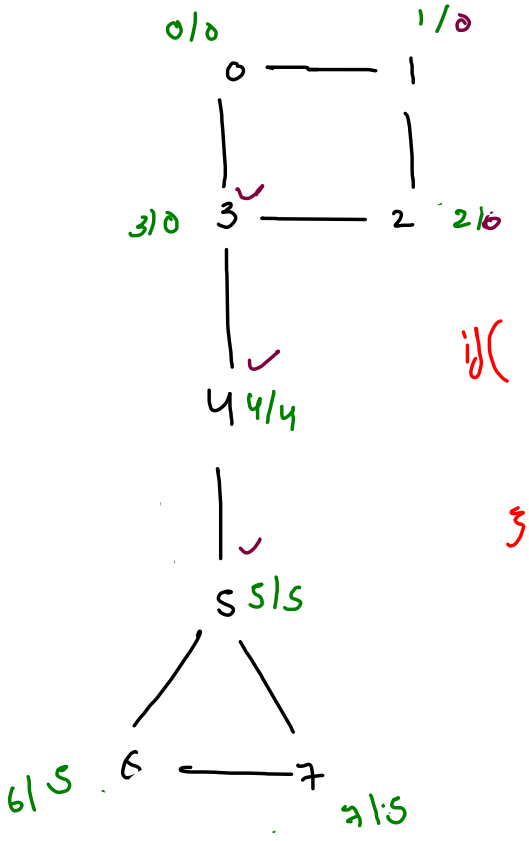
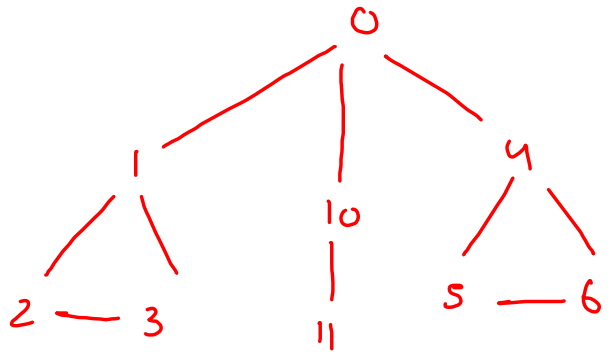
Articulation point

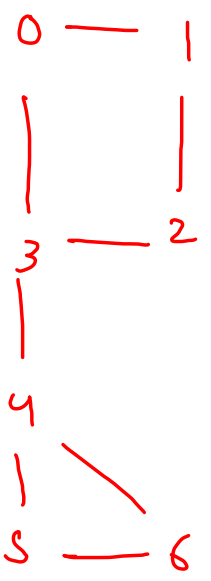



```
if( disc[src] <= low[nbr]) {
    src is an ap.
}
```

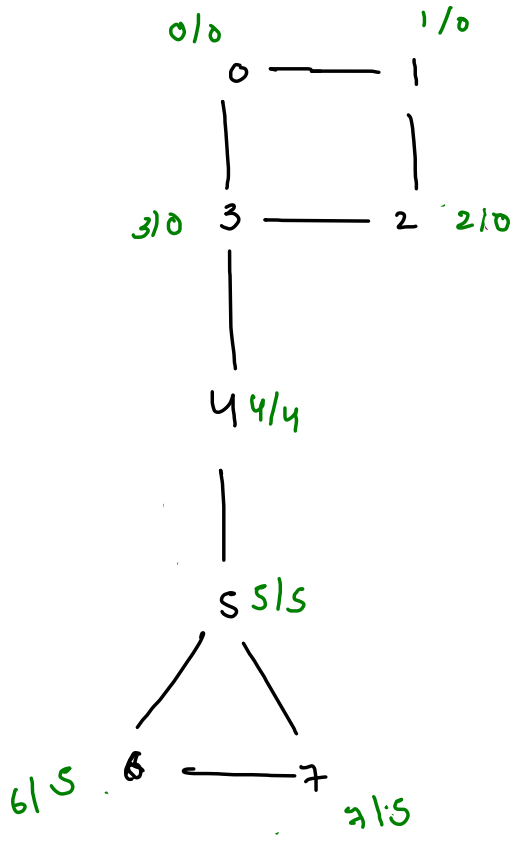
```
if( nbr == par) {
    //ignore
}
else if (nbr == unvisited) {
    -> go to the nbr
    low[src] = Math.min (low[src],
                         low[nbr])
}
else if (nbr == visited) {
    low[src] = Math.min (low[src],
                         disc[nbr])
}
```

0 is an ap





0 is not ap



bridge

```
if( disc[src] < low[nbr]) {
        src ..... nbr -> bridge
}
```

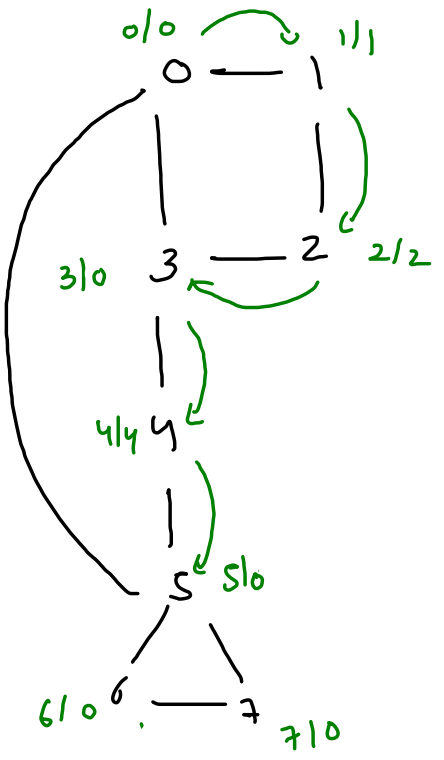
```java
if(nbr == par) {

}
else if(vis[nbr] == false) {
    bridge(graph,nbr,src,vis,ans);
    low[src]  = Math.min(low[src],low[nbr]);
}
else if(vis[nbr] == true) {
    low[src] = Math.min(low[src],disc[nbr]);
}
```

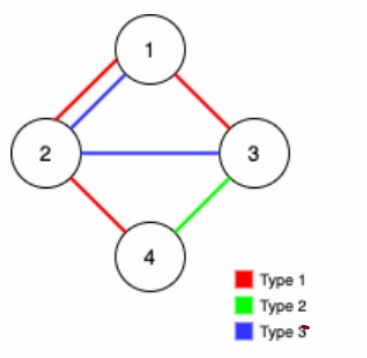
low

```java
if(src != 0 && disc[src] <= low[nbr]) {
    ap[src] = true;
}
```

why not to use vis nbr's low time ?

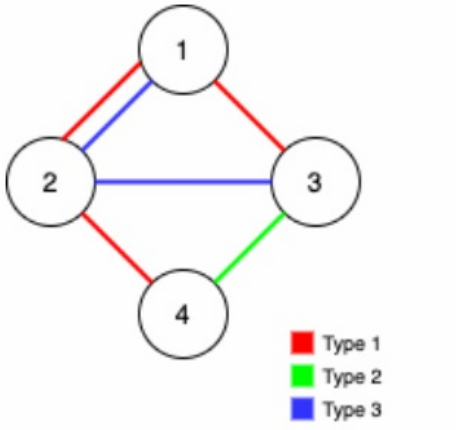


1579. Remove Max Number of Edges to Keep Graph Fully Traversable



max edges removal

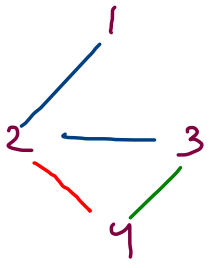
→ use min edges through which graph is fully traversable by alice & bob.



1 → alice

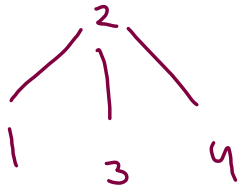
2 → Bob

3 → both

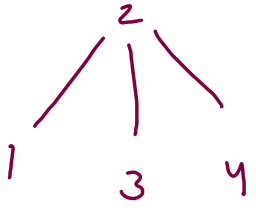


type	u	v	
3	2	3	✓
3	1	2	✓
2	3	4	✓
1	1	3	✗
1	2	4	✓
1	1	2	✗

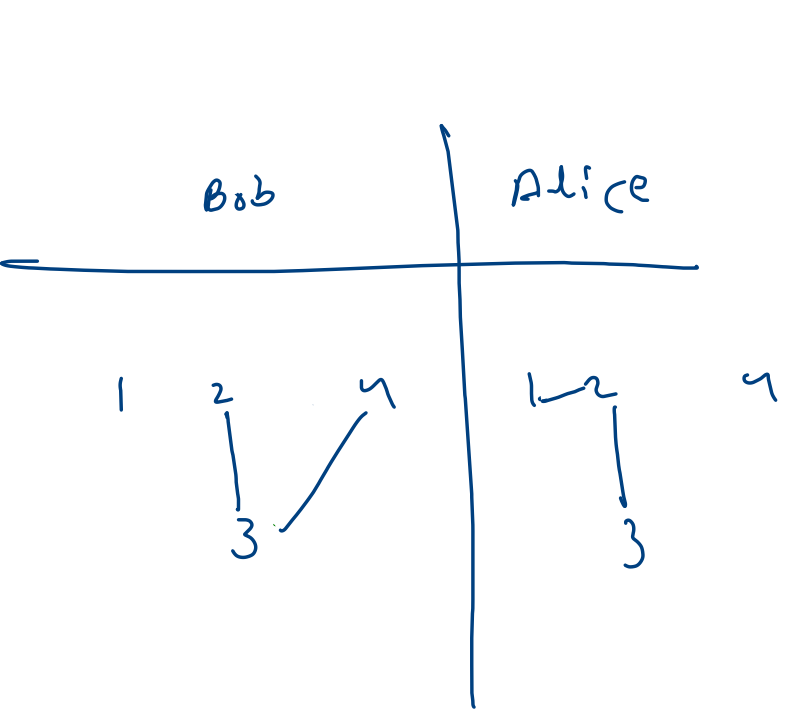
Bob

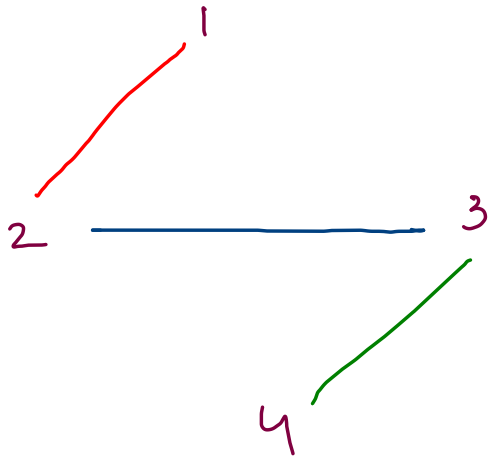


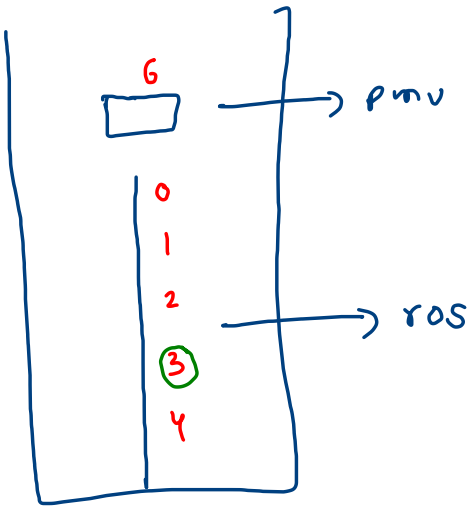
Alice



4 [[3,1,2],[3,2,3],[1,1,3],[1,2,4],[1,1,2],[2,3,4]] 4 [[3,1,2],[3,2,3],[1,1,4],[2,1,4]] 4 [[3,2,3],[1,1,2],[2,3,4]] Output 2







✓✓ pmv ✓ ros X

✓✓ pmv ✓ ros ✓

✓✓ pmv X ros X

pmv X ros ✓

↳ impossible