UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 9

Queues

Submitted by:
Balaoro, Judge Wayne B.

Instructor:
Engr. Maria Rizette H. Sayo

October 11, 2025

I. Objectives

Introduction

Another fundamental data structure is the queue. It is a close “the same” of the stack, as a queue is a collection of objects that are inserted and removed according to the first-in, first-out (FIFO) principle. That is, elements can be inserted at any time, but only the element that has been in the queue the longest can be next removed.

The Queue Abstract Data Type

Formally, the queue abstract data type defines a collection that keeps objects in a sequence, where element access and deletion are restricted to the first element in the queue, and element insertion is restricted to the back of the sequence. This restriction enforces the rule that items are inserted and deleted in a queue according to the first-in, first-out (FIFO) principle. The queue abstract data type (ADT) supports the following two fundamental methods for a queue Q:

Q.enqueue(e): Add element e to the back of queue Q.

Q.dequeue( ): Remove and return the first element from queue Q;
an error occurs if the queue is empty.

The queue ADT also includes the following supporting methods (with first being analogous to the stack’s top method):

Q.first(): Return a reference to the element at the front of queue Q, without removing it;
an error occurs if the queue is empty.

Q.is empty( ): Return True if queue Q does not contain any elements.

len(Q): Return the number of elements in queue Q; in Python, we implement this with the special method len .

This laboratory activity aims to implement the principles and techniques in:

- Writing Python program using Queues

Writing a Python program that will implement Queues operations

II. Methods

Instruction: Type the python codes below in your Colab. Reconstruct them by implementing Queues (FIFO) algorithm. Hint: You may use Array or Linked List

```
# Stack implementation in python

# Creating a stack
def create_stack():
    stack = []
    return stack
```

```
# Creating an empty stack
def is_empty(stack):
    return len(stack) == 0

# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("Pushed Element: " + item)

# Removing an element from the stack
def pop(stack):
    if (is_empty(stack)):
        return "The stack is empty"
    return stack.pop()

stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))

print("The elements in the stack are:"+ str(stack))
```

Answer the following questions:

1. What is the main difference between the stack and queue implementations in terms of element removal?
2. What would happen if we try to dequeue from an empty queue, and how is this handled in the code?
3. If we modify the enqueue operation to add elements at the beginning instead of the end, how would that change the queue behavior?
4. What are the advantages and disadvantages of implementing a queue using linked lists versus arrays?
5. In real-world applications, what are some practical use cases where queues are preferred over stacks?

III. Results

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

    def enqueue(self, dataval):
        if dataval not in self.queue:
            self.queue.append(dataval)
            return True
        else:
            return False

    def dequeue(self):
        if len(self.queue) <= 0:
            return "Queue is empty"
        else:
            return self.queue.pop(0)

    def empty_stack(self):
        while self.queue:
          self.queue.pop()
        print("The Queue is empty.")


AQueue = Queue()
AQueue.enqueue("1")
AQueue.enqueue("2")
AQueue.enqueue("3")
AQueue.enqueue("4")
AQueue.enqueue("5")

print("The elements in the Queue are:")
print(AQueue.queue)

AQueue.dequeue()
print("After removing an element from the Queue:")
print(AQueue.queue)

AQueue.empty_stack()
```

```
The elements in the Queue are:
['1', '2', '3', '4', '5']
After removing an element from the Queue:
['2', '3', '4', '5']
The Queue is empty.
```

Figure 1. Implementation of Queue

1. The difference is that on stack the removed element is from the recent one that you added in your elements. In this case, it's the last element. While on Queue, the removed element is the first element that you enqueue in your list and also on Stack we use pop while on Queue we call it dequeue.
2. It returns that the queue is empty since we have an if-else statement that checks if the length of elements is less than 0, it will say that the Queue is empty.
3. In my understanding, it will still be a queue only if we dequeue at the end of the Queue so that it will implement the FIFO (First In, First Out)
4. Using a linked list is the best choice since it's fast at both adding an element and removing an element. Also it can grow to any size that you need. In array, it might be a bad choice since its slow in removing an element in the front. Although there's a faster

way for it to be fast. It is still complex and fixed in size. So, the linked list is the best combination for Queue.

5. The first one I could think of is printing documents, since printing the last document that you sent doesn’t make it confusing and illogical since the last document you sent will be printed while the first one will be stuck at the bottom. If this is applied to printing shops, it will enrage the first customer that asks you to print his/her document while the recently arrived will get their print first..

IV. Conclusion

In this Laboratory Report 9 entitled Queue. I was able to review the Queues that we learned back then. I learned to convert a stack program into a queue program. To answer the questions. It requires an understanding of both stack and queue. Now, I was able to clearly differentiate the two and apply them in the program. In the last question, it asks me to think of a real world application and in my experience since back then we have a printing shop, sometimes there's a customer complaining when they submit their document to be printed while the one who just arrived get their printed documents first.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.

[2] GeeksforGeeks. (2025, July 23). *Queue data structure*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/queue-data-structure/

[3]*W3Schools.com*. (n.d.). https://www.w3schools.com/dsa/dsa_data_queues.php