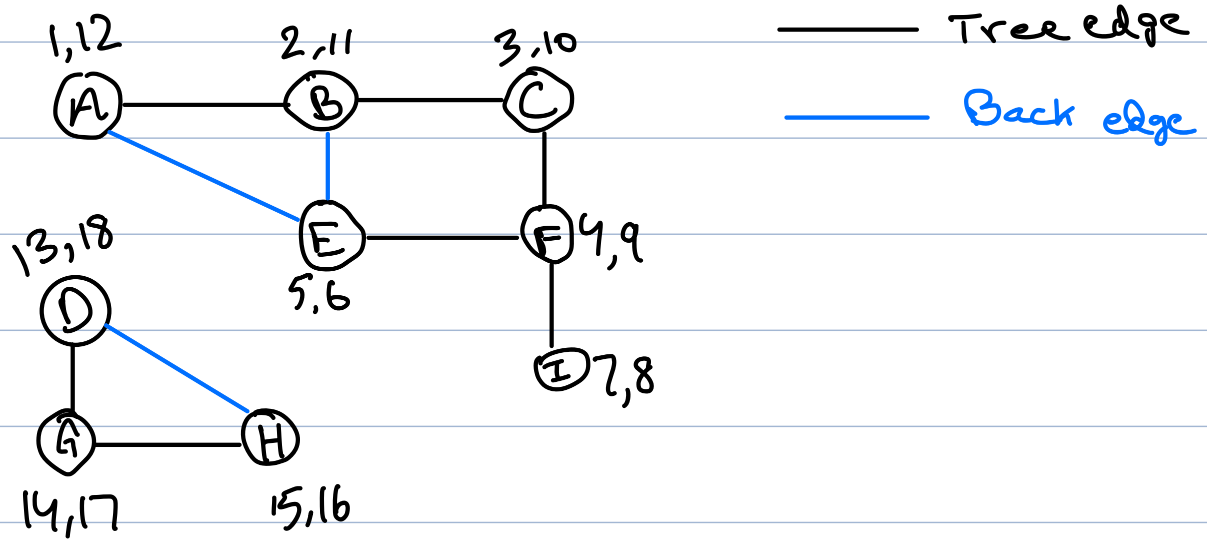
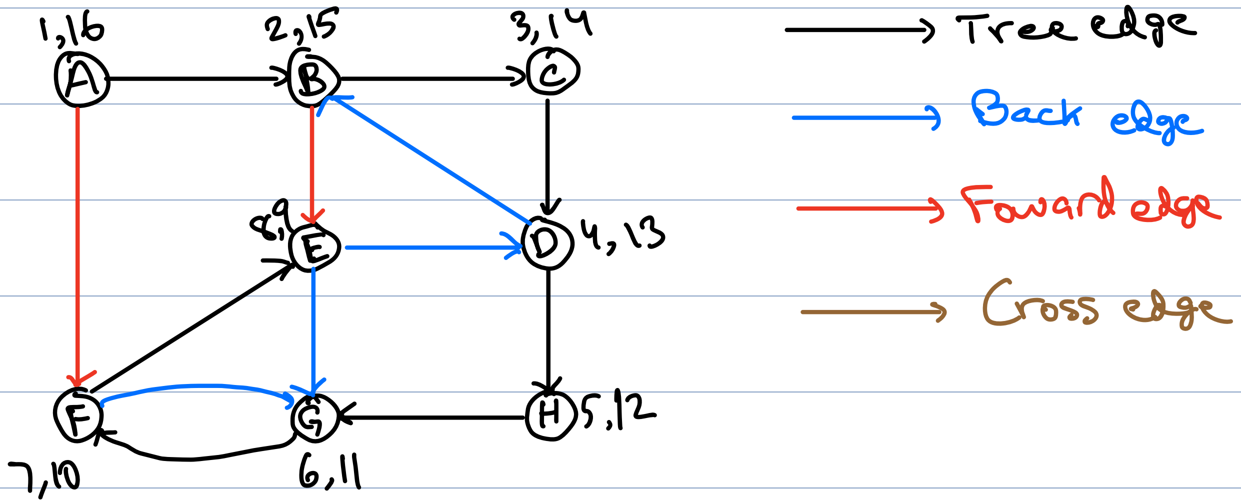
3.1



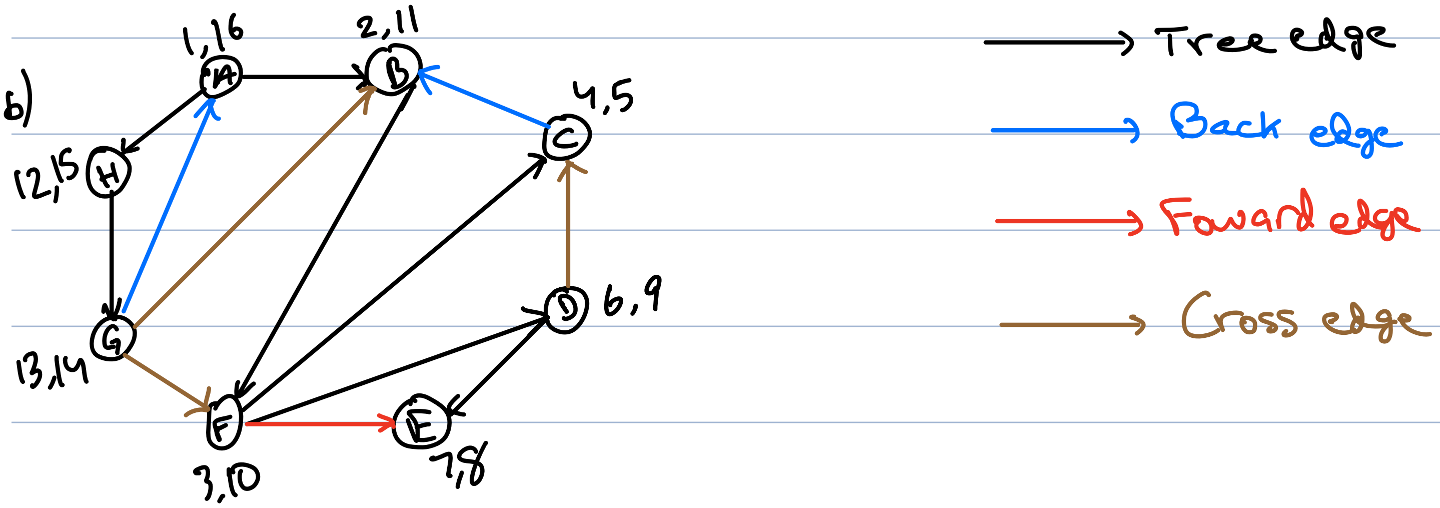
1,12
A
2,11
B
3,10
C
13,18
D
E
5,6
F
4,9
I
7,8
G
H
14,17
15,16
Tree edge
Back edge

3.2 a)



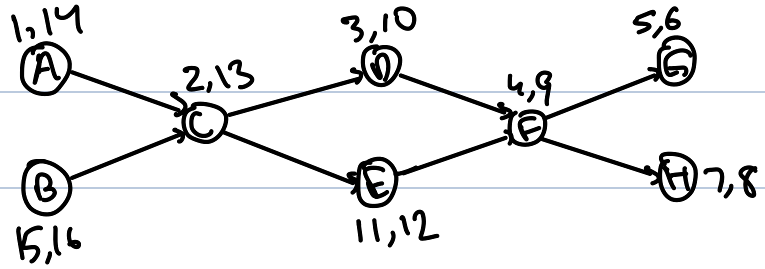
1,16
A
2,15
B
3,14
C
8,9
E
D
4,13
F
G
H
5,12
7,10
6,11
Tree edge
Back edge
Foward edge
Cross edge

b)



1,16
A
2,11
B
4,5
C
12,15
H
G
13,14
D
6,9
F
E
3,10
7,8
Tree edge
Back edge
Foward edge
Cross edge

3.3 a)



b) Sources: A and B

Sinks: G and H

c) B, A, C, E, D, F, H, G

d) The graph must be in any order of the following form:

{A, B}, C, {D, E}, F, {G, H}

The vertices inside the brackets can be swapped

Since there are three sets of vertices that can be swapped which loads to two combinations of each, there are $\boxed{8}$ total possible orderings

3.5 function reverse(G)

```
Input: A directed graph $G = (V, E)$ in adjacency list format
Output: Rererse of Graph
Create the graph $G^R = (V, E^R)$ with edge-set $E^R$
for each $u \in V$:
    for each $u, w \in E$:
        add edge $(w, u)$ to $E^R$
    return $G^R$
```

Time complexity: $O(V+E)$ where V is the number of vertices and E is the number of edges in G

3.9

```
function twodegree(adj_list)
Input: An adjacency list of the undirected graph
Output: Array containing the twodegree values for each node
n = length(adj_list)   //Number of nodes
twodegree = array with size n, initialized to all zeros
for u=0 to n-1:
    for each v in adj_list[u]:
        twodegree[u] += length(adj_list[v])
return twodegree
```