**PrimePath程序说明：**

**程序：**primepath.py

**输入：**所有的结点编号以及结点对应的出边，没有出边的结点则不输入出边。

**输出：**所有的Prime Path以及总数

**程序流程步骤：**

1. 输入所有结点以及结点对应的出边
2. 调用primepath(nodes, edges)函数，返回对应于输入的图的一个包含所有Prime Path的列表。
3. 在primepath(nodes, edges)函数中算法步骤：
4. 计算输入的结点数目n，初始化路径列表，其中的路径数目为输入的结点数目n，每条路径中仅包含一个结点。
5. 遍历路径列表中的每条路径，以每条路径中的最后一个结点为起点寻找该结点对应的下一个结点，并将其附加到原路径中构成一条新的路径添加到路径列表中。将得到的新的路径列表更新之前的路径列表。循环执行n次（n为结点数目）。
6. 从上一步骤得到的路径列表中去除那些存在内部循环的路径。具体步骤：

遍历每条路径，对每条路径中的每个结点检查该路径中是否存在其他相同的结点，如果存在两个相同的结点，再判断这两个结点是否是首尾结点，若相同的两个结点不是首尾结点，即代表该路径存在内部循环，则从路径列表中去除该路径。其他情况则保留该路径。

1. 从第3步骤得到路径列表中去除子路径。具体步骤：

先对所得到的路径进行排序，从短到长排序。排序的目的是为了减少比较次数，排完序后只需将当前路径与其后面的路径进行判断是否是子路径，而不需要与路径列表中的所有路径进行比较。设置一个冗余标志，初始值为False，用来标志当前路径是否是其他路径的子路径，如果是则为True。

遍历路径列表中的每条路径，将当前路径其后面的路径进行比较，判断当前路径是否出现在其他路径中，如果出现了，则将冗余标志设为True，跳出循环。每条路径检查完之后判断冗余标志，如果为False则代表该路径不是其他路径的子集，则将该路径添加到新的路径列表中。遍历完之后得到一个路径列表即为所有的Prime Path。

（5）输出所有的Prime Path以及总数目