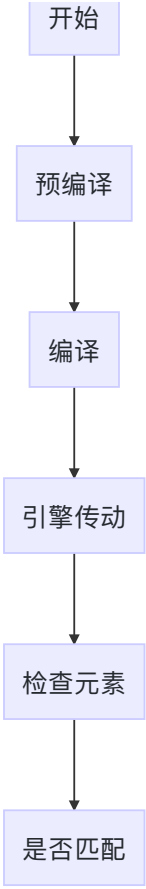
正则表达式

正则起源

最初来自于神经学家们的一种用数学方式来描述神经网络的模型，后来数学科学家Stephen kleene发表《神经网事件的表示法》，利用此正则集合的数学符号来描述此模型，从而引入了正则表达式的概念。c语言之父、unix之父肯·汤普森将正则用于做一下搜索算法的研究，描述一种正则表达式的编译器，这就是最早的正则表达式的编译器qed（这成为后来的grep编辑器）。unix使用正则之后，正则表达式不断发展壮大。后面产生了perl流派，现在的语言中都参考的perl。

匹配过程

可见每次匹配都有预编译



引擎

在这里只看了NFA引擎

NFA是一个非确定型有穷自动机

- 非确定 不能确定字符匹配的顺序
- 有穷 有限次数能够拿到结果
- 自动机 自动完成，设置好匹配规则后引擎自动完成

NFA的回溯机制

当某个分支匹配不成功时，文本的位置需要回退，换另一个分支匹配 就好像是走迷宫时，走不通时原路返回，走另一个方向，直到走出去。

常见优化措施

- 提前预编译
- 不要滥用括号和字符组
- 等其他正则表达式本身的一些优化措施
  https://developer.aliyun.com/article/292257

编译缓存

```
//Java中预编译正则（?
Pattern pattern = Pattern.compile(“规则”);
pattern.matcher(value)
//python中的（?
内置re模块
1、re.compile(pattern)预编译返回pattern对象码字后面代码直接引用
2、re.match(pattern,text)即用编译，虽然也会缓存pattern对象，但每次从缓存中取，比预编译多一步取的操作。
一般来说预编译比即用编译快
```

```
//javascript
js中实例化一个正则表达式会自动编译，存储在变量中，则不会重新编译
var re = /[Ff]oo|[Bb]ar/;
"Foo".match(re); // ["Foo"]
"Baz".match(re); // null
```