# 实验报告七

给定一个字符串 (s) 和一个字符模式 (p) ，实现一个支持 '?' 和 '\*' 的通配符匹配。

'?' 可以匹配任何单个字符。

'\*' 可以匹配任意字符串（包括空字符串）。

两个字符串完全匹配才算匹配成功。

说明:

s 可能为空，且只包含从 a-z 的小写字母。

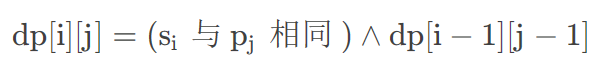
p 可能为空，且只包含从 a-z 的小写字母，以及字符 ? 和 \*。

方法接口如下：

public static boolean isMatch(String s, String p) {}

1. 题目分析

本题采用动态规划算法求解。在给定的模式p中，只会有三种类型的字符出现：小写字母a-z ,可以匹配对应的一个小写字母；问号 ? ，可以匹配任意一个小写字母；星号\*，可以匹配任意字符串，可以为空，也就是匹配零个或任意个小写字母。其中小写字母和问号的匹配是确定的，而星号的匹配是不确定的，因此需要枚举所有的匹配情况。为了减少枚举，可以使用动态规划来解决。用dp[i][j] 表示字符串s的前i个字符与模式p 的前j个字符是否能匹配。在进行状态转移时，可以考虑模式p的第j个字符pj和与之对应的字符串s的第i个字符si：如果pj 是小写字母，那么si必须也为相同的小写字母，状态转移方程为：

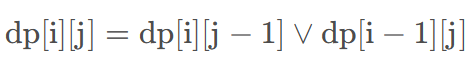


其中∧ \wedge∧ 表示逻辑与运算，也就是说，dp[i][j]为真，当且仅当dp[i-1][j-1]为真，并且si与pj相同。

如果pj 是问号，那么对si没有任何要求，状态转移方程为：

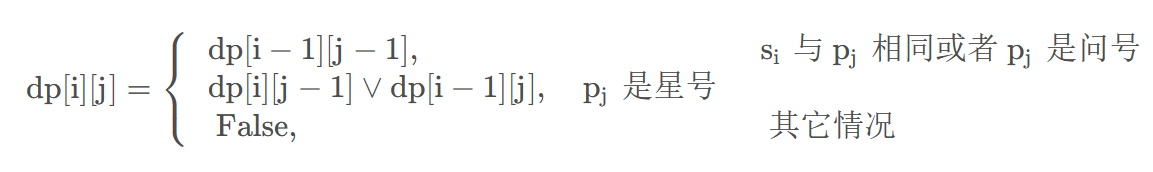


如果pj为星号，那么同样对si没有任何要求，但是星号可以匹配零或任意多个小写字母，因此状态转移方程分为俩种情况，即使用或者不使用：



其中∨ \vee∨ 表示逻辑或运算。如果我们不使用这个星号，那么就会从dp[i][j-1]转移而来，如果使用这个星号，那么就从dp[i-1][j]转移而来。

最终的状态转移方程为：



只有确定了边界条件，才能进行动态规划。在上述的状态转移方程中，由于dp[i][j]对应着s的前i个字符和模式p的前j个字符，因此所有的dp[0][j]和dp[i][0]都是边界条件。

dp[0][0] = True ,即当字符串s和模式串p均为空，匹配成功

dp[i][0] = False, 即空模式无法匹配非空字符串

dp[0][j] 需要分情况讨论：因为星号才能匹配空字符串，所以只有当模式匹配p的前j个字符都是星号是，dp[0][j]为True。

1. 代码实现

主要实现代码（字符串匹配）

**package** char\_match;

**public** **class** match\_char **implements** isMatch {

@Override

**public** **boolean** Match(String s, String p) {

**boolean**[][] dp = **new** **boolean**[s.length() + 1][p.length() + 1];

//空字符串和空模式

dp[0][0] = **true**;

//空字符串和非空模式

**for** (**int** i = 1; i < p.length() + 1; i++) {

**if** (p.charAt(i - 1) == '\*') {

dp[0][i] = dp[0][i - 1];

}

}

**for** (**int** i = 1; i < s.length() + 1; i++) {

**for** (**int** j = 1; j < p.length() + 1; j++) {

**if** (s.charAt(i - 1) == p.charAt(j - 1) || p.charAt(j - 1) == '?') {

dp[i][j] = dp[i - 1][j - 1];

} **else** **if** (p.charAt(j - 1) == '\*') {

dp[i][j] = dp[i][j - 1] || dp[i - 1][j];

} **else** {

dp[i][j] = **false**;

}

}

}

**return** dp[s.length()][p.length()];

}

}

接口类：

**package** char\_match;

**public** **interface** isMatch {

**boolean** Match(String s, String p);

}

测试类：

**package** char\_match;

**import** java.util.Scanner;

**public** **class** ceshi\_char {

**public** **static** **void** main(String[] args) {

**var** sc = **new** Scanner(System.***in***);

System.***out***.println("请输入模式串：");

String s = sc.nextLine();

System.***out***.println("请输入目标串：");

String p = sc.nextLine();

match\_char chop = **new** match\_char();

**if**(chop.Match(s, p)) {

System.***out***.println("模式匹配");

}**else** {

System.***out***.println("模式不匹配");

}

}

}

1. 运行结果

