1. 说明
   1. 展示



table.txt包含三列，第一列State表示状态栈，第二列Stack表示符号栈，第三列Input表示输入流中剩余的字符串。

* 1. 约定

特殊字符’->’,’|’,’e’分别表示产生式推导符号、或和空。

单词之间通过一个空格隔开，’e’被认为是一个单词。

使用lr.add\_input()添加产生式，使用lr.set\_string()设置输入单词流。

1. 实现
   1. 切割

对输入的字符串切割，切割方式与LL1（类似），不同点在于对于‘e’，此处将推导出’e’的原子产生式将的右部的单词列表处理为空列表，即在最后推导的初始状态栈和符号栈需要规约空时不会弹栈，可以顺利进行下一步。

* 1. 项目组推导
     1. 概述

项目组推导包含两个部分，第一部分为补全当前项目组，第二部分为根据当前项目组向外新建项目组，当遍历完所有项目组，迭代结束。

* + 1. 补全

每个项目组根据一个或若干个“种子”生成完整的项目组。对于第一个项目组，该项目组的第一个项目为’**S’->.S**’,从该项目派生出该项目组的其他项目，对每一个项目重复上述操作，每次项目派生结束后，更新所有项目的展望字符，所有项目派生完成后，对项目和每个项目的展望单词去重。

* + 1. 外推

当前项目组补全（包括去重）结束后，重新遍历该项目组的每个项目，对于点没有遍历到末尾且点当前指向的单词为非终结符的项目，根据当前点指向的非终结符根据原子产生式产生新的项目集和新的项目集的“种子”，对“种子”查重，查重通过后将该新项目组添加进项目组列表。

* 1. Parsing\_table推导

在2.2.3构建新的项目组时，可以构建[原状态, 目标状态, 跳转值]的三元组列表，这个三元组列表构成一张无向图，对于所有出度为0的端点即为发生规约的端点。Parsing\_table使用二维字典存储，第一维的key是状态，整型数，第二维的key是单词，第二维的value根据跳转和规约分为整型数和列表，对于终结符和非终结符根据key区分。对于无向图中出度为0的端点在二维字典中以[左部, [右部单词]]的列表形式表示。

* 1. 分析推导

对于Parsing\_table,第一维的key即为状态栈栈顶，第二维的key即为当前输入单词流索引为0的单词。如果对应的value为int，将该value压入状态栈，将当前输入单词流索引为0的单词压入符号栈，将当前输入单词流索引为0的单词从输入单词流除去。如果对应的value为列表，列表索引为0的元素为产生式左部，即规约的目标，右边为产生式右部单词列表，符号栈和状态栈同时弹出右部单词列表长度（数量）的次数，将左部压入符号栈，将以当前状态栈和符号栈栈顶为key的二维字典的对应的值压入状态栈，当当前符号栈栈顶为’S’时迭代结束。