5. Longest Palindromic Substring

Add to List

QuestionEditorial Solution

[My Submissions](https://leetcode.com/problems/longest-palindromic-substring/submissions/)

* Total Accepted: **165978**
* Total Submissions: **673580**
* Difficulty: **Medium**
* Contributors: **Admin**

Given a string **s**, find the longest palindromic substring in **s**. You may assume that the maximum length of **s** is 1000.

**Example:**

**Input:** "babad"

**Output:** "bab"

**Note:** "aba" is also a valid answer.

思路：中間抽一個字母，按照奇數搜一遍，偶數也搜一遍。得出這個字母為中間的longest parlidrome substring的起點和長度。

public class Solution {

private int lo, maxLen;

public String longestPalindrome(String s) {

int len = s.length();

if (len < 2)

return s;

for (int i = 0; i < len-1; i++) {

extendPalindrome(s, i, i); //assume odd length, try to extend Palindrome as possible

extendPalindrome(s, i, i+1); //assume even length.

}

return s.substring(lo, lo + maxLen);

}

private void extendPalindrome(String s, int j, int k) {

while (j >= 0 && k < s.length() && s.charAt(j) == s.charAt(k)) {

j--;

k++;

}

if (maxLen < k - j - 1) {

lo = j + 1;

maxLen = k - j - 1;

}

}}

43. Multiply Strings

思路：如果任意一個string是0，返回0；

兩層forloop，逐位相乘。左邊數，第i位與第j位相乘，product = carry + num3[i + j + 1] +

Character.getNumericValue(num1.charAt(i)) \*

Character.getNumericValue(num2.charAt(j));

num3[i + j + 1] = product % 10;

carry = product / 10;

**public** **class** Solution {

**public** String multiply(String num1, String num2) {

**if** (num1.equals(“0”) || num2.equals(“0”)) {

**return** **“0”**;

}

**int** len1 = num1.length(), len2 = num2.length();

**int** len3 = len1 + len2;

**int** i, j, product, carry;

**int**[] num3 = **new** **int**[len3];

**for** (i = len1 - 1; i >= 0; i--) {

carry = 0;

**for** (j = len2 - 1; j >= 0; j--) {

product = carry + num3[i + j + 1] +

Character.getNumericValue(num1.charAt(i)) \*

Character.getNumericValue(num2.charAt(j));

num3[i + j + 1] = product % 10;

carry = product / 10;

}

num3[i + j + 1] = carry;

}

StringBuilder sb = **new** StringBuilder();

i = 0;

**while** (i < len3 - 1 && num3[i] == 0) {

i++;

}

**while** (i < len3) {

sb.append(num3[i++]);

}

**return** sb.toString();

}

}

564. Find the Closest Palindrome

Given an integer n, find the closest integer (not including itself), which is a palindrome.

The 'closest' is defined as absolute difference minimized between two integers.

**Example 1:**

**Input:** "123"**Output:** "121"

如果有多个解，返回最小值

思路：

有四个可能性：

1. 少一位的数里面最大的palindrome number。例如 input 10 output可以是9
2. 多一位的数里面最小的。例如input：10， output：11
3. 将右半部分翻转，整个palindrome数的中间1或2位加1
4. 将右半部分翻转，整个palindrome数的中间1或2位减1

以下程序，用循环做加1和减1.所以才需要将原来数字从set除去

class Solution {public:

string nearestPalindromic(string n) {

long len = n.size(), num = stol(n), res, minDiff = LONG\_MAX;

unordered\_set<long> s;

s.insert(pow(10, len) + 1);

s.insert(pow(10, len - 1) - 1);

long prefix = stol(n.substr(0, (len + 1) / 2));

for (long i = -1; i <= 1; ++i) {

string pre = to\_string(prefix + i);

string str = pre + string(pre.rbegin() + (len & 1), pre.rend());

s.insert(stol(str));

}

s.erase(num);

for (auto a : s) {

long diff = abs(a - num);

if (diff < minDiff) {

minDiff = diff;

res = a;

} else if (diff == minDiff) {

res = min(res, a);

}

}

return to\_string(res);

}

};