# LeetCode

## 10 Regular expression matching

**题意**：源串s : “aaa”，模式串p : ”.\*”，能用p表示s否? 全匹配

**题解思路**：采用动态规划求解

令dp[i][j]代表s : 0 ~ i-1和p : 0 ~ j-1的匹配结果，即不包含s[i]和p[j]，这样原问题就转换为求解dp[sLen][pLen]。具体情况如下：

* 当p[j-1] != ’\*’时，那么p[j-1]必须和s[i-1]匹配，那就将匹配过程分成了两部分，即s : 0 ~ i-2 和p : 0 ~ j-2的匹配结果【即dp[i-1][j-1]】与p[j-1]和s[i-1]的匹配结果【p[j-1] == s[i-1] || p[j-1] == ‘.’】，所以dp[i][j] = dp[i-1][j-1] && (p[j-1] == s[i-1] || p[j-1] == ‘.’)
* 当p[j-1] == ‘\*’时,情况复杂一些，如下：

1. ‘\*’代表p[j-2]出现了0次，即s:0~i-1和p:0~j-3匹配【即dp[i][j-2]】
2. ‘\*’代表p[j-2]出现了1次或1次以上，那么s[i-1]一定要与p[j-1]匹配【p[j-1] == s[i-1] || p[j-1] == ‘.’】，因为p[j-1]至少出现了一次，例如：s:”aaaa”和p:”a\*”，’a’和’.’一定是匹配的。还有，s:0~i-2一定要与p:0~j-1匹配【即dp[i-1][j]】，可以用反证法，如果s:0~i-2不与p:0~ j-1匹配，那加上s[j-1]也肯定不匹配。正向思维就是只有在s:0~i-2和p:0~j-1匹配了，那我们再加上满足条件的s[i-1]也肯定匹配，因为p[j-1]为’\*’。
3. 这样，当p[j-1] == ‘\*’时，dp[i][j] = dp[i][j-2] || dp[i-1][j] && (p[j-1] == s[i-1] || p[j-1] == ‘.’)

**注意事项**：

* 审题，题目要求是全匹配
* 初始化dp[i][0]，不需要，除了dp[0][0]外，其余dp[i][0]=false默认
* 初始化dp[0][j]，当然dp[0][0] = true，当j>0时，dp[0][2K+1] = false,因为奇数位置是不可能匹配s为空字符串的，即s:””和p:”a”肯定是不能匹配的。出现匹配的情况只有是p:”a\*a\*a\*”，其中’a’可以换为任何字符。程序实现:

|  |
| --- |
| //初始化dp[0][2],dp[0][4],...,dp[0][2k],...  for(int i=2; i<=pLen; i+=2) {  if(!(p.charAt(i-1) == '\*' && dp[0][i-2])) {  break;  }else {  dp[0][i] = true;  }  } |

代码实现

|  |
| --- |
| **public** **class** Solution3 {  **public** **boolean** isMatch(String s, String p) {  **if**(s == **null** || p == **null**) **return** **false**;  **int** sLen = s.length();  **int** pLen = p.length();  **boolean**[][] dp = **new** **boolean**[sLen+1][pLen+1];  dp[0][0] = **true**;  //初始化dp[0][2],dp[0][4],...,dp[0][2k],...  **for**(**int** i=2; i<=pLen; i+=2) {  **if**(!(p.charAt(i-1) == '\*' && dp[0][i-2])) {  **break**;  }**else** {  dp[0][i] = **true**;  }  }  **for**(**int** i=1; i<=sLen; i++) {  **for**(**int** j=1; j<=pLen; j++) {  //dp[i][j] 代表s:0~i-1,p:0~j-1,而当j=1时,j-1=0,p不可能是以'\*'开头  **if**(p.charAt(j-1) == '\*') {  //dp[i][j] = dp[i][j-2] || dp[i][j-1] || dp[i-1][j-1] && (s.charAt(i-1) == p.charAt(j-2) || p.charAt(j-2) == '.');  dp[i][j] = dp[i][j-2] || i > 0 && dp[i-1][j] && (p.charAt(j-2) == s.charAt(i-1) || p.charAt(j-2) == '.');  }**else** {  dp[i][j] = dp[i-1][j-1] && (p.charAt(j-1) == s.charAt(i-1) || p.charAt(j-1) == '.');  }  }  }    **return** dp[sLen][pLen];  }  } |