Parvatibai Chowgule College of Arts and Science
Autonomous
Accredited by NAAC with Grade 'A' (CGPA Score 3.41 on a 4 Point Scale in 3rd cycle)
Best affiliated College-Goa University Silver Jubilee Year Award

Department of Computer Science

2020 - 2021

M.Sc-IT (Part 1)

Course: `Data Structures and Algorithms`

Course Code: `MIT11`

Name: `Alexander Roque Rodrigues`
Roll Number: `202202`

Problem Statements.

Contents

1 Design a stack to add new function GetMinimum(), which retrieves minimum number from stack in O(1)

Date: 21-09-2020

1.1 Problem Statement

Design a stack to add new function GetMinimum(), which retrieves minimum number from stack in $O(1)$.

1.2 Input

- First line of input contains an intger T denoting count of the numbers.
- Next T lines of input contains a integer number S.

1.3 Conditions

- $1 \leq T \leq 10$
- $1 \leq |S| \leq 10^5$

1.4 Source Code

1.4.1 MinStack Class Code

```python
#!/usr/bin/env python3
from typing import Any
from collections import deque

class MinStack:

  # constructor function
  def __init__(self, factor: int):
    self.factor = factor
    self.s = deque()
    self.currentMinimum = None
    print("Factor: " ,self.factor)

  # push to stack
  def push(self, element: int) -> None:
    # if new stack with no elements
    if not self.s:
      self.s.append(element)
      self.currentMinimum = element

    # if greater than min, directly append to the top of the stack
    elif element > self.currentMinimum:
      self.s.append(element)

    # else is new minimum, append after calculations
    else:
      self.s.append(self.factor * element - self.currentMinimum)
      self.currentMinimum = element

  def pop(self) -> None:
    # no element on stack
    if not self.s:
      print("Under Flow Occured.")

    # top is the smallest
    if self.s[-1] < self.currentMinimum:
      # update with new minimum from the stack
      self.currentMinimum = self.factor * self.currentMinimum - self.s[-1]

    # invoke pop method after checking for new minimum value
    self.s.pop()

  def minimum(self):
    # return minimum
    return self.currentMinimum

# end of class definition #
```

1.5 Output

[1]:
```python
# import class
import minStack as m
from random import randint
stack = m.MinStack()
```

[2]:
```python
# Push and Pop Operations begin on the stack.
stack.push(-10)
stack.minimum()
# new minimum = -10
```

[2]:
```
-10
```

[3]:
```
# push 10
stack.push(10)
stack.minimum()
```

[3]:
```
-10
```

[4]:
```
# new minimum = -10
stack.push(-20)
stack.minimum()
```

[4]:
```
-20
```

[5]:
```
# pop -20, now minimum should go back to -10
stack.pop()
stack.minimum()
```

[5]:
```
-10
```

[6]:
```
# push and dont change minimum
stack.push(15)
stack.minimum()
```

[6]:
```
-10
```

[7]:
```
# new minimum so store diff and new min = -15
stack.push(-15)
stack.minimum()
```

[7]:
```
-15
```

[8]:
```
# pop -15, therefore min = -10
stack.pop()
stack.minimum()
```

[8]:
```
-10
```

1.6 References

1. Ladd, S., Xin, Y., Yang, J., Liu, P., & Wu, L. (1998). Java suan fa = JAVA ALGORITHMS. Beijing: Dian Zi Gong ye Chu Ban She.

2 Book Stack

You have three stacks of books where each book has the same length, but they may vary in height. You can change the height of a stack by removing and discarding its topmost book any number of times.