Here assumtion is that the graph is connected and undirected

In [1]:
```
#graph
G={1:{5},
   2:{5,6},
   3:{5,7},
   4:{7,8},
   5:{1,3,2},
   6:{3},
   7:{3,4},
   8:{4}
  }
```

In [2]:
```
visited={}
time={}
```

In [3]:
```
# marking all visited equals to zero
for a in G.keys():
    visited[a]=0
```

In [4]:
```
# marking all time equals to zero
for a in G.keys():
    time[a]=0
```

Here for starting node we count 0 and when we visited a DFS tree(edge) we increse the count and take modulas 1.

Since for a biparteite graph there should not edge betwees a group of vertices so only tree edge occurs when we

visit different group

In [5]:
```
def DFS(a,i):
    visited[a]=1
    for u in G[a]:
        if visited[u]==0:
            i=i+1 #when first visit we increse count 1
            time[u]=i%2
            DFS(u,i)
            i=i+1 #when the node is finished(exploied) we increse count 1
```

DFS

In [6]:
```
DFS(1,0)
```

As the time divided the marked into 0 and 1 group we collect that data into set C and D

In [7]:
```
A={}
B={}
i=0
j=0
for a in time.keys():
    if time[a]==0:
        A[a]=i
        i+1
    if time[a]==1:
        B[a]=j
        j+1
C={*A}
D={*B}
```

Now we went c node in C abd check is there is any node which is not in D. If this happen then there is a

edge between C to C self. So for a baipartrite graph the set theoritic difference nust be zero.

In [8]:
```
l=0
for a in C:
    P={*G[a]}
    L=P-D
    k=len(L)
    l=l+k

for a in D:
    P={*G[a]}
    L=P-C
    k=len(L)
    l=l+k
```

Final concluasion

In [9]:
```
if l==0:
    print(" the Graph is baipartritr")
    print("First set of vartices   "+str(C))
    print("Second set vartices  "+str(D))
if l>0:
    print("the graph is not baipartrite")
```

```
the graph is not baipartrite
```