Description | Editorial | Solutions (6.2K) | Submissions

8. String to Integer (atoi)

Medium 3.3K 10.1K

Companies

Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s `atoi` function).

The algorithm for `myAtoi(string s)` is as follows:

Read in and ignore any leading whitespace.

Check if the next character (if not already at the end of the string) is `'-'` or `'+'`. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.

Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.

Convert these digits into an integer (i.e. `"123" -> 123`, `"0032" -> 32`). If no digits were read, then the integer is `0`. Change the sign as necessary (from step 2).

If the integer is out of the 32-bit signed integer range $[-2^{31}, 2^{31} - 1]$, then clamp the integer so that it remains in the range. Specifically, integers less than -2^{31} should be clamped to -2^{31}, and integers greater than $2^{31} - 1$ should be clamped to $2^{31} - 1$.

Return the integer as the final result.

Note:

- Only the space character `' '` is considered a whitespace character.
- **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

Example 1:

```
Input: s = "42"
Output: 42
```

Java | Auto

```java
class Solution {
    public int myAtoi(String s) {
        int len = s.length();
        if (len == 0) {
            return 0; // Handle empty string case
        }
        double num = 0;
        int i = 0;
        while (i < len && s.charAt(i) == ' ') {
            i++;
        }
        if (i == len) {
            return 0; // All characters are whitespace
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s =
"42"

Output

42

Expected

42

Contribute a testcase

Console | Run | Submit

Description | Editorial | Solutions (11.3K) | Submissions

7. Reverse Integer

Medium | 10.6K | 12.1K

Companies

Given a signed 32-bit integer `x`, return `x` *with its digits reversed*. If reversing `x` causes the value to go outside the signed 32-bit integer range `[-2^31, 2^31 - 1]`, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123
Output: 321
```

Example 2:

```
Input: x = -123
Output: -321
```

Example 3:

```
Input: x = 120
Output: 21
```

Constraints:

- `-2^31 <= x <= 2^31 - 1`

Accepted 2.6M | Submissions 9.4M | Acceptance Rate 27.5%

Java | Auto

```java
class Solution {
    public int reverse(int x) {
        long reverse = 0;
        while (x != 0) {
            int digit = x % 10;
            reverse = reverse * 10 + digit;
            x = x / 10;
        }
        if (reverse > Integer.MAX_VALUE || reverse < Integer.MIN_VALUE) return 0;
        return (int) reverse;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

x =
123

Output

321

Expected

321

Contribute a testcase

Console | Run | Submit

Description | Editorial | Solutions (5.9K) | Submissions

6. Zigzag Conversion

Medium | 6.2K | 12.3K

Companies

The string `"PAYPALISHIRING"` is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

```
P   A   H   N
A P L S I I G
Y   I   R
```

And then read line by line: `"PAHNAPLSIIGYIR"`

Write the code that will take a string and make this conversion given a number of rows:

```
string convert(string s, int numRows);
```

Example 1:

```
Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"
```

Example 2:

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:
P     I    N
A   L S  I G
Y A   H R
P     I
```

Java | Auto

```java
class Solution {
    public String convert(String s, int numRows) {
        if (numRows == 1) {
            return s;
        }
        StringBuilder result = new StringBuilder();
        int n = s.length();
        int cycleLen = 2 * numRows - 2;
        for (int i = 0; i < numRows; i++) {
            for (int j = 0; j + i < n; j += cycleLen) {
                result.append(s.charAt(j + i));
                if (i != 0 && i != numRows - 1 && j + cycleLen - i < n) {
                    result.append(s.charAt(j + cycleLen - i));
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s =
"PAYPALISHIRING"

numRows =
3

Output

"PAHNAPLSIIGYIR"

Expected

"PAHNAPLSIIGYIR"

Console | Run | Submit

Description | Editorial | Solutions (7.9K) | Submissions

5. Longest Palindromic Substring

Hint

Medium | 25.2K | 1.5K

Companies

Given a string `s`, return *the longest palindromic substring* in `s`.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consist of only digits and English letters.

Accepted **2.4M** | Submissions **7.4M** | Acceptance Rate **32.4%**

Seen this question in a real interview before? 1/4

Yes | No

Discussion (148)

Java | Auto

```java
class Solution {
    int maxLen = 0;
    int lo = 0;
    public String longestPalindrome(String s) {
        char[] input = s.toCharArray();
        if(s.length() < 2) {
            return s;
        }

        for(int i = 0; i<input.length; i++) {
            expandPalindrome(input, i, i);
            expandPalindrome(input, i, i+1);
        }
        return s.substring(lo, lo+maxLen);
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

s =
"babad"

Output

"bab"

Expected

"bab"

Contribute a testcase

Console | Run | Submit