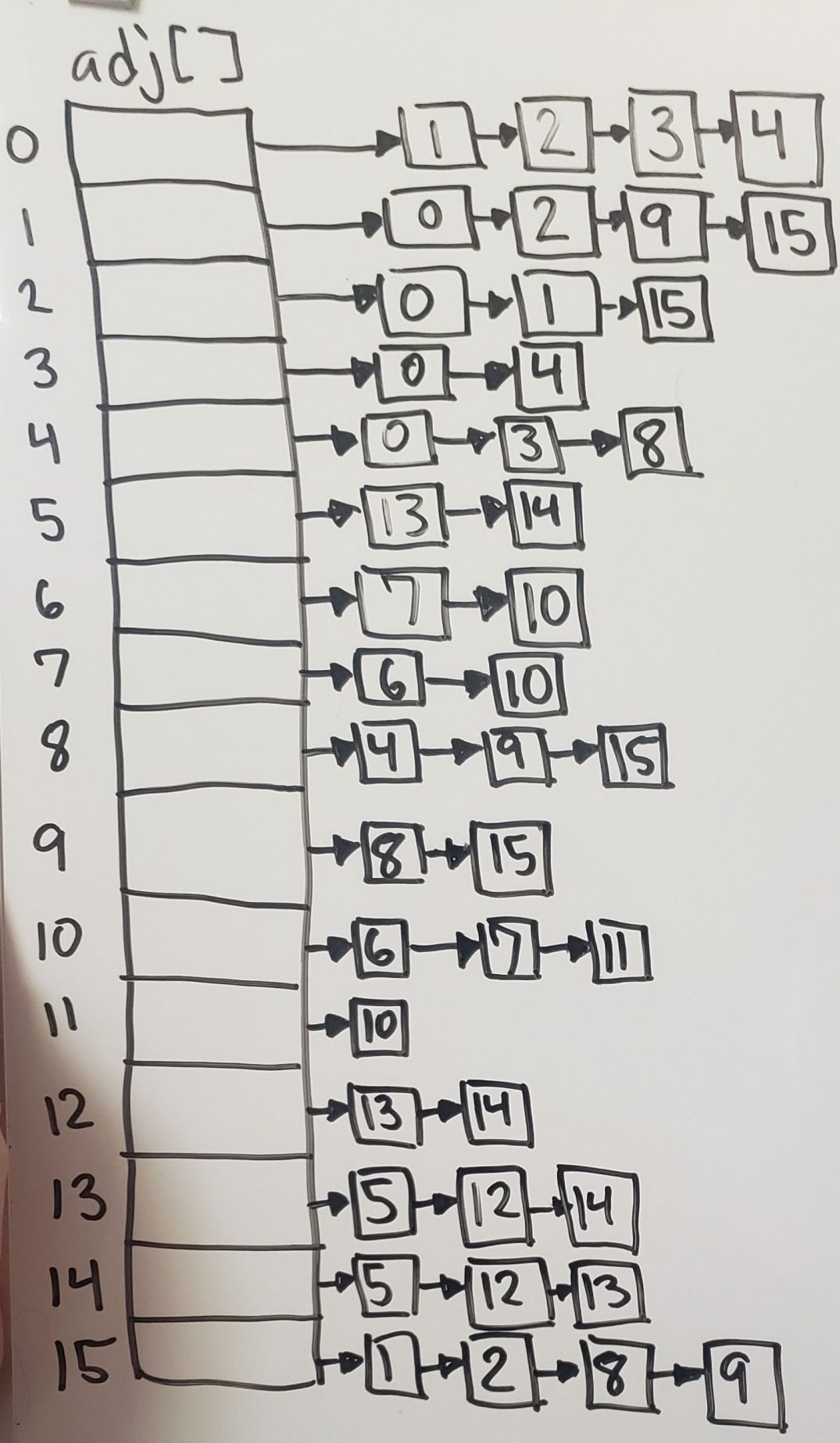
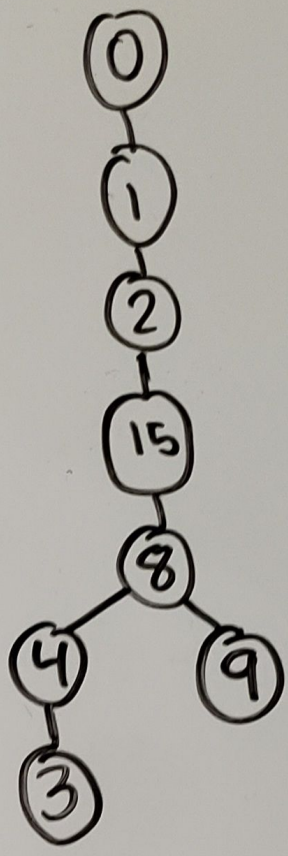
Undirected Graph
#1

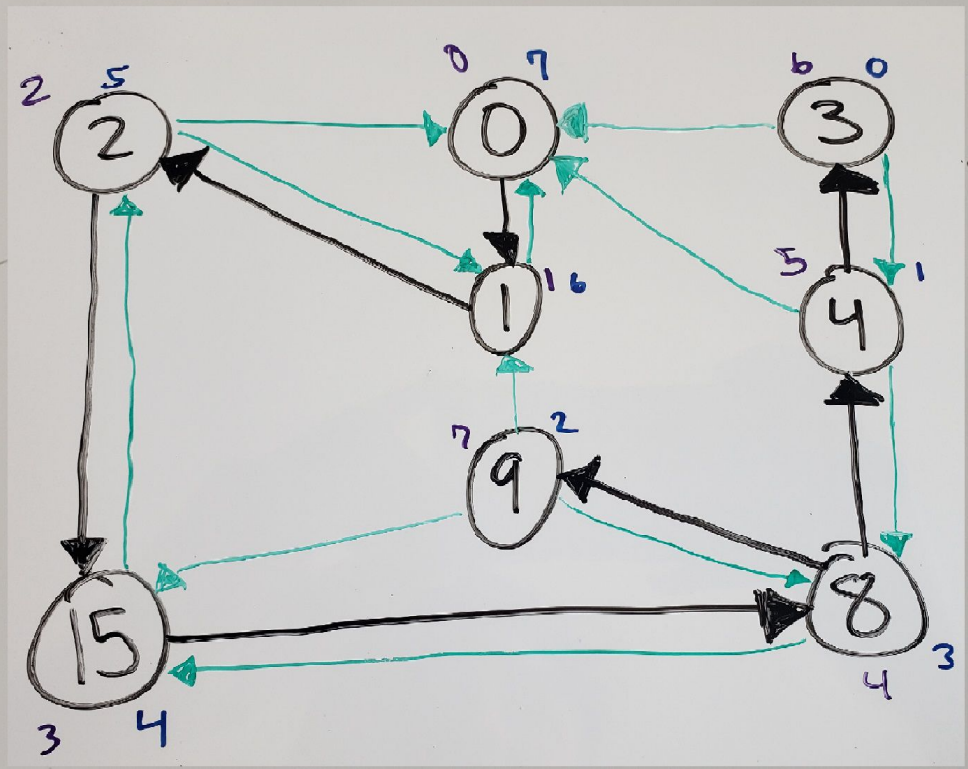


Undirected Graphs #2

```
dfs(0)
  dfs(1)
    check 0
    dfs(2)
      check 0
      check 1
      dfs(15)
        check 1
        check 2
        dfs(8)
          dfs(4)
            check 0
            dfs(3)
              check 0
              check 4
            3 done
            check 8
          4 done
          dfs(9)
            check 1
            check 8
            check 15
          9 done
          check 15
        8 done
        check 9
      15 done
    2 done
    check 9
    check 15
  1 done
  check 2
  check 3
  check 4
0 done
```

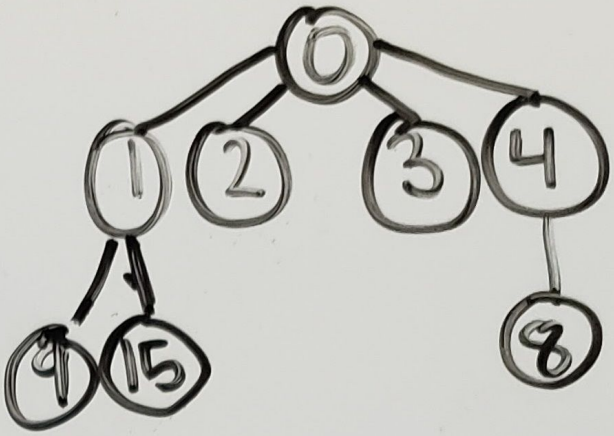
V	marked[]	edgeTo[]
0	T	—
1	T	0
2	T	1
3	T	4
4	T	8
5	F	
6	F	
7	F	
8	T	15
9	T	8
10	F	
11	F	
12	F	
13	F	
14	F	
15	T	2

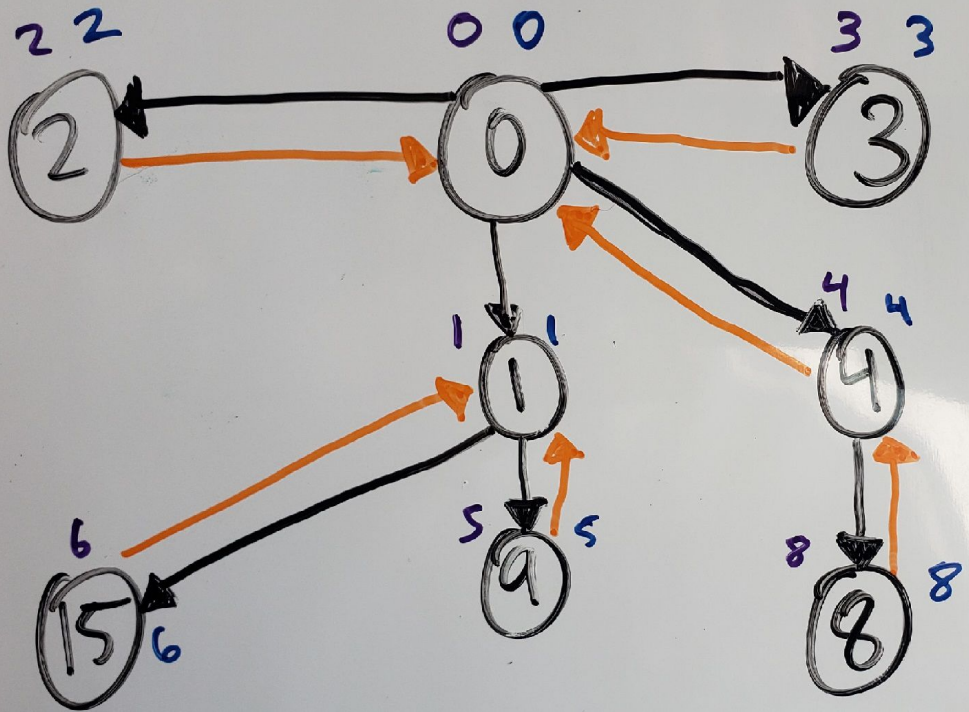




Undirected Graphs #3

V	marked[]	edgeTo[]
0	T	—
1	T	0
2	T	0
3	T	0
4	T	0
5		
6		
7		
8	T	4
9	T	1
10		
11		
12		
13		
14		
15	T	1





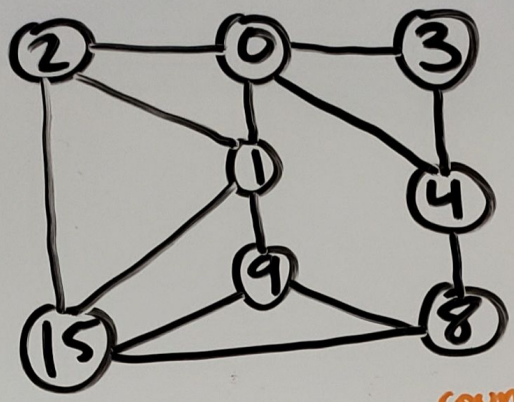
Tree Edge

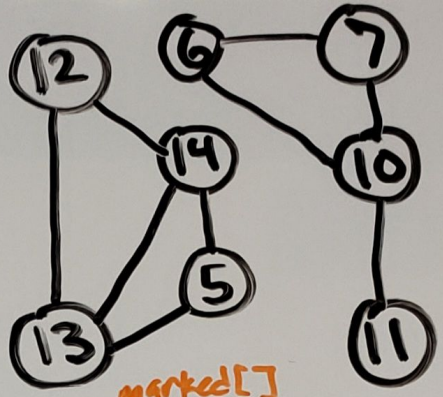
Back Edge

pre number

post number

Undirected Graphs
#4

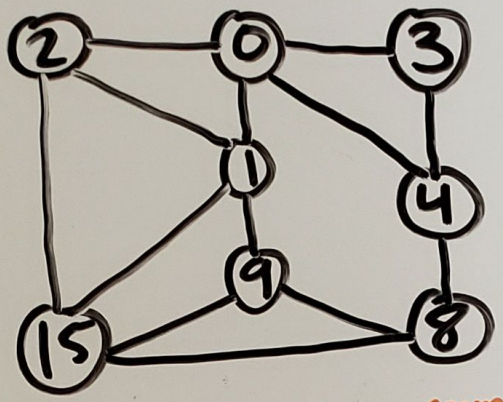


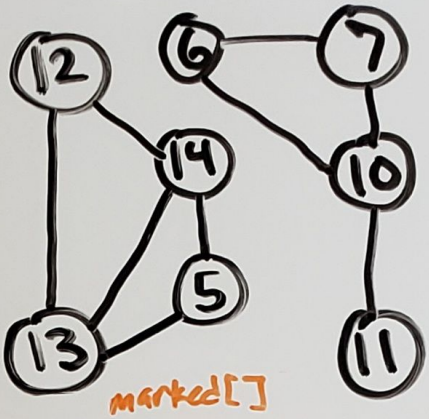



```
dfs(0)
  dfs(1)
    check 0
    dfs(2)
      check 0
      check 1
      dfs(15)
        check 1
        check 2
        dfs(8)
          dfs(4)
            check 0
            dfs(3)
              check 0
              check 4
            3 done
            check 8
          4 done
          dfs(9)
            check 1
            check 8
            check 15
          9 done
          check 15
        8 done
        check 9
      15 done
    2 done
    check 9
    check 15
  1 done
  check 2
  check 3
  check 4
0 done
```

count	marked[] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	id[] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0	T	0
0	TT	00
0	TTT	000
0	TTT	T000
0	TTT T	T000 0
0	TTT T T	T000 0 0
0	TTTT T T	T0000 0
0	TTTTT TT	T0000 00

Undirected Graphs #4

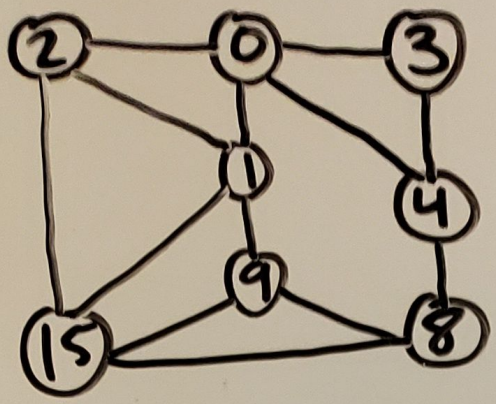


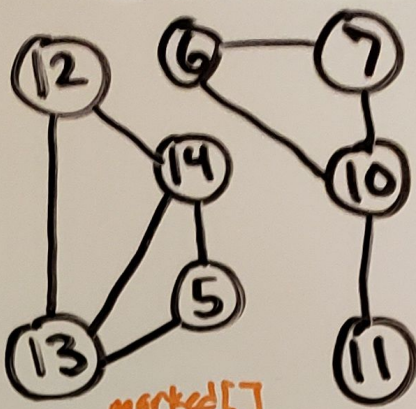



```
dfs(5)
  dfs(13)
    check 5
    dfs(12)
      check 13
      dfs(14)
        check 5
        check 12
        check 13
      14 done
    12 done
    check 14
  13 done
  check 14
5 done
dfs(6)
  dfs(7)
    check 6
    dfs(10)
      check 6
      check 7
      dfs(11)
        check 10
      11 done
    10 done
  7 done
  check 10
6 done
```

count	marked[] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	id[] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1	T T T T T T	T T	T	0 0 0 0 0 1	0 0	0
1	T T T T T T	T T	T	0 0 0 0 0 1	0 0	1 0
1	T T T T T T	T T	T T	T	0 0 0 0 0 1	0 0	1 1	0
1	T T T T T T	T T	T T T T	0 0 0 0 0 1	1 1	1 1 1 0
2	T T T T T T T	T T	T T T T	0 0 0 0 0 1 2	1 1	1 1 1 0
2	T T T T T T T T T T	T T T	T T T T	0 0 0 0 0 1 2 2 1 1	1 1 1 0
2	T T T T T T T	T T T T	T T T T	0 0 0 0 0 1 2 2 1 1 2	1 1 1 0
2	T T T T T T T T T T T T T T T T	0 0 0 0 0 1 2 2 1 1 2 2 1 1 1 0

Undirected Graphs #5





marked[]

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

T

T T

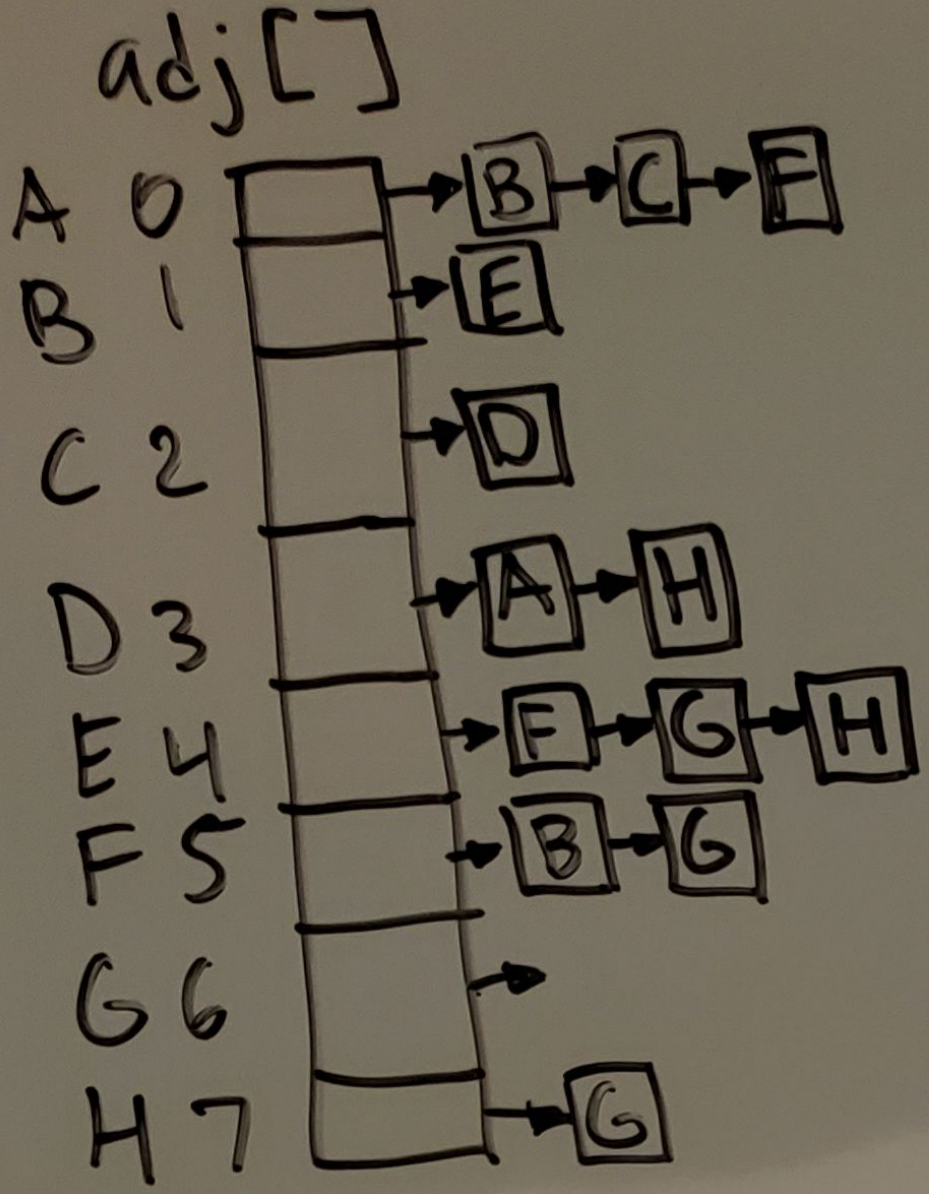
T T T

dfs(G,0,0)
dfs(G,1,0)
dfs(G,2,1)

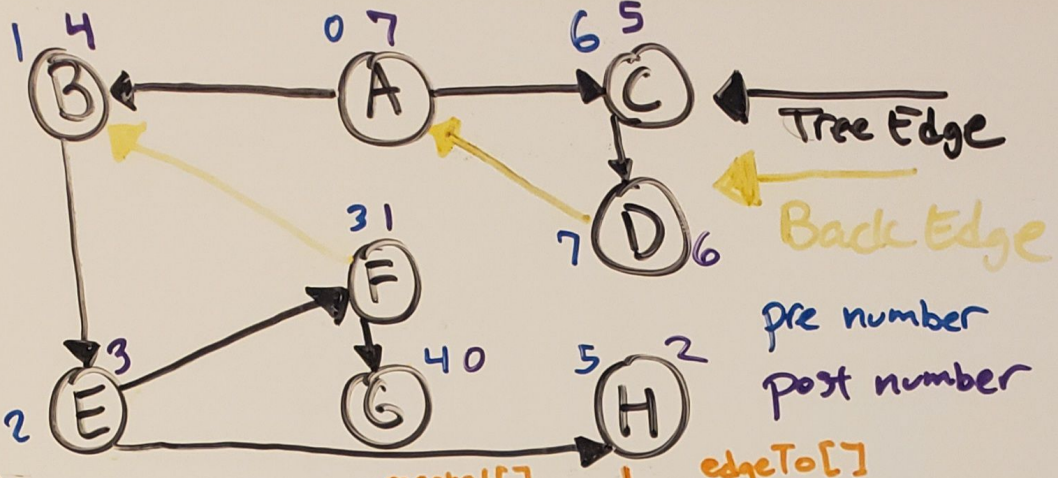
```
dfs(G,0,0)
    V=0
    U=0
    marked[0]=T
    W=1
      dfs(G,1,0)
        V=1
        U=0
        marked[1]=T
        W=0
        W=2
        dfs(G,2,1)
          V=2
          U=0
          marked[2]=T
          ...
          elseif(0!=1) hasCycle=True
```

Directed Graphs

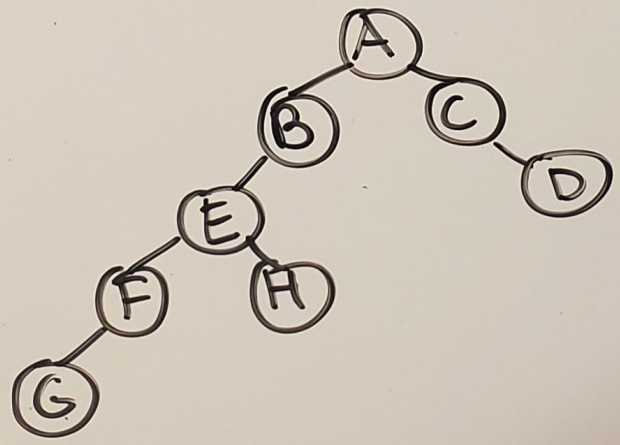
#1



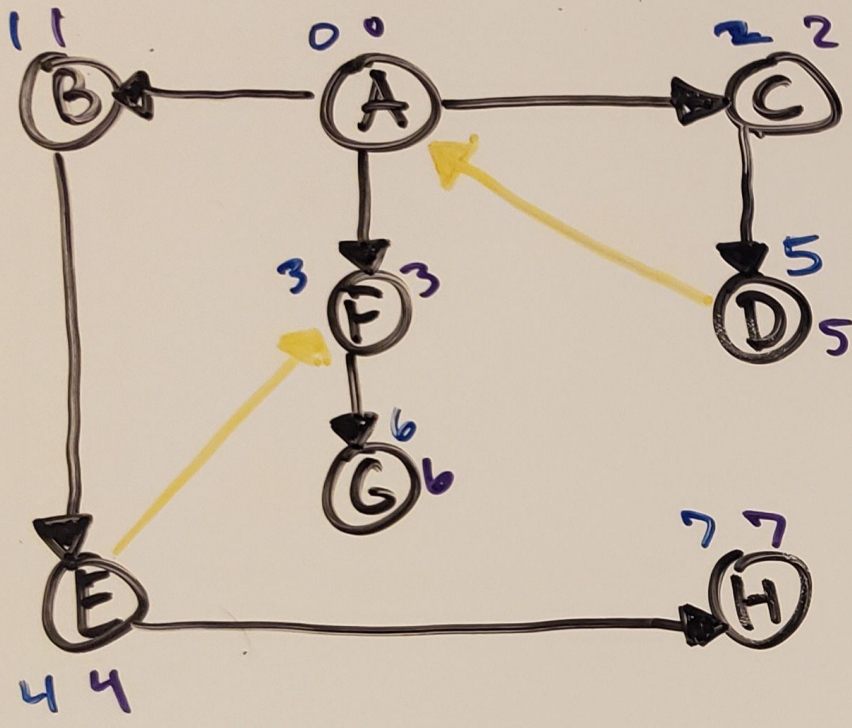
Directed Graphs #2



	marked[] 0 1 2 3 4 5 6 7	edgeTo[] 0 1 2 3 4 5 6 7
dfs(A)	T	-
dfs(B)	T T	- 0
dfs(E)	T T T	- 0 1
dfs(F)	T T T T	- 0 1 4
check B		
dfs(G)	T T T T T	- 0 1 4 5
G done		
F done		
check G		
dfs(H)	T T T T T T	- 0 1 4 5 6
check G		
H done		
E done		
B done	T T T T T T	- 0 0 1 4 5 6
dfs(C)	T T T T T T T	- 0 0 2 1 4 5 6
dfs(D)		
check A		
check H		
D done		
C done		
A done		



Directed Graphs #3

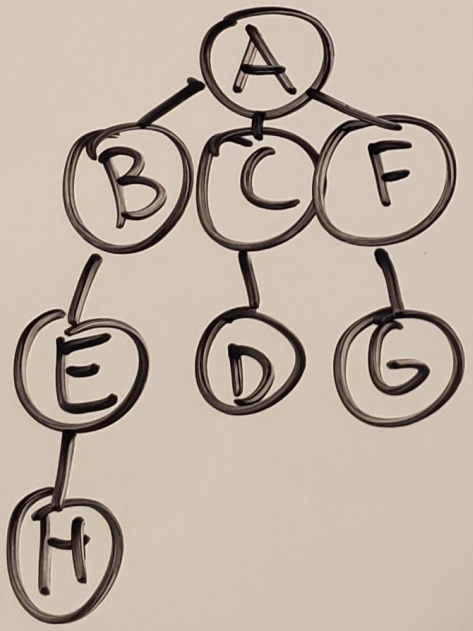


Tree Edge

Back Edge

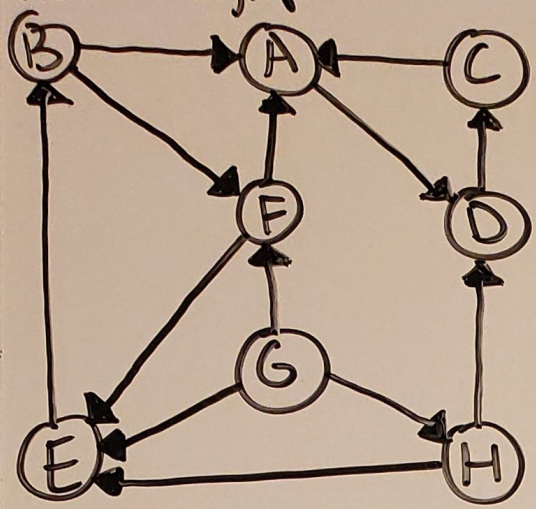
pre number

post number



Directed Graphs
#4

Reverse Digraph



dfs(A)
- dfs(D)
  - dfs(C)
    - checkA
    - C done
  - D done
- A done

dfs(B)
- checkA
- dfs(F)
  - checkA
  - dfs(E)
    - checkB
    - E done
  - F done

B done

check C
check D
check E
check F

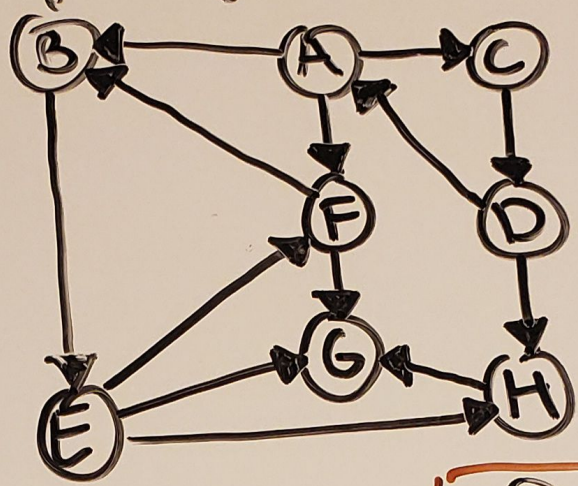
dfs(G)
- check E
- check F
- dfs(H)
  - check D
  - check E
- H done

G done

Reverse Postorder:

G H B F E A D C

Original Digraph



dfs(G)
- G done

dfs(H)
- checkG

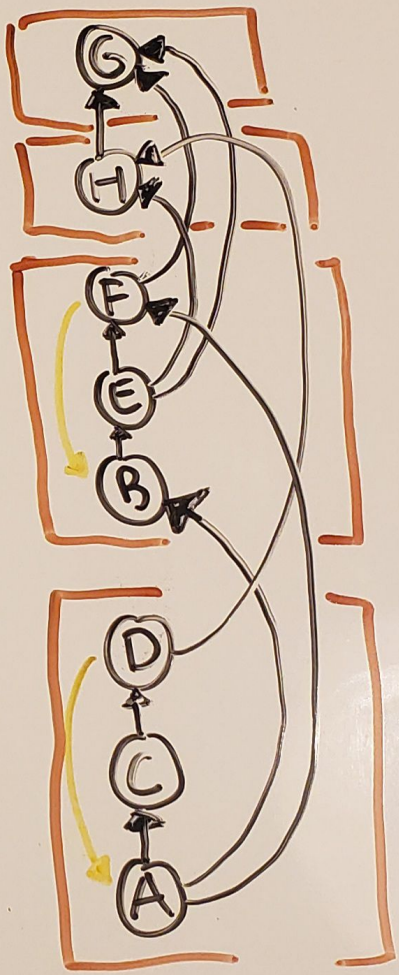
H done

dfs(B)
- dfs(E)
  - dfs(F)
    - checkB
    - checkG
  - F done
  - checkG
  - checkH
- E done

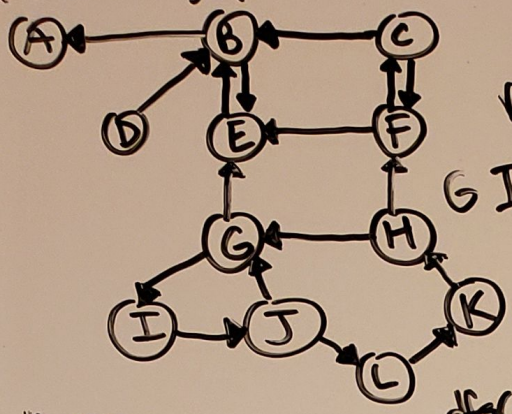
B done
checkF
checkE

dfs(A)
- checkB
- dfs(C)
  - dfs(D)
    - checkA
    - checkH
  - D done
  - C done
  - checkF

A done



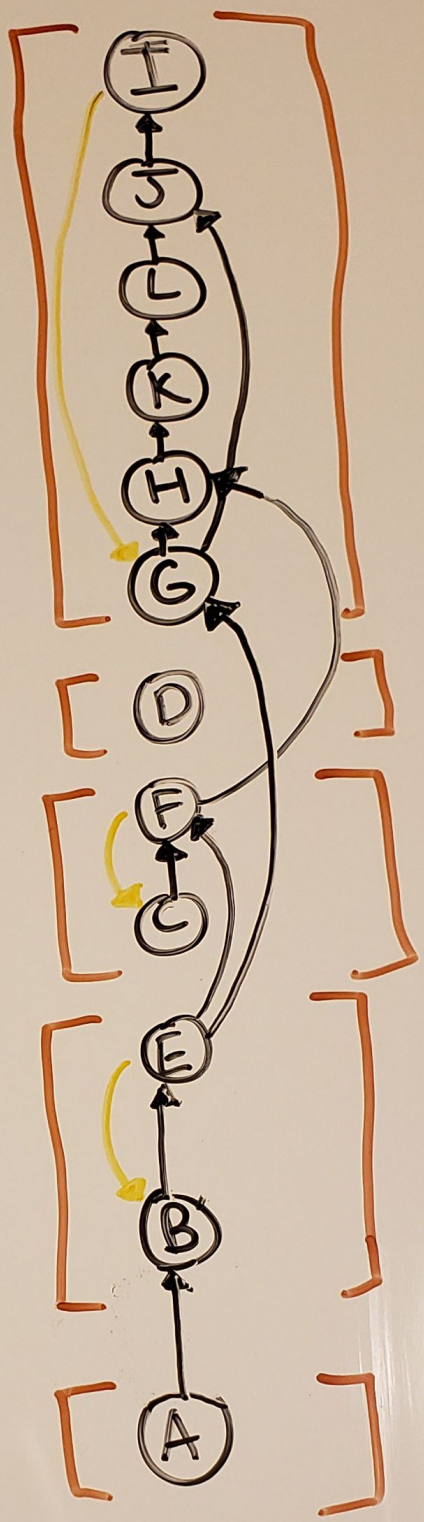
Reversed Digraph



Strongly Connected Components

Reverse Postorder

G I J L K H D C F B E A



dfs(A)
A done
dfs(B)
| dfs(E)
| | check B
| E done
B done
dfs(C)
| check B
| dfs(F)
| | check C
| | check E
| F done
C done
dfs(D)
| check B
D done
check E
check F
dfs(G)
| check E
| dfs(I)
| | dfs(J)
| | | check G
| | | dfs(L)
| | | | dfs(K)
| | | | | dfs(H)
| | | | | | check F
| | | | | | check G
| | | | | H done
| | | | K done
| | | L done
| | J done
| I done
G done
check H
check I
check J
check K
check L

dfs(G)
| dfs(H)
| | dfs(K)
| | | dfs(L)
| | | | dfs(J)
| | | | | dfs(I)
| | | | | | check G
| | | | | I done
| | | | J done
| | | L done
| | K done
| H done
| check J
G done
check I
check J
check L
check K
check H
dfs(D)
D done
dfs(C)
| dfs(F)
| | check C
| | check H
| F done
C done
check F
dfs(B)
| check C
| dfs(E)
| | check B
| | check F
| | check G
| E done
B done
check E
dfs(A)
| check B
A done