正则表达式的匹配\_剑指Offer\_50

# 正则表达式的匹配\_剑指Offer\_50

## 题目描述

/\*\*\*\*\*\*\*\*\*\*\*剑指Offer50:正则表达式的匹配\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\* 题目描述

\* 请实现一个函数用来匹配包括'.'和'\*'的正则表达式。

\* 模式中的字符'.'表示任意一个字符，而'\*'表示它前面的字符可以出现任意次（包含0次）。

\* 在本题中，匹配是指字符串的所有字符匹配整个模式。

\* 例如，字符串"aaa"与模式"a.a"和"ab\*ac\*a"匹配，但是与"aa.a"和"ab\*a"均不匹配

## 思路分析

\* 思路分析：利用递归思想。重点考虑匹配符.和\*

\* . 可以匹配任意字符，因此当前字符匹配两种情况：

\* str[strIndex] == pattern[patternIndex]||pattern[patternIndex]=='.'

\* 考虑\*的问题，\*主要影响匹配字符个数问题。主要以下一字符是否为\*分为两大类：

\* （1）若下一字符为\*：

\* 分成两种情况：①若当前字符匹配，则分成三种情况递归下去：

\* \*只匹配1个字符，\*匹配多余1个，\*匹配0个。

\* ②若当前字符不匹配，代表\*匹配0个，递归。

\* （2）若下一字符非\*:

\* 当前字符匹配，则递归；否则直接返回false。

\*/

## Java代码

public boolean **match**(char[] str, char[] pattern){

if(str == null||pattern == null) return false;

return match(str,0,str.length,pattern,0,pattern.length);

}

public boolean **match**(char[] str, int strIndex,int strLen,char[] pattern,int patternIndex,int patternLen) {

//递归终止条件

if(strIndex == strLen){//字符结束了

//str结束且pattern结束完全匹配，str结束且pattern只剩下x\*两个字符也完全匹配；其他情况返回false

if(patternIndex == patternLen||(patternIndex+1==patternLen-1&&pattern[patternIndex+1]=='\*'))

return true;

else

return false;

}

//匹配结束

if(strIndex < strLen && patternIndex == patternLen) return false;//匹配结束

//操作

if(patternIndex+1 < patternLen&&pattern[patternIndex+1]=='\*'){//下一个匹配字符为'\*'

if(str[strIndex] == pattern[patternIndex]||pattern[patternIndex]=='.'){//当前字符相匹配

return match(str,strIndex+1,strLen,pattern,patternIndex+2,patternLen)||//只匹配1个

match(str,strIndex+1,strLen,pattern,patternIndex,patternLen)||//\*匹配为多余1个

match(str,strIndex,strLen,pattern,patternIndex+2,patternLen);//\*匹配为0

}else{//当前字符不匹配

return match(str,strIndex,strLen,pattern,patternIndex+2,patternLen);//只能认为\*匹配为0

}

}else{//patternIndex+1==patternLen或者pattern下一字符不为'\*'

if(str[strIndex]==pattern[patternIndex]||pattern[patternIndex]=='.')

return match(str,strIndex+1,strLen,pattern,patternIndex+1,patternLen);

else

return false;

}

}