Assignment 2: Double Linked List for Queue

Part A

Files

1. queue.cpp
   - My queue implementation

2. driver.cpp
   - Correctness and time complexity tests written with the help of Claude Opus 4.6.

Driver Output

```
=== Correctness Tests ===
[PASS] Basic enqueue/dequeue (FIFO order)
[PASS] Newly created queue is empty
[PASS] Queue is empty after draining
[PASS] Single element enqueue/dequeue
[PASS] Interleaved enqueue/dequeue
[PASS] Large volume (100000 elements) FIFO

=== O(1) Complexity Tests ===

--- Enqueue timing (avg ns) ---
n=1000        => 27.8372 ns/op
n=10000       => 36.441 ns/op
n=100000      => 27.7499 ns/op
n=1000000     => 28.0417 ns/op

--- Dequeue timing (avg ns) ---
n=1000        => 5.77278 ns/op
n=10000       => 5.66613 ns/op
n=100000      => 5.86472 ns/op
n=1000000     => 5.93044 ns/op

--- isEmpty timing (avg ns) ---
n=1000        => 0.91676 ns/op
n=10000       => 0.9173 ns/op
n=100000      => 0.91684 ns/op
n=1000000     => 0.91697 ns/op

enqueue ratio (n=1000000 / n=1000) = 1.00735
[PASS] enqueue is O(1)

dequeue ratio (n=1000000 / n=1000) = 1.02731
[PASS] dequeue is O(1)
```

isEmpty ratio (n=1000000 / n=1000) = 1.00023
[PASS] isEmpty is O(1)

Reflection

It is my understanding that you only need a singly linked list in order it implement a queue with an enqueue and dequeue time complexity of O(1). This is because you only ever add to the end of a queue and remove from the front. By storing a pointer to the first and last item in the queue you avoid the need to traverse the list when enqueuing or dequeuing.

A doubly linked list is not required and only increases the memory usage of each node in the case of a regular queue. The only reason a doubly linked list would be needed is if you wanted to created a double ended queue that you could enqueue or dequeue from either end with O(1) time complexity.

Part B

Question

If we do in array with two extra pointers, it seems more efficient than doing the double linked list. what's the con for doing array? (think about overlow) and discuss about it (extra cost)

Response

Same as with the linked list implementation you only need two pointers to allow for O(1) time complexity enqueues and dequeues. This is true because no matter how many items we have in our array our pointers will always lead to the first and last item in the queue without needing to iterate through the array.

The main downside of using an array as opposed to a linked list is that arrays are of a fixed size. This means that if the queue is full you will be unable to add more items to it until some are removed or the queue is copied to a new larger array. This can get quickly become a costly operation as the size of the queue grows.

The main advantage of using an array instead of a linked list is memory usage. In our linked list implementation we require each node in the list to contain a pointer to the next node. This adds a small but not insignificant amount of memory required to store each item. With the array implementation entries are stored sequentially in memory allowing us to manage the queue using only two pointers.

Part C

Question

Please derive the closed form F(N) $F(N) = k F(N/2) + 1$ Assume: $F(1) = 1$ $N = 2^n$.

Response

$$F(N) = \begin{cases} \frac{k \cdot N^{\log_3 k} - 1}{k - 1} & \text{if } k \neq 1 \\ \log_2 N + 1 & \text{if } k = 1 \end{cases}$$