Generate Parentheses

Difficulty: Medium

这道问题的解法不是我想的，但我觉得非常巧妙。

大致思路是，从集合$\{\varepsilon\}$开始，对集合中的每一个串w，分别在w的开头，以及每一个"("后面，插入一对括号"()"。不断迭代，直到达到要求的长度。

举个例子

初始状态只有一个空字符串：

> *{ "" }*

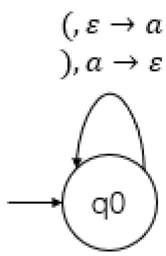
之后每一次迭代：

> *{ "()" }*
>
> *{ "()()", "(())" }*
>
> *{ "()()()", "(())()", "()(())", "(()())", "((()))" }*

但是为什么这样的算法就能够保证生成所有的括号序列呢？

我们知道，给定一个串$w \in \{"(",")"\}^*$，判断w是否为合法括号序列最常用的方法就是将串中的字符依次入栈，如果匹配就弹出，如果最后栈空说明这一序列是合法的。

这提示我们，存在一个接受空栈状态的PDA (Pushdown Automata with Empty State) P= $(\{q_0\},\{(,)\},\{a\},\delta,q_0,Z_0)$，P接受的语言L(P)是所有的合法括号序列。可以利用P来判定是否有$w \in L(P)$. P的状态转移图如下：



这个PDA对应着一个等价的CFG (Context Free Gramma). 直接将PDA转化成等价的CFG比较复杂，这里写出另一种等价的CFG G：

$$S \to SS|(S)|\varepsilon$$

举个例子，串"(()())()"的推导过程为：

> $S \to SS \to (S)S \to (SS)S \to ((S)S)S \to (()S)S$
>
> $\to (()(S))S \to (()())S \to (()())(S) \to (()())()$

下面来证明一下为什么G的语言L(G)与P的语言L(P)是等价的。

首先，G生成的都是合法的括号序列。

这是因为：

- ε是合法的括号序列
- 假设w_1, w_2是合法的括号序列，那么$w_1w_2, (w_1)$都是合法的括号序列

其次，所有的合法括号序列都可以由G生成。

这是因为，$\forall w, w$是合法的括号序列，要么$w = (w_1)$，要么$w = w_1w_2$，要么$w = \varepsilon$，其中w_1, w_2是合法的括号序列。

- ε可以由G生成
- 给定w_1, w_2，那么$(w_1), w_1w_2$可以由G生成

因此L(G)=L(P).

那么再回到最开始的算法中。

- 初始集合为$R = \{\varepsilon\}$，对应着$S \to \varepsilon$
- 对$\forall w \in R$，都会进行以下操作：

  a. $w' = ()w$
  b. 将w分解为$w_1(w_2)w_3, w' = w_1(()w_2)w_3$
- a操作对应着$S \to SS$
- 当$w_2 \neq \varepsilon$时，b操作对应着$S \to SS$，否则对应着$S \to (S)$

因此可以说，这样的算法确实能生成所有的合法序列并且只会生成合法的序列。