Stack minimum:

stack has function of push and pop . can you also add function ‘min’ to the stack and it solution also execute in O(1)

Solution:

```
class MinStack:
    def __init__(self):
        # Initialize the main stack and the min stack
        self.stack = []       # The main stack to store values
        self.min_stack = []   # The min stack to keep track of minimum values

    def push(self, value):
        # Add the value to the main stack
        self.stack.append(value)

        # If the min stack is empty or the value is less than or equal to the top value of
the min stack,
        # add the value to the min stack to keep track of the minimum values
        if not self.min_stack or value <= self.min_stack[-1]:
            self.min_stack.append(value)

    def pop(self):
        # Check if the main stack is empty
        if not self.stack:
            return None

        # Remove the top value from the main stack
        value = self.stack.pop()

        # If the popped value is the same as the top value of the min stack,
        # it means the minimum value has been removed, so remove it from the min stack as
well
        if value == self.min_stack[-1]:
            self.min_stack.pop()

        return value

    def min(self):
        # Check if the min stack is empty
        if not self.min_stack:
            return None

        # Return the top value of the min stack, which represents the minimum value in the
stack
        return self.min_stack[-1]
```

Explanation:

1. The push function adds the value to the main stack. If the min stack is empty or the value is less than or equal to the top value of the min stack, the value is also added to the min stack.

2. The pop function removes the top value from the main stack. If the popped value is the same as the top value of the min stack, it means the minimum value has been removed, so it is also removed from the min stack.

3. The min function returns the top value of the min stack, which represents the minimum value in the stack.

4. Both push and pop operations have a time complexity of O(1), and the min operation also executes in O(1) time complexity since it only involves accessing the top value of the min stack.

Bonus 1:

Explain one real world use case where stack is better used Data Structure than array

One real-world use case where a stack data structure is better used than an array is in implementing the "undo" functionality in text editor.

In text editors , users often perform various operations like typing, deleting, formatting, etc. Sometimes users may make mistakes or want to revert back to a previous state. The "undo" functionality allows users to undo their recent actions and restore the previous state.

1. Whenever a user enters or deletes text, for example, the specifics of that action (such as the text input or deleted) are saved in a data structure known as a "undo stack."
2. The most recent action is the one that is pushed to the top of the stack.
3. The top element of the stack is popped, and the related operation is reversed, if the user wants to undo the previous action. For instance, if typing a character was the previous action, erasing that character would be the undo operation.
4. In case the user decides to undo an operation later, the undone activities can be saved in a different data structure known as a "redo stack."

Undo actions are possible because the most recent action is easily available at the top of the stack. The stack's LIFO (Last-In-First-Out) feature also fits nicely with the user actions order.

An array, on the other hand, wouldn't work as well in this situation. To keep track of the most recent action and its index with an array, you would either need to shift elements or keep additional pointers.