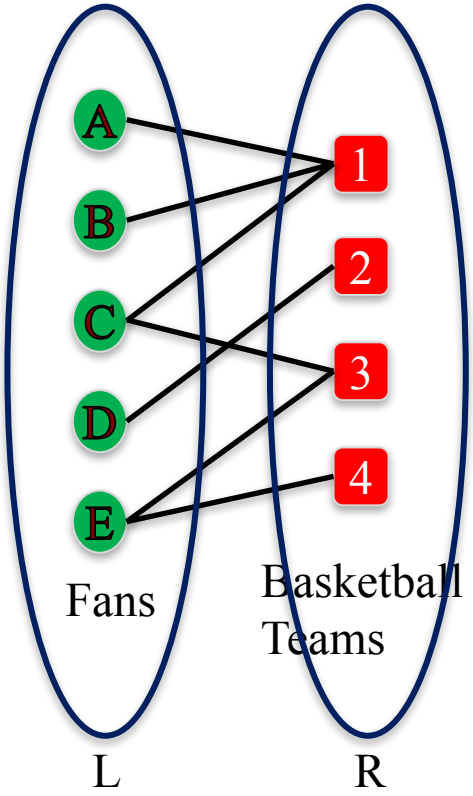
Bipartite Graphs

Bipartite Graph: a graph whose nodes can be split into two sets *L* and *R* and every edge connects an node in *L* with a node in *R*.

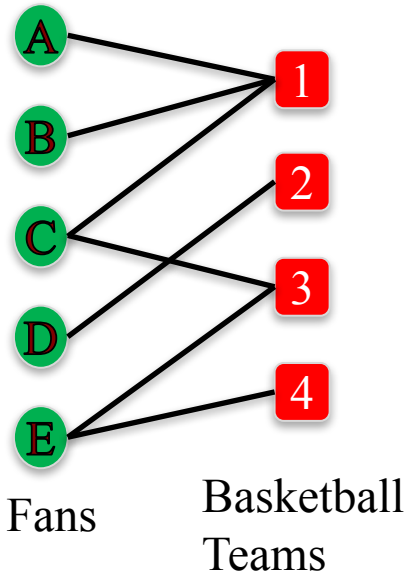
```
from networkx.algorithms import bipartite
B = nx.Graph() #No separate class for bipartite graphs
B.add_nodes_from(['A','B','C','D', 'E'], bipartite=0) #label one set of nodes 0
B.add_nodes_from([1,2,3,4], bipartite=1) #label other set of nodes 1
B.add_edges_from([('A',1), ('B',1), ('C',1), ('C',3), ('D',2), ('E',3), ('E', 4)])
```



Bipartite Graphs

Checking if a graph is bipartite:

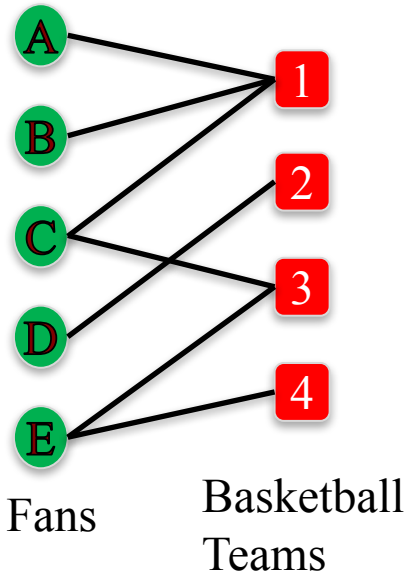
```
In: bipartite.is_bipartite(B) # Check if B is bipartite
Out: True

In: B.add_edge('A', 'B')
In: bipartite.is_bipartite(B)
Out: False

B.remove_edge('A', 'B')
```



Bipartite Graphs

Checking if a set of nodes is a bipartition of a graph:

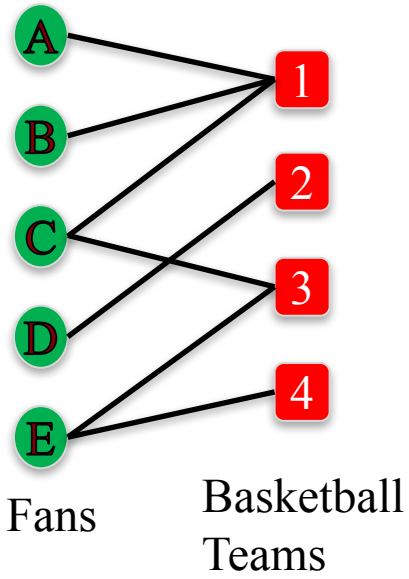
```
In: X = set([1,2,3,4])
In: bipartite.is_bipartite_node_set(B,X)
Out: True

X = set(['A', 'B', 'C', 'D', 'E'])
In: bipartite.is_bipartite_node_set(B,X)
Out: True

X = set([1,2,3,4, 'A'])
In: bipartite.is_bipartite_node_set(B,X)
Out: False
```



Bipartite Graphs

Getting each set of nodes of a bipartite graph:

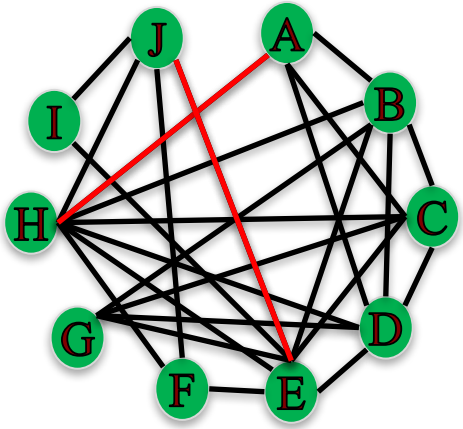
```
In: bipartite.sets(B)
Out: ({'A', 'B', 'C', 'D', 'E'}, {1, 2, 3, 4})

In: B.add_edge('A', 'B')
In: bipartite.sets(B)
Out: NetworkXError: Graph is not bipartite.

B.remove_edge('A', 'B')
```



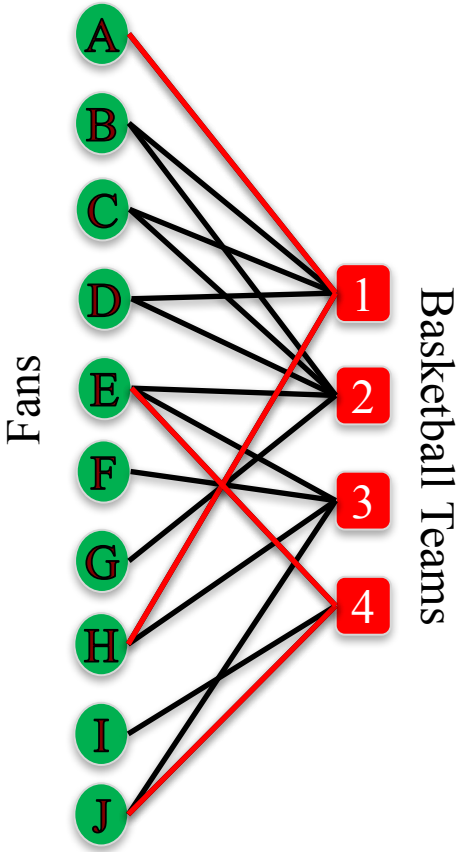
Projected Graphs

***L*-Bipartite graph projection:** Network of nodes in group *L*, where a pair of nodes is connected if they have a common neighbor in *R* in the bipartite graph.

Similar definition for *R*-Bipartite graph projection

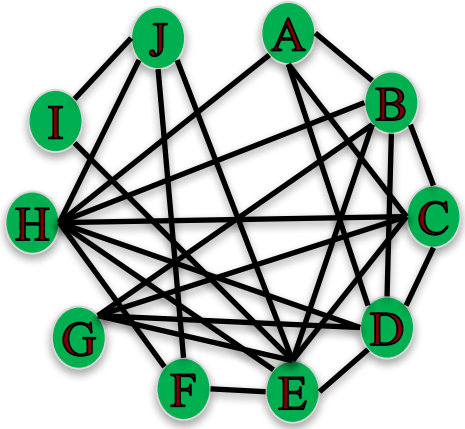


Network of fans who have a team in common

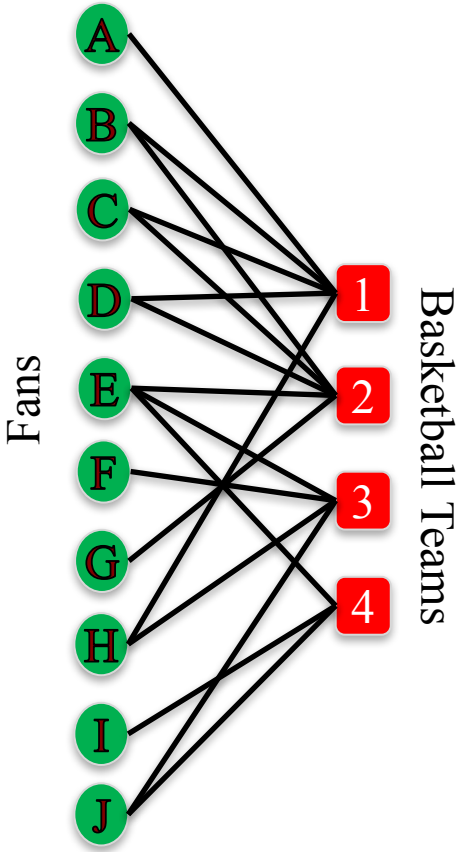


Projected Graphs

```
B = nx.Graph()
B.add_edges_from([('A',1), ('B',1),
('C',1),('D',1),('H',1), ('B', 2), ('C', 2), ('D',
2),('E', 2), ('G', 2), ('E', 3), ('F', 3), ('H', 3),
('J', 3), ('E', 4), ('I', 4), ('J', 4) ])

X = set(['A','B','C','D', 'E', 'F','G', 'H', 'I','J'])
P = bipartite.projected_graph(B, X)
```

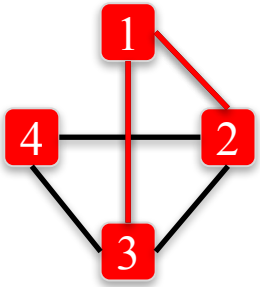


Network of fans who have a team in common

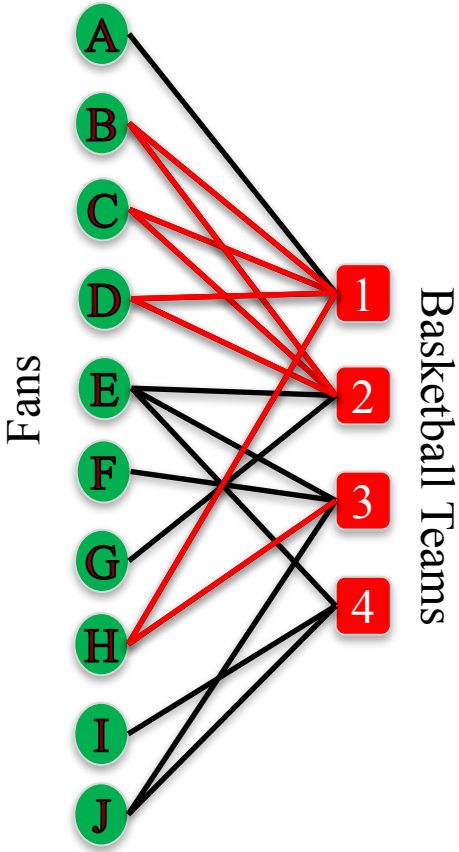


Projected Graphs

```
B = nx.Graph()
B.add_edges_from([('A',1), ('B',1),
('C',1),('D',1),('H',1), ('B', 2), ('C', 2), ('D',
2),('E', 2), ('G', 2), ('E', 3), ('F', 3), ('H', 3),
('J', 3), ('E', 4), ('I', 4), ('J', 4) ])

X = set([1,2,3,4])
P = bipartite.projected_graph(B, X)
```



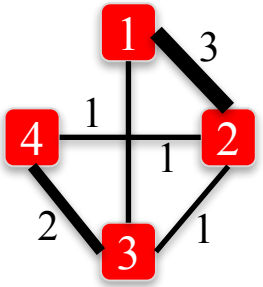
Network of teams who have a fan common



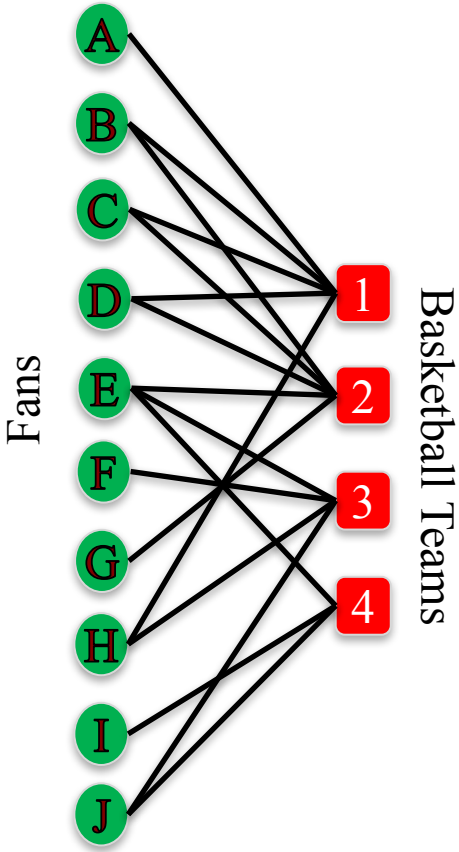
We need weights on the edges!

Projected Graphs

***L*-Bipartite weighted graph projection:** An *L*-Bipartite graph projection with weights on the edges that are proportional to the number of common neighbors between the nodes.



Weighted Network of teams who have a fan common




```
X = set([1,2,3,4])
P = bipartite.weighted_projected_graph(B, X)
```

Summary

No separate class for bipartite graphs in NetworkX
Use Graph(), DiGraph(), MultiGraph(), etc.
Use from networkx.algorithms import bipartite for bipartite related algorithms (Many algorithms only work on Graph()).

- nx.bipartite.is_bipartite(B) # Check if B is bipartite
- bipartite.is_bipartite_node_set(B,X) # Check if node set X is a bipartition
- bipartite.sets(B) # Get each set of nodes of bipartite graph B
- bipartite.projected_graph(B, X) # Get the bipartite projection of node set X
- bipartite.weighted_projected_graph(B, X) # Get the weighted bipartite projection of node set X