Data Structures

Queue Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Queue using 2 Stacks: **O(1) dequeue**

- Implement Queue functionalities using 2 stack objects
- However, the **dequeue**() function must remain O(1)
- Implement only these requested functionalities

```
qu = Queue()

for i in range(1, 5):
    qu.enqueue(i)

while not qu.empty():
    print(qu.dequeue(), end = ' ')
# 1 2 3 4
```

```
class Queue:
    def __init__(self):
        self.stk1, self.stk2 = Stack(), Stack()
```

Problem #2: Priority Queue

- Priority queue is a queue in which each element has a "**priority**" associated with it. Elements of **higher priority** are ALWAYS SERVED before elements of lower priority
- Assume that we have an OS comprised of tasks, each of priority 1, 2, or 3
  - Assume we enqueued them as follows:
  - Enqueue (task_id = 1131, priority = 1)
  - Enqueue (task_id = 3111, priority = 3)
  - Enqueue (task_id = 2211, priority = 2)
  - Enqueue (task_id = 3161, priority = 3)
  - Let's print the tasks in order: 3111 3161 2211 1131
    - That is: to dequeue we must first get from priority 3, if nothing from 2, if nothing from 1
- Implement a priority queue class by **black box utilization** of linked-list queue

- Priority Queue for tasks
- Priority is from 1 to 3
- Display: 1 row per priority
- dequeue()
  - asks of higher priority should be returned first
  - So, priority 3 tasks are retrieved first, then 2, then 1
- Time complexity
  - O(1) for all operations

```
tasks = PriorityQueue()

tasks.enqueue(1131, 1)
tasks.enqueue(3111, 3)
tasks.enqueue(2211, 2)
tasks.enqueue(3161, 3)

tasks.display()
# Priority #3 tasks 3111, 3161
# Priority #2 tasks 2211
# Priority #1 tasks 1131

print(tasks.dequeue())  # 3111
print(tasks.dequeue())  # 3161
```

- In the future, we will learns the **heap data structure**, which can be used for priority queues
  - But priority > 1 [not limited]

```
tasks.enqueue(1535, 1)
tasks.enqueue(2815, 2)
tasks.enqueue(3845, 3)
tasks.enqueue(3145, 3)

tasks.display()
# Priority #3 tasks 3845, 3145
# Priority #2 tasks 2211, 2815
# Priority #1 tasks 1131, 1535

while not tasks.empty():
    print(tasks.dequeue(), end = ' ')
# 3845 3145 2211 2815 1131 1535
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."