算法讨论班第二期题目——2015-09-11

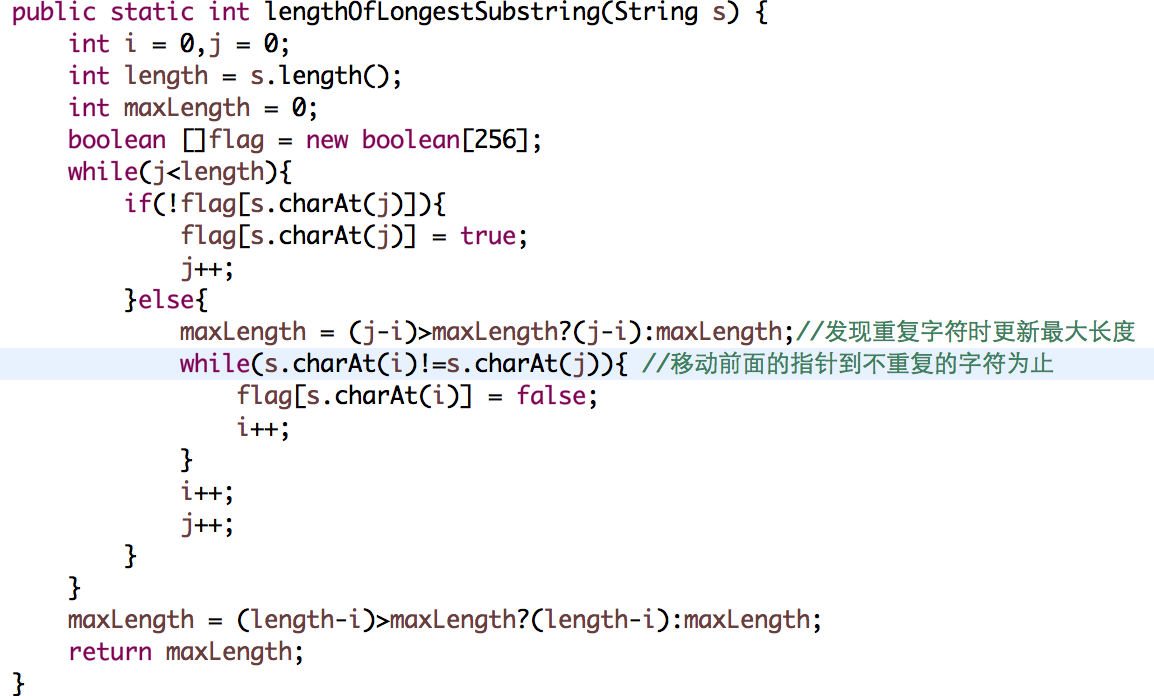
3)Given a string, find the length of the longest substring without repeating characters. For example, the longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3. For "bbbbb" the longest substring is "b", with the length of 1.

思路：

设置两个指针，只要后面的值比前面的值小，后面的指针就后移，发现字符相等时，更新最大长度，再更新前面的指针到不重复的字符为止。

时间复杂度：O(n)

附java代码：



5)Given a string *S*, find the longest palindromic substring in *S*. You may assume that the maximum length of *S* is 1000, and there exists one unique longest palindromic substring.

思路一：动态规划

opt[i][j]＝1表示从i到j为回文子序列

opt[i][j] = opt[i+1][j-1] 当S[i]==S[j]

时间复杂度：O(n\*n),空间复杂度:O(n\*n)

附java代码：



结果发现超时

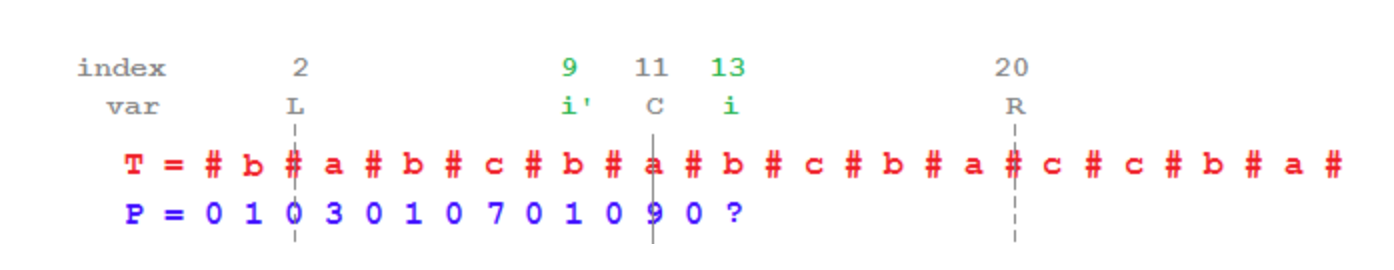
思路二：

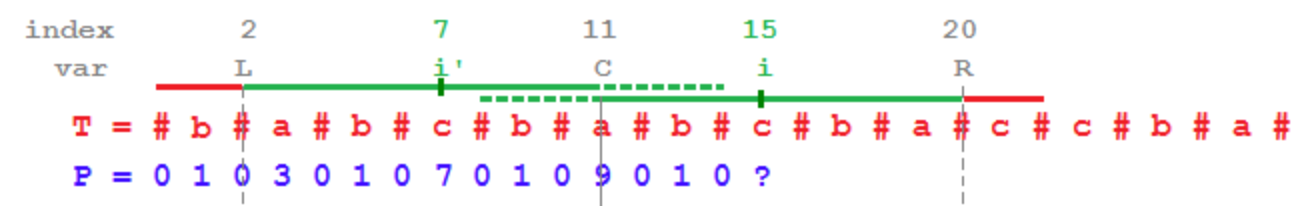
在思路一中会遇到奇偶性的问题，abcba回文字符串长度为奇数时会以某个字符作为中心，abba当回文字符串长度为偶数时会以中间的空格为中心。思路二的方法就是利用加入#的方式消除奇偶性的问题，将＃代替空格参与回文字符串长度的计算。

变量介绍：C代表中心点指针

R代表以C为中心点最大回文子串的右边界

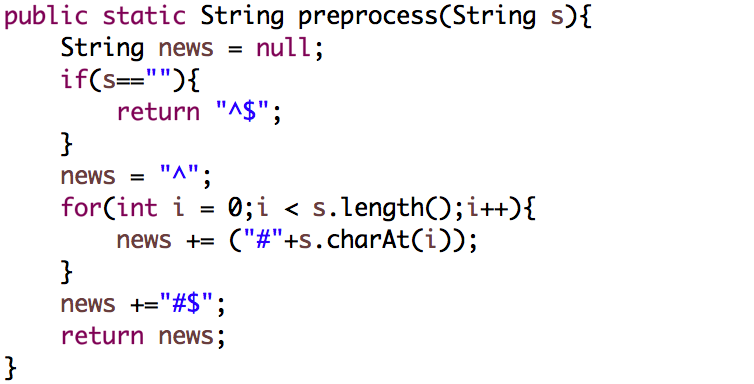
i代表遍历指针

p[i]代表以T[i]为中心的最大回文子串的长度

上图涉及到的问题： b的值为7吗？NO！Why?因为

以T[i]为中心的长度为7的字符串右边界已经超过了C，而只能保证以C为中心的左右区域相等，超过的部分不能保证，所以在给p[i] 赋值的时候需要比较一下R-i和p[i’]的大小。

如果发现在计算p[i]过程中，i+p[i]>R，需要更新R的值为i+p[i]，来增大步长提高计算速度。

附java代码：

时间复杂度：O(n) 空间复杂度：O(n)

8)Implement atoi to convert a string to an integer.

思路：

去掉空格，当遇到+,-或者数字的时候开始转化。但是要注意溢出的问题。

附java代码：