任务3.2: 文法左公共因子提取方法及实现

一、实验目的

- 理解上下文无关文法中的左公共因子的概念及其对语法分析的影响。
- 掌握从上下文无关文法中提取左公共因子的算法，形成无二义性的语法结构。
- 熟练运用数据结构（如 Trie 树）处理和优化文法。

二、实验内容

实现从上下文无关文法中提取左公共因子的算法，具体步骤包括：

1. 对每个非终结符的候选式，识别最长的公共前缀。
2. 构建字典树（Trie），辅助提取最长公共前缀，将公共前缀提取为新非终结符的候选式。
3. 输出去除左公共因子的等价文法。

三、实验要求

1. 输入一个上下文无关文法，包括非终结符、终结符和产生式。
2. 输出提取左公共因子后的文法。
3. 使用适当的数据结构（如 Trie 树）提高提取效率。
4. 确保输出文法无二义性，且与输入文法等价。

四、实验指南

输入输出示例

输入示例【注意：此处为了方便，大部分公共前缀用了终结符号，实际中也可能会有非终结符号】

```
S -> apple | apply | application | ball | bat | bath | Xb
X -> ab | ac | ad
```

过程说明

1. **构建 Trie**：将 `S` 的候选式插入 Trie 树中，逐字符生成路径：

```
S
├── a
│   └── p
│       └── p
│           └── l
│               ├── e
│               ├── y
│               └── i
│                   └── c
│                       └── a
```

```
│                           └── t
│                               └── i
│                                   └── o
│                                       └── n
├── b
│   └── a
│       ├── l
│       │   └── l
│       └── t
│           ├── h
└── X
    └── b
```

2. **提取左公共因子**：
   - 在 Trie 中，`appl` 是 `S` 的最长公共前缀，其后续分支为 `e`，`y`，`ication`。
   - `ba` 是另一个公共前缀，其后续分支为 `ll`，`t`，`th`。
3. **生成新的非终结符**：
   - 提取最长公共前缀 `appl` 和 `ba`，并将其分别作为 `S` 的候选式，生成新非终结符 `S'` 和 `S''`。
   - 对 `X` 的候选式，提取公共前缀 `a`，生成新的候选式 `X'`。

输出示例

```
S -> applS' | baS'' | Xb
S' -> e | y | ication
S'' -> ll | t | th
X -> aX'
X' -> b | c | d
```

示范流程

1. **Trie 结构的使用**：采用 Trie 树帮助分析和识别候选式中的公共前缀。
   - 每个节点表示一个字符。
   - 通过沿路径检查每条候选式的前缀分支情况，提取最长公共前缀。
2. **算法流程**：
   - 遍历每个非终结符的候选式，将它们插入 对应的Trie 树。
   - 在 Trie 中查找每个非终结符候选式的公共前缀。
   - 以最长公共前缀生成新的非终结符，形成新的候选式，确保生成的文法无二义性且易于解析。
3. **示例流程**：
   - 对 `S -> apple | apply | application | ball | bat | bath | Xb` 进行公共因子提取，构建新的非终结符候选式。
   - 输出消去左公共因子的等价文法，确保表达清晰和无二义性。

实验难点

- **确定最长公共前缀**：在候选式较多时，识别最长公共前缀并在提取过程中保持文法的等价性是难点。
- **字典树的实现与优化**：Trie 树构建和遍历过程较为复杂，特别是在有大量候选式时，需注意效率问题。