Strobogrammatic Numbers

Simple & Visual Guide

Mausam Kar

February 4, 2026

Contents

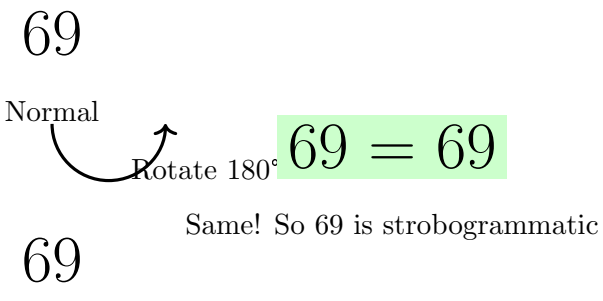
1 What is Strobogrammatic Number?

Simple Definition

A number that looks THE SAME when rotated 180°!
Like flipping upside down.

1.1 Visual Example

Example: 69



1.2 Which Digits Work?

Only some digits look the same when rotated:

Digit	Rotated 180°	Valid?
0	0	YES
1	1	YES
6	9	YES
8	8	YES
9	6	YES
2,3,4,5,7	?	NO

Valid pairs:

- 0 0
- 1 1
- 6 9
- 8 8
- 9 6

1.3 Examples

Strobogrammatic:

- 0 → Rotated = 0
- 1 → Rotated = 1
- 8 → Rotated = 8
- 11 → Rotated = 11
- 69 → Rotated = 69
- 88 → Rotated = 88
- 818 → Rotated = 818
- 69006 → Rotated = 69006

NOT Strobogrammatic:

- 2 → Can't rotate
- 12 → 2 can't rotate
- 69 → Rotated = 69
- 690 → Rotated = 096 690

2 Hash Map - Simple Explanation

2.1 What is Hash Map?

Simple Idea

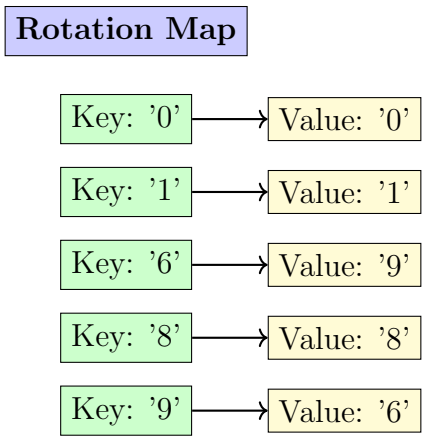
A Hash Map stores pairs: KEY → VALUE
Like a dictionary:

- Word (KEY) → Definition (VALUE)

In our case:

- Digit (KEY) → Rotated digit (VALUE)

2.2 Visual: Hash Map for Rotation



2.3 How to Use It

Example: Check if '6' can rotate

1. Look up '6' in map
2. Find: map['6'] = '9'
3. So '6' rotates to '9'

Example: Check if '2' can rotate

1. Look up '2' in map

2. NOT FOUND!

3. So ’2’ cannot rotate

3 Algorithm - Detailed Explanation

3.1 The Strategy

Two-Pointer Approach

Use two pointers: LEFT and RIGHT
Check from both ends moving inward!
LEFT starts at beginning
RIGHT starts at end

3.2 Step-by-Step Logic

For number "818":

1. **Initialize:**
   - left = 0 (first index)
   - right = 2 (last index)
   - Create rotation map
2. **Loop while left right:**
   - Get left digit: num[left]
   - Get right digit: num[right]
   - Check: map[left digit] = right digit?
   - If YES → Move inward (left++, right–)
   - If NO → Return FALSE
3. **If all pairs match → Return TRUE**

3.3 Visual: Two-Pointer Movement

Check: map['8']='8' and num[R]='8'? YES Check: map['1']='1' and num[R]='1'? YES

All pairs match → TRUE!

4 Flowchart

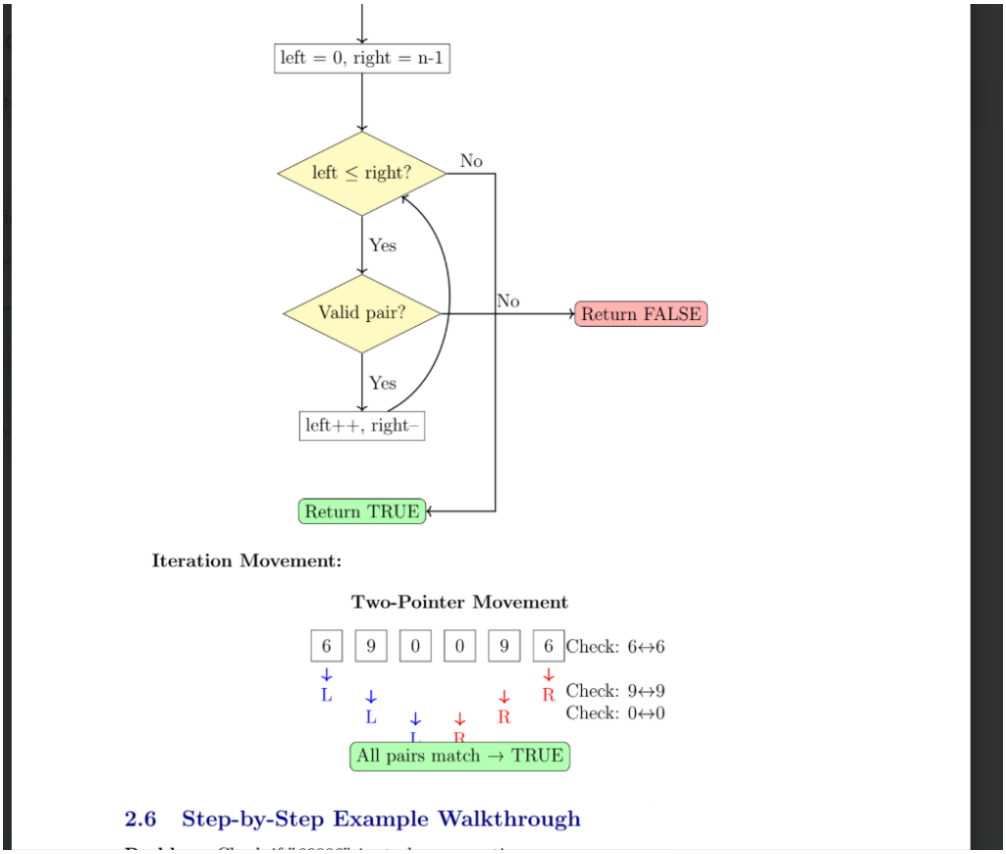


Figure: Algorithm flowchart showing the validation process

5 Complete Iteration Tables

5.1 Example 1: ”818”

Number: ”818”

Iteration	L	R	num[L]	num[R]	Check
1	0	2	’8’	’8’	map[’8’]=’8’, num[R]=’8’
2	1	1	’1’	’1’	map[’1’]=’1’, num[R]=’1’
End	2	0	-	-	L ¿ R, STOP

Result: All pairs match → ”818” is strobogrammatic

5.2 Example 2: ”69006”

Number: ”69006”

Iteration	L	R	num[L]	num[R]	Check
1	0	4	’6’	’6’	map[’6’]=’9’, num[R]=’6’
STOP	-	-	-	-	Mismatch! Return FALSE

Result: ”69006” is NOT strobogrammatic

5.3 Example 3: ”69” (Detailed)

Number: ”69”
Rotation Map: {00, 11, 69, 88, 96}

Step 1: Initialize

- left = 0
- right = 1

Iteration 1:

- num[0] = ’6’, num[1] = ’9’
- Check: map[’6’] = ’9’
- Does ’9’ == ’9’? YES
- Move: left = 1, right = 0

Loop ends: left ¿ right
Result: All matched → TRUE

6 Pseudocode

```
IS_STROBOGRAMMATIC(num):

1. Create map:
   map = {
       '0': '0',
       '1': '1',
       '6': '9',
       '8': '8',
       '9': '6'
   }

2. left = 0
3. right = length(num) - 1

4. While left <= right:
5.     leftDigit = num[left]
6.     rightDigit = num[right]

7.     If leftDigit NOT in map:
8.         Return FALSE

9.     If map[leftDigit] != rightDigit:
10.        Return FALSE

11.    left = left + 1
12.    right = right - 1

13. Return TRUE
```

6.1 Line Explanation

Lines 1-8: Create rotation map with valid pairs
Lines 2-3: Initialize pointers at both ends
Line 4: Loop while pointers haven't crossed
Lines 5-6: Get digits at current positions
Lines 7-8: Check if left digit can rotate
Lines 9-10: Check if rotation matches right digit
Lines 11-12: Move pointers inward
Line 13: All checks passed → strobogrammatic!

7 Java Code - Simple & Working

Listing 1: Strobogrammatic Number - Complete Code

```java
import java.util.*;

public class StrobogrammaticNumber {

    public static boolean isStrobogrammatic(String num) {
        // Step 1: Create rotation map
        Map<Character, Character> map = new HashMap<>();
        map.put('0', '0');
        map.put('1', '1');
        map.put('6', '9');
        map.put('8', '8');
        map.put('9', '6');

        // Step 2: Initialize two pointers
        int left = 0;
        int right = num.length() - 1;

        // Step 3: Check from both ends
        while (left <= right) {
            char leftDigit = num.charAt(left);
            char rightDigit = num.charAt(right);

            // Check if left digit can rotate
            if (!map.containsKey(leftDigit)) {
                return false;  // Can't rotate
            }

            // Check if rotation matches right digit
            if (map.get(leftDigit) != rightDigit) {
                return false;  // Doesn't match
            }

            // Move pointers inward
            left++;
            right--;
        }

        // All checks passed!
        return true;
    }

    public static void main(String[] args) {
        // Test cases
```

```
        String[] tests = {"0", "1", "8", "11", "69", "88", "818",
                          "69006", "96", "609"};

        System.out.println("Checking␣Strobogrammatic␣Numbers:");
        System.out.println("=".repeat(40));

        for (String num : tests) {
            boolean result = isStrobogrammatic(num);
            System.out.println(num + "␣   ␣" +
                               (result ? "   ␣Strobogrammatic" : "   ␣
                                  NOT␣strobogrammatic"));
        }
    }
}
```

Output:

```
Checking Strobogrammatic Numbers:
========================================
0 →  Strobogrammatic
1 →  Strobogrammatic
8 →  Strobogrammatic
11 →  Strobogrammatic
69 →  Strobogrammatic
88 →  Strobogrammatic
818 →  Strobogrammatic
69006 →  NOT strobogrammatic
96 →  NOT strobogrammatic
609 →  NOT strobogrammatic
```

8 Summary

Strobogrammatic Number:
A number that looks the same when rotated 180°

Valid digits:

- 00, 11, 69, 88, 96

Algorithm:

1. Create rotation map
2. Use two pointers (left, right)
3. Check each pair from ends moving inward
4. All pairs must match rotation

Time: O(n) - Check each digit once
Space: O(1) - Fixed size map

8.1 Key Points

Hash Map: Stores digit → rotated digit
Two Pointers: Check from both ends simultaneously
Why it works: Rotation reverses AND rotates digits!