## 算法讨论班 第十期 2015-10-30

### 主讲人：李玥珮

### 44．Wildcard Matching

———————————————————————————————————————

Implement wildcard pattern matching with support for '?' and '\*'.

'?' Matches any single character.

'\*' Matches any sequence of characters (including the empty sequence).

Some examples:

isMatch("aa","a") → false

isMatch("aa","aa") → true

isMatch("aaa","aa") → false

isMatch("aa", "\*") → true

isMatch("aa", "a\*") → true

isMatch("ab", "?\*") → true

isMatch("aab", "c\*a\*b") → false

———————————————————————————————————————

思路：

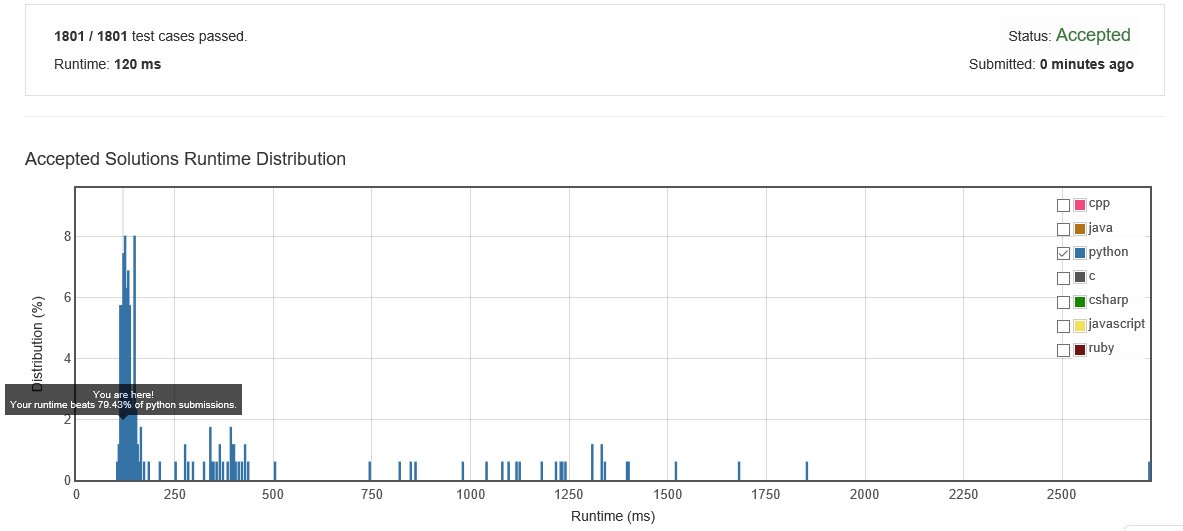
分别用i、j索引字符串s与p，则每个字符的匹配过程可以分为：

1. p[j]不是特殊字符，且s[i] == p[j]，则匹配成功；i++，j++，即直接处理下一个字符。
2. p[j] == ‘?’，能够匹配单个字符，此时与s是什么无关，总是匹配成功，处理方法与上一条一样。
3. p[j] == ‘\*’，相当于0~n个 ’?’。所以，首先假设为0个 ’?’，即把 ’\*’ 替换为 ‘’；如果之后的字符串匹配成功，则返回结果True；否则，可能是之前的假设出现问题，重新假设 ‘\*’ 为一个 ‘?’。为了方便的回溯，我们要记录下 ‘\*’ 出现时，i，j 的值。
4. 其他情况。这就意味着单个字符在i、j这里匹配不成功。如果之前没有出现过 ’\*’，则说明匹配不成功，函数结束。如果出现过 ‘\*’，则可能是之前 ‘\*’ 匹配出现问题，回到3重新假设，知道s结束为止。

循环的出口条件是s结束。此时，p不一定结束，需要遍历p剩余的字符，如果有非 ’\*’ 字符出现，则匹配不成功。

如果以上条件都满足，则字符串匹配成功。





### 96．Unique Binary Search Trees

———————————————————————————————————————

Given n, how many structurally unique BST's (binary search trees) that store values 1...n?

For example,  
Given n = 3, there are a total of 5 unique BST's.

1 3 3 2 1

\ / / / \ \

3 2 1 1 3 2

/ / \ \

2 1 2 3

———————————————————————————————————————

二叉排序树：

每个节点的左子树的所有节点都小于该节点，右子树的所有节点都大于该节点。

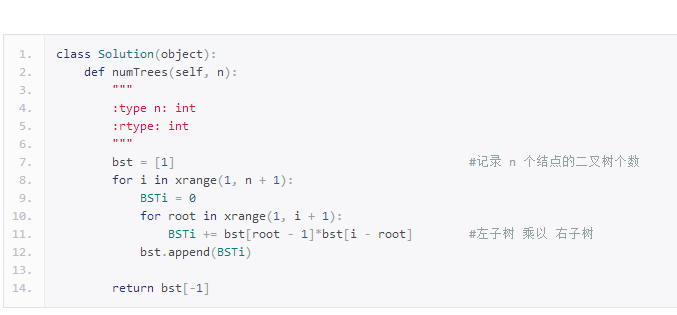
动态规划思想：

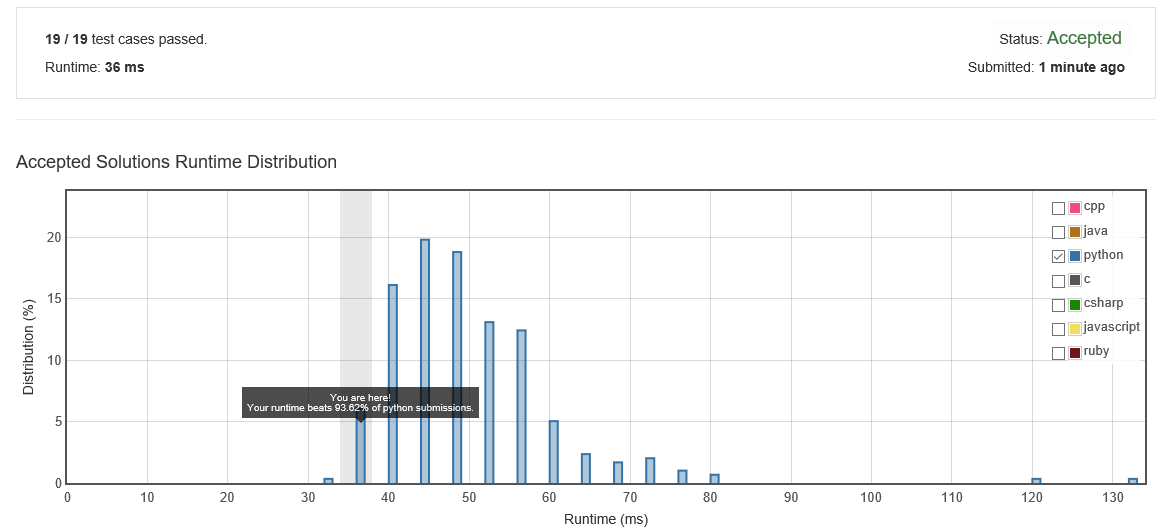
1. 分解问题：求二叉排序树左子树或者右子树的不同的结构数（以下简称SUB）。
2. 递推公式：

其中，i的范围是1——n，为根节点的取值。SUB左i为根节点是i时左子树的SUB数。根节点是i时，左子树的节点个数为i-1个，正好是原问题的子问题，i-1个不同节点的SUB数是多少。题目中，我们用一个数组bst[] 记录i个节点的SUB数，这样方便查找之前计算的SUB数值。

1. 递推求解：就本题而言，对于0个节点的情况，我们应该返回1，因为树的SUB数关系是相乘，所以任何子树的SUB是0，都将导致整树的SUB是0。

代码：





### 139．Word Break

———————————————————————————————————————

Given a string s and a dictionary of words dict, determine if s can be segmented into a space-separated sequence of one or more dictionary words.

For example, given  
s = "leetcode",  
dict = ["leet", "code"].

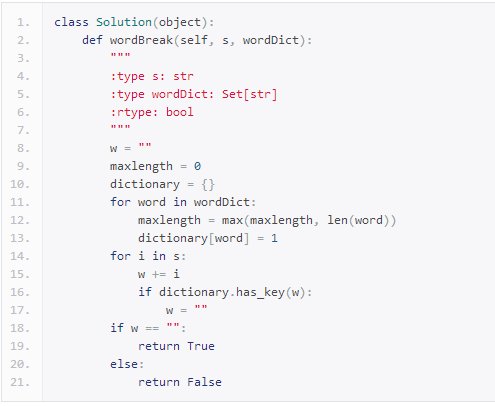
Return true because "leetcode" can be segmented as "leet code".

———————————————————————————————————————

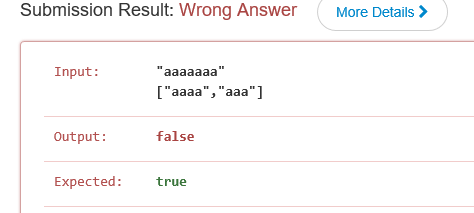
思路：

1. 直接循环

依次从字符串重选取字符，并连接成单词，如果该单词存在于字典中，则查找成功，继续处理以后的字符串。如果之后的单词都能查找成功则返回 True。



结果：

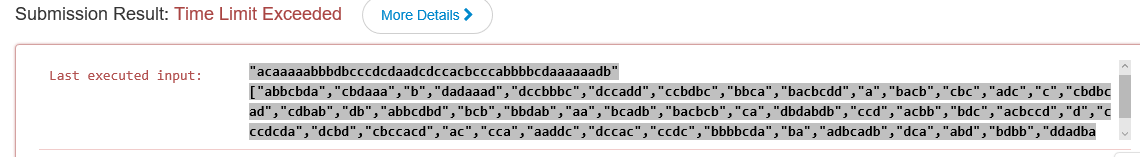


1. 递归与回溯

上面的方法错误在于连续匹配了两个 ”aaa”，很简单的想法是利用递归的思想解决问题。每次递归都查找字符串开头的几个字符，把剩余的字符串交给下一层递归解决。



结果：



1. 动态规划

动态规划问题解题的关键点在于找到问题的分解方案，即找到问题及其子问题之间的递推公式。就本题而言：

如果字符串s[0:i] 是可以分解的，则必然存在s[0:j] 是可以分解的、且s[j+1:i] 在字典中。所以，我们用一个数组match[] 标记字符串s[0:i] 的可分解情况，则可以推算出i之后字符串的可分解情况。

具体的：

0——i 是否可解，只要遍历0——i 之间的可解点，找到j 使得 s[j:i] in worddict；而得出的i的可解情况，同样能帮助求解 i 之后的点的可解性。

这样通过一遍扫描就能得到所有 i 点的可解性

