Exploration: Queue

1. What are the similarities and differences between a queue and a stack?

Similarities are that they are both Non-primitive data structures, they are both linear storing sequentially, and operations such as delete and insert can be placed on both of them. Differences, are the way in which they go about inserting and removing or the methods Stacks are LIFO while queues are FIFO.

1. Your reading discussed implementing a queue as a list. What are the performance implications of this implementation? List each operation and explain its associated time complexity.

Because the queue is FIFO (first in first out), when items get added to the end of the queue (Enqueue()) and remove items from the front of the queue(dequeue()) calling the function front() for example will return the first value without removing it. This means that if implemented as a List we need to decide which is the front end and which is the rear end. In the first implementation we assumed that the front of the queue is position 0. These time complexities dequeue() is O(1) whole enqueue() is O(n) since there are only two pointers/locations that these functions can access: front of the queue and the back. It doesn’t work like a loop.

1. How might you modify this implementation to ensure adding and removing elements happens in constant time?

Instead of the first time we implemented where we assumed that the front of the queue is position 0 we can instead assume that the rear of the dequeue is at position 0 in the list which will make adding and removing items from the list O(1)

1. What are some common ways that queues are used in computing?

Let’s say you have a website with thousands and thousands of users, you can organize the way in which they’re serves by FIFO, Also despite our thinking that computers are multitaskers they still (like humans) generally handle each request one by one. The order in which we do so is FIFO. If I click three things while the wheel of death is rolling, when it finally does load the tasks run in order.