

■ String to Integer (atoi)

Parse an integer from a string — handle ALL the edge cases

● String

■ MEDIUM

■ What You Will Learn

- How to parse integers character by character
- Handling leading spaces, sign (+/-), and non-digit chars
- Integer overflow detection BEFORE it happens
- The ordered checklist approach to tricky parsing

■ Intuition First

atoi is really an exercise in edge cases. The steps: skip spaces → read sign → read digits → stop at non-digit → clamp to int range. The key challenge is overflow: if your number grows bigger than 2,147,483,647 (INT_MAX), clamp it — don't let it explode.

Input variations to handle:

```
'42'          → 42
' -42'        → -42   (leading spaces + sign)
'4193 words'  → 4193  (stop at non-digit)
'-91283472332' → -2147483648 (overflow → clamp to MIN)
'words 123'   → 0     (starts with non-digit)
```

Algorithm flow:

```
[skip spaces] → [read sign] → [read digits] → [overflow check]
    ↓           ↓           ↓           ↓
    i++       sign=+1/-1   result=result*10   clamp to
while ' '           +digit           INT range
```

■ Java Code

```
1  class Solution {
2  public int myAtoi(String s) {
3      int i = 0, n = s.length(), sign = 1;
4      long result = 0;
5
6      // Step 1: Skip leading whitespace
7      while (i < n && s.charAt(i) == ' ') i++;
8
9      // Step 2: Read optional sign
10     if (i < n && (s.charAt(i) == '+' || s.charAt(i) == '-')) {
11         sign = (s.charAt(i) == '-') ? -1 : 1;
12         i++;
```

```

13     }
14
15     // Step 3: Read digits
16     while (i < n && Character.isDigit(s.charAt(i))) {
17         result = result * 10 + (s.charAt(i) - '0');
18         // Step 4: Overflow check
19         if (result * sign > Integer.MAX_VALUE) return Integer.MAX_VALUE;
20         if (result * sign < Integer.MIN_VALUE) return Integer.MIN_VALUE;
21         i++;
22     }
23     return (int)(result * sign);
24 }
25 }

```

■ Complexity

Type	Value	Meaning
Time	$O(n)$	One pass through the string
Space	$O(1)$	Only a few variables

■ Common Mistakes

■ Mistake 1

Using int instead of long for result. The number could overflow int before you have a chance to clamp it. Use long during calculation.

■ Mistake 2

Checking overflow AFTER adding the digit. Check BEFORE or during, otherwise you've already overflowed.

■ Quick Revision

✓	4 steps in order: skip spaces → sign → digits → overflow
✓	Use long for result to safely detect overflow
✓	Stop reading at first non-digit character
✓	Clamp: $> \text{INT_MAX} \rightarrow \text{return MAX}$, $< \text{INT_MIN} \rightarrow \text{return MIN}$
✓	Time $O(n)$, Space $O(1)$