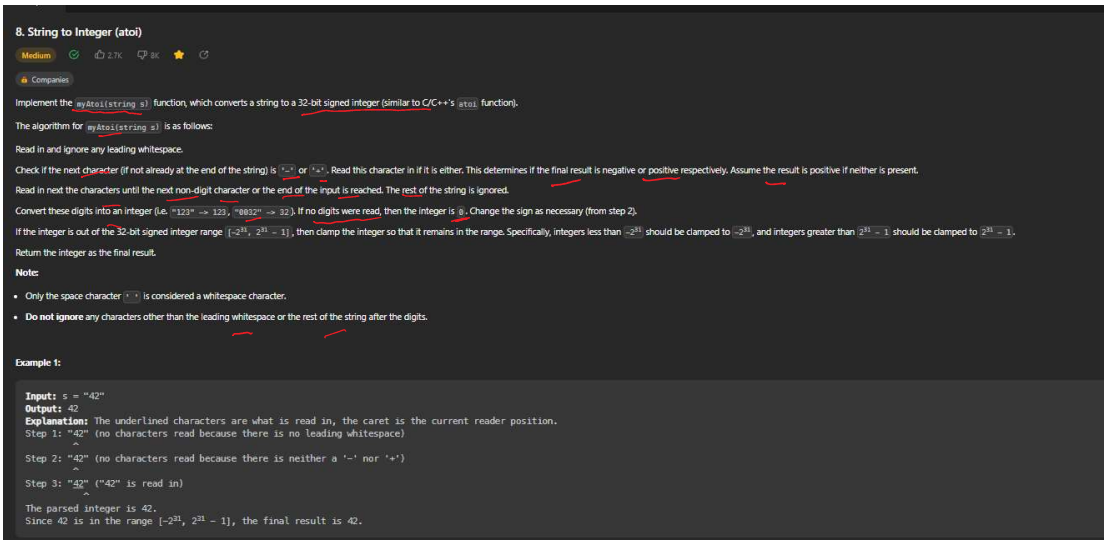
Recursion

26 December 2022 08:17



8. String to Integer (atoi)

Medium 2.7K 8K

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
Explanation: The underlined characters are what is read in, the caret is the current reader position.
Step 1: "42" (no characters read because there is no leading whitespace)
         ^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
         ^
Step 3: "42" ("42" is read in)
           ^
The parsed integer is 42.
Since 42 is in the range [-2^31, 2^31 - 1], the final result is 42.
```

Screen clipping taken: 26-12-2022 08:17

Case should be taken care :-

1. null or empty string.
2. White spaces
3. +/- sign.
4. calculate real values
5. handle min or max.

Exp: 42 → 42

① str == null ⊗ str.length < 1 ⊗

str = str.trim() → 42.

+42.

-42 → i=1

char flg = '+' i=0.

if (str.charAt(0) == '+')

flag = '+'

i++

else flag = '-'

i++;

double res=0.

while (str.length > i && str.charAt(i) >= '0' && str.charAt(i) <= '9')

2 > 1

res = res * 10 + (str.charAt(i) - '0');

i++;

}

if (flag == '-')

res = -res;

if (res > Integer.MAX_VALUE)

{ return Integer.MAX_VALUE;

0*10 + (4 - '0')

res

40 + (2 - '0')

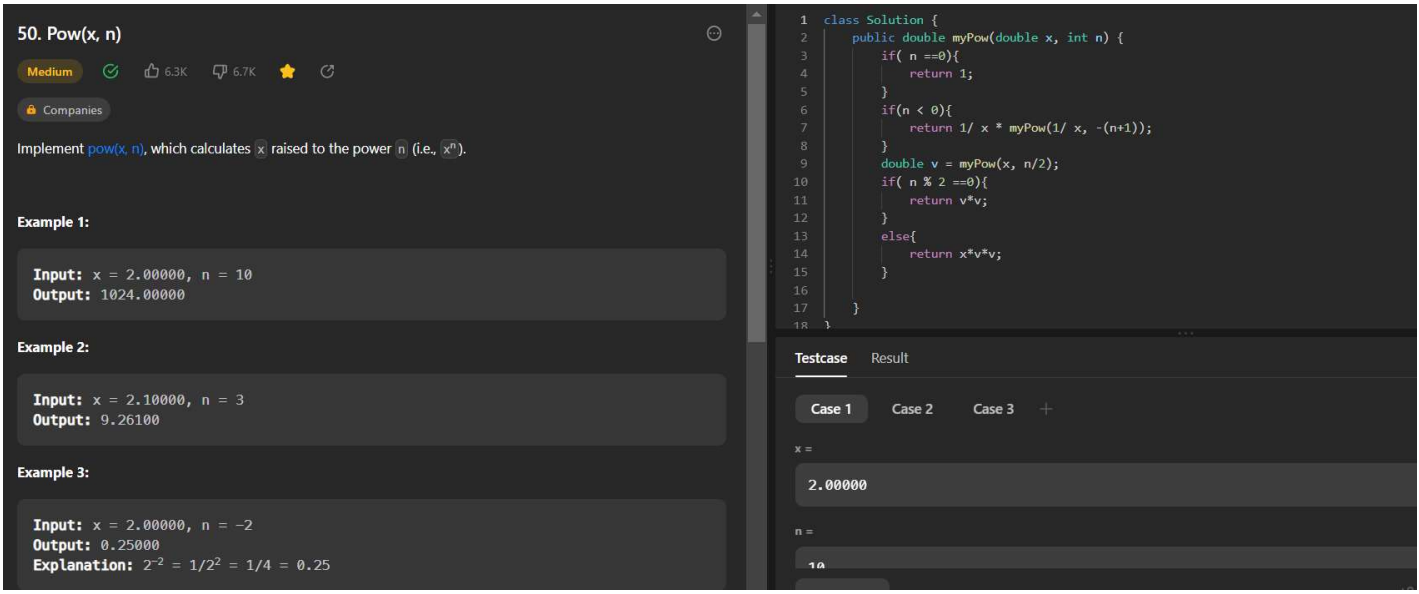
= 42

One test getting failed:

```java
class Solution {
    public int myAtoi(String str) {

      if(str == null || str.length()==0) return 0;
        // trim white spaces
        str = str.trim();

        // check negative or positive
        char flag = '+';
        int i=0;


        if(str.charAt(0)=='-'){
            flag = '-';
            i++;
        }
        else if(str.charAt(0)=='+')
            i++;
        // else str = "+" + str;

        // Store Result
        double res = 0;
        while(i < str.length() && str.charAt(i) >= '0' && str.charAt(i) <= '9'){
            res = res * 10 + (str.charAt(i)-'0');
            i++;
        }

        if(flag == '-')
            res = -res;

        // handle max and min
        if(res>Integer.MAX_VALUE) return Integer.MAX_VALUE;
        if(res<Integer.MIN_VALUE) return Integer.MIN_VALUE;

        return (int)res;
    }
}
```

Screen clipping taken: 26-12-2022 09:32

Que 2:-



Screen clipping taken: 26-12-2022 09:56

Dry run:- ①

double finalSection

v = pow(x, n/2) *

pow(2.00, 5)

32 x 32

x = 2.000000 , n = 10

check is (n == 0) ⊗

check is (n < 0) ⊗

check is (n % 2 == 0) ↳

return v * v; = 1024 ⇒ 32 * 32

else if (n % 2 != 0)

return x * v * v

n = 3

x = 2.1000

↳ (32 x 32)

x = 2.1000

return x * x * x

2.100 * 2.100 * 2.100

= 9.2610

2^{-2}

$= \frac{1}{2^2}$

$= \frac{1}{4}$

if (n < 0)

↳ return $\frac{1}{x * (\frac{1}{x}, (-n+1))}$

x = 2.000 , n = -2

$\frac{1}{2 * pow(\frac{1}{2}, (-1))}$

$\frac{1}{2} = 0.25$

n = -3

if ($\frac{1}{2 * (\frac{1}{2}, (-3+1))} = \frac{1}{2 * (\frac{1}{2}, (-2))} = \frac{1}{2 * 4} = \frac{1}{8} = 0.125$

que 3:- count Good numbers

Screen clipping taken: 26-12-2022 10:52

1922. Count Good Numbers

Hint

Medium 590 280

Companies

A digit string is **good** if the digits **(0-indexed)** at **even** indices are **even** and the digits at **odd** indices are **prime** (`2`, `3`, `5`, or `7`).

- For example, `"2582"` is good because the digits (`2` and `8`) at even positions are even and the digits (`5` and `2`) at odd positions are prime. However, `"3245"` is **not** good because `3` is at an even index but is not even.

Given an integer `n`, return *the **total** number of good digit strings of length* `n`. Since the answer may be large, **return it modulo** $10^9 + 7$.

A **digit string** is a string consisting of digits `0` through `9` that may contain leading zeros.

Example 1:

```
Input: n = 1
Output: 5
Explanation: The good numbers of length 1 are "0", "2", "4", "6", "8".
```

Example 2:

```
Input: n = 4
Output: 400
```

```java
class Solution {
    int mod = (int)1e9+7;
    public long pow(long x, long n){
        if(n == 0)
            return 1;
        long v= pow(x, n/2);
        if(n %2==0){
            return (v*v) % mod;
        }
        else{
            return (x*v*v) % mod;
        }
    }
    public int countGoodNumbers(long n) {
        long even = n /2 + n%2;
        long odd = n /2;
        long mul1 = pow(5, even) % mod;
        long mul2 = pow(4, odd) % mod;
        return (int) ((mul1 * mul2) % mod);
    }
}
```

Console

for n=1 _

_ one blank.

since 0 is even index, we can place either 0 or 2 or 4 or 6 or 8

5C1 out of 5 choices choose 1

for n=2 _ _ Two blanks -

0 2 4 6 8 since 1 is odd place.

2 3 5 7 4C1

So total 0, 2, 4, 6, 8 placing first blank.
2, 3, 5, 7 second blank

$5C_1 \times 4C_1 = 20$ Ans

for n=3 :- — — — three blanks-

$5C_1 * 4C_1 * 5C_1 = 100$

for n=4 :- $5C_1 * 4C_1 * 5C_1 * 4C_1 = 400$

If n = even → formula $5^{n/2} + 4^{n/2}$.

If n = Odd → $5^{n/2+1} \times 4^{n/2}$

Time complex O(logn)

que :- Reverse stack :-

Sort a stack

Easy Accuracy: **69.19%** Submissions: **79K+** Points: **2**

Bag Offers from Top Product Companies. Explore Exclusive Problems Now!

Given a stack, the task is to sort it such that the top of the stack has the greatest element.

Example 1:

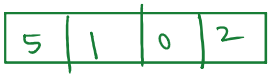
Input:
Stack: 3 2 1
Output: 3 2 1

Example 2:

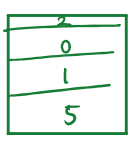
Input:
Stack: 11 2 32 3 41
Output: 41 32 11 3 2

Screen clipping taken: 26-12-2022 12:15

Sort an stack :-

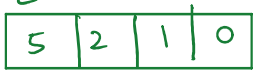


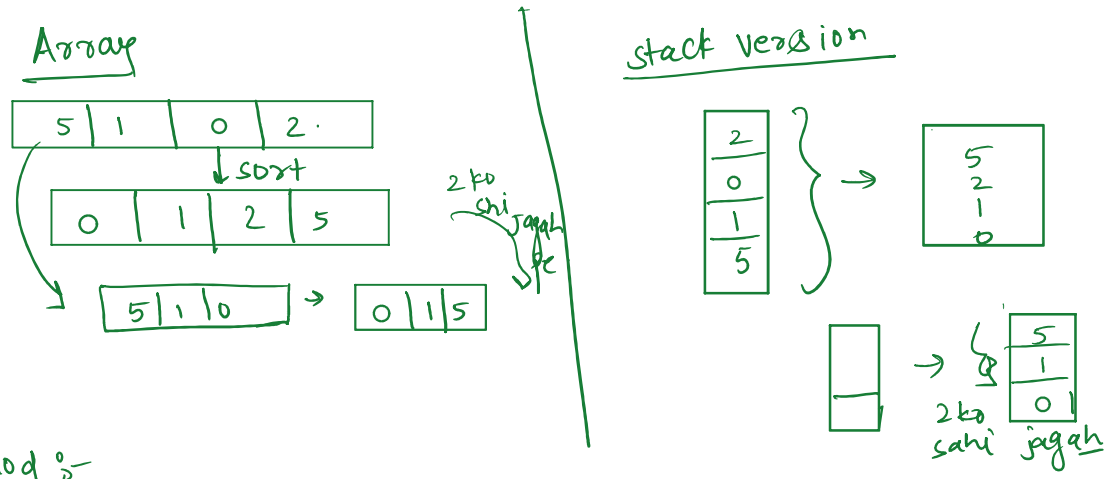
If



stack →

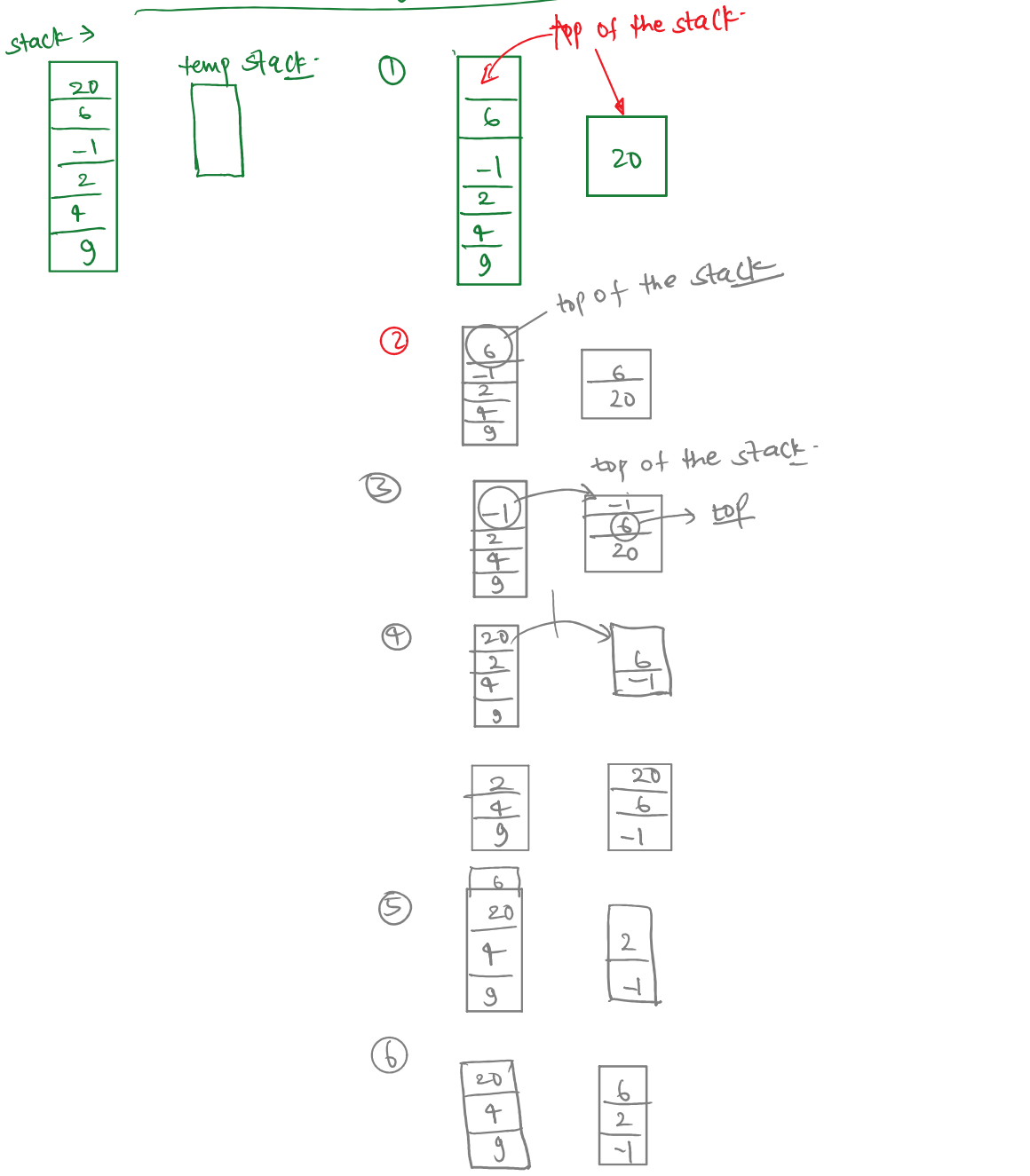
print

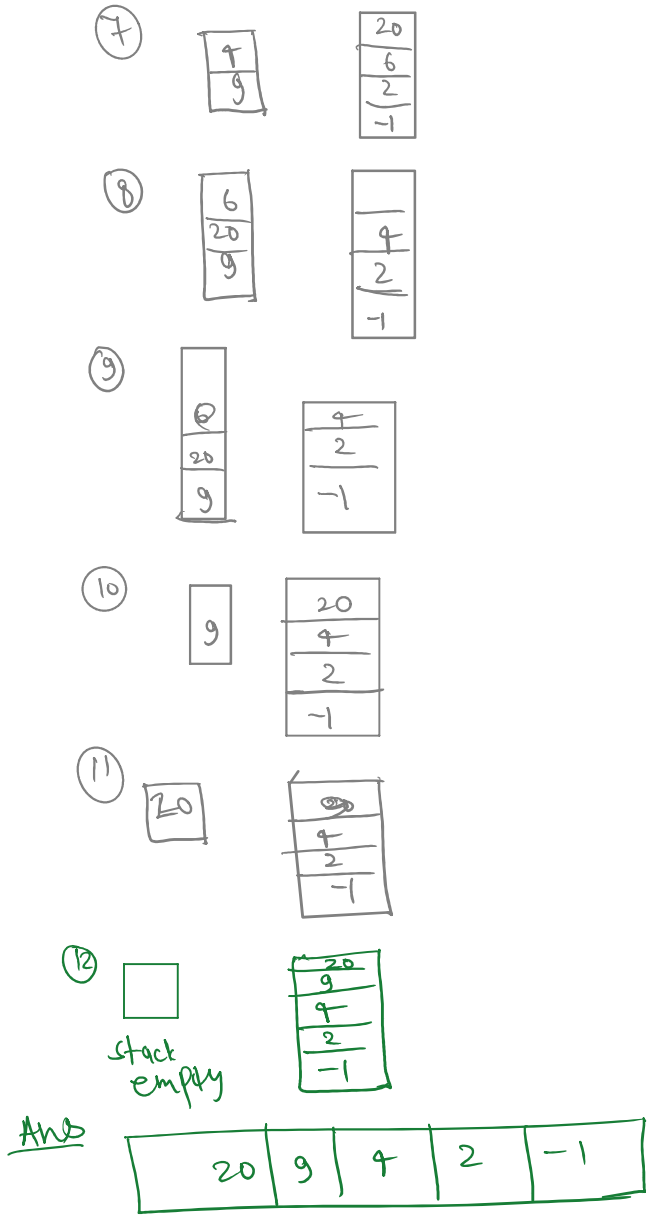




Method :-

sort a stack using temp stack :-





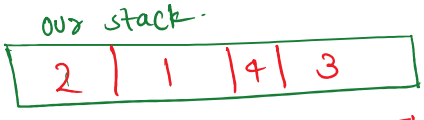

```java
/*Complete the function below*/
class GfG{
    public Stack<Integer> sort(Stack<Integer> s)
    {
        //add code here.
        Stack<Integer>tempstack = new Stack<Integer>();
        while(!s.isEmpty()){
            int temp = s.peek();
            s.pop();
            while(!tempstack.isEmpty() && tempstack.peek() > temp){
                s.push(tempstack.pop());
            }
            tempstack.push(temp);
        }
        return tempstack;
    }
}
```

TC:- O(n^2)

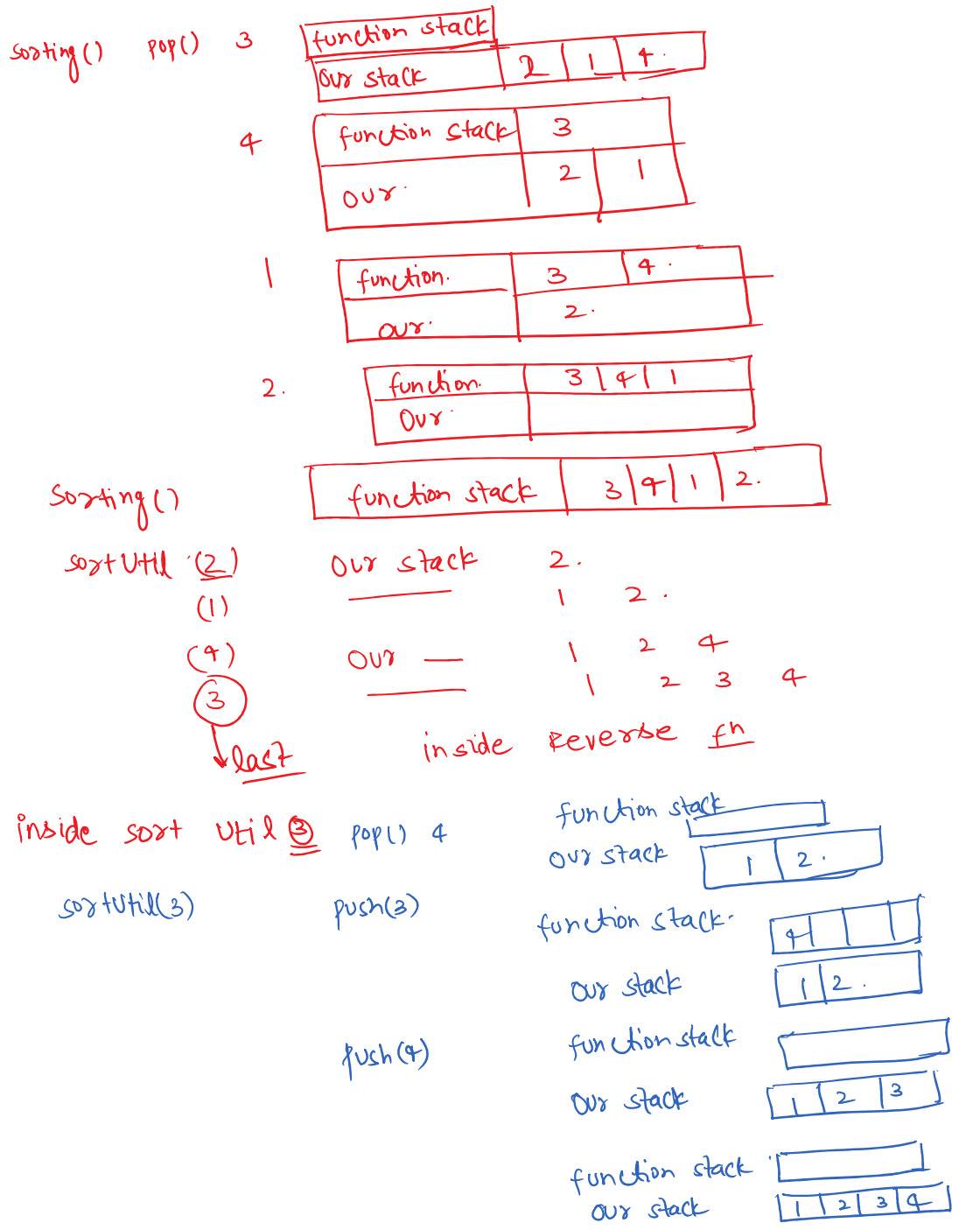
SC:- O(n)

Screen clipping taken: 26-12-2022 13:16

Using Recursion:-



sorting() pop() 3




```
6
26  /*Complete the function below*/
27  class GfG{
28
29      public void sortUtil(Stack<Integer> stack, int x){
30
31          //if stack is empty OR top of stack element is greater than the element x
32          //push item to stack, this will maintain the sorted order.
33          if(stack.size()==0 || stack.peek()>x){
34              stack.push(x);
35              return;
36          }
37          //if here means insertion of new element is breaking the sorted order,
38          //so pop it out for so that element x can be pushed to its right position
39
40          int y = stack.pop();
41          sortUtil(stack, x);
42
43          //once element x is placed then push y back to stack
44          stack.push(y);
45      }
46      public Stack<Integer> sort(Stack<Integer> original)
47      {
48          //add code here.
```

```java
/*Complete the function below*/
class GfG{

    public void sortUtil(Stack<Integer> stack, int x){

        //if stack is empty OR top of stack element is greater than the element x
        //push item to stack, this will maintain the sorted order.
        if(stack.size()==0 || stack.peek()>x){
            stack.push(x);
            return;
        }
        //if here means insertion of new element is breaking the sorted order,
        //so pop it out for so that element x can be pushed to its right position

        int y = stack.pop();
        sortUtil(stack, x);

        //once element x is placed then push y back to stack
        stack.push(y);
    }
    public Stack<Integer> sort(Stack<Integer> original)
    {
        //add code here.
      if(original.size()>0){

            ////pop out all the items from the stack and store it in function stack
            int x = original.pop();
            sort(original);

            //now insert the items into stack in sorted order
            sortUtil(original, x);
        }

        return original;
    }
}
```

Screen clipping taken: 26-12-2022 14:41

que 5 :- Reverse stack :-

Screen clipping taken: 26-12-2022 15:09

Reverse a Stack

Medium Accuracy: **81.66%** Submissions: **15K+** Points: **4**

You are given a stack **St**. You have to reverse the stack using recursion.

Example 1:

Input:
St = {3,2,1,7,6}
Output:
{6,7,1,2,3}

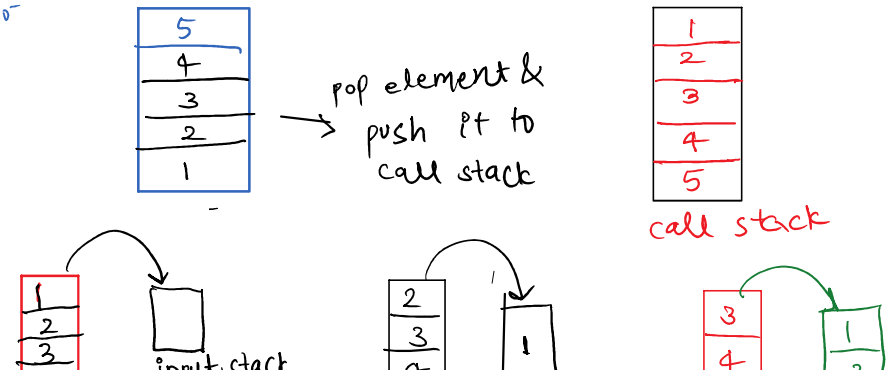
Example 2:

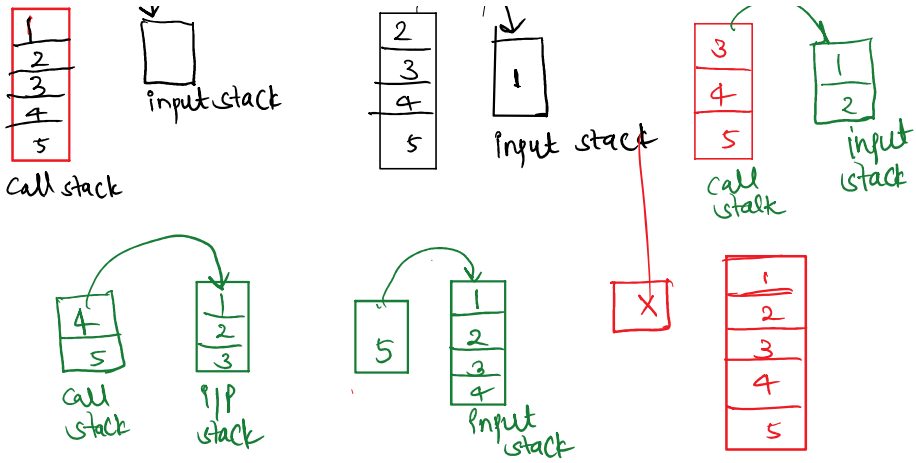
Input:
St = {4,3,9,6}
Output:
{6,9,3,4}

```java
//User function Template for Java

class Solution
{
    static void insertAtbottom(Stack<Integer>s , int item){
        if(s.empty()){
            s.push(item);
            return;

        }
        int top  = s.peek();
        s.pop();
        insertAtbottom(s, item);
        s.push(top);

    }
    static void reverse(Stack<Integer> s)
    {
        // add your code here
        if(s.empty()){
            return;
        }
        int item = s.peek();
        s.pop();
        reverse(s);
        insertAtbottom(s, item);
    }
}
```

Custom Input Compile &

Dry :-




1
2
3
4
5
call stack
input stack
2
3
4
5
1
input stack
3
4
5
call stack
1
2
input stack
4
5
call stack
1
2
3
i/p stack
5
1
2
3
4
input stack
X
1
2
3
4
5