18CS601
Foundations of Computer Science : Data Structures & Algorithms

CASE STUDY PROBLEM # 9

DEEP PANCHOLI [CB.EN.P2CSE20011]

SURAJ SINGH [CB.EN.P2CSE20028]

Contents

Objective Statement

- At the Bank of Amritanagar, a bank teller decides to go to lunch and a line forms with m people. Number the customers 1, 2, ... , m.
- When the teller returns, the teller tells the person at the head of the line to go to the back of the line, doing this n time. Then the teller serves the person at the head of the line.
- The teller repeats this process until the line is empty (assuming no one else enters the bank).
- Write a method lastCustomer that has two parameters that specify the number of customers initially in line (m) and the number of customers sent to the back of the line each time (n).
- The method should return the number of the customer that is served LAST. You may assume that $m > 0$ and $1 \leq n \leq m$.

Example

```
Enter the number of customers: 6
Enter the number of rotations of queue: 2
[1, 2, 3, 4, 5, 6]
Customer 1 sent back.
[2, 3, 4, 5, 6, 1]
Customer 2 sent back.
[3, 4, 5, 6, 1, 2]
Served Customer 3
Customer 4 sent back.
[5, 6, 1, 2, 4]
Customer 5 sent back.
[6, 1, 2, 4, 5]
Served Customer 6
Customer 1 sent back.
[2, 4, 5, 1]
Customer 2 sent back.
[4, 5, 1, 2]
Served Customer 4
Customer 5 sent back.
[1, 2, 5]
Customer 1 sent back.
[2, 5, 1]
Served Customer 2
Customer 5 sent back.
[1, 5]
Customer 1 sent back.
[5, 1]
Served Customer 5
The last customer to be served: 1.
```

Strategy used to solve the problem

1. Define a class Queue.
2. Data members: one list to store the customer queue at any point of time.
3. Member functions:
   - enqueue: take a number and enqueue in the front of queue.
   - dequeue : dequeues and element from the front of queue.
   - first: returns the first element in the queue without dequeue operation.
4. Driver Code: takes the number of customers(m), and rotations in one go (n) as input from user and computes the rotations on queue as per m and n and finds the last customer to be served by teller.

```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, data):
        self.items.append(data)

    def dequeue(self):
        return self.items.pop(0)

    def first(self):
        return self.items[0]
```

```
for i in range(customers,0,-1):
    for _ in range(n,0,-1):
        temp = q.dequeue()
        q.enqueue(temp)
        print(q.items)
```

Which Data Structure used and Why?

Queue:

The customers sequence is stored in a queue.

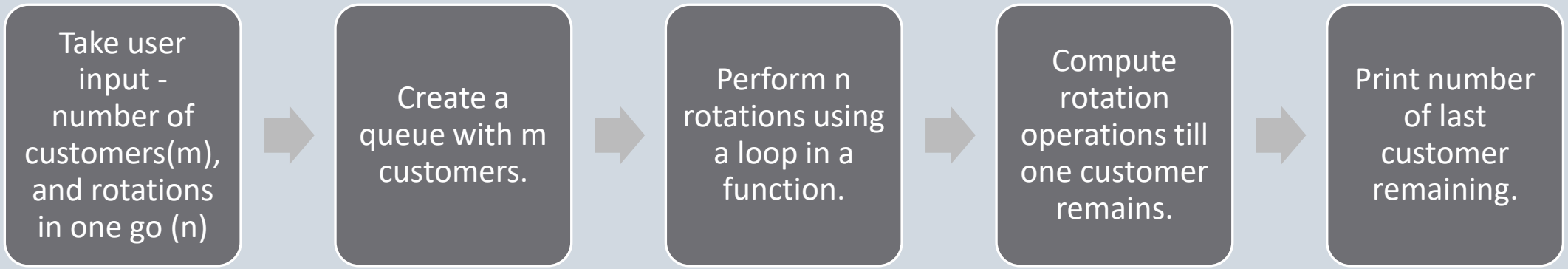
```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, data):
        self.items.append(data)

    def dequeue(self):
        return self.items.pop(0)
```

Why?

Because using a queue it is easy to perform rotation operation (i.e., dequeue n elements from the front and enqueue them at the back).

Performing the above operation with a looping condition will allow us to find the last customer.

Algorithm


Take user input - number of customers(m), and rotations in one go (n)
Create a queue with m customers.
Perform n rotations using a loop in a function.
Compute rotation operations till one customer remains.
Print number of last customer remaining.

Expected Outcome

The program is expected to show the correctly rotated states of the customers queue and also the last one to be served by the teller.

->Code Demo and Output

Thank You!

```
Enter the number of customers: 6
Enter the number of rotations of queue: 2
[1, 2, 3, 4, 5, 6]
Customer 1 sent back.
[2, 3, 4, 5, 6, 1]
Customer 2 sent back.
[3, 4, 5, 6, 1, 2]
Served Customer 3
Customer 4 sent back.
[5, 6, 1, 2, 4]
Customer 5 sent back.
[6, 1, 2, 4, 5]
Served Customer 6
Customer 1 sent back.
[2, 4, 5, 1]
Customer 2 sent back.
[4, 5, 1, 2]
Served Customer 4
Customer 5 sent back.
[1, 2, 5]
Customer 1 sent back.
[2, 5, 1]
Served Customer 2
Customer 5 sent back.
[1, 5]
Customer 1 sent back.
[5, 1]
Served Customer 5
The last customer to be served: 1.
```