Queue using Stacks

Day 8

This method takes sure that oldest entered element is always at top of the stack 1. so that dequeue operation just pops from Stack 1. To put the element at top of Stack 1, Stack 2 used.

Programme :-

```
class queue:
    def __init__(self):
        self.s1 = []
        self.s2 = []

    def enqueue(self,x):
        while(len(self.s1) != 0):
            self.s2.append(self.s1[-1])
            self.s1.pop()
        self.s1.append(x)
        while(len(self.s2) != 0):
            self.s1.append(self.s2[-1])
            self.s2.pop()
```

```
def dequeue(self):
    if len(self.s1)==0:
        Print("Queue is empty")
    else:
        x=self.s1[-1]
        self.s1.pop()
        return x
```

Complexity Analysis:

Time Complexity:

Push operation : O(N)

In the worst case we have ~~one~~ empty whole of stack 1 into stack 2

pop operation : O(1)

Same as pop operation in stack

Auxiliary Space:

We of stack for storing value

Reversing a Queue

* enqueue : Add an item x to rear of queue.

* dequeue : Remove item from front of queue.

* empty : Check if a queue is empty or not.

Programme :-

```
from queue import Queue

def Print(Queue):
    while(not queue.empty()):
        print(queue.queue[0], end=" ")
        queue.get()

def reverse queue:
    Stack = []
    while (not queue.empty()):
        Stack.append(queue.queue[0])
        queue.get()
```

```
while (len(stack) != 0):
    queue.pop(Stack[-1])
    Stack.pop()
```

Reversing a queue using recursion:

```
queue reverse function(queue)
{
  if (queue is Empty)
      return queue
  else:
      data.front()
      queue.pop()
      queue.reversefunction(queue):
      q.push(data)
      return queue
}
```

Programme

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

    def isempty(self):
        return self.items ==[]

    def add(self, items):
        self.items.append(item)

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

    def front(self)
        return self.items[0]

    def frontQueue(self):
        for i in self.items:
            print[i, end = " "]
        print(" ")
```

```
def reverseQueue(q)
    if q.isEmpty():
        return
    data = q.front()
    q.pop()
    reverseQueue(q)
    q.add(data)
```