首先，我们创建了一个类，用来存储NFA，有两个成员变量，分别是节点数（也就是状态数），还有存储所有边的数组

类中有三个方法，一个是通过起点和邻接点获取边，也就是根据两个状态来获取非终结符，如果这两个状态无法到达，则返回-1，否则返回终结符；一个是根据起点和边来获取所有能到达的邻接点，如果没有则返回一个空数组；最后一个是根据起点、邻接点、边来构建图

初始化NFA，e代表ε

wait代表待处理的状态集，state\_name用来存储最终状态转换矩阵的表的第一列，new\_set用来存储整个状态转换矩阵，name就是对最终求得的状态转换矩阵的第一列的状态集重新标号

先求初始状态的闭包，eps\_cover函数就是用来求某个状态集的闭包。先将该状态集加入结果，然后从这些状态出发，求走一条ε能到哪些状态，去重后加入结果，然后再从新的结果集出发继续求闭包，一直递归，直到闭包结束。

然后进入循环，终止条件为没有要处理的状态集了，也就是说状态转换矩阵中第一列全部处理完了。

从待处理的状态集中取出第一个，然后看经过每个终结符它能到达哪些状态，然后对这些状态取闭包。这样就求得了状态转换矩阵的一行，然后加入状态转换矩阵。对第一列状态重新标号。

然后判断，如果状态转换矩阵的第二列和第三列已经处理过了，则不将他们加入到待处理的状态集

dfa是一个字典，用来存储还未最少化的dfa，因为使用的算法需要用到字典，所以没有再使用自己定义的类；然后初始化dfa 、终态集和非终态集

最小化函数。用set来表示等价类，set这个数据结构就相当于集合，不会存在重复的元素，还能进行集合的运算，会用到。

P用来存储每个阶段划分好的等价类集合，W是目的等价类(判断哪些状态可以经过char到达该等价类)。

然后进入循环，终止条件为没有要目的等价类了，也就是说等价类已经划分好了。

从W中随变取一个目的等价类A，然后移除掉。对于每个终结符，判断上一个阶段求得的等价类Y中，哪些状态可以经过该终结符到达A，如果有不属于等价类Y的状态，则将等价类Y进行拆分，把能到目的等价类A的和不能到的拆开。如果等价类Y在目的等价类中，则先把它从中删除掉，否则直接把拆分后的结果加入到目的等价类；如果没有不属于等价类Y的状态，说明等价类Y是暂时不需要拆分的，直接将Y加入新划分的等价类集合

把所有终结符处理完后，更新P，也就是新划分的等价类集合