Team: JVJ

Big-O Efficiency

**Main Operation:** Our main operation is to ultimately search for the rooms that lead to the desired room. Ultimately, we scan for all possible rooms by using six different operations to change the coordinates. Once all rooms are found, loop through the various paths found and choose the path that leads to the desired destination.

**Analysis:** Using the brute force approach, the algorithm we used searches all the rooms. Because all rooms are initially found, the operation starts off with O(N) search. Since there are 6 operations for every room searched, this multiplies the search by 6. There is a queue of various paths, one of which contains the destination we want. Additionally, we loop through the queue to find the correct path, creating an additional O(N/2) search the algorithm must perform.

**Complexity:** Since we used brute force to search for the desired destination, the runtime for the main operation is **O(N)**.

= 6N + N/2

= 6.5N

= 1N

O(N)

However, in later realization, we discovered a more optimal algorithm that could have been used to produce a more efficient run time. Rather than using a Breadth-First Search, we realized we could have used a Depth-First Search instead which would have skipped repeated rooms, consequently improving the Big-O efficiency of the algorithm.