**Computational Costs of Queue’**

So, considering the aspect of a pool table simulation, if the drive ball is struck then there are a number of collisions that could take place, so the balls on the outer set of the triangle would be struck first. If we use a priority sorted queue then we could sort based off the concept of which collisions would take place first. The outer balls would be struck first, then the inner set then the one that has two layers of balls before it. If all of these possible collisions are loaded into a Priority Queue by the time at which an event or collision takes place, then when you need to access which collisions would occur first then it’s simple to dequeue to the Queue to get the first collisions. Once the first collisions have been dequeued then the next dequeue would be the second level of collisions. We need a Priority Queue because if we didn’t sort by the time at which events occurred then there would not be able to keep track of which collisions happened at what times. Overall it would be significantly difficult to figure out which collisions occurred with solely event driven simulations.

The computational costs for using a priority queue would significantly less than that of an unsorted queue, as all collisions wouldn’t be stored in a queue sorted by the event at which each collision takes place, having to dequeue each event and sorting it based on which ones will occur first. If it is already sorted based on the time at which a collision can take place, then it’s only one dequeue operation to get the specified collision versus if the collision events are not sorted then the worst case may be that you have to dequeue every possible event before getting to the event that will occur, which can be anywhere from 1 to the amount of possible collisions. Which is greatly higher in computational cost.