本程序运行需要输入对应**文法的txt文件路径**以及需要进行语法分析的对应的**字符串**

运行后的待输入状态如图：





Tip：如果文法目录为空，系统则会自动检索本程序py文件同目录下的grammar.txt文件，如果依然不存在则会自动按照["E -> E + T","E -> T","T -> T \* F","T -> F","F -> ( E )","F -> id"]文法进行创建grammar.txt文件。

Tip：字符串的输入则要以空格隔开每一个字符，文法文件中的文法也是同理。

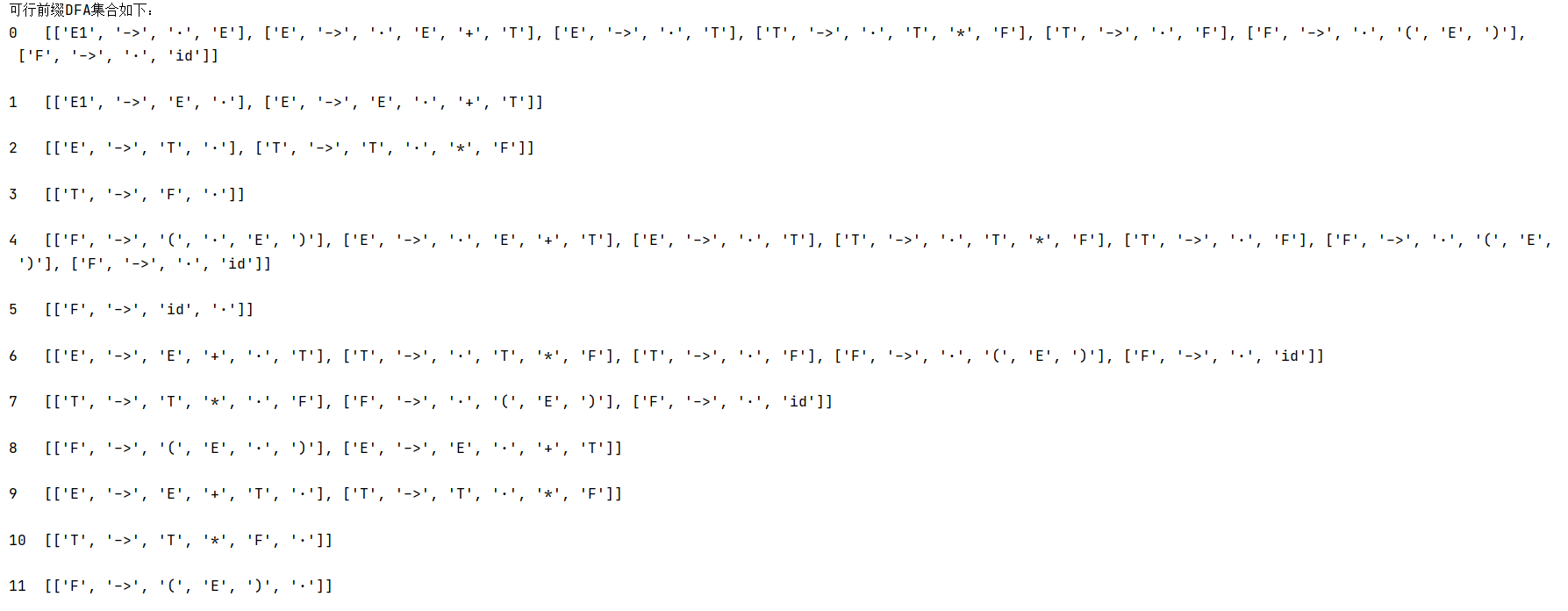
Tip：截图以默认文法：["E -> E + T","E -> T","T -> T \* F","T -> F","F -> ( E )","F -> id"]

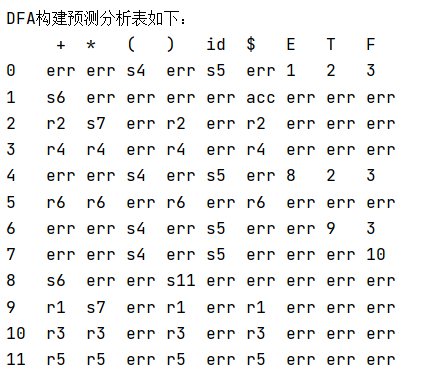
字符串：id + id \* id \* ( id + id )生成。

Tip：本程序由于纯手写可能部分算法执行效率略低，请谅解！

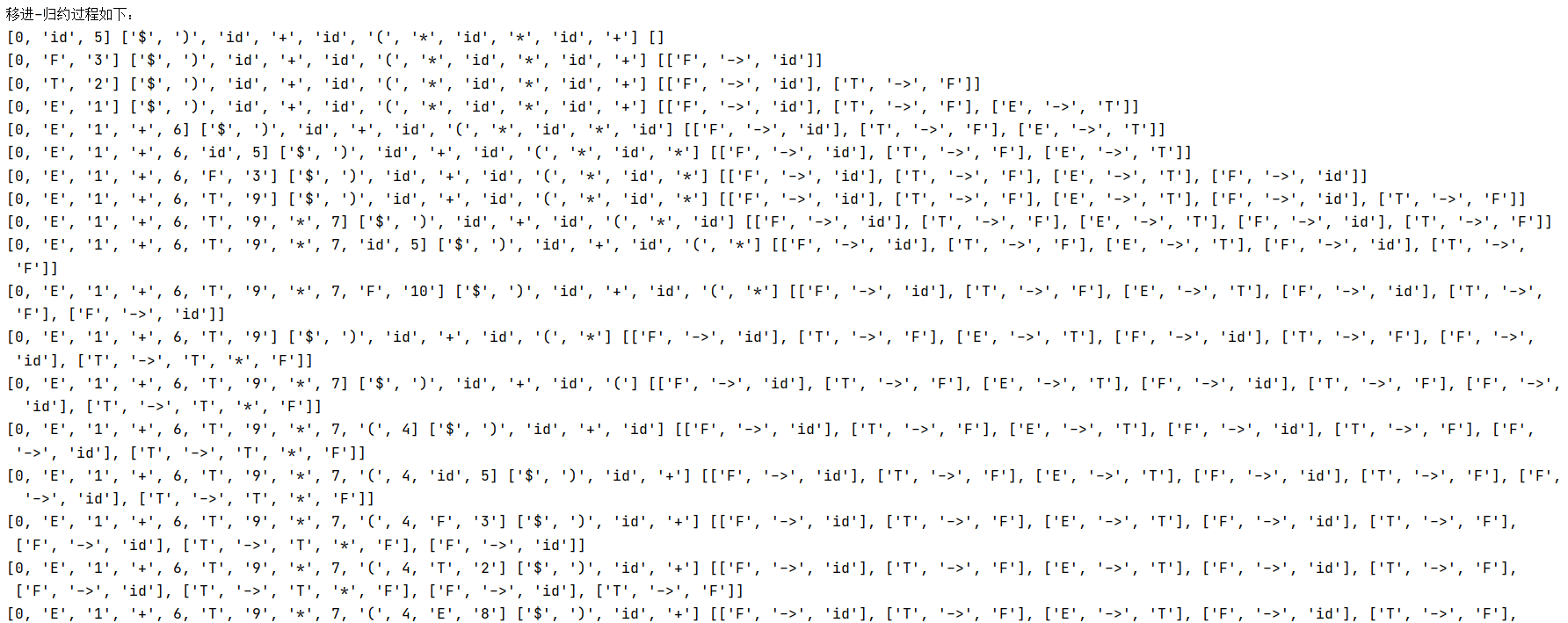
运行后程序会在控制台上显示：可行前缀DFA集合，DFA构建预测分析表，对应归约集合，移进-归约过程以及结果字符串（结果字符串也会保存在SLR.py同目录下result.txt文件中）。

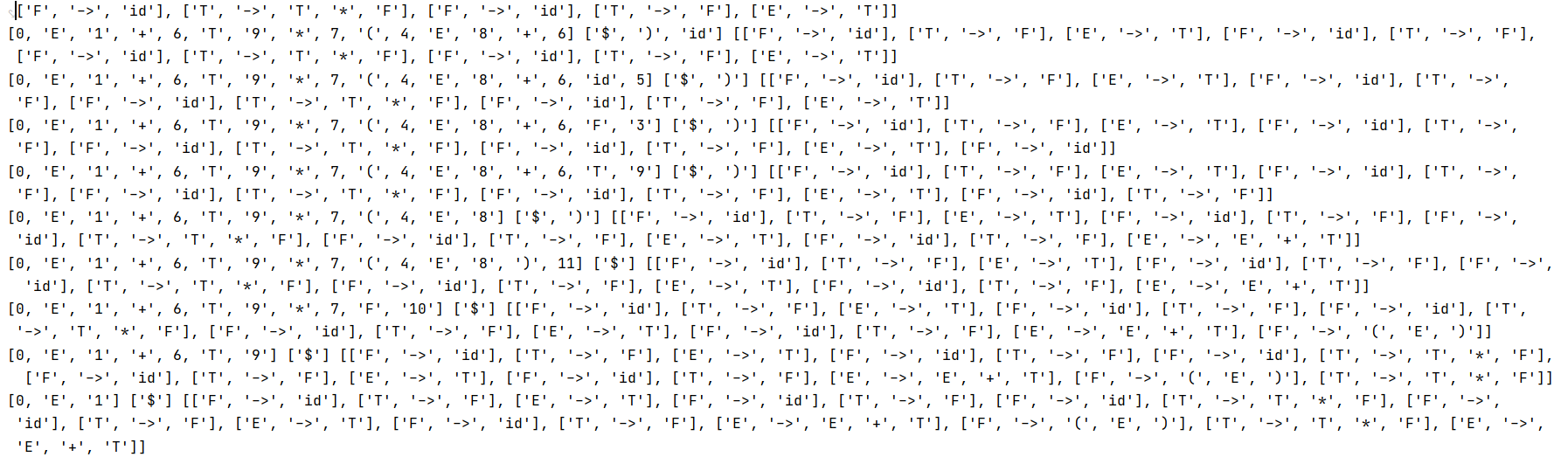
对应结果如下图：

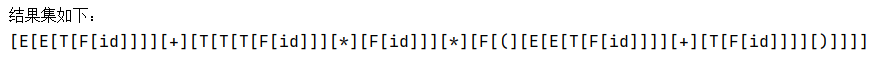


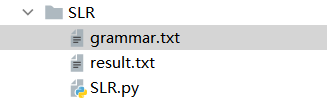


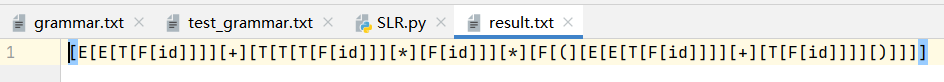


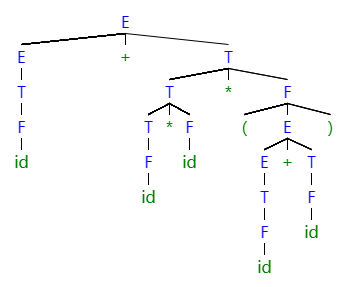






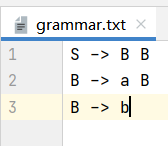






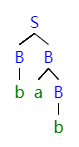
其他运行结果：

文法： 字符串：b a b

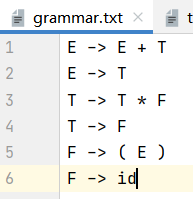


结果集：[S[B[b]][B[a][B[b]]]]

分析树：



文法： 字符串：id + ( id + id ) \* id



结果集：[E[E[T[F[id]]]][+][T[T[F[(][E[E[T[F[id]]]][+][T[F[id]]]][)]]][\*][F[id]]]]

分析树：

