Previous Next

564. Find the Closest Palindrome

April 22, 2017 | 50.5K views

Average Rating: 3.11 (37 votes)

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"
```

Note:

1. The input **n** is a positive integer represented by string, whose length will not exceed 18.
2. If there is a tie, return the smaller one as answer.

Solution

Approach #1 Brute Force[Time Limit Exceeded]

The simplest solution is to consider every possible number smaller than the given number n, starting by decrementing 1 from the given number and go on in descending order. Similarly, we can consider every possible number greater than n starting by incrementing 1 from the given number and going in ascending order. We can continue doing so in an alternate manner till we find a number which is a palindrome.

Java

```java
public class Solution {
    public String nearestPalindromic(String n) {
        long num = Long.parseLong(n);
        for (long i = 1;; i++) {
            if (isPalindrome(num - i))
                return "" + (num - i);
            if (isPalindrome(num + i))
                return "" + (num + i);
        }
    }
    boolean isPalindrome(long x) {
        long t = x, rev = 0;
        while (t > 0) {
            rev = 10 * rev + t % 10;
            t /= 10;
        }
        return rev == x;
    }
}
```

Complexity Analysis

- Time complexity : $O(\sqrt{n})$. Upto $2 * \sqrt{n}$ numbers could be generated in the worst case.
- Space complexity : $O(1)$. Constant space is used.

Approach #2 Using Math[Accepted]

Algorithm

To understand this method, let's start with a simple illustration. Assume that the number given to us is "abcxy". One way to convert this number into a palindrome is to replicate one half of the string to the other half. If we try replicating the second half to the first half, the new palindrome obtained will be "yxcxy" which lies at an absolute of $|10000(a-y)+1000(b-x)|$ from the original number. But, if we replicate the first half to the second half of the string, we obtain "abcba", which lies at an absolute difference of $|10(x-b)+(y-a)|$. Trying to change c additionaly in either case would incur an additional value of atleast 100 in the absolute difference.

From the above illustration, we can conclude that if replication is used to generate the palindromic number, we should always replicate the first half to the second half. In this implementation, we've stored such a number in a at a difference of $diff1$ from n.

But, there exists another case as well, where the digit at the middle index is incremented or decremented. In such cases, it could be profitable to make changes to the central digit only since such changes could lead to a palindrome formation nearer to the original digit. e.g. 10987. Using the above criteria, the palindrome obtained will be 10901 which is at a more difference from 10987 than 11011. A similar situation occurs if a 0 occurs at the middle digit. But, again as discussed previously, we need to consider only the first half digits to obtain the new palindrome. This special effect occurs with 0 or 9 at the middle digit since, only decrementing 0 and incrementing 9 at that digit place can lead to the change in the rest of the digits towards their left. In any other case, the situation boils down to the one discussed in the first paragraph.

Now, whenever we find a 0 near the middle index, in order to consider the palindromes which are lesser than n, we subtract a 1 from the first half of the number to obtain a new palindromic half e.g. If the given number n is 20001, we subtract a 1 from 200 creating a number of the form 199xx. To obtain the new palindrome, we replicate the first half to obtain 19991. Taking another example of 10000, (with a 1 at the MSB), we subtract a 1 from 100 creating 099xx as the new number transforming to a 9999 as the new palindrome. This number is stored in b having a difference of $diff2$ from n

Similar treatment needs to be done with a 9 at the middle digit, except that this time we need to consider the numbers larger than the current number. For this, we add a 1 to the first half. e.g. Taking the number 10987, we add a 1 to 109 creating a number of the form 110xx(11011 is the new palindrome). This palindrome is stored in c having a difference of $diff3$ from n.

Out of these three palindromes, we can choose the one with a minimum difference from n. Further, in case of a tie, we need to return the smallest palindrome obtained. For resolving this tie's conflict, we can observe that a tie is possible only if one number is larger than n and another is lesser than n. Further, we know that the number b is obtained by decreasing n. Thus, in case of conflict between b and any other number, we need to choose b. Similarly, c is obtained by increasing n. Thus, in case of a tie between c and any other number, we need to choose the number other than c.

Java

```java
public class Solution {
    public String mirroring(String s) {
        String x = s.substring(0, (s.length()) / 2);
        return x + (s.length() % 2 == 1 ? s.charAt(s.length() / 2) : "") + new
StringBuilder(x).reverse().toString();
    }
    public String nearestPalindromic(String n) {
        if (n.equals("1"))
            return "0";

        String a = mirroring(n);
        long diff1 = Long.MAX_VALUE;
        diff1 = Math.abs(Long.parseLong(n) - Long.parseLong(a));
        if (diff1 == 0)
            diff1 = Long.MAX_VALUE;

        StringBuilder s = new StringBuilder(n);
        int i = (s.length() - 1) / 2;
        while (i >= 0 && s.charAt(i) == '0') {
            s.replace(i, i + 1, "9");
            i--;
        }
        if (i == 0 && s.charAt(i) == '1') {
            s.delete(0, 1);
            int mid = (s.length() - 1) / 2;
            s.replace(mid, mid + 1, "9");
        } else
            s.replace(i, i + 1, "" + (char)(s.charAt(i) - 1));
```

Complexity Analysis

- Time complexity : $O(l)$. Scanning, insertion, deletion,, mirroring takes $O(l)$, where l is the length of the string.
- Space complexity : $O(l)$. Temporary variables are used to store the strings.

Rate this article: ★★★☆☆

Previous Next

Comments: 16 Sort By

Type comment here... (Markdown is supported)

Preview Post

pavangm ★ 37 November 12, 2018 4:26 AM
The closest palindrome of 20001 is 20002. But, the explanation says as 19991
36 Share Reply
SHOW 6 REPLIES

sudhanshumittaliitg ★ 21 November 1, 2017 8:52 PM
This is easier to understand -
class Solution {

```
public String nearestPalindromic(String n) {
    int order = (int) Math.pow(10, n.length()/2);
```

Read More
21 Share Reply
SHOW 4 REPLIES

bbdnx ★ 12 September 13, 2018 11:01 PM
The explanation is some kind of complicated. It just construct 3 Palindromes, first is mirror the fist half to the last, second is let first half plus one and mirror to the last, third is let firs half minus one and mirror to the last. Then compare the three and return the nearest.
12 Share Reply
SHOW 2 REPLIES

delta45724 ★ 167 October 21, 2018 9:28 AM
why the complexity of Brute Force approach O(sqrtn)?
6 Share Reply
SHOW 2 REPLIES

leonbnu ★ 45 October 21, 2018 10:44 AM
the explanation of the solution is wrong. say, 88211, the closest is 88188, so increment/decrement by 1 is certainly not only for the case of '0' or '9'.
5 Share Reply

mnirk1609 ★ 5 June 26, 2018 5:18 PM
What do you have to say about 832
5 Share Reply
SHOW 3 REPLIES

plzplzspammeplzplz ★ 7 December 23, 2017 10:34 AM
Python makes this look like a joke:

def closestPalindrome(x: str) -> str:
return x[:(len(x) + 1) // 2] + "".join(reversed(x[:len(x) // 2]))
6 Share Reply
SHOW 2 REPLIES

miuralu670 ★ 1 October 16, 2018 2:34 AM
For the "math" solution if you use a number that is itself a palindrome, say "1001", the solution returns "999" since it doesn't start the check with i = 0, but with i = 1. Shouldn't the nearest palindrome to "1001" be itself? based on the problem definition.
1 Share Reply
SHOW 3 REPLIES

dckeybridge ★ 2 November 1, 2018 9:23 AM
The code works while the explanation is wrong.
1 Share Reply

plzplzspammeplzplz ★ 7 December 23, 2017 10:39 AM
Even better:
def closestPalindrome(x):
return x[:(len(x) + 1) // 2] + x[len(x) // 2 - 1:: -1]
1 Share Reply
SHOW 1 REPLY

‹ 1 2 ›