Алгоритм Эрли

Следующий алгоритм для выяснения того, порождается ли строка данной контекстно-свободной грамматикой или нет, называется в честь его первооткрывателя «алгоритм Эрли». Этот алгоритм может работать с любой контекстно-свободной грамматикой, причем никакого преобразования грамматики не требуется. Время его работы в худшем случае Cn^3, где n — длина входной строки. Чтобы его описать, нам понадобятся несколько определений.

Пусть есть некоторая строка. Под позицией в этой строке мы будем понимать номер промежутка между соседними символами, причем номер 0 соответствует позиции перед первым символом нашей строки, 1 — между первым и вторым символами, и так далее. Изображать позицию мы будем иногда при помощи точки, стоящей между соседними символами нашей строки. Например, позиция 0 в строке «abcd» изображается как «•abcd», позиция 1 в той же строке — «a•bcd».

Пусть у нас есть теперь некоторая контекстно-свободная грамматика. Под состоянием понимается некоторая ее продукция вместе с двумя позициями: позицией в правой части этой продукции и позиция в исходной строке. Проще всего представлять состояние в программе как 3 числа: номер продукции и две позиции. Позиция в исходной строке означает то место, откуда мы начали разбирать текст, соответствующий нетерминальному символу, стоящему в левой части продукции, а позиция в правой части продукции означает, что мы уже считали, а что еще осталось считать.

Теперь мы рассмотрим сам алгоритм Эрли. Как и в случае алгоритма Кока-Янгера-Касами, мы рассмотрим для простоты только тот алгоритм, который решает, принадлежит ли строка языку или нет. Построение аналогичного алгоритма, строящего дерево разбора, остается вам в качестве упражнения.

Для удобства объяснения алгоритма, мы добавим в грамматику один дополнительный нетерминальный символ P и одну дополнительную продукцию $P \to S$, где S — стартовый нетерминальный символ нашей грамматики.

Для разбора строки с каждой позицией в ней мы свяжем множество состояний, которое будет обозначаться $S(k)$, где $0 \le k \le n$ — позиция в строке, n — длина строки.

Изначально, множество $S(0)$ состоит из единственного состояния $(P \to \bullet S, 0)$. Здесь продукция и позиция в ней указаны строкой $P \to \bullet S$, а позиция в исходной строке — номером 0.

Теперь, для каждой позиции k в исходной строке от 0 до n мы строим множества $S(k)$ при помощи следующих трех правил, применяя их до тех пор, пока они не перестанут давать дополнительные состояния в $S(k)$:

1) Предсказание. Для каждого состояния из $S(k)$ вида $(N \to \alpha \bullet M \beta, i)$, где α, β — произвольные строки, а M, N — нетерминальные символы, в $S(k)$ добавляется состояние $(M \to \bullet \gamma, k)$ для любой продукции $M \to \gamma$ из нашей грамматики (γ — строка).

2) Считывание. Если $k > 0$, и между позициями $k-1$ и k в исходной строке стоит символ c, то для любого состояния из $S(k-1)$ вида $(N \to \alpha \bullet c \beta, i)$ в $S(k)$ добавляется состояние $(N \to \alpha c \bullet \beta, i)$.

3) Завершение. Для любого состояния из $S(k)$ вида $(N \to \gamma \bullet, i)$ и любого состояния из $S(i)$ вида $(M \to \alpha \bullet N \beta, j)$ в $S(k)$ добавляется состояние $(M \to \alpha N \bullet \beta, j)$.

После построения таких множеств состояний, мы проверяем множество $S(n)$. Если в нем есть состояние $(P \to S \bullet, 0)$, то строка принадлежит языку, иначе — нет.

Для примера рассмотрим ту же грамматику, для которой в предыдущем файле мы использовали алгоритм Кока-Янгера-Касами. После добавления дополнительного нетерминального символа и продукции эта грамматика выглядит так:

```
<P> ::= <A>
<A> ::= ""
<A> ::= "0" <A> "1"
```

Программа разбора строки по этой грамматике находится в файле earley.cpp

Задачи

1. Для задач 2–5 из файла cpp19 написать функции на C++, проверяющие принадлежность строки-параметра соответствующему языку в соответствии с алгоритмом Эрли.