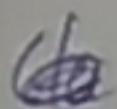


## Queue Using Stacks

Day 8

This method takes care that oldest entered element is always at top of the stack 1, so that dequeue operation just pops from Stack1. To put the element at top of stack1, Stack2 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  $\text{len}(\text{self}.s1) == 0$ :

Print ("Queue is empty")

else:

$x = \text{self}.s1[-1]$

$\text{self}.s1.\text{POP}()$

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:

use 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 reverseQueue :
```

```
    stack = []
```

```
    while (not queue.empty()):
```

```
        stack.append(queue.queue[0])
```

```
        queue.get()
```

while (len(stack) != 0):

queue.pop(stack[-1])

stack.pop()

: (H02) 2nd

: (H02) - 3rd - fib

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, item):

    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)