Documentation for Find the Closest Palindrome

📃 Table of Contents

1. Problem Statement

Given a string n representing an integer, return the closest integer (not including itself), which is a palindrome. If there is a tie (two palindromes equally close), return the smaller one.

Constraints:

- 1 <= n.length <= 18
- n consists only of digits (0-9)
- n has no leading zeros
- The number is within the range $[1, 10^{18} - 1]$

2. Intuition

Rather than checking all palindromes around the number (which is inefficient for large inputs), we realize that the closest palindrome must be structurally similar to the given number. We can manipulate the first half (prefix) of the number and mirror it to generate candidates.

3. Key Observations

- Mirroring the first half of the number creates a potential palindrome.
- Only a few candidates are required: mirrored version of prefix, prefix+1, and prefix-1.
- Edge cases like 999...9 and 100...001 can be closer than structural mirrors.
- Exclude the number n itself when comparing.

4. Approach

- Extract the prefix from n.
- Generate three palindrome candidates:
  - Using prefix
  - Using prefix - 1
  - Using prefix + 1
- Add two special palindromes:
  - One less digit: all 9's (e.g., 999 for 1000)
  - One more digit: 100...001 (e.g., 10001 for 9999)
- Filter out n itself.
- Return the palindrome with:
  - Minimum absolute difference from n
  - Smaller value in case of a tie

5. Edge Cases

Input	Explanation
"1"	Closest palindromes: 0 and 2 → return 0
"1000"	Mirror of prefix → 1001, also consider 999
"999"	Candidates: 1001, 989, 100 etc.
"10"	Candidates: 9, 11 → return 9

6. Complexity Analysis

 Time Complexity

- $O(1)$ – We only generate a constant number (~ 5) of palindromic candidates.

 Space Complexity

- $O(1)$ – No extra space used apart from a small candidate set.

7. Alternative Approaches

Approach	Pros	Cons
Brute Force (Check palindromes around n)	Simple to implement	Not scalable for 18-digit numbers
Convert to int and iterate	Works for small inputs	Very slow for large n
String mirroring and edge case checks (our method)	Fast and efficient	Slightly tricky logic

8. Algorithm

- Convert string n to an integer num.
- Extract the prefix of n.
- Create palindromes by:
  - Mirroring prefix
  - Mirroring prefix + 1
  - Mirroring prefix - 1
- Add edge palindromes:
  - 10^(length-1) - 1 (e.g., 999)
  - 10^length + 1 (e.g., 10001)
- Remove the original number n from candidates.
- Return the closest based on:
  - Minimum absolute difference
  - Smaller value if tie

9. Test Cases

Input	Output	Explanation
"123"	"121"	Closer than 131
"1"	"0"	Only valid smaller palindrome
"11"	"9"	Closer than 22
"100"	"99"	Closer than 101
"999"	"1001"	Equal diff with 999 itself (ignored), 1001 and 989 → 989 is closer

10. Final Thoughts

- This problem is an excellent example of combining string manipulation and numeric reasoning.
- Optimizing by reducing the search space using mirrored prefix and edge cases is key to solving large inputs efficiently.
- Always remember to exclude the original number and resolve ties by choosing the smaller value.