
A	1	E	7	7	T
B	0	A	8	8	F
C	2	S	2	2	T
D	1	S	5	5	F
E	1	D	6	6	T
F	0	C	3	2	F
G	0	C	4	4	F