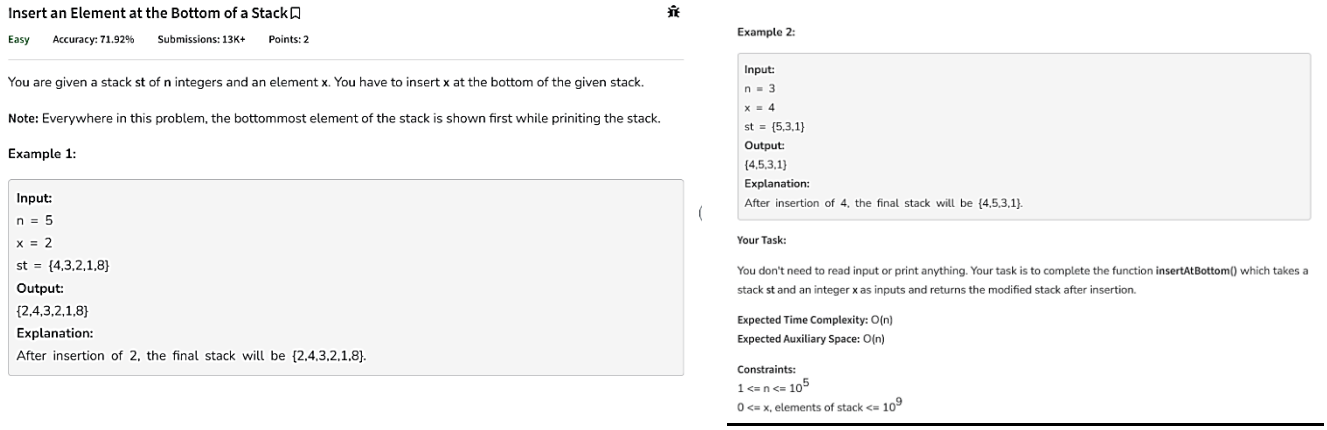
PROBLEM



Insert an Element at the Bottom of a Stack

Easy Accuracy: 71.92% Submissions: 13K+ Points: 2

You are given a stack **st** of **n** integers and an element **x**. You have to insert **x** at the bottom of the given stack.

Note: Everywhere in this problem, the bottommost element of the stack is shown first while priniting the stack.

Example 1:

Input:
n = 5
x = 2
st = {4,3,2,1,8}
Output:
{2,4,3,2,1,8}
Explanation:
After insertion of 2, the final stack will be {2,4,3,2,1,8}.

Example 2:

Input:
n = 3
x = 4
st = {5,3,1}
Output:
{4,5,3,1}
Explanation:
After insertion of 4, the final stack will be {4,5,3,1}.

Your Task:

You don't need to read input or print anything. Your task is to complete the function **insertAtBottom()** which takes a stack **st** and an integer **x** as inputs and returns the modified stack after insertion.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(n)

Constraints:
$1 <= n <= 10^5$
$0 <= x$, elements of stack $<= 10^9$

CODE

```
class Solution:
  def insertAtBottom(self,st,x):
    temp=[]

    while st:
      temp.append(st.pop())

    temp.append(x)

    while temp:
      st.append(temp.pop())

    return st
```

EXPLANATION

This code is a Python function defined inside a class called Solution. The purpose of this function, insertAtBottom, is to insert an element x at the bottom of a stack st.

Here's a line-by-line explanation:

temp = []: This line initializes an empty list called temp. This list will be used to temporarily hold elements from the stack.

while st:: This initiates a loop that continues as long as there are elements in the stack st.

temp.append(st.pop()): Inside the loop, each element from the stack st is popped off (removed from the top) and appended to the temp list. This effectively reverses the order of elements from st.

temp.append(x): After all elements from st are moved to temp, the element x that needs to be inserted at the bottom is appended to temp.

while temp:: This initiates another loop that continues as long as there are elements in the temp list.

st.append(temp.pop()): Inside this loop, each element from temp is popped off (removed from the top, since lists in Python are indexed from the left) and appended to the stack st. This effectively restores the original order of elements in st, with the newly added element x at the bottom.

Finally, the function returns the modified stack st with the element x inserted at the bottom.

In summary, this function first empties the stack st into a temporary list temp, then adds the new element x to the bottom of temp, and finally refills the stack st from temp, effectively placing x at the bottom of the original stack st.

Imagine you have a stack represented by a vertical arrangement of elements. Initially, the stack is empty:

|

|

|

|

Now, let's say you want to insert the element 5 at the bottom of the stack. Using the provided function insertAtBottom, here's what would happen step by step:

We start with an empty temporary list temp:

temp: []

We iterate through the stack, which is initially empty, so nothing happens here.

We append 5 to the temporary list temp:

temp: [5]

Now, we iterate through the temporary list temp, which contains 5.

We pop 5 from temp and push it onto the stack:

|

|

|

| 5

And that's it! We have successfully inserted 5 at the bottom of the stack.

Let's say now we want to insert 3 at the bottom of the stack. Following the same steps:

We start with an empty temporary list temp.

We iterate through the stack, which currently contains only 5.

We append 5 to the temporary list temp.

We append 3 to the temporary list temp.

temp: [5, 3]

Now, we iterate through the temporary list temp.

We pop 3 from temp and push it onto the stack:

|

|

| 3

| 5

We pop 5 from temp and push it onto the stack:

|

| 5

| 3