**pyg基本概念**

1. **数据处理**

data.x：具有形状的节点特征矩阵[num\_nodes, num\_node\_features]

data.edge\_index：具有形状和类型的COO 格式的图形连接[2, num\_edges]torch.long 2：起点到终点

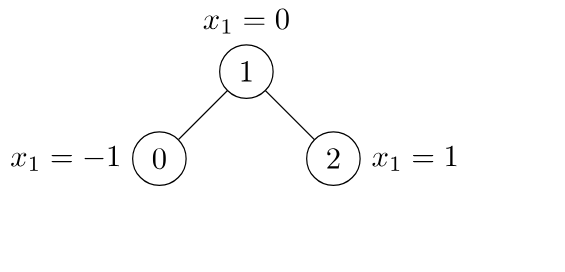
data.edge\_attr：具有形状的边缘特征矩阵[num\_edges, num\_edge\_features]

data.y：训练目标（可以具有任意形状），例如形状的节点级目标或形状的图形级目标[num\_nodes, \*][1, \*]

data.pos：具有形状的节点位置矩阵[num\_nodes, num\_dimensions]

例子：

1. import torch
2. from torch\_geometric.data import Data
3. edge\_index = torch.tensor([[0, 1, 1, 2],
4. [1, 0, 2, 1]], dtype=torch.long)
5. x = torch.tensor([[-1], [0], [1]], dtype=torch.float)
6. data = Data(x=x, edge\_index=edge\_index)
7. >>> Data(edge\_index=[2, 4], x=[3, 1])



尽管图上只有两条边，但实际上是四条（有方向）

注意:

定义所有边的源节点和目标节点的张量，不是索引元组的列表。如果您想以这种方式编写索引，则应该contiguous在将它们传递给数据构造函数之前转置并调用它：

1. import torch
2. from torch\_geometric.data import Data
3. edge\_index = torch.tensor([[0, 1],
4. [1, 0],
5. [1, 2],
6. [2, 1]], dtype=torch.long)
7. x = torch.tensor([[-1], [0], [1]], dtype=torch.float)
8. data = Data(x=x, edge\_index=edge\_index.t().contiguous())
9. >>> Data(edge\_index=[2, 4], x=[3, 1])

**图学习方法**