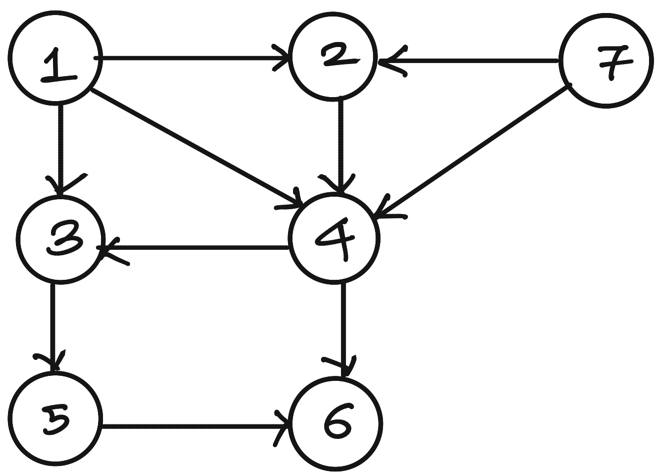
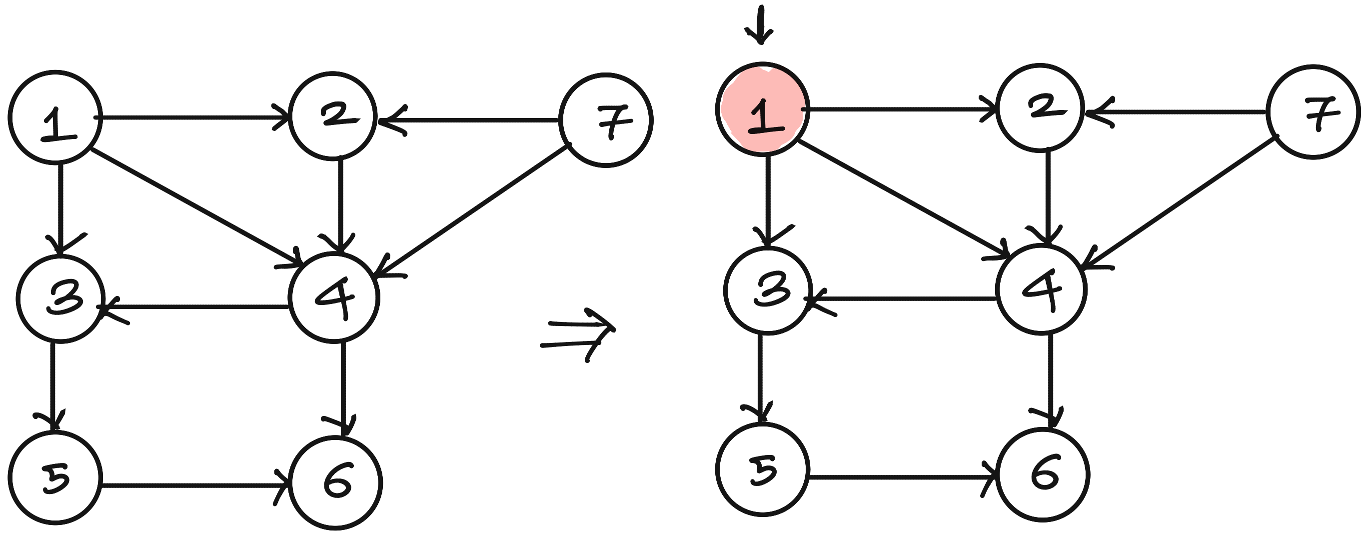
ANOTHER GRAPH TRAVERSAL TECHNIQUE

DEPTH FIRST SEARCH.



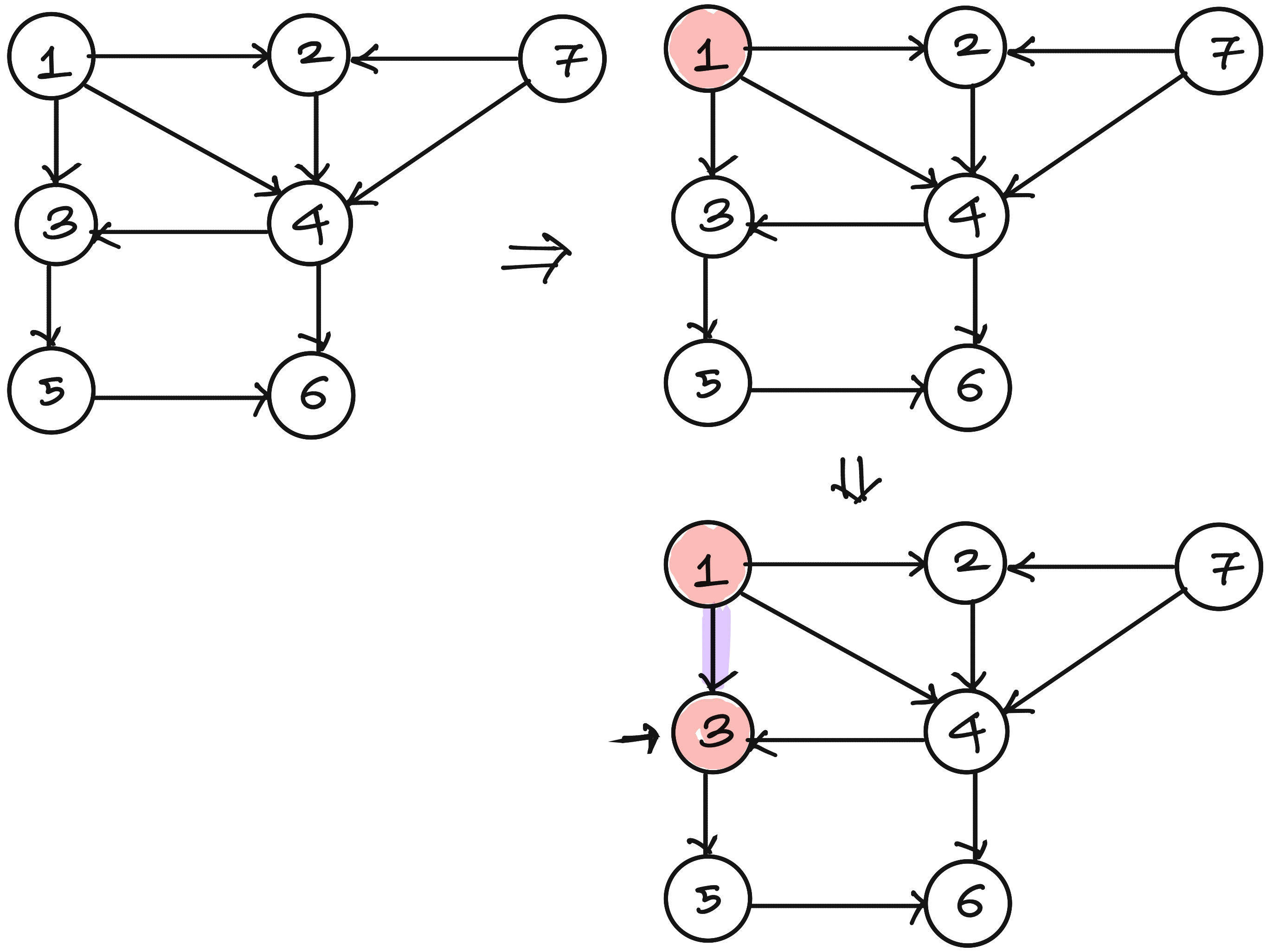
ANOTHER GRAPH TRAVERSAL TECHNIQUE

DEPTH FIRST SEARCH.



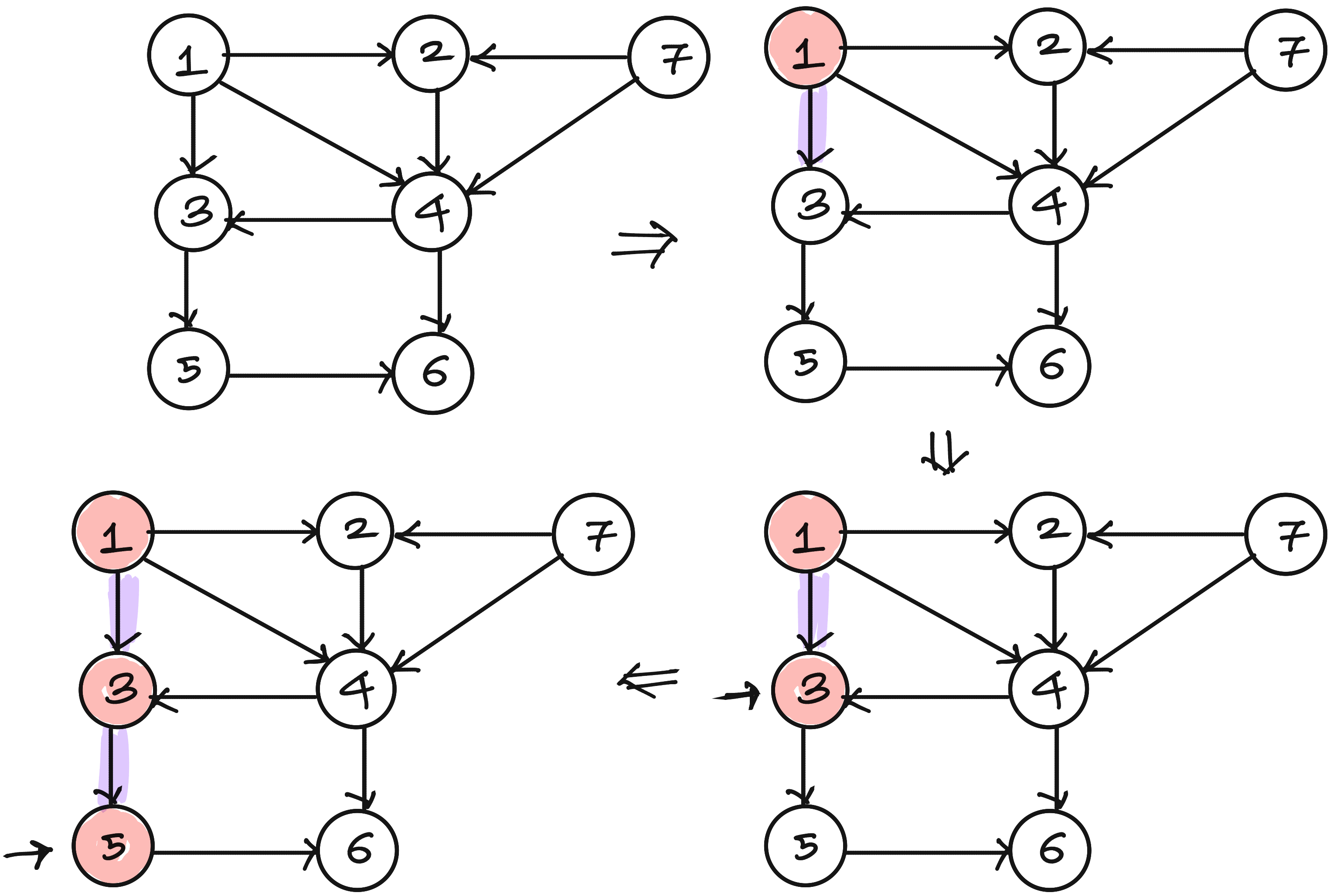
Another Graph Traversal Technique

Depth First Search.



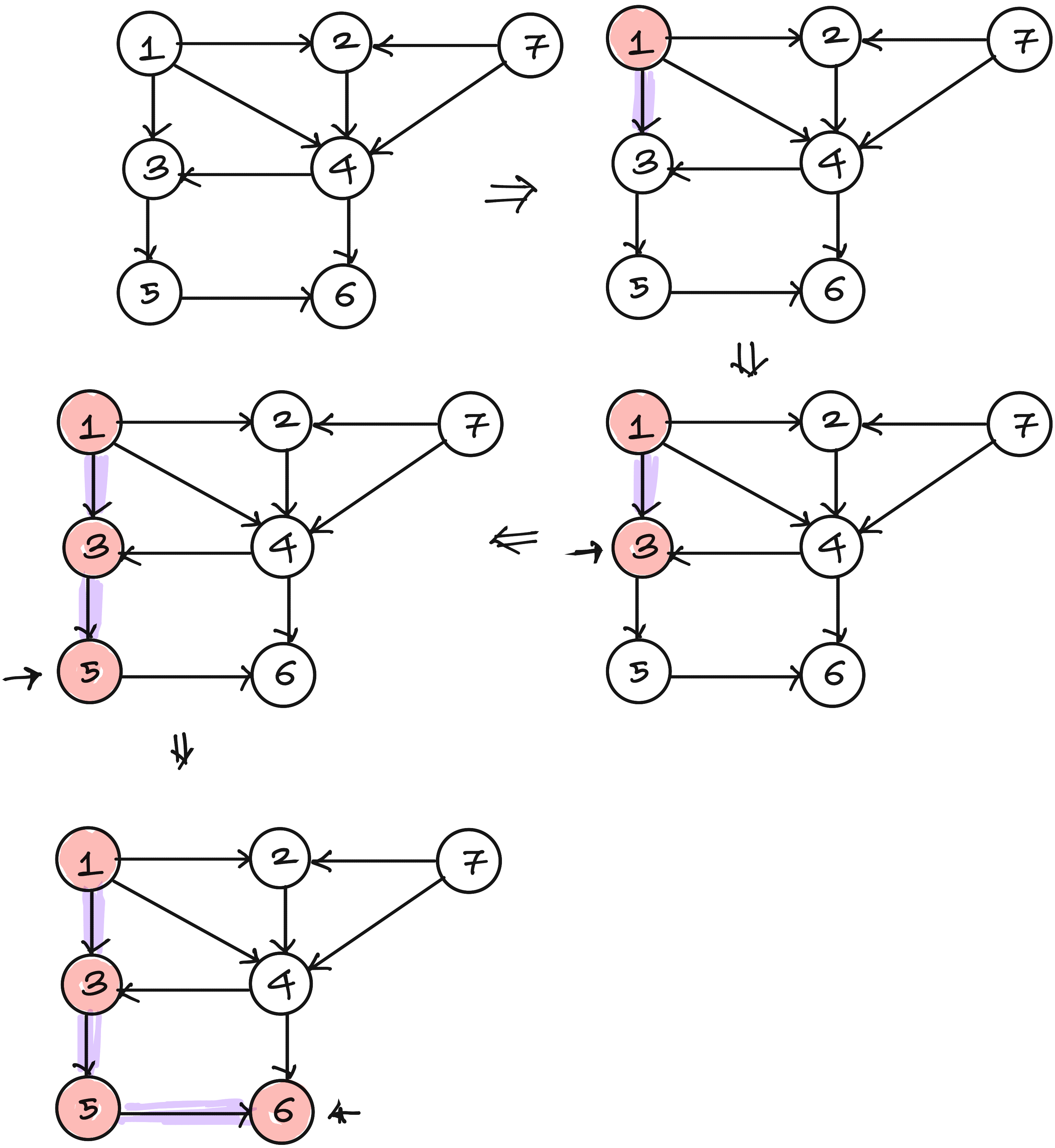
ANOTHER GRAPH TRAVERSAL TECHNIQUE

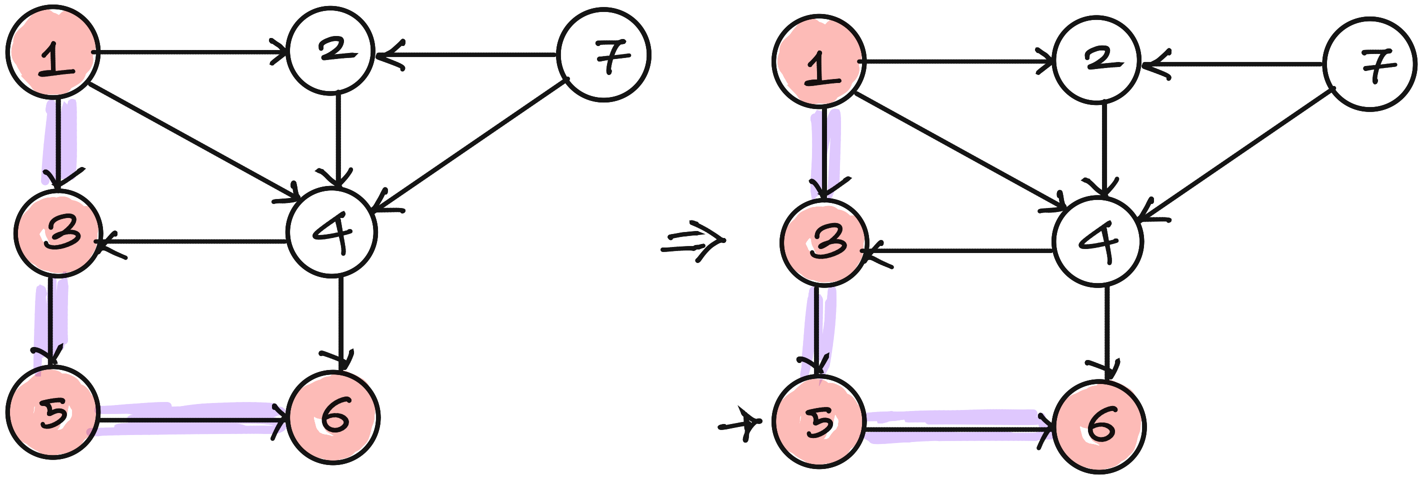
DEPTH FIRST SEARCH.

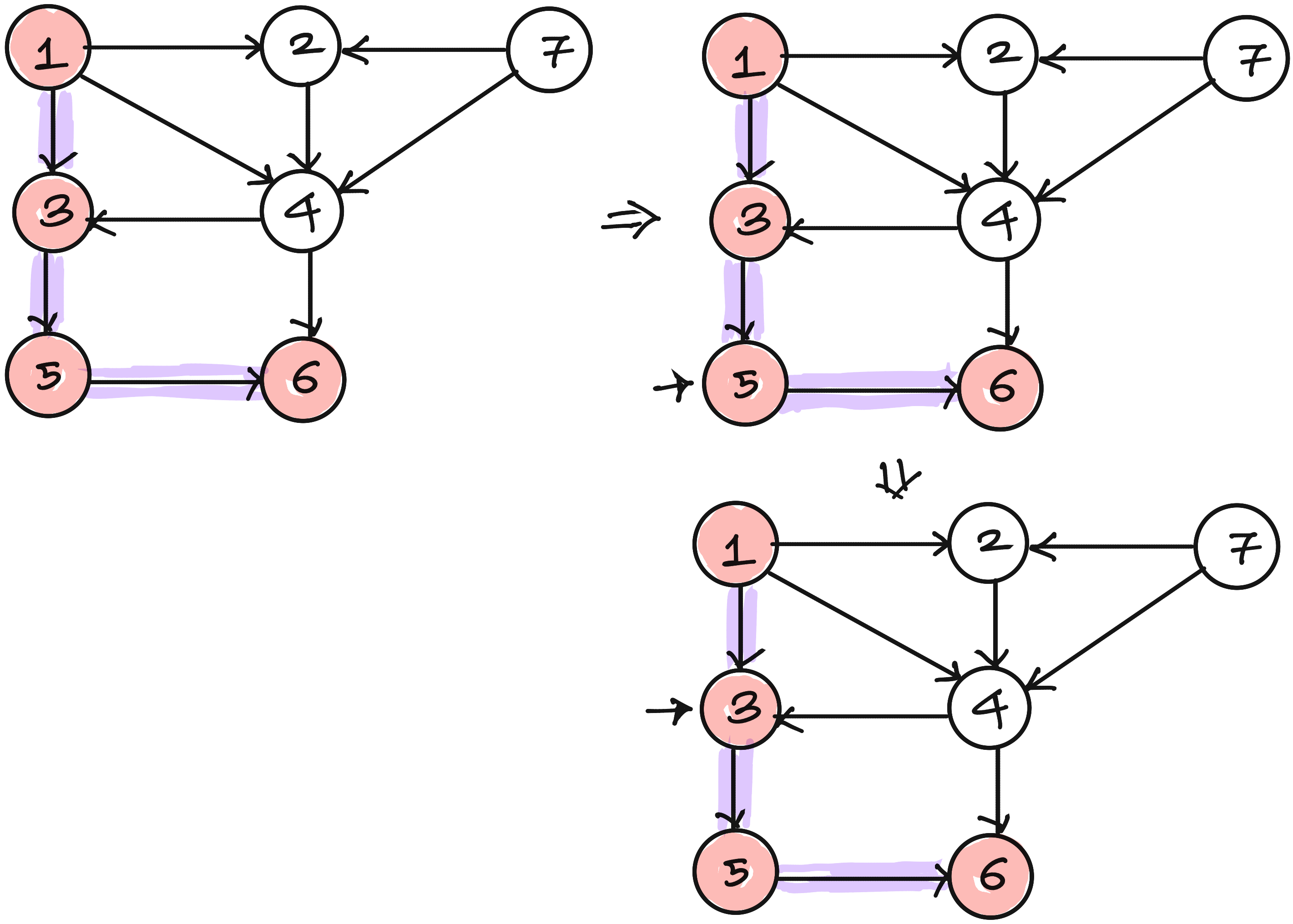


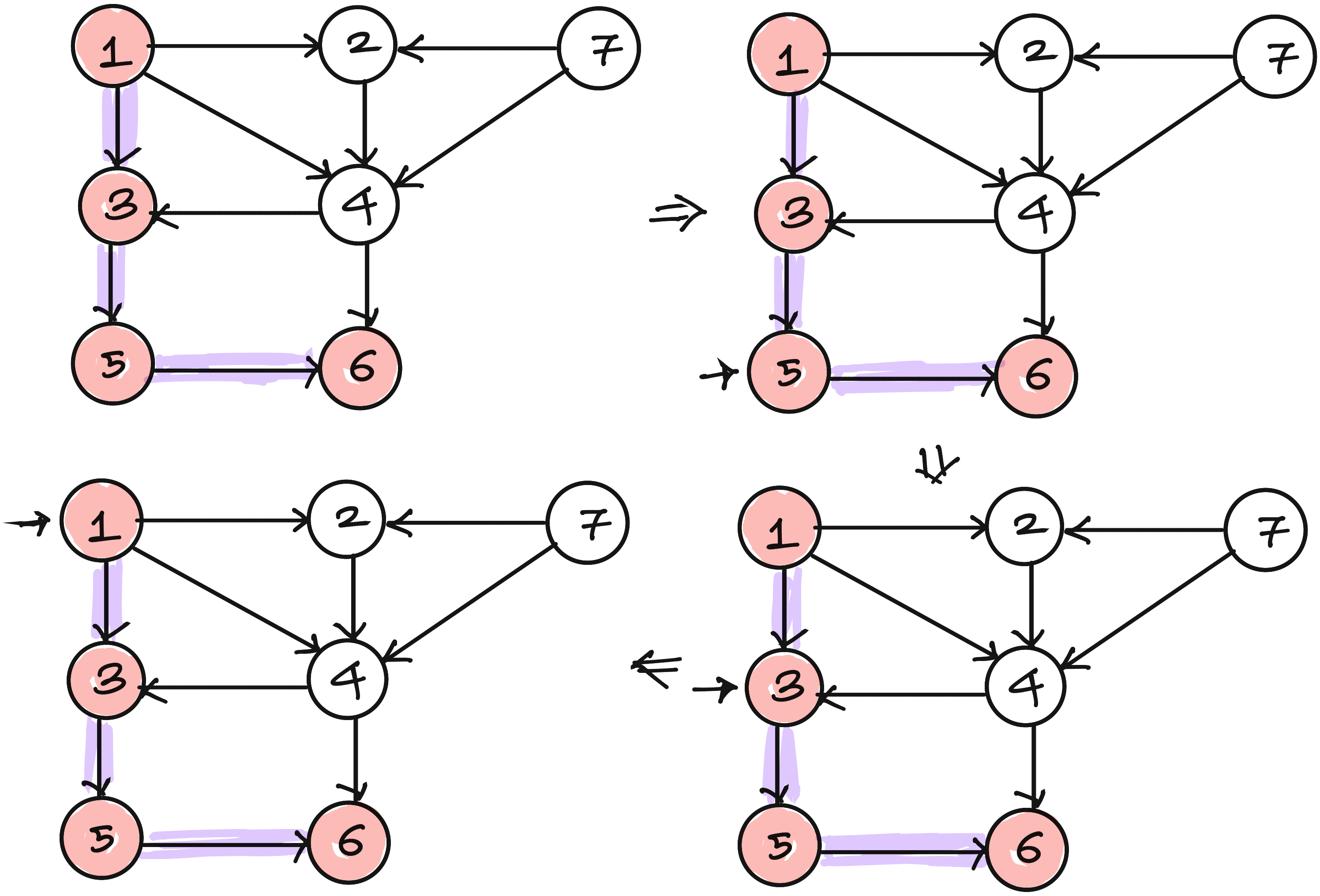
ANOTHER GRAPH TRAVERSAL TECHNIQUE

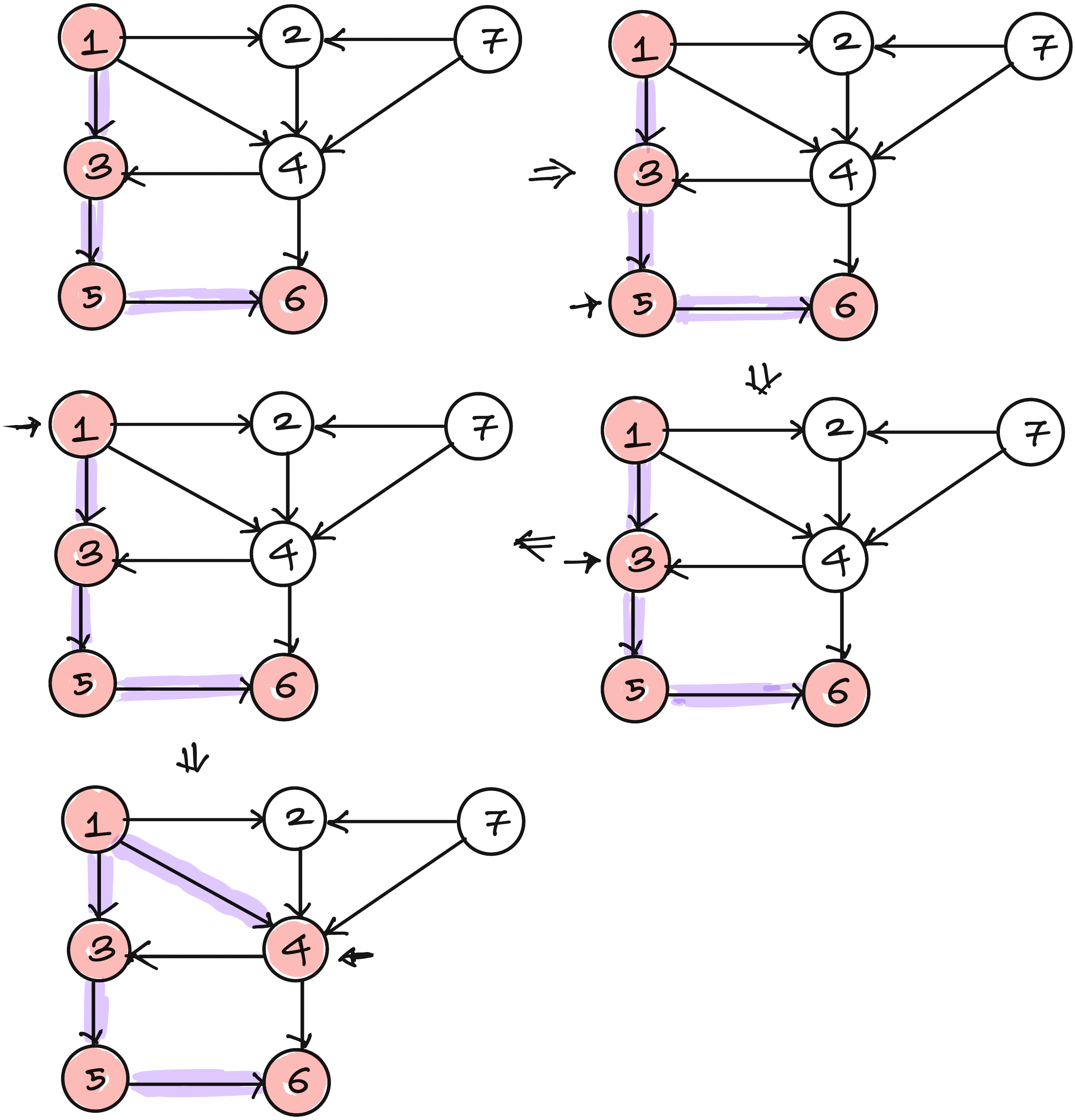
DEPTH FIRST SEARCH.

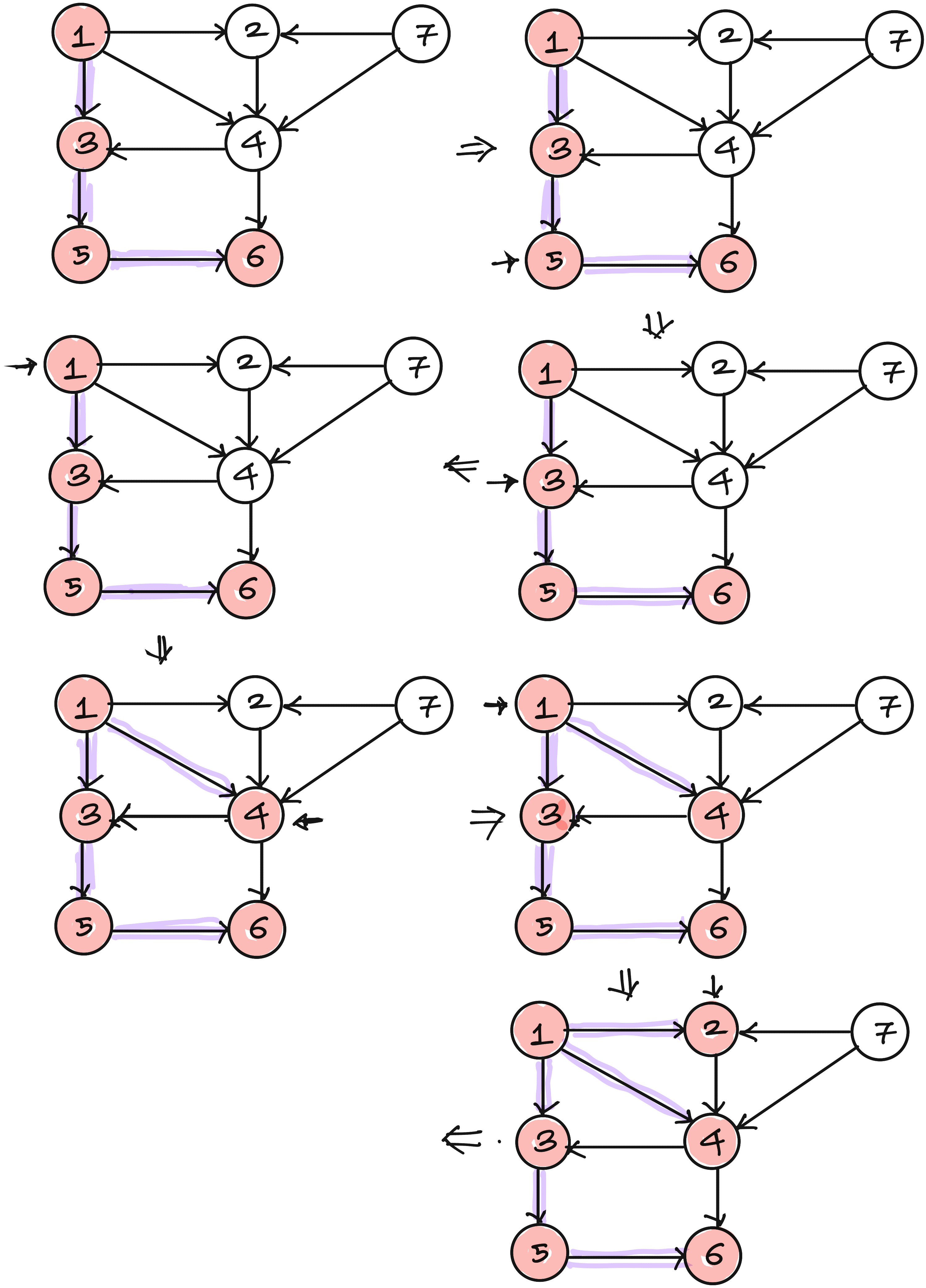


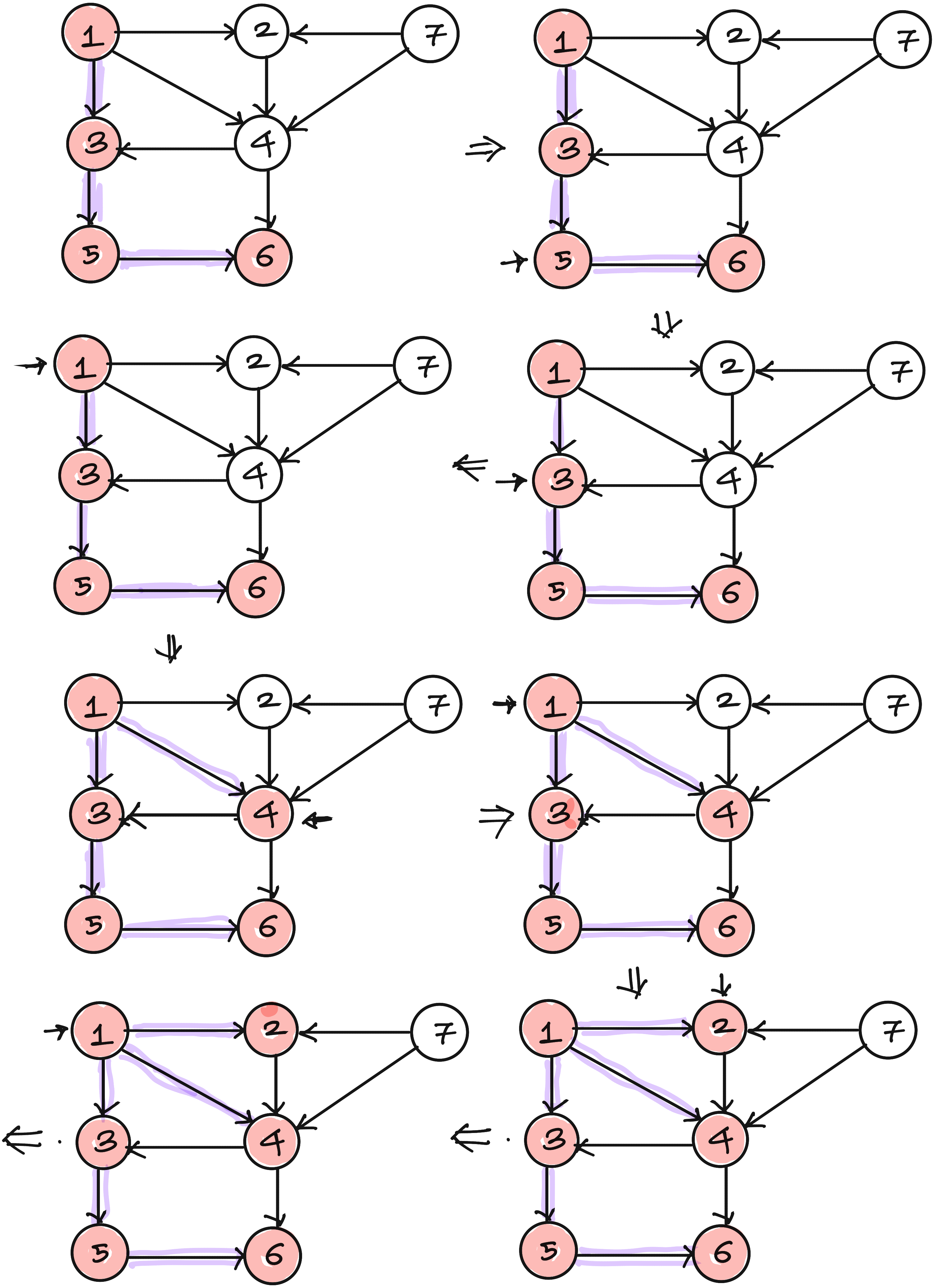

1
2
7
3
4
5
6
1
2
7
3
4
5
6


1
2
7
3
4
5
6
1
2
7
3
4
5
6
1
2
7
3
4
5
6


1 2 7
3 4
5 6
1 2 7
3 4
5 6
1 2 7
3 4
5 6
1 2 7
3 4
5 6


1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6


1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6


1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6
1 2 7 3 4 5 6

Q: CAN YOU WRITE THE CODE FOR DFS

Q: CAN YOU WRITE THE CODE FOR DFS

```
FOREACH v ∈ V
    VISTED[v] ← FALSE
FOREACH v ∈ V
{   If ( VISITED[v] = FALSE)
        DFS(v)
}
```

Q: CAN YOU WRITE THE CODE FOR DFS

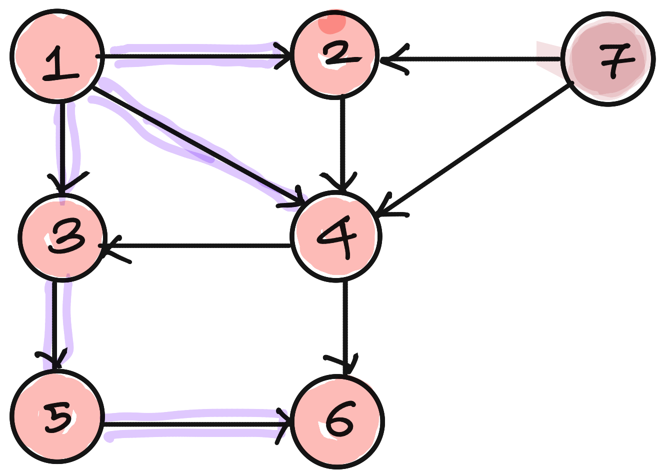
```
FOREACH v ∈ V
    VISITED[v] ← FALSE
FOREACH v ∈ V
{   IF ( VISITED[v] = FALSE)
        DFS(v)
}


DFS(v)
{

    VISITED[v] ← TRUE
    FOREACH OUTGOING EDGE (v,w)
    {    IF ( VISITED[w] = FALSE)
         {
         }    DFS(w)
    }


}
```

Q: CAN YOU WRITE THE CODE FOR DFS

```
TIME ← 0
FOREACH v ∈ V
        VISITED[v] ← FALSE
FOREACH v ∈ V
{    IF ( VISITED[v] = FALSE)
            DFS(v)
}


DFS(v)
{    ARRIVAL[v] ← TIME
     TIME ← TIME+1;
     VISITED[v] ← TRUE
     FOREACH OUTGOING EDGE (v,w)
     {     IF ( VISITED[w] = FALSE)
           {
           }     DFS(w)
     }
     DEPARTURE[v] ← TIME;
     TIME ← TIME+1 ;
}
```

Q: CAN YOU WRITE THE CODE FOR DFS

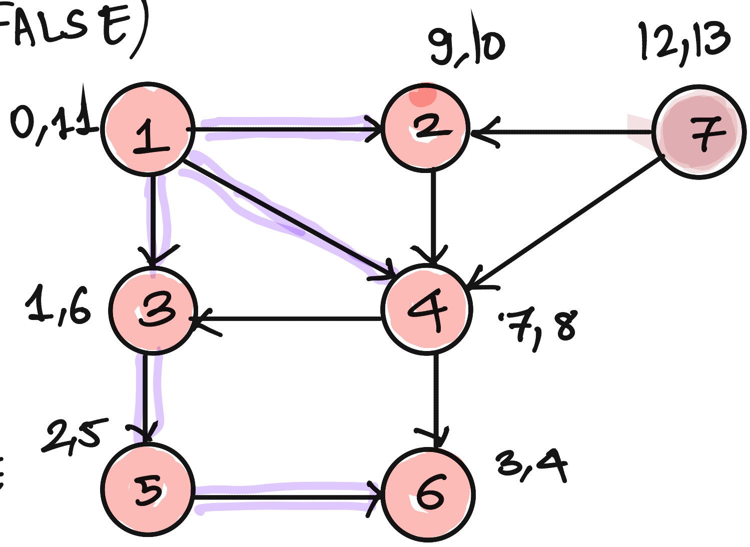
```
TIME ← 0
FOREACH v ∈ V
    VISITED[v] ← FALSE
FOREACH v ∈ V
{   IF ( VISITED[v] = FALSE)
        DFS(v)
}
```




```
DFS(v)
{   ARRIVAL[v] ← TIME
    TIME ← TIME+1 ;
    VISITED[v] ← TRUE
    FOREACH OUTGOING EDGE (v,w)
    {   IF ( VISITED[w] = FALSE)
        {
        }   DFS(w)
    }
    DEPARTURE[v] ← TIME ;
    TIME ← TIME+1 ;
}
```

Q: CAN YOU WRITE THE CODE FOR DFS

```
TIME ← 0
FOREACH v ∈ V
      VISITED[v] ← FALSE
FOREACH v ∈ V
{    If ( VISITED[v] = FALSE)
          DFS(v)
}
```

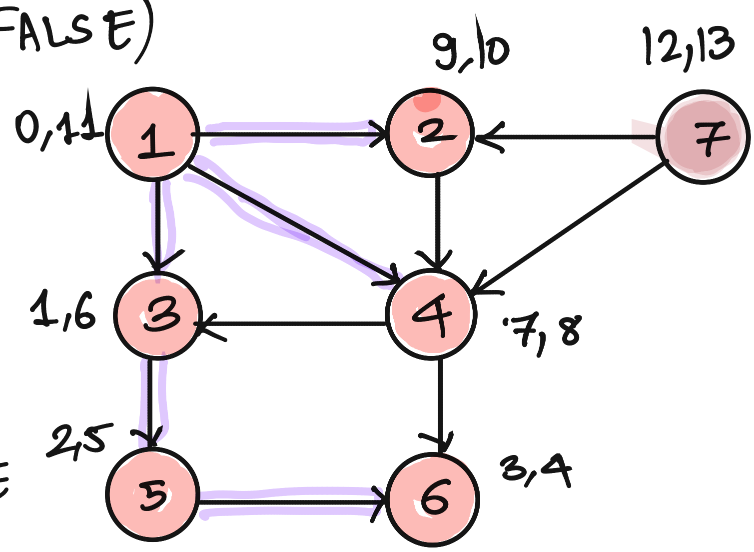



```
DFS(v)
{    ARRIVAL[v] ← TIME
     TIME ← TIME+1;
     VISITED[v] ← TRUE
     FOREACH OUTGOING EDGE (v,w)
     {     IF ( VISITED[w] = FALSE)
           {
           }     DFS(w)
     }
     DEPARTURE[v] ← TIME;
     TIME ← TIME+1 ;
}
```

RUNNING TIME =

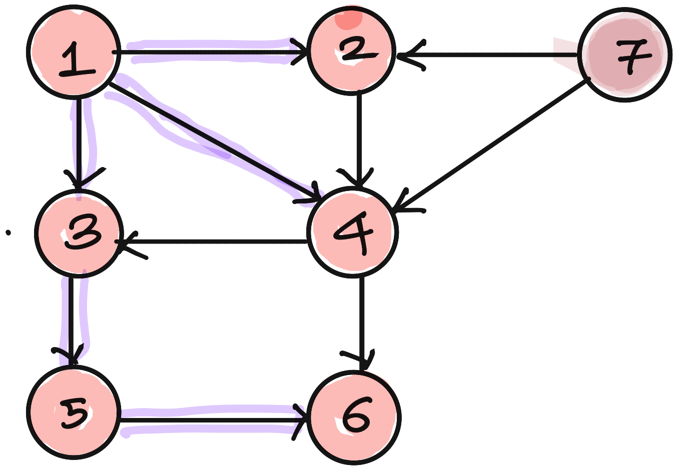
Q: CAN YOU WRITE THE CODE FOR DFS

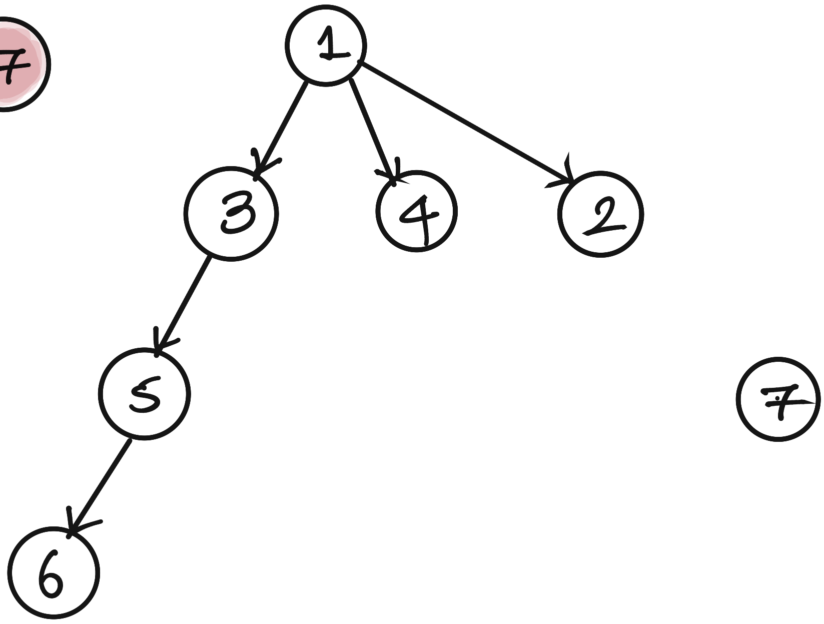
```
TIME ← 0
FOREACH v ∈ V
    VISITED[v] ← FALSE
FOREACH v ∈ V
{   IF ( VISITED[v] = FALSE)
        DFS(v)
}
```

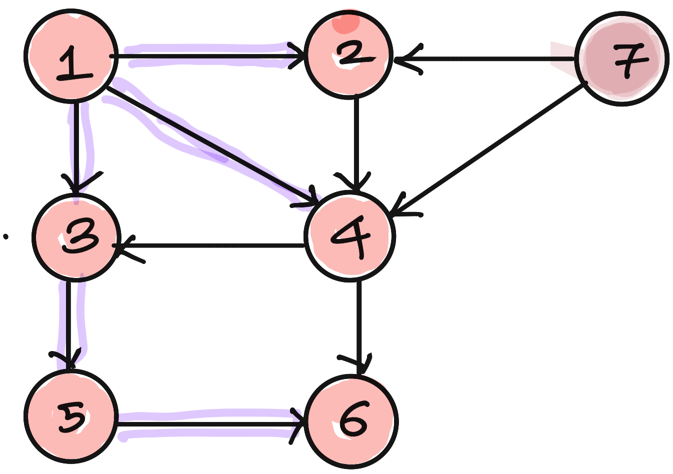


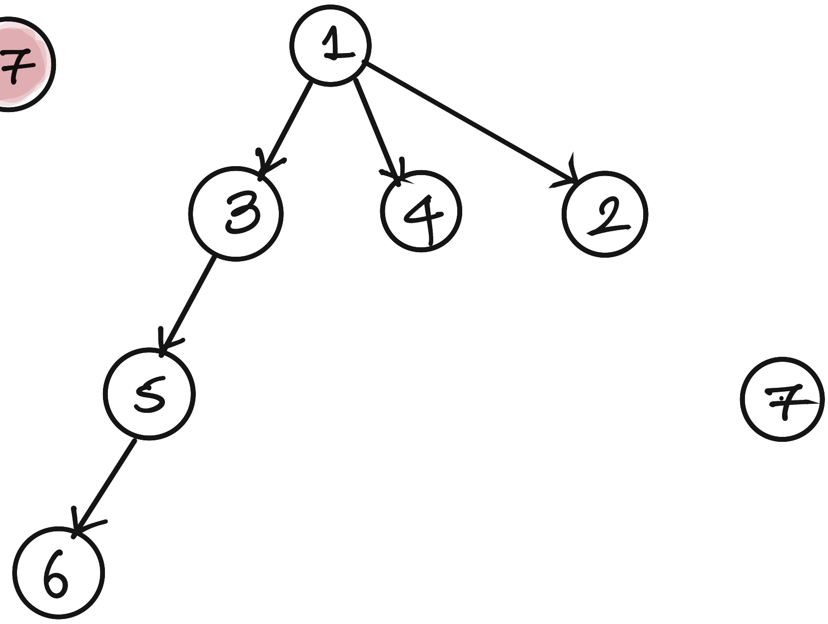

```
DFS(v)
{   ARRIVAL[v] ← TIME
    TIME ← TIME+1;
    VISITED[v] ← TRUE
    FOREACH OUTGOING EDGE (v,w)
    {   IF ( VISITED[w] = FALSE)
        {
        }   DFS(w)
    }
    DEPARTURE[v] ← TIME;
    TIME ← TIME+1;
}
```

RUNNING TIME = $O(m+n)$


1
2
7
3
4
5
6


1
3
4
2
5
7
6

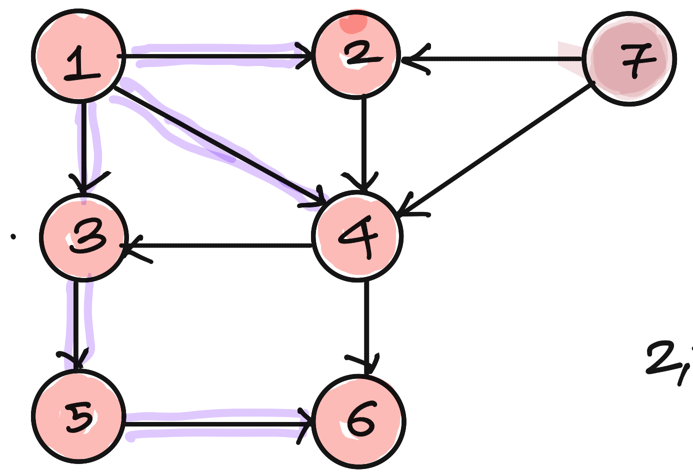


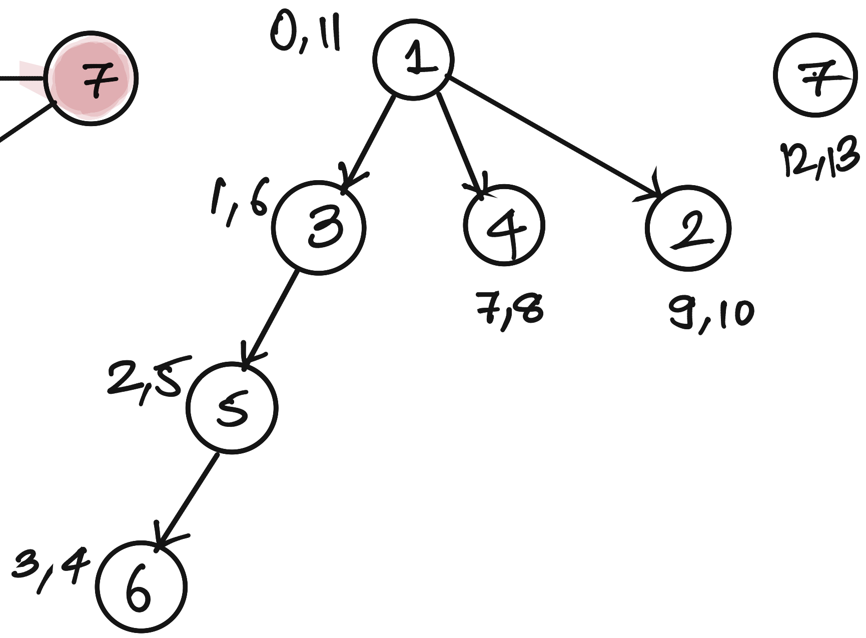


LET US LOOK AT ALL POSSIBLE EDGES OF THE GRAPH

1) EDGES THAT ARE PART OF DFS TREE
(TREE EDGES)

2) EDGES THAT ARE NOT PART OF DFS TREE
(NON- TREE EDGES).

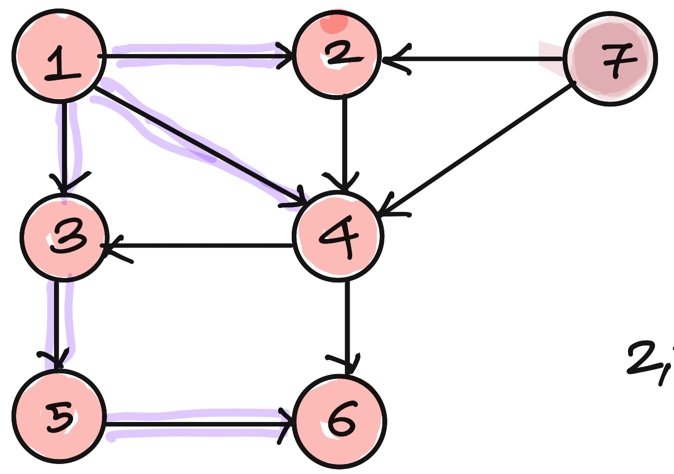


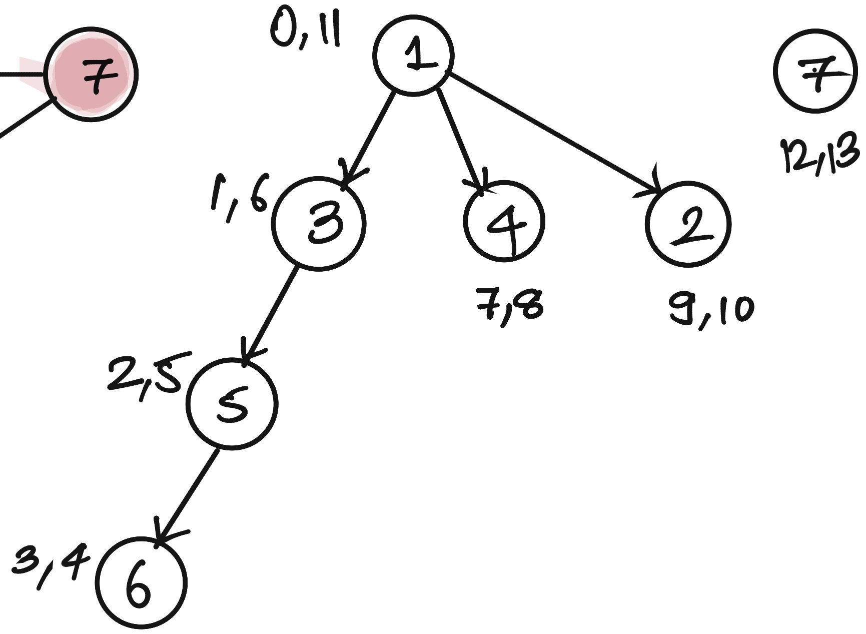


LET US LOOK AT ALL POSSIBLE EDGES OF THE GRAPH

1) TREE EDGE (u, v)

IS THERE ANY RELATION BETWEEN ARRIVAL & DEPARTURE TIME OF u & v ?



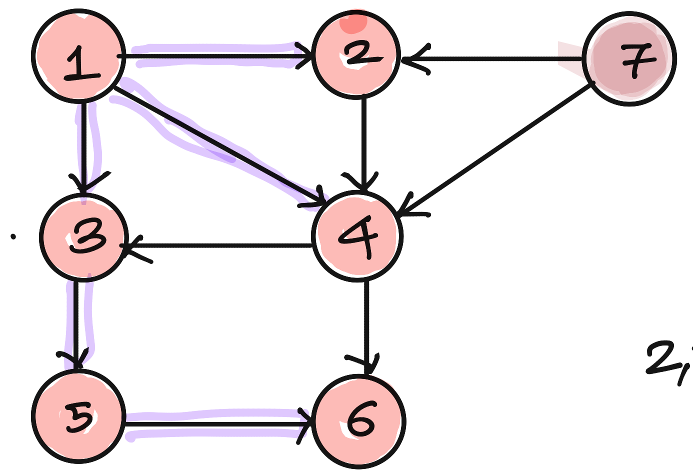


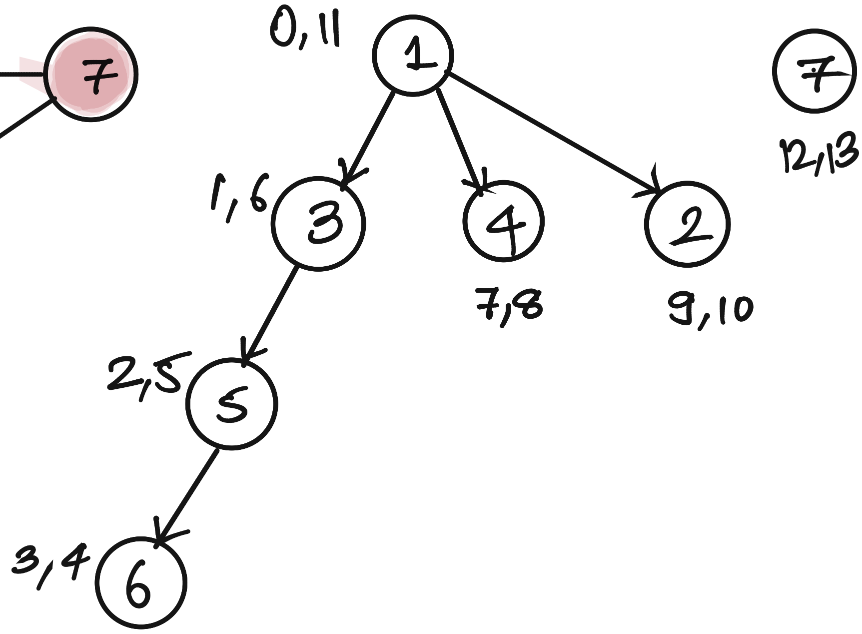
LET US LOOK AT ALL POSSIBLE EDGES OF THE GRAPH

1) TREE EDGE (u, v)

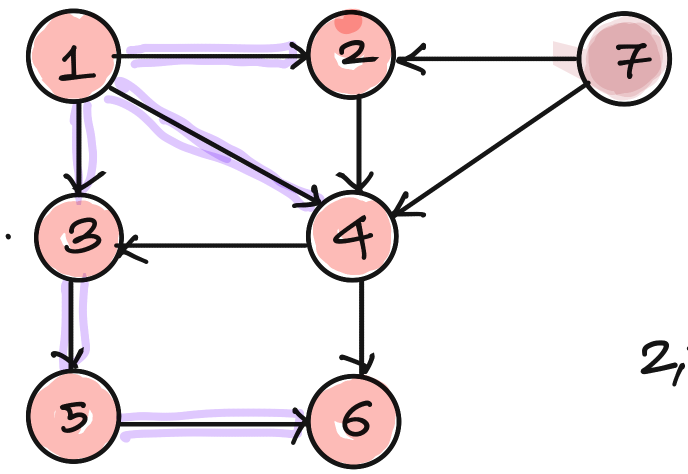
IS THERE ANY RELATION BETWEEN ARRIVAL & DEPARTURE TIME OF u & v ?

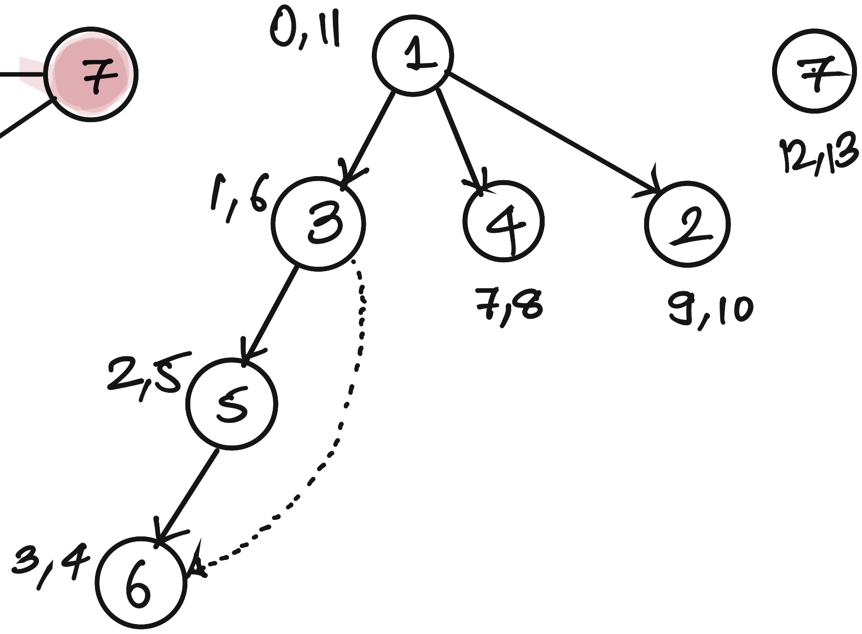
A : ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]



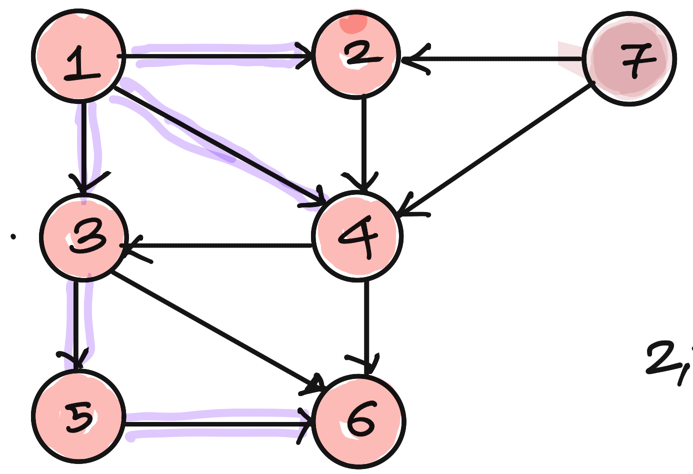


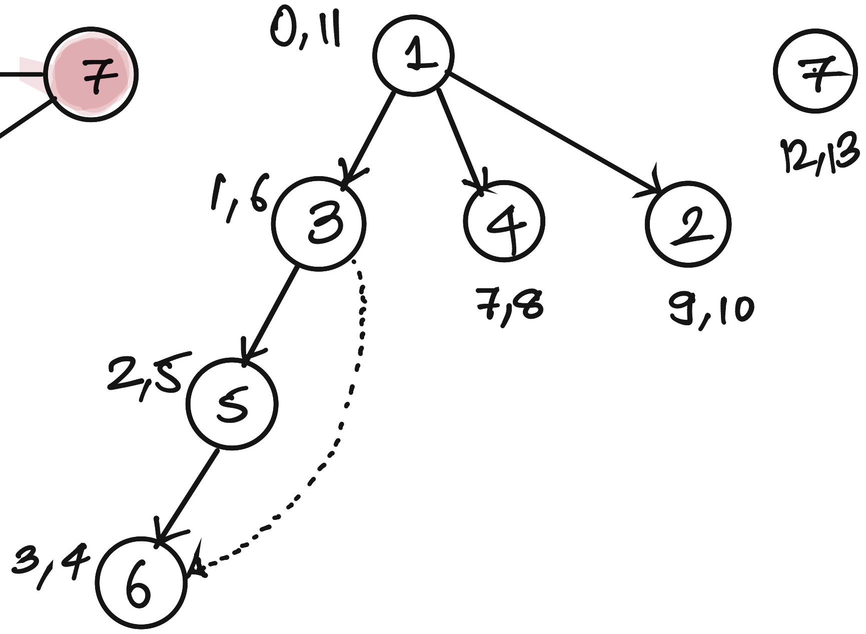
2) NON-TREE EDGES



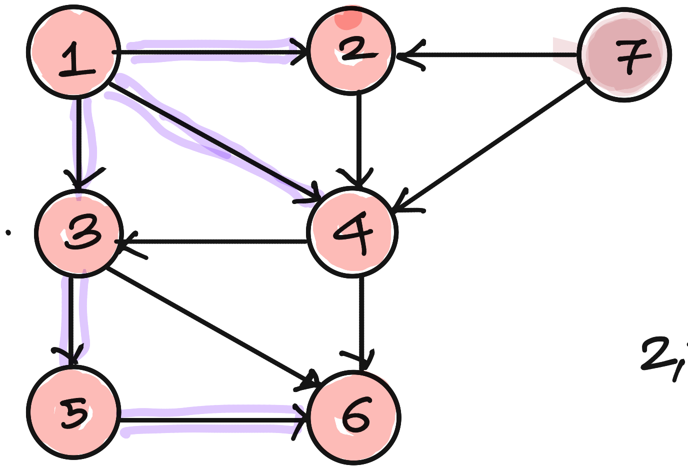


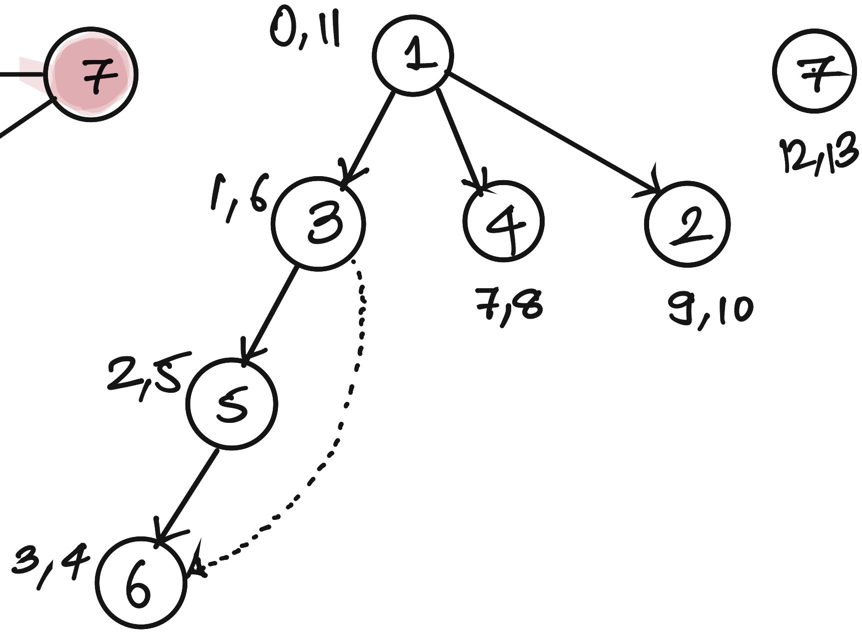
2) NON - TREE EDGES





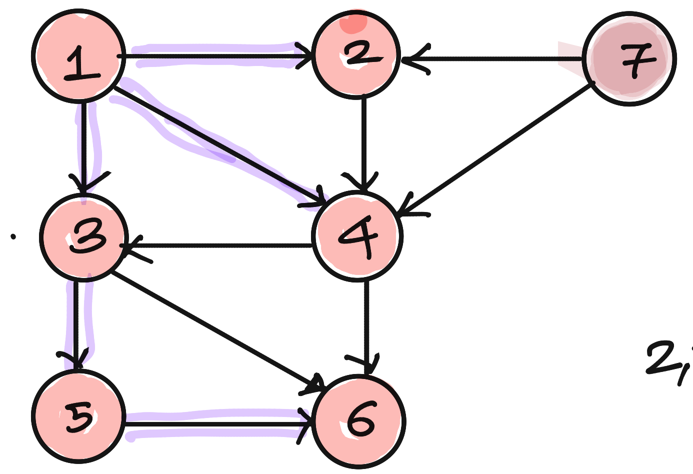
2) NON - TREE EDGES

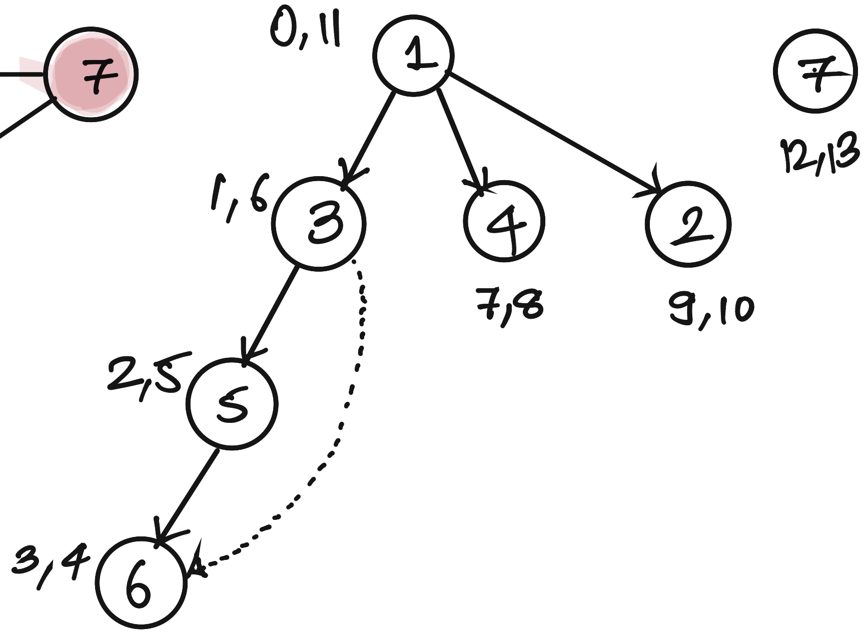




2) NON - TREE EDGES

a) FORWARD EDGE (u,v)





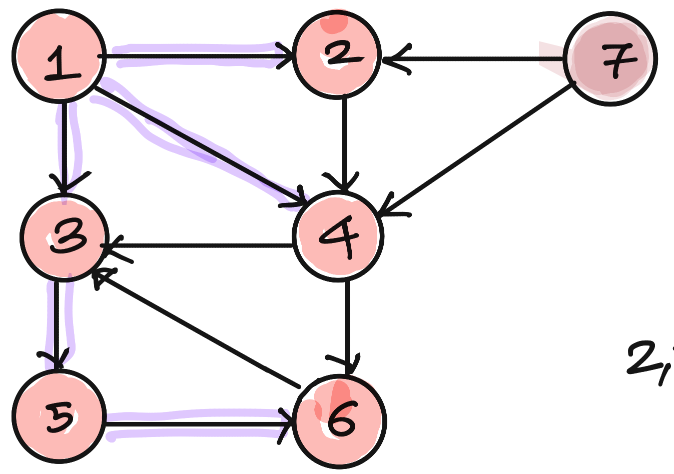
2) NON - TREE EDGES

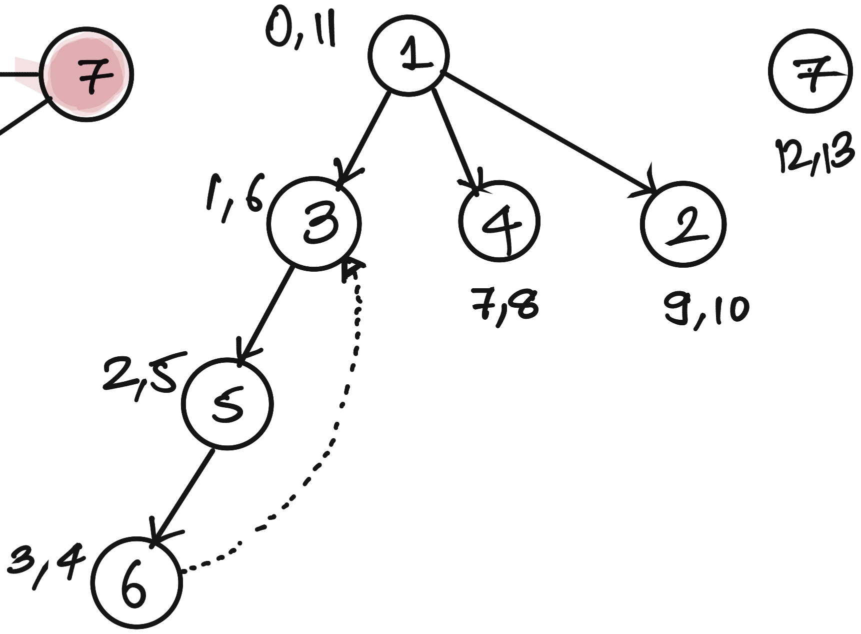
a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v].

< DEPARTURE [u]

(b)





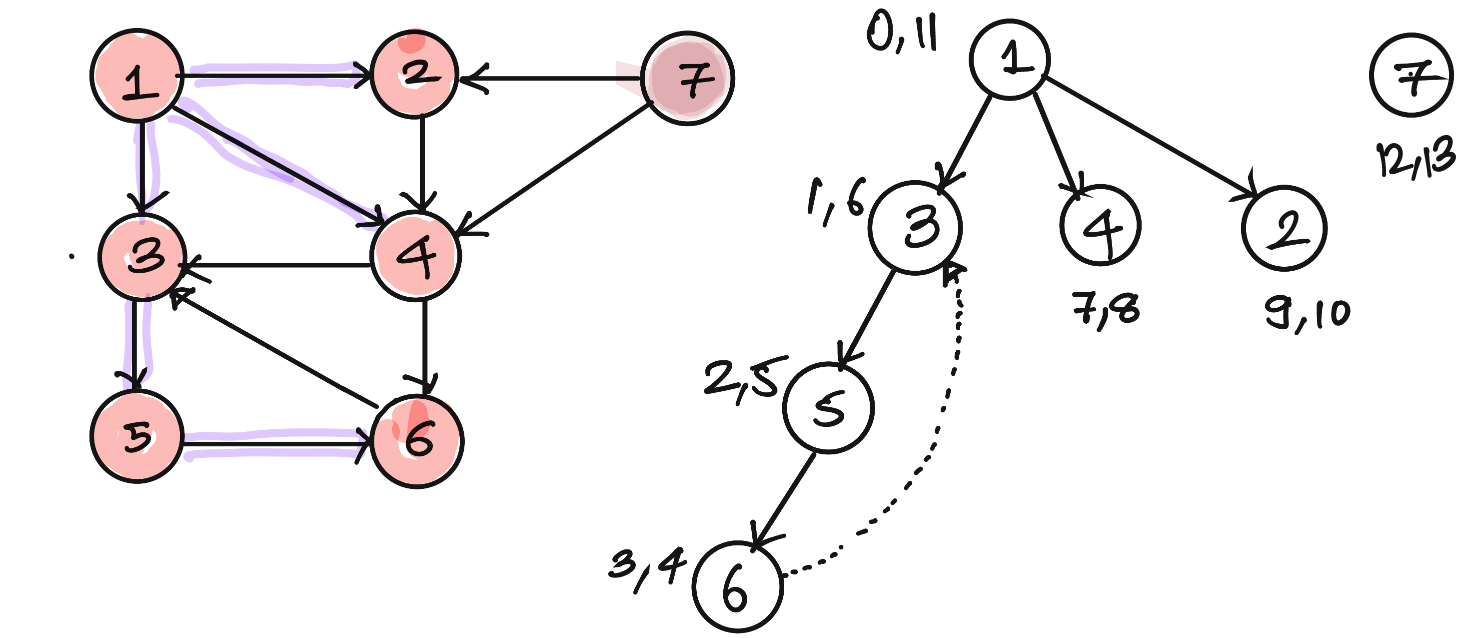
2) NON - TREE EDGES

a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v].

< DEPARTURE[u]

(b) BACK EDGE (u,v)



2) NON - TREE EDGES

a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v].

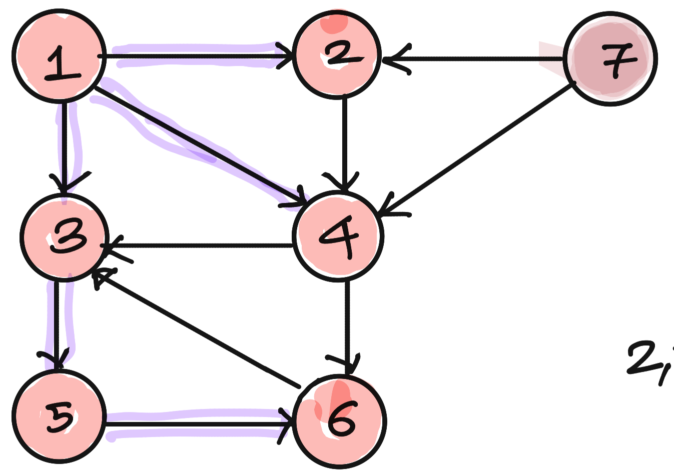
< DEPARTURE [u]

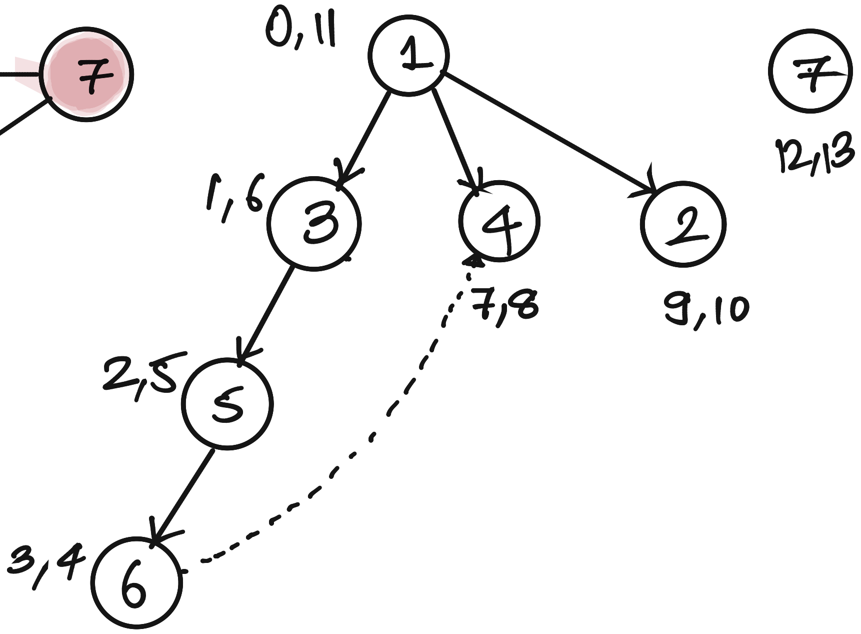
(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u]

< DEPARTURE[v]

(c)





2) NON - TREE EDGES

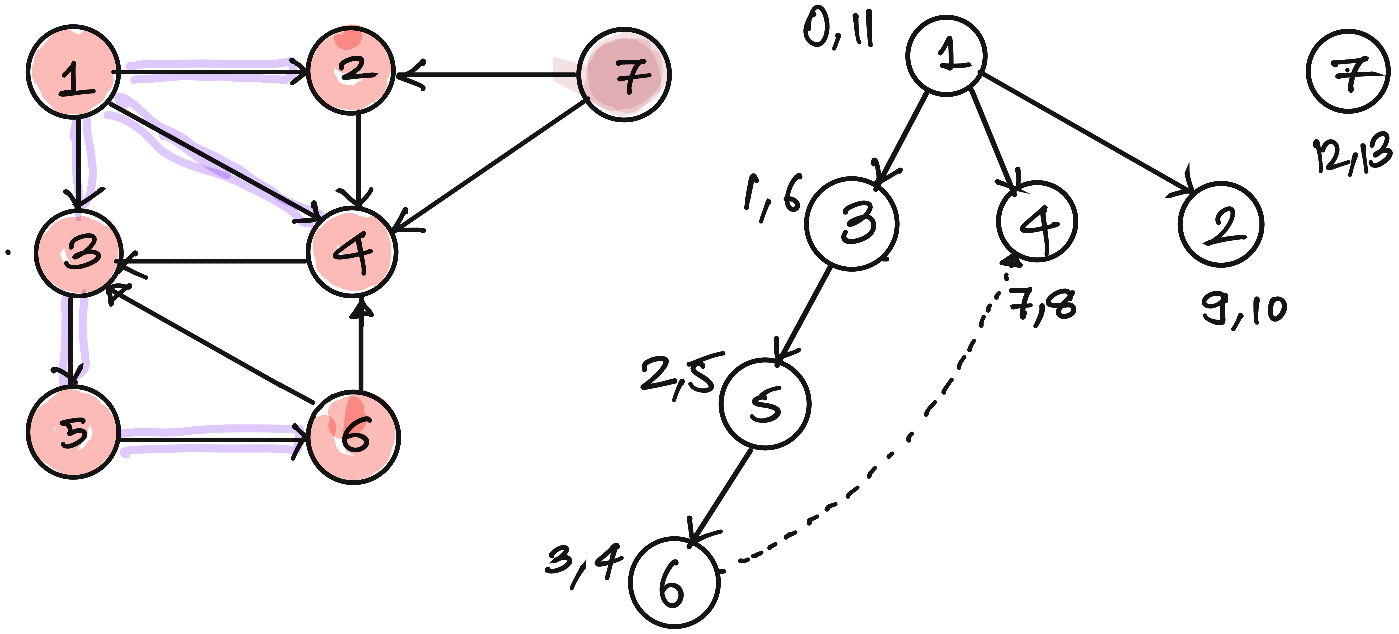
a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v].
< DEPARTURE[u]

(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u]
< DEPARTURE[v]

(c)



2) NON - TREE EDGES

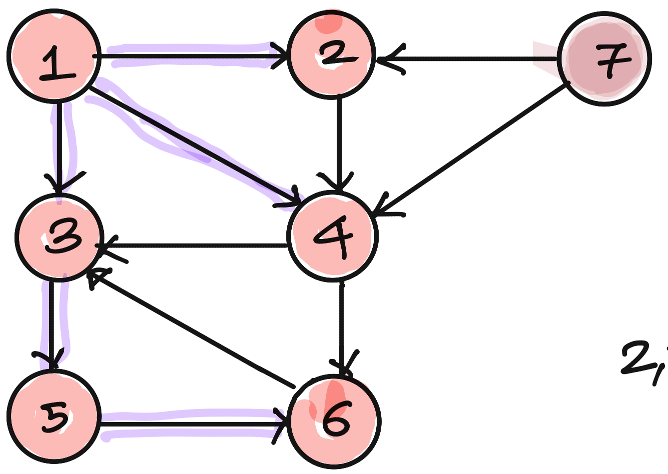
a) FORWARD EDGE (u,v)

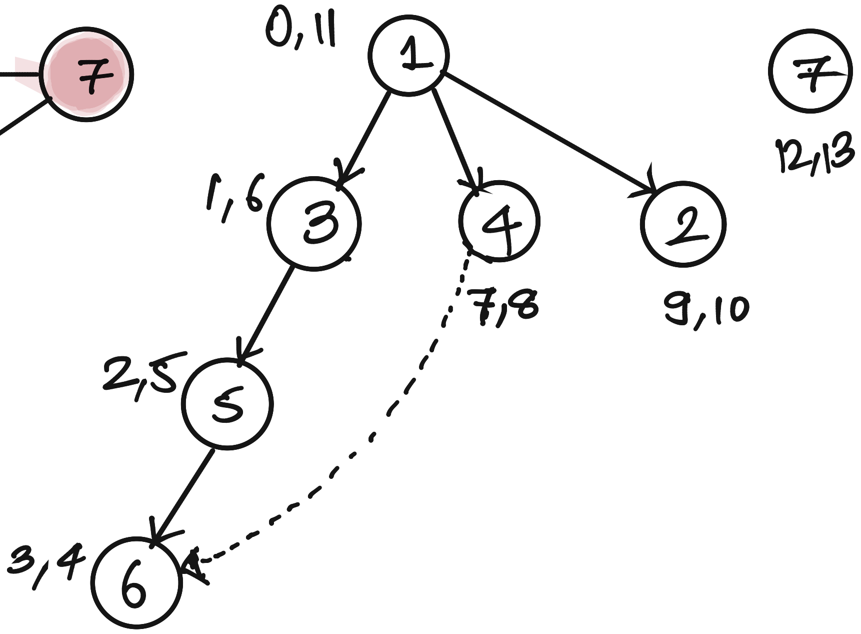
$$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{DEPARTURE}[u]$$

(b) BACK EDGE (u,v)

$$\text{ARRIVAL}[v] < \text{ARRIVAL}[u] < \text{DEPARTURE}[u] < \text{DEPARTURE}[v]$$

(c)





2) NON-TREE EDGES

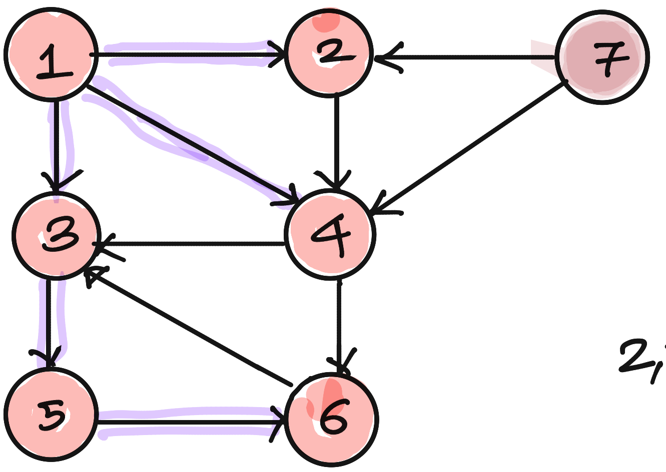
a) FORWARD EDGE (u,v)

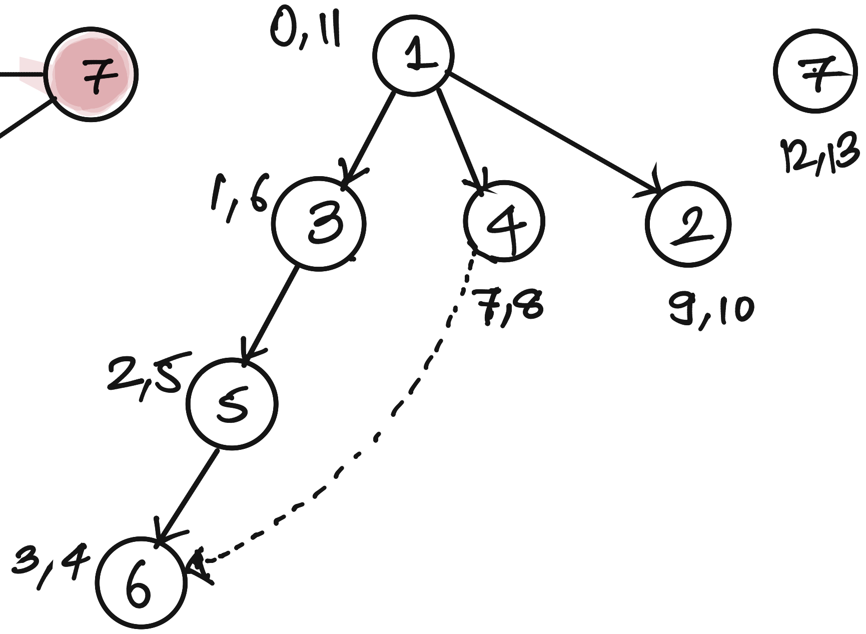
ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u] < DEPARTURE[v]

(c) CROSS EDGE (u,v)





2) NON - TREE EDGES

a) FORWARD EDGE (u, v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

(b) BACK EDGE (u, v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u] < DEPARTURE[v]

(c) CROSS EDGE (u, v)

ARRIVAL[v] < DEPARTURE[v] < ARRIVAL[u] < DEPARTURE[u]

Q: GIVEN A DIRECTED GRAPH, FIND IF IT CONTAINS A DIRECTED CYCLE.

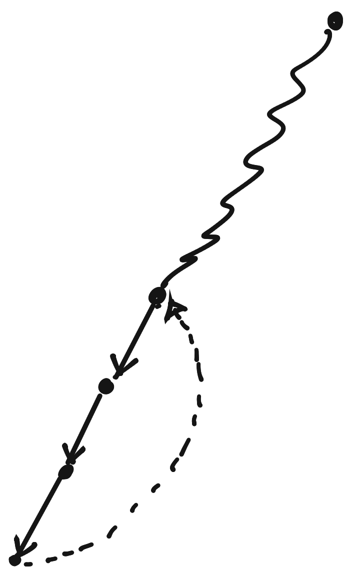
Q: GIVEN A DIRECTED GRAPH, FIND IF IT CONTAINS A DIRECTED CYCLE.

LEMMA: IF THERE IS A BACK-EDGE, THEN THERE IS A DIRECTED CYCLE.

Q: GIVEN A DIRECTED GRAPH, FIND IF IT CONTAINS A DIRECTED CYCLE.

LEMMA: IF THERE IS A BACK-EDGE, THEN THERE IS A DIRECTED CYCLE.

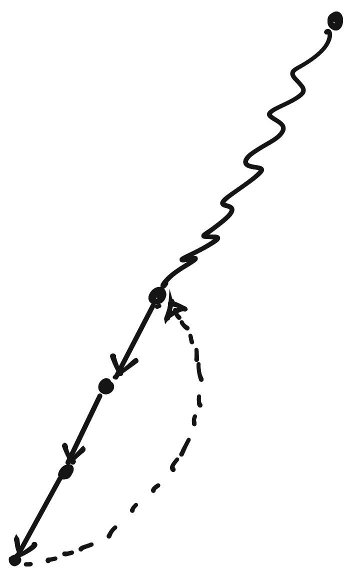
PROOF: BY PICTURE



Q: GIVEN A DIRECTED GRAPH, FIND IF IT CONTAINS A DIRECTED CYCLE.

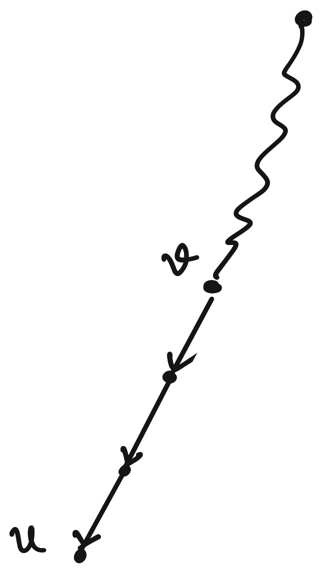
LEMMA: IF THERE IS A BACK-EDGE, THEN THERE IS A DIRECTED CYCLE.

PROOF: BY PICTURE

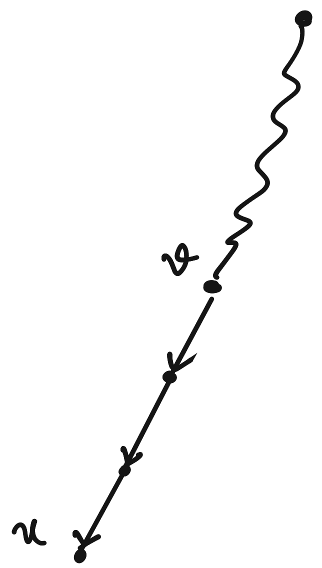


IS THE REVERSE TRUE

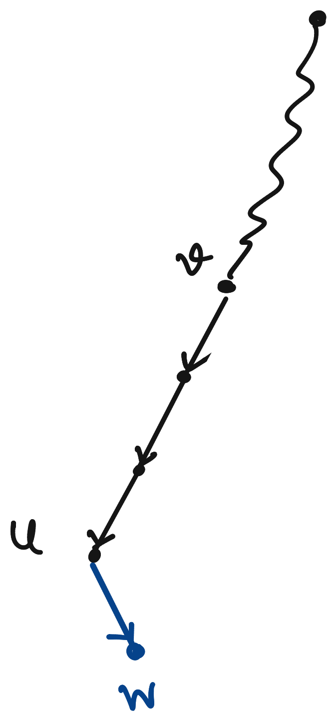
IF THERE IS NO BACK EDGE, THERE IS NO CYCLE IN THE GRAPH.



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.



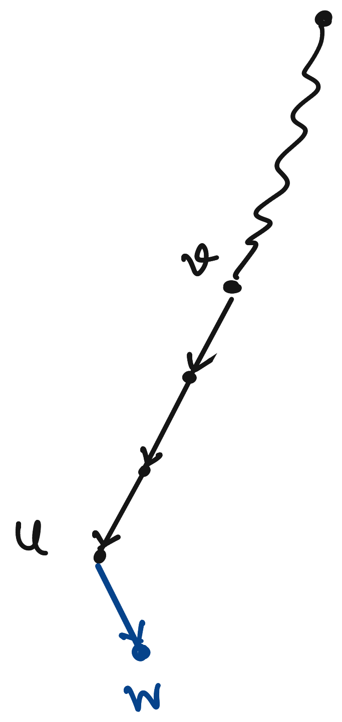
TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.
BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE



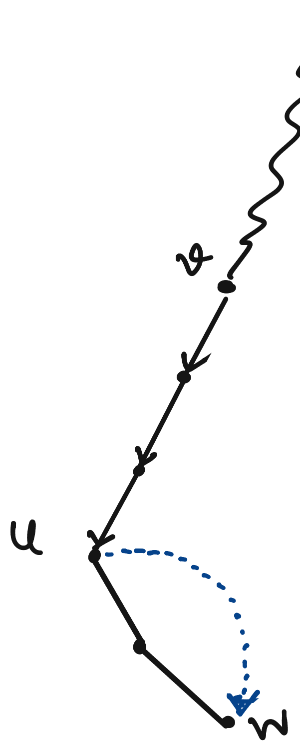
TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE

$\rightarrow$ YOU GO AWAY FROM v

(2) NON-TREE EDGE



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

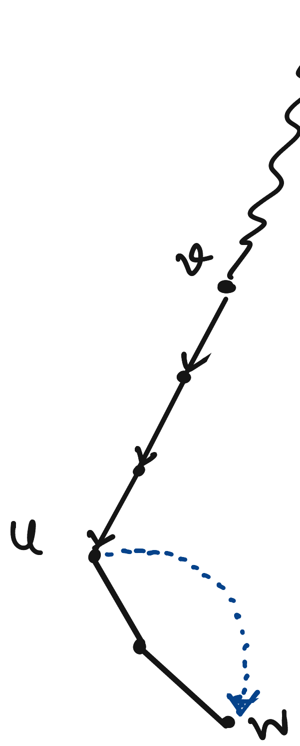
BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE

→ YOU GO AWAY FROM v

(2) NON-TREE EDGE

(a) FORWARD EDGE



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE

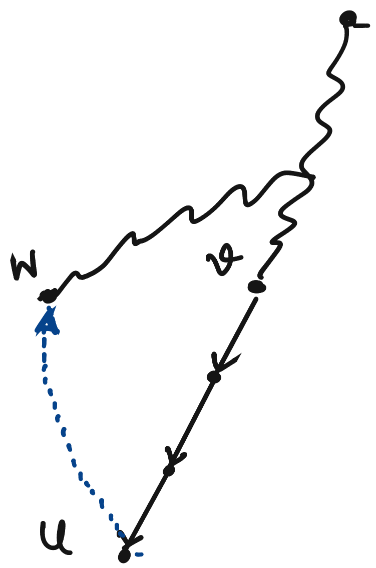
→ YOU GO AWAY FROM v

(2) NON-TREE EDGE (FORWARD)

(a) FORWARD EDGE

→ YOU GO AWAY FROM v

(b) CROSS EDGE



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE

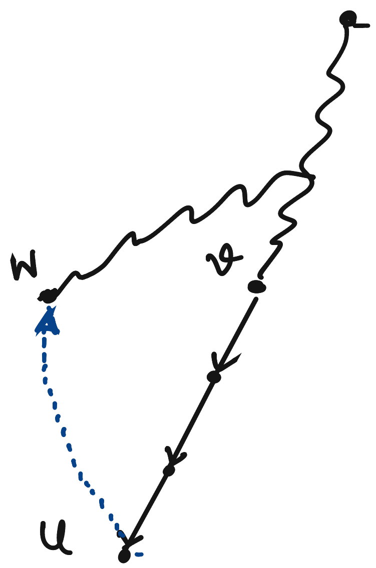
→ YOU GO AWAY FROM v

(2) NON-TREE EDGE (FORWARD)

(a) FORWARD EDGE

→ YOU GO AWAY FROM v

(b) CROSS EDGE



TO COMPLETE THE CYCLE, YOU HAVE TO COME BACK TO v.

BUT THERE ARE THREE POSSIBLE EDGE YOU CAN TAKE FROM u

(1) TREE EDGE

→ YOU GO AWAY FROM v

(2) NON-TREE EDGE (FORWARD)

(a) FORWARD EDGE

→ YOU GO AWAY FROM v

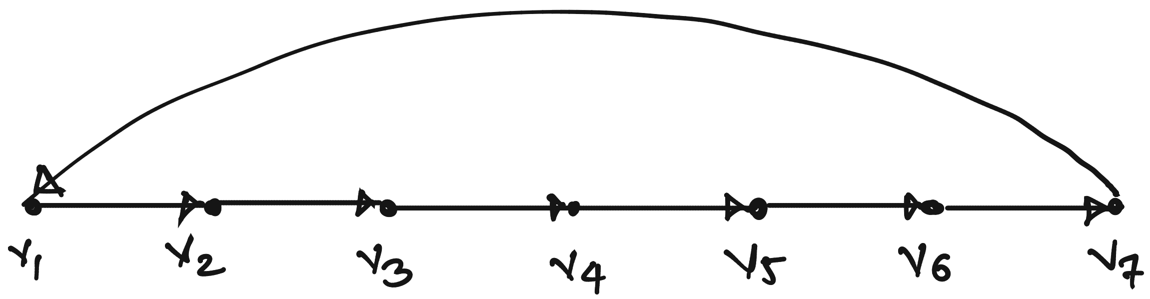
(b) CROSS EDGE

→ YOU HAVE GONE LEFT USING A CROSS EDGE, NOW YOU CANNOT COME RIGHT.

ALTERNATE NON-INTUITIVE PROOF

<u>LEMMA</u>: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

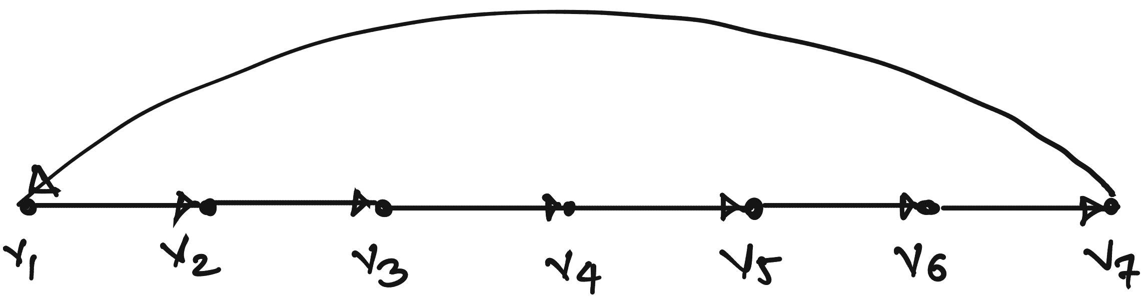
<u>PROOF</u>: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



ALTERNATE NON-INTUITIVE PROOF

<u>LEMMA</u>: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

<u>PROOF</u>: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



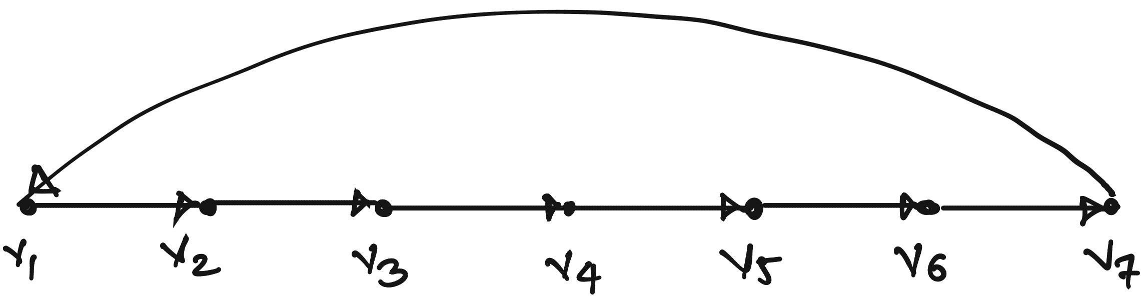
1) TREE EDGE (u,v)

$$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{DEPARTURE}[u]$$

ALTERNATE NON-INTUITIVE PROOF

LEMMA: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

PROOF: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



1) TREE EDGE (u,v)

$$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{DEPARTURE}[u]$$

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

$$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{DEPARTURE}[u]$$

~~(b) BACK EDGE (u,v)~~

$$\cancel{\text{ARRIVAL}[v] < \text{ARRIVAL}[u] < \text{DEPARTURE}[u] < \text{DEPARTURE}[v]}$$

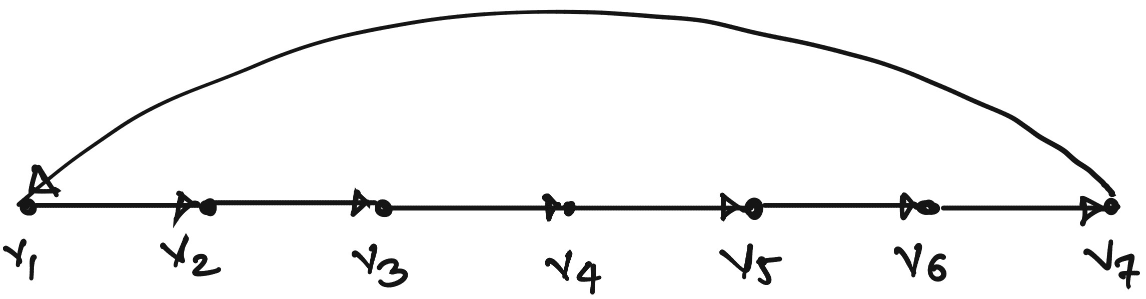
(c) CROSS EDGE (u,v)

$$\text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{ARRIVAL}[u] < \text{DEPARTURE}[u]$$

ALTERNATE NON-INTUITIVE PROOF

<u>LEMMA</u>: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

<u>PROOF</u>: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



1) TREE EDGE (u, v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

2) NON TREE EDGE

a) FORWARD EDGE (u, v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

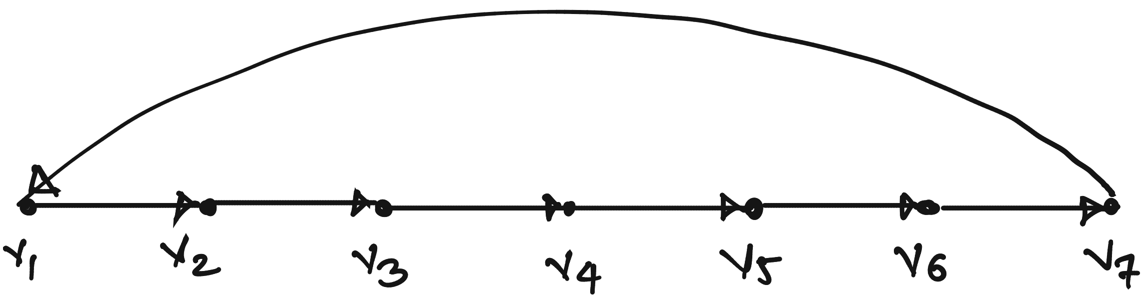
b) CROSS EDGE (u, v)

ARRIVAL[v] < DEPARTURE[v] < ARRIVAL[u] < DEPARTURE[u]

ALTERNATE NON-INTUITIVE PROOF

LEMMA: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

PROOF: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



1) TREE EDGE (u,v)

DEPARTURE[v] < DEPARTURE[u]

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

DEPARTURE[v] < DEPARTURE[u]

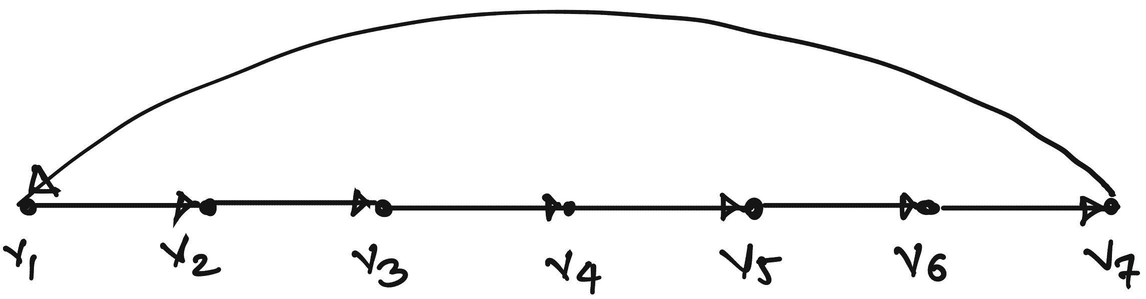
b) CROSS EDGE (u,v)

DEPARTURE[v] < DEPARTURE[u]

ALTERNATE NON-INTUITIVE PROOF

LEMMA: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

PROOF: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



$D[v_1] > D[v_2]$

1) TREE EDGE (u,v)

   DEPARTURE$[v]$ < DEPARTURE$[u]$

2) NON TREE EDGE

   a) FORWARD EDGE (u,v)

      DEPARTURE$[v]$ < DEPARTURE$[u]$

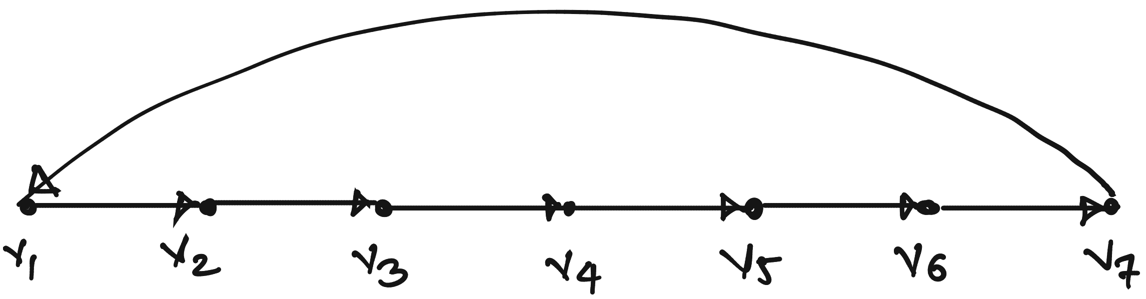
   b) CROSS EDGE (u,v)

      DEPARTURE$[v]$ < DEPARTURE$[u]$

ALTERNATE NON-INTUITIVE PROOF

<u>LEMMA</u>: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

<u>PROOF</u>: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



$D[v_1] > D[v_2] > D[v_3]$

1) TREE EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

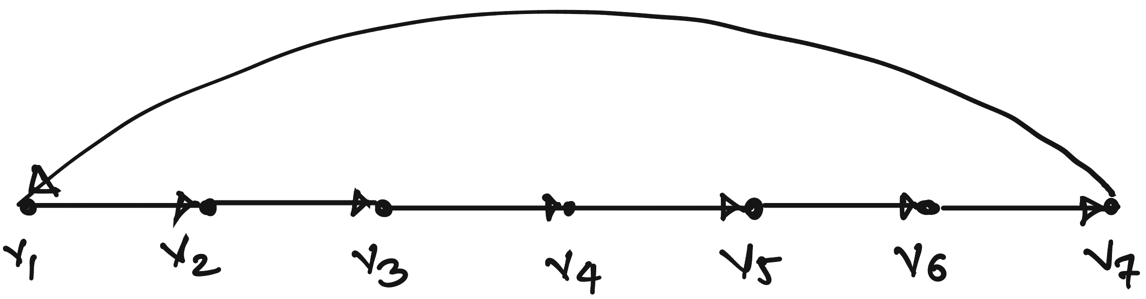
b) CROSS EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

ALTERNATE NON-INTUITIVE PROOF

<u>LEMMA</u>: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

<u>PROOF</u>: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



$D[v_1] > D[v_2] > D[v_3] > D[v_4] > D[v_5] > D[v_6] > D[v_7]$

1) TREE EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

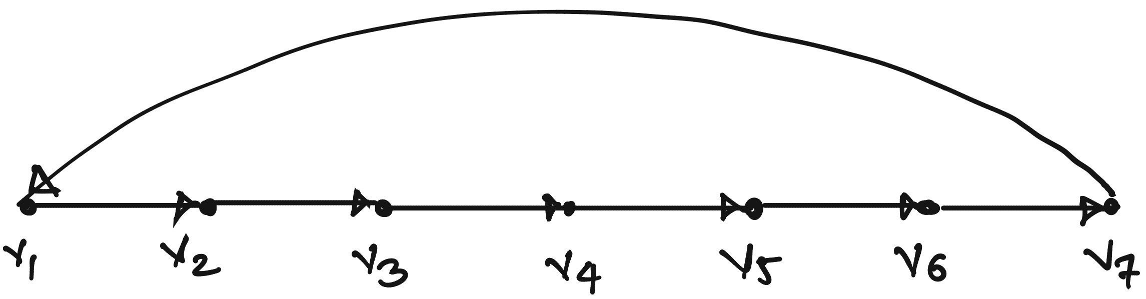
b) CROSS EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

ALTERNATE NON-INTUITIVE PROOF

LEMMA: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

PROOF: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



$D[v_1] > D[v_2] > D[v_3] > D[v_4] > D[v_5] > D[v_6] > D[v_7] > D[v_1]$

1) TREE EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

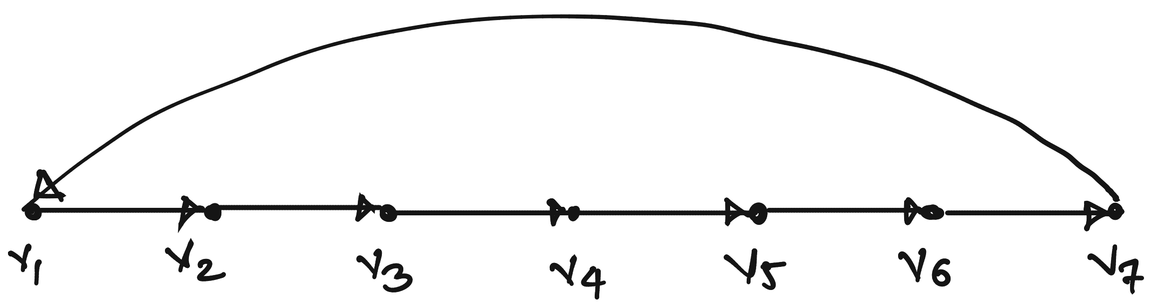
b) CROSS EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

ALTERNATE NON-INTUITIVE PROOF

LEMMA: IF THERE IS NO BACK EDGE, THEN THERE IS NO CYCLE IN THE GRAPH

PROOF: ASSUME FOR CONTRADICTION THAT THERE IS A CYCLE IN THE GRAPH



$D[v_1] > D[v_2] > D[v_3] > D[v_4] > D[v_5] > D[v_6] > D[v_7] > D[v_1]$

A CONTRADICTION

1) TREE EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

2) NON TREE EDGE

a) FORWARD EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

b) CROSS EDGE (u,v)

DEPARTURE$[v]$ < DEPARTURE$[u]$

DO DFS IN THE GRAPH

If THERE EXISTS A BACK EDGE

THERE IS A CYCLE

ELSE

THERE IS NO CYCLE

RUNNING TIME

DO DFS IN THE GRAPH

IF THERE EXISTS A BACK EDGE

THERE IS A CYCLE

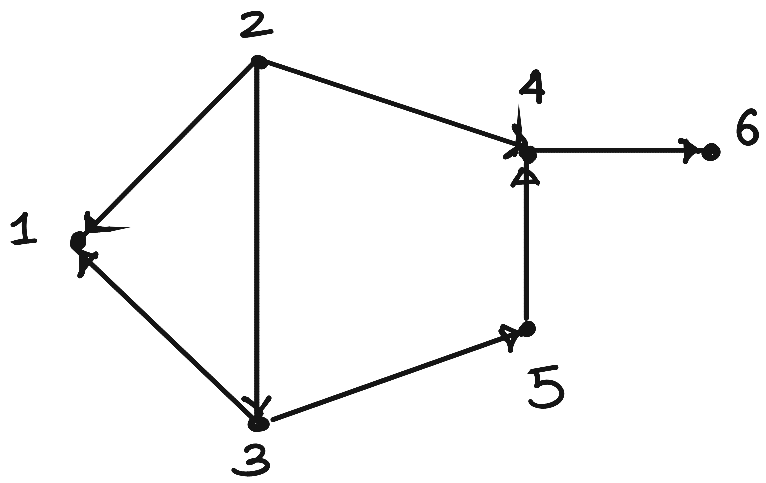
ELSE

THERE IS NO CYCLE

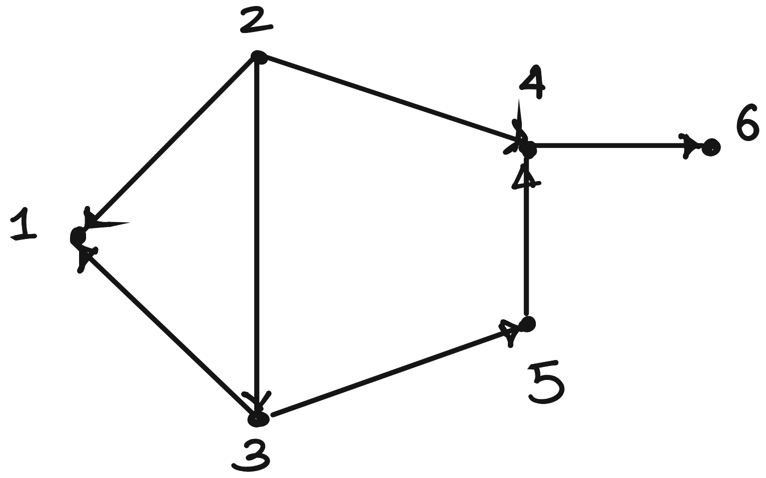
RUNNING TIME : $O(m+n)$.

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

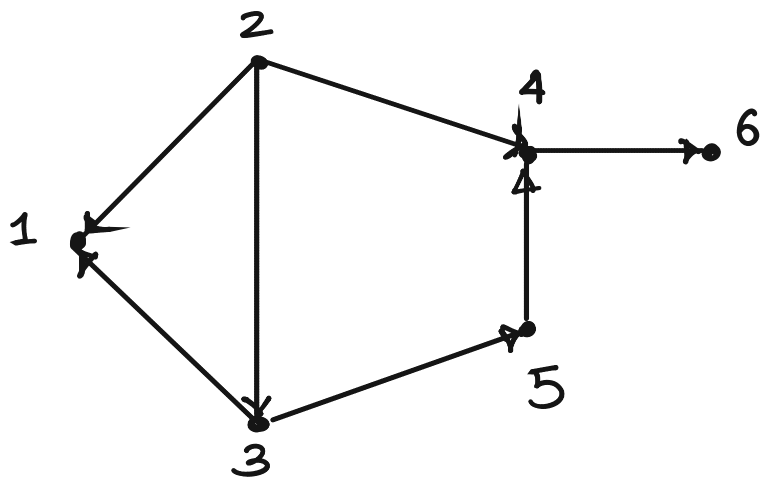
PROBLEM 2 : GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u, v), u LIES TO THE LEFT OF v.

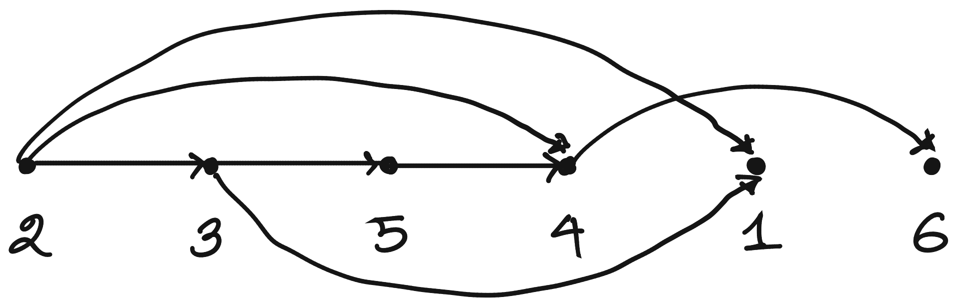


PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

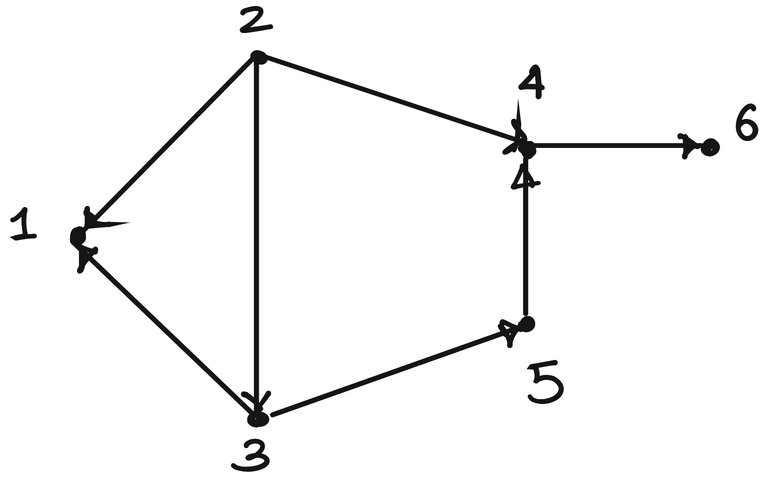


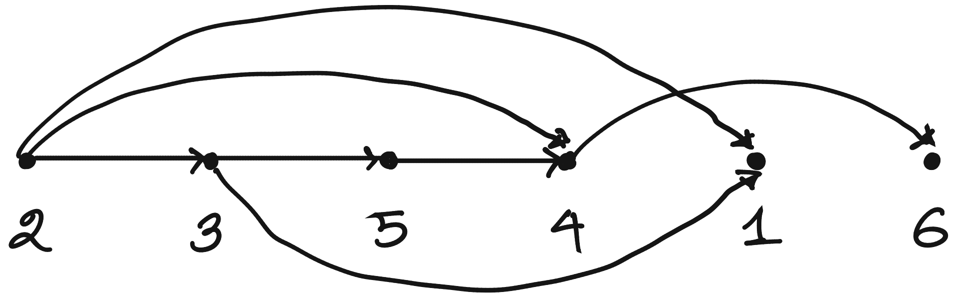
PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.



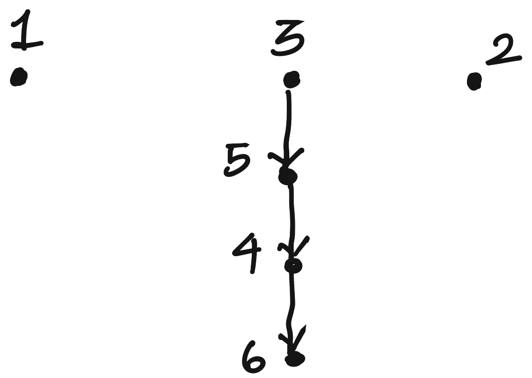


PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u, v), u LIES TO THE LEFT OF v.

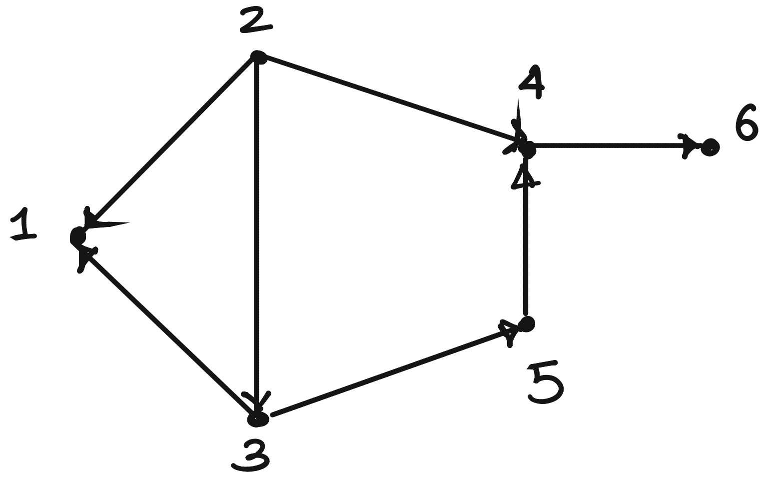




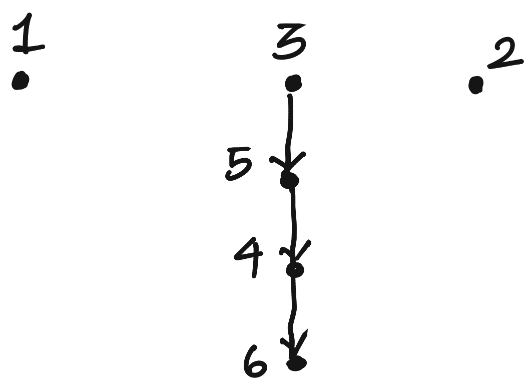
DFS



PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

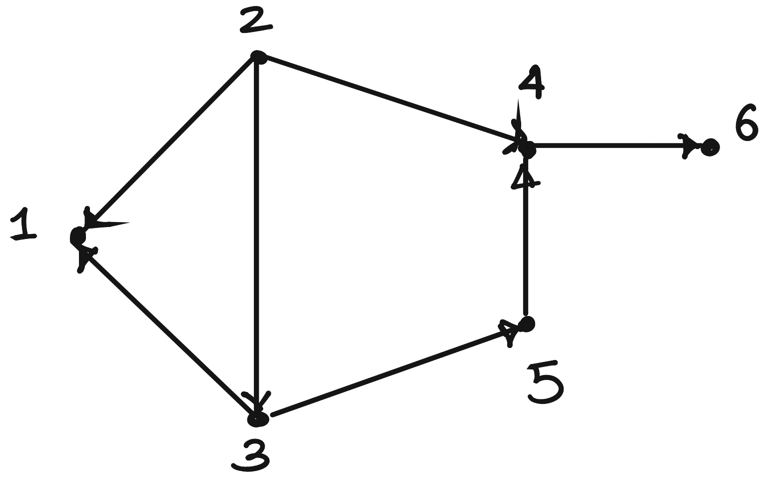


DFS

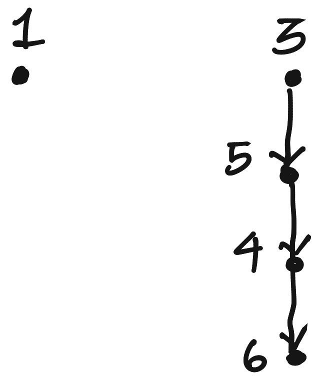


Q: IS THERE ANY INCOMING EDGE AT 2?

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u, v), u LIES TO THE LEFT OF v.



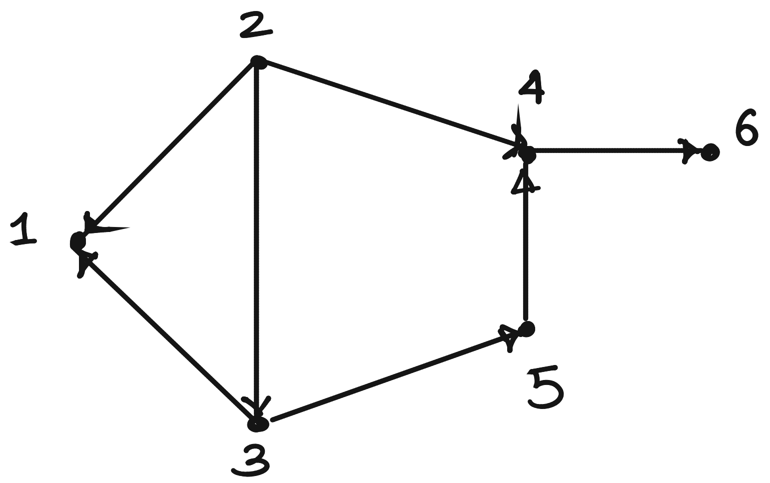
DFS



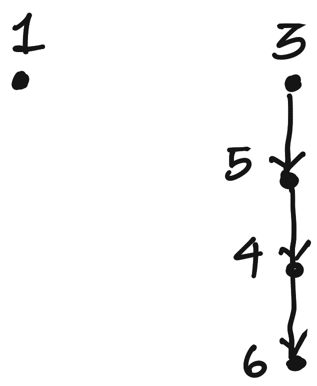
Q: IS THERE ANY INCOMING EDGE AT 2?

A: NO

PROBLEM 2 : GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

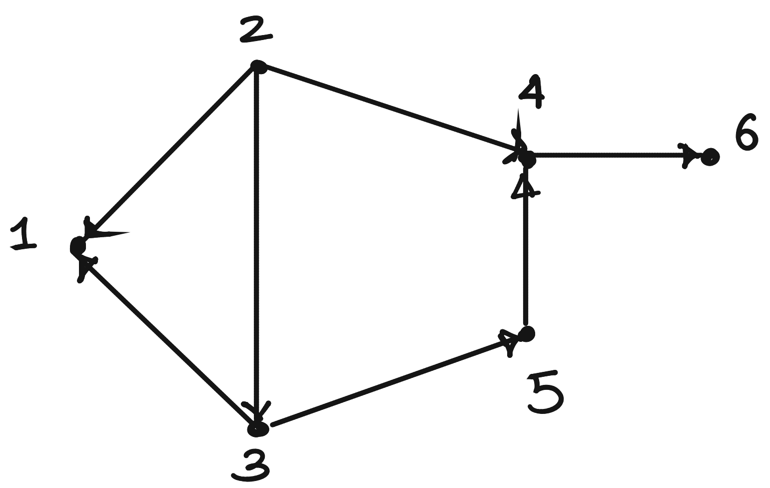


DFS

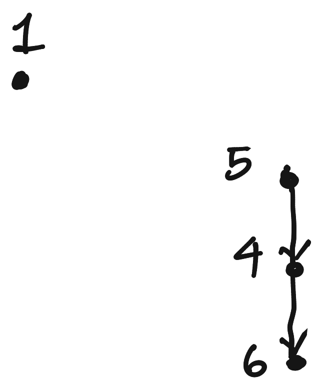


Q: IS THERE ANY INCOMING EDGE AT 3!

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.



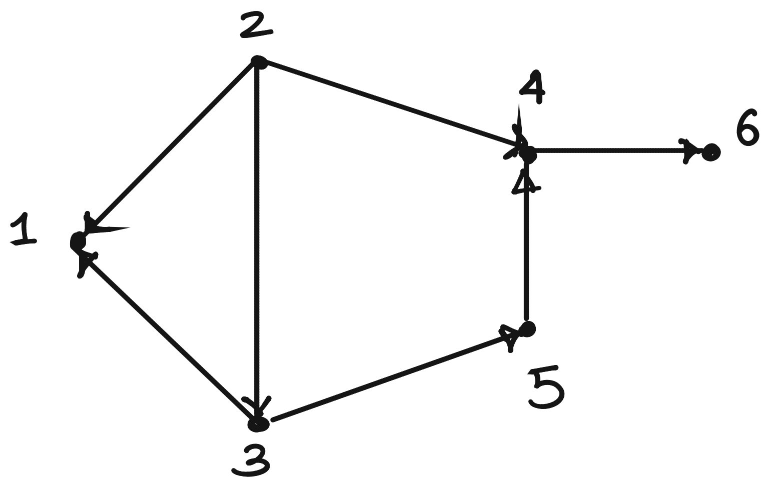
DFS



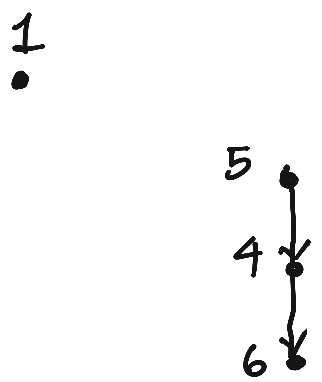
Q: IS THERE ANY INCOMING EDGE AT 3?

A: NO

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

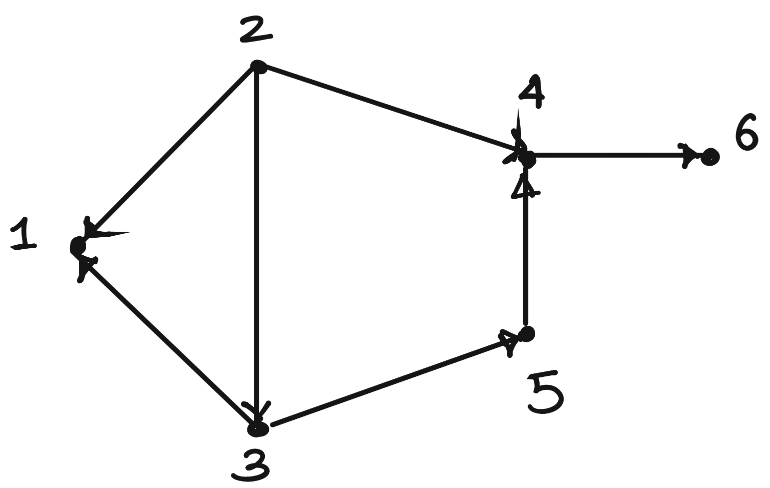


DFS

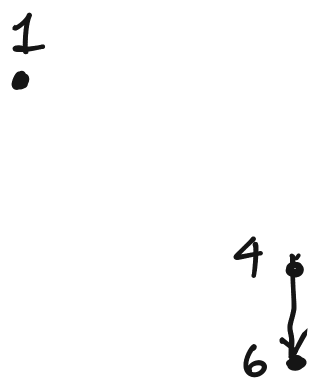


Q: IS THERE ANY INCOMING EDGE AT 5?

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

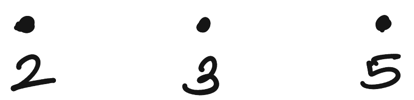


DFS

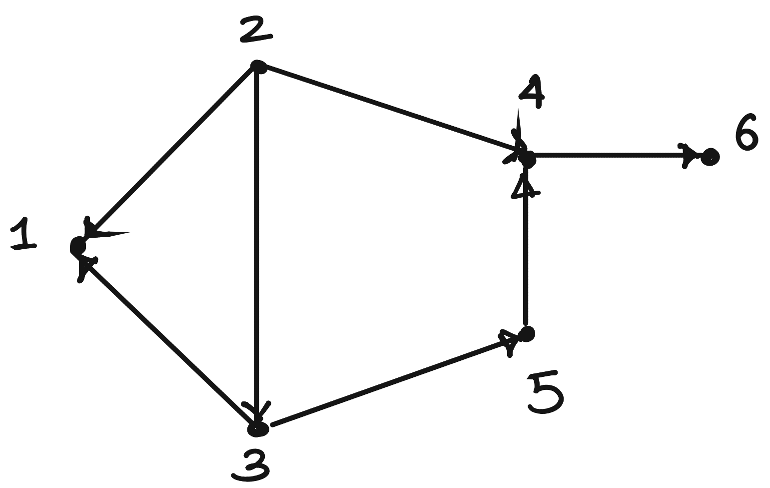


Q: IS THERE ANY INCOMING EDGE AT 5?

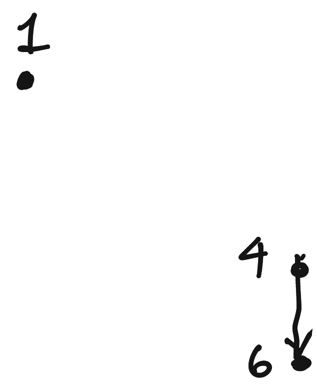
A: NO



PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u,v), u LIES TO THE LEFT OF v.

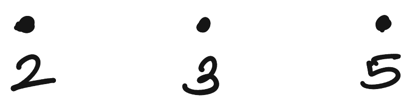


DFS



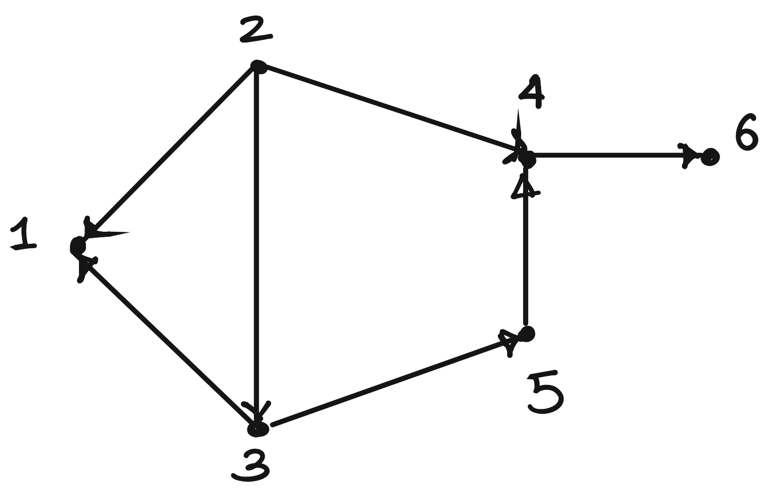
Q: IS THERE ANY INCOMING EDGE AT 5?

A: NO

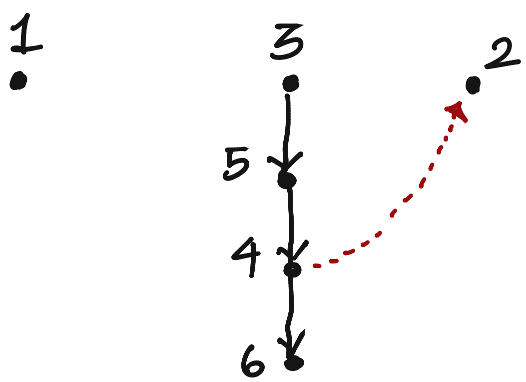


..... AND SO ON

PROBLEM 2: GIVEN A DIRECTED ACYCLIC GRAPH, ARRANGE THE NODES SUCH THAT FOR EACH (u, v), u LIES TO THE LEFT OF v.



Q: IS THERE AN EDGE FROM, say 4 to 2



THIS IS A CROSS EDGE GOING TO THE RIGHT CAN NOT EXIST.

NON-INTUITIVE BUT SIMILAR ALGO.

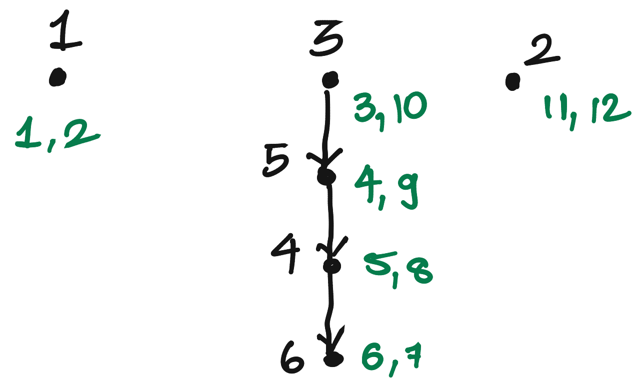
DO DFS IN G

ARRANGE THE NODES IN DECREASING ORDER OF THEIR DEPARTURE TIME

NON-INTUITIVE BUT SIMILAR ALGO.

DO DFS IN G

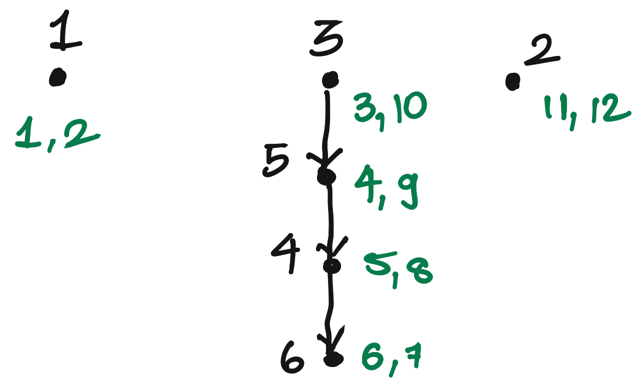
ARRANGE THE NODES IN DECREASING ORDER OF THEIR DEPARTURE TIME



NON- INTUITIVE BUT SIMILAR ALGO.

DO DFS IN G

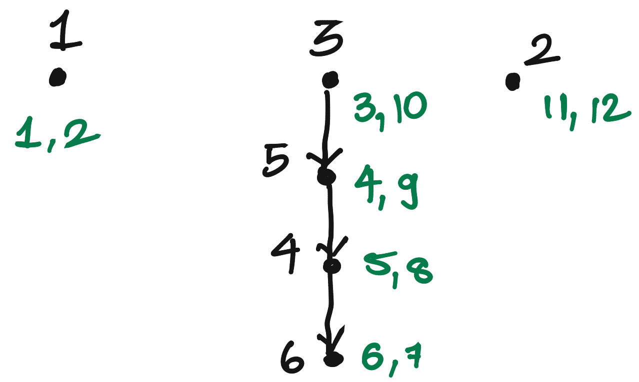
ARRANGE THE NODES IN DECREASING ORDER OF THEIR DEPARTURE TIME



GIVES SAME ORDER

NON-INTUITIVE BUT SIMILAR ALGO.

DO DFS IN G
ARRANGE THE NODES IN DECREASING ORDER OF THEIR DEPARTURE TIME



GIVES SAME ORDER

2 3 5 4 6 1

LEMMA : FOR EACH $(u,v) \in G$, $D(u) > D(v)$

LEMMA : FOR EACH $(u,v) \in G$, $D(u) > D(v)$

1) TREE EDGE (u, v)

$$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{DEPARTURE}[u]$$

LEMMA : FOR EACH $(u,v) \in G$, $D(u) > D(v)$

1) TREE EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v]
< DEPARTURE[u]

2) NON-TREE EDGES

a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v].
< DEPARTURE[u]

(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u]
< DEPARTURE[v]

(c) CROSS EDGE (u,v)

ARRIVAL[v] < DEPARTURE[v] < ARRIVAL[u]
< DEPARTURE[u]

LEMMA : For each $(u,v) \in G$, $D(u) > D(v)$

1) TREE EDGE (u,v)

✓ $\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v]$
$< \text{DEPARTURE}[u]$

2) NON-TREE EDGES

✓ a) FORWARD EDGE (u,v)

$\text{ARRIVAL}[u] < \text{ARRIVAL}[v] < \text{DEPARTURE}[v].$
$< \text{DEPARTURE}[u]$

~~(b) BACK EDGE (u,v)~~

~~ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u]~~
~~< DEPARTURE[v]~~

✓ (c) CROSS EDGE (u,v)

$\text{ARRIVAL}[v] < \text{DEPARTURE}[v] < \text{ARRIVAL}[u]$
$< \text{DEPARTURE}[u]$

DFS IN UNDIRECTED GRAPH.

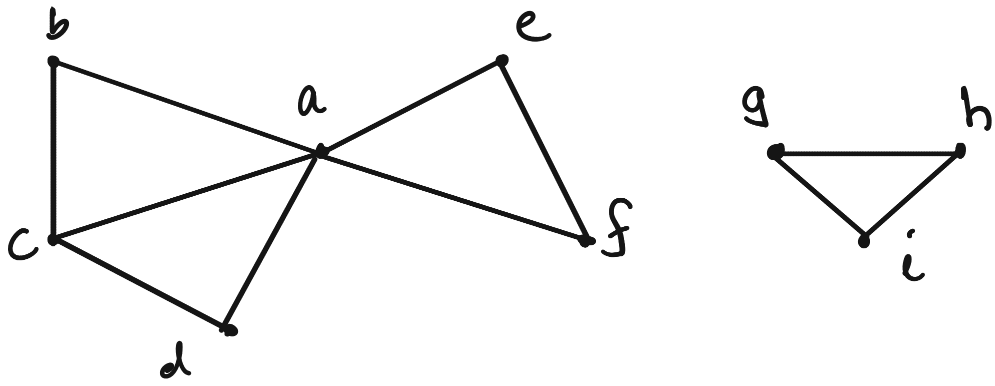
IS THERE ANY CHANGE IN THE ALGORITHM AND OBSERVATION.

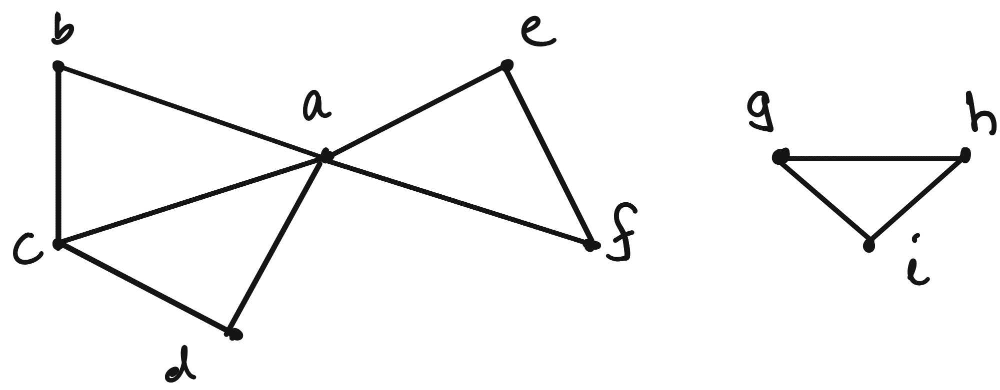
```
TIME ← 0
FOREACH v ∈ V
    VISITED[v] ← FALSE
FOREACH v ∈ V
{   IF ( VISITED[v] = FALSE)
        DFS(v)
}


DFS(v)
{   ARRIVAL[v] ← TIME
    TIME ← TIME+1;
    VISITED[v] ← TRUE
    FOREACH OUTGOING EDGE (v,w)
    {   IF ( VISITED[w] = FALSE)
        {
        }   DFS(w)
    }
    DEPARTURE[v] ← TIME;
    TIME ← TIME+1;
}
```

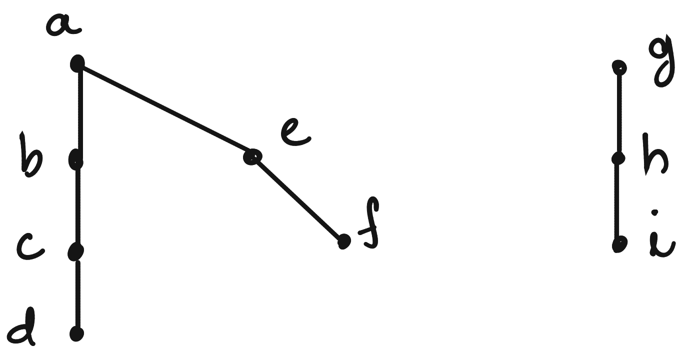
DFS IN UNDIRECTED GRAPH.

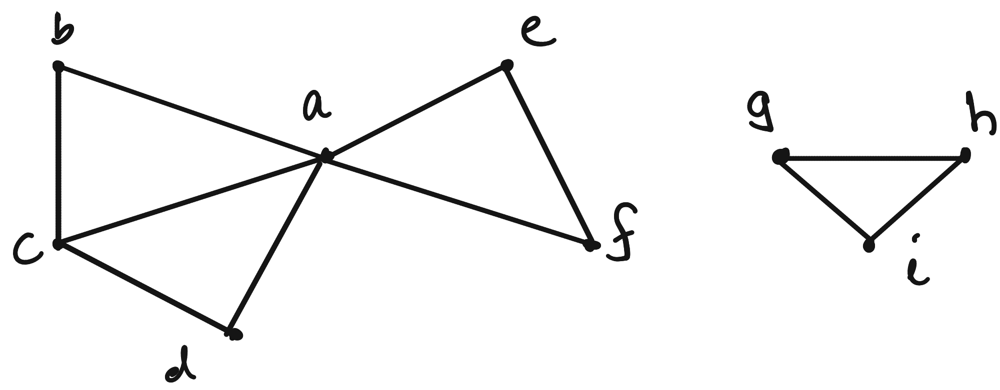
IS THERE ANY CHANGE IN THE ALGORITHM AND OBSERVATION.

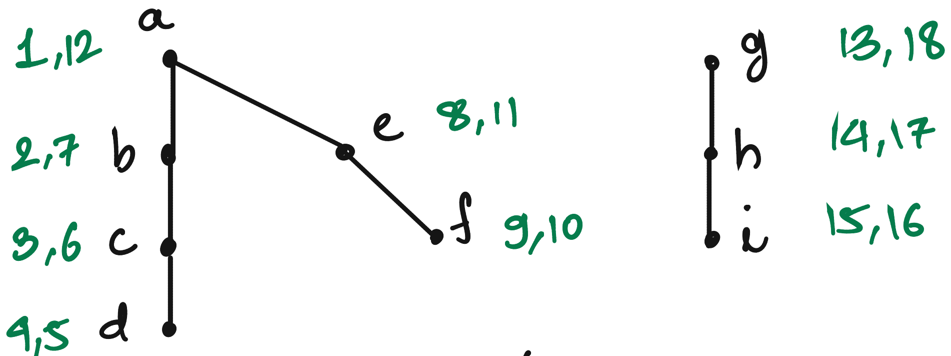
```
TIME ← 0
FOREACH v ∈ V
        VISITED[v] ← FALSE
FOREACH v ∈ V
{    IF ( VISITED[v] = FALSE)
          DFS(v)
}


DFS(v)
{    ARRIVAL[v] ← TIME
     TIME ← TIME+1;
     VISITED[v] ← TRUE
     FOREACH ~~OUTGOING~~ EDGE (v,w)
     {    IF ( VISITED[w] = FALSE)
          {
          }    DFS(w)
     }
     DEPARTURE[v] ← TIME;
     TIME ← TIME+1;
}
```


b
e
a
g
h
c
f
i
d


b
e
a
g
h
c
f
i
d


a
g
b
e
h
c
f
i
d





1) TREE EDGE (u, v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

2) NON - TREE EDGES

a) FORWARD EDGE (u,v)

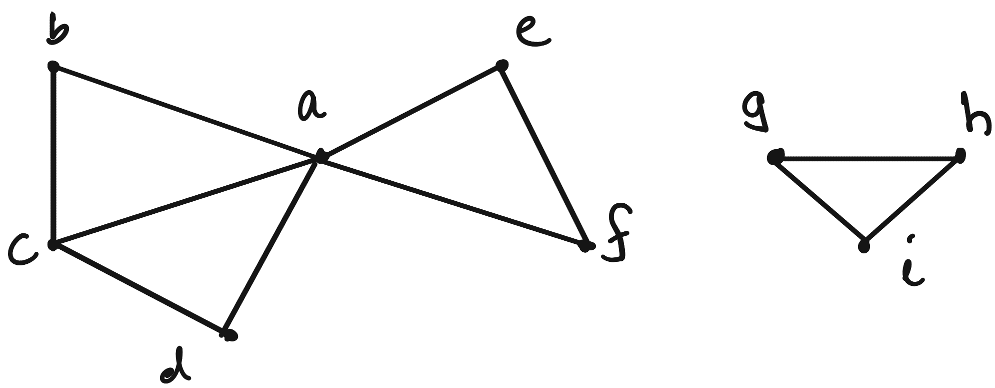
ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v]. < DEPARTURE[u]

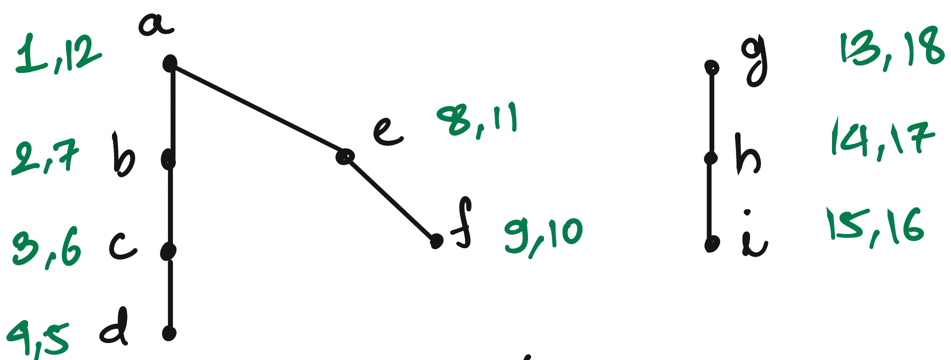
(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u] < DEPARTURE[v]

(c) CROSS EDGE (u,v)

ARRIVAL[v] < DEPARTURE[v] < ARRIVAL[u] < DEPARTURE[u]





1) TREE EDGE (u, v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v] < DEPARTURE[u]

2) NON - TREE EDGES

SAME

a) FORWARD EDGE (u,v)

ARRIVAL[u] < ARRIVAL[v] < DEPARTURE[v]. < DEPARTURE[u]

(b) BACK EDGE (u,v)

ARRIVAL[v] < ARRIVAL[u] < DEPARTURE[u] < DEPARTURE[v]

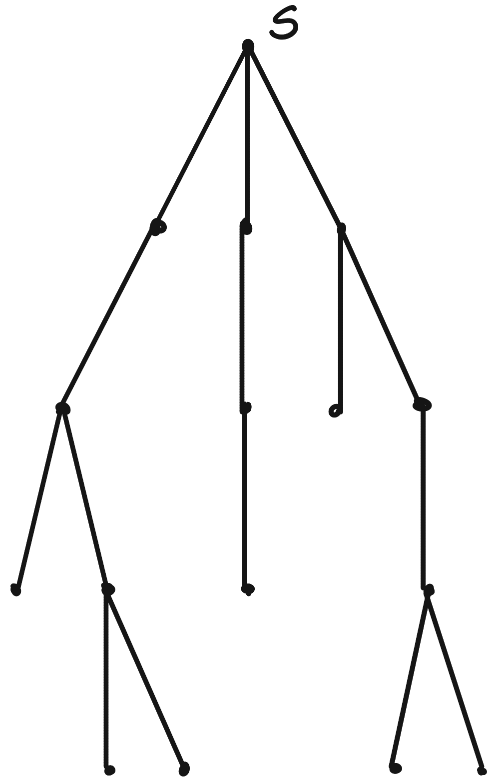
CANNOT EXIST.

(c) CROSS EDGE (u,v)

X ARRIVAL[v] < DEPARTURE[v] < ARRIVAL[u] < DEPARTURE[u]

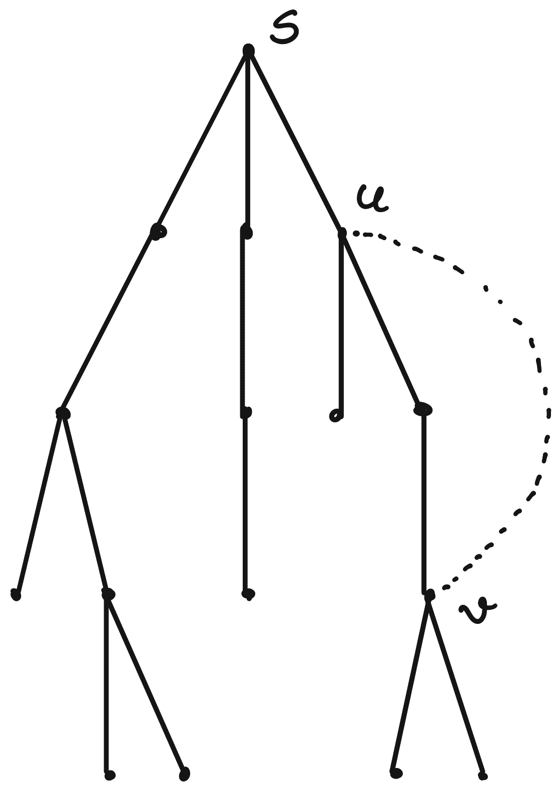
Q: GIVEN A CONNECTED GRAPH G, ITS DFS - TREE & BFS - TREE FROM A NODE S IS SAME, THEN G IS A ______

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE S IS SAME, THEN G IS A ______



LETS LOOK AT ALL NON TREE EDGES.

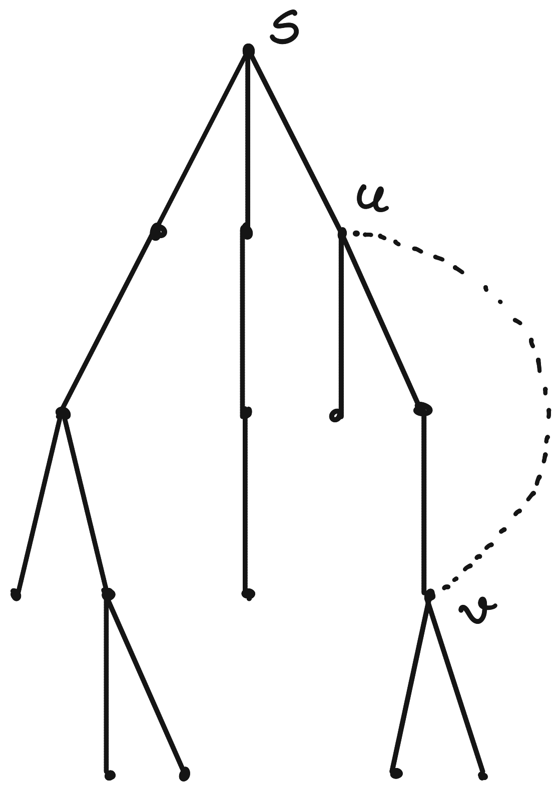
Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE S IS SAME, THEN G IS A ______



LETS LOOK AT ALL NON TREE EDGES.

Q: CAN THERE BE A NON TREE EDGE (u,v) ABOVE

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE s IS SAME, THEN G IS A ________

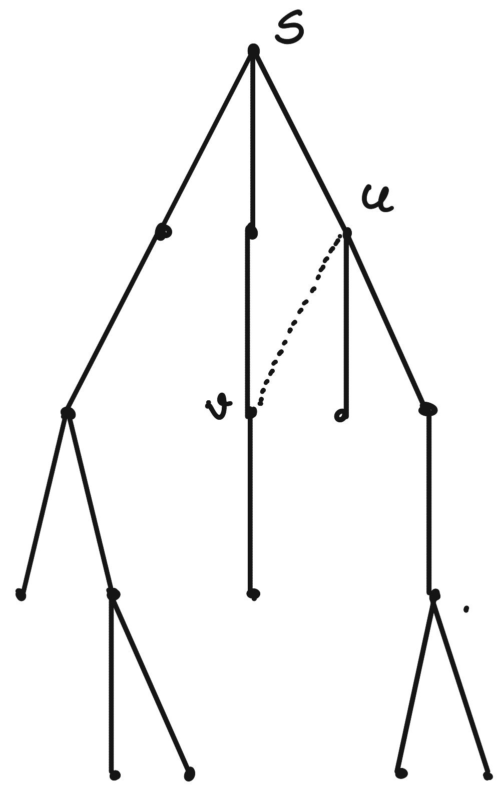


LETS LOOK AT ALL NON TREE EDGES.

Q: CAN THERE BE A NON TREE EDGE (u,v) ABOVE

A: NO NON-TREE EDGE WITH LEVEL DIFFERENCE >1 AS THIS VIOLATES BFS PROPERTY.

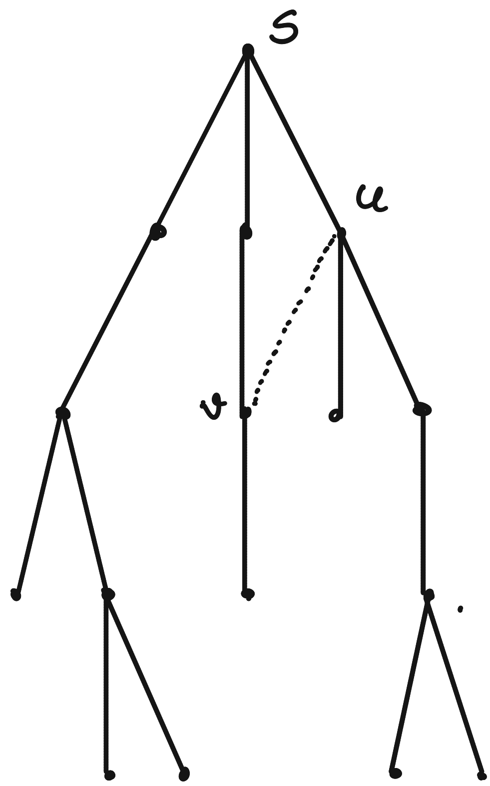
Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE S IS SAME, THEN G IS A ______



(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1

Q: CAN THERE BE A NON TREE EDGE WITH LEVEL DIFFERENCE 1 ?

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE s IS SAME, THEN G IS A ______

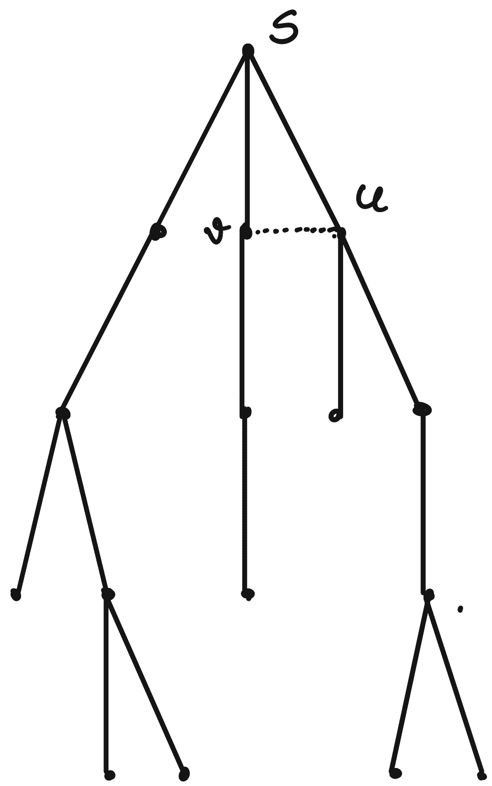


(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1

Q: CAN THERE BE A NON TREE EDGE WITH LEVEL DIFFERENCE 1?

A: NO CROSS EDGE CAN EXIST IN DFS TREE

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE s IS SAME, THEN G IS A ______

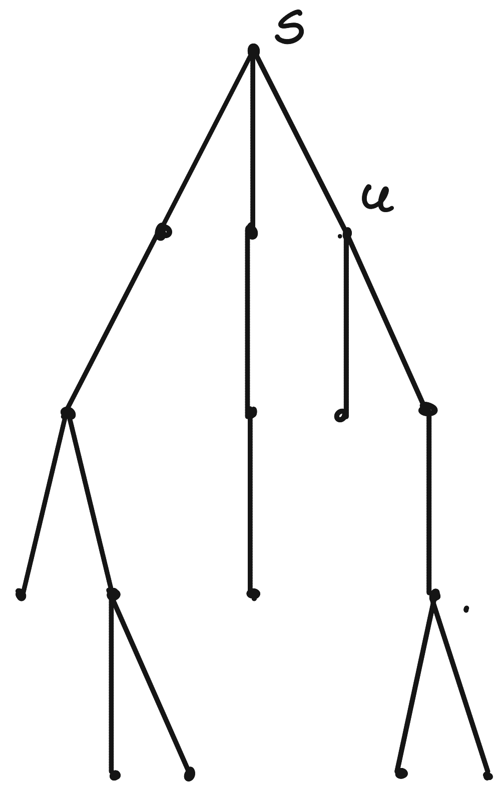


(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1

(2) NO NON-TREE EDGE WITH LEVEL DIFFERENCE = 1

Q: CAN THERE BE A NON TREE EDGE WITH LEVEL DIFFERENCE 0 ?

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE s IS SAME, THEN G IS A ______

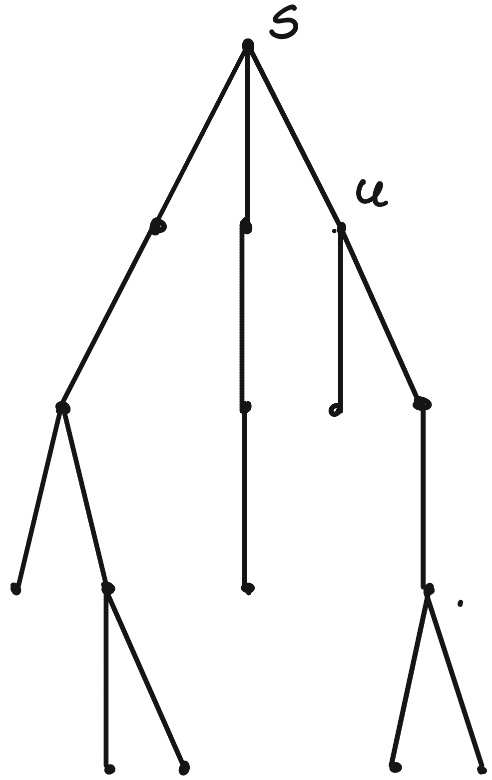


(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1
(2) NO NON-TREE EDGE WITH LEVEL DIFFERENCE = 1

Q: CAN THERE BE A NON TREE EDGE WITH LEVEL DIFFERENCE 0 ?

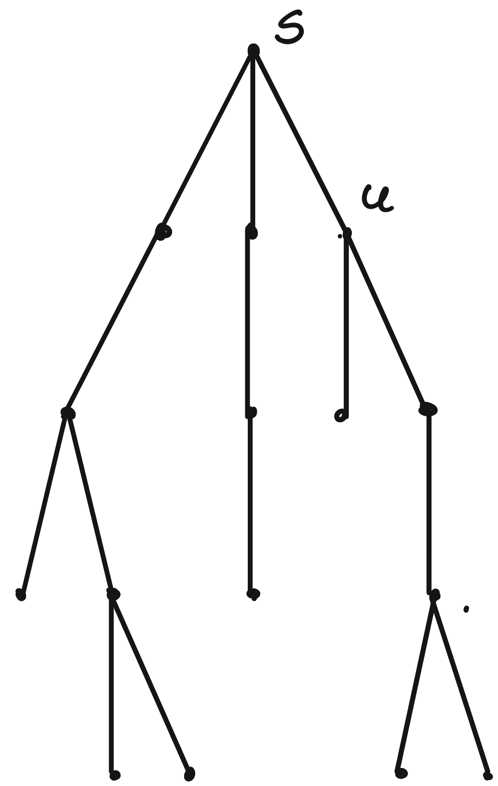
A: NO CROSS EDGE CAN EXIST IN DFS TREE

Q: GIVEN A CONNECTED GRAPH G, ITS DFS - TREE & BFS - TREE FROM A NODE S IS SAME, THEN G IS A ______



(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1
(2) NO NON-TREE EDGE WITH LEVEL DIFFERENCE = 1
(3) NO NON-TREE EDGE WITH LEVEL DIFFERENCE 0.

Q: GIVEN A CONNECTED GRAPH G, ITS DFS-TREE & BFS-TREE FROM A NODE S IS SAME, THEN G IS A ______



(1) NO NON-TREE EDGE WITH LEVEL DIFFERENCE > 1
(2) NO NON-TREE EDGE WITH LEVEL DIFFERENCE = 1
(3) NO NON-TREE EDGE WITH LEVEL DIFFERENCE 0.

$\Rightarrow$ NO NON-TREE EDGE IN G,
$\Rightarrow$ G IS A TREE.