Jenny Nguyen
CSCE313-591
September 21, 2021
PA1

According to the Ackermann function, the time taken to complete the allocation and free operations grow exponentially as you increase the parameters. For example, when keeping n the same number (n = 3 in this example) while increasing m incrementally, it is shown in the graph that the total time grows exponentially. I increased m by 1 every time I ran the program and I saw that it took longer for the program to run and for the output to display than its precedent. The graph in figure 1 clearly shows these observations through the trendline that is being displayed.

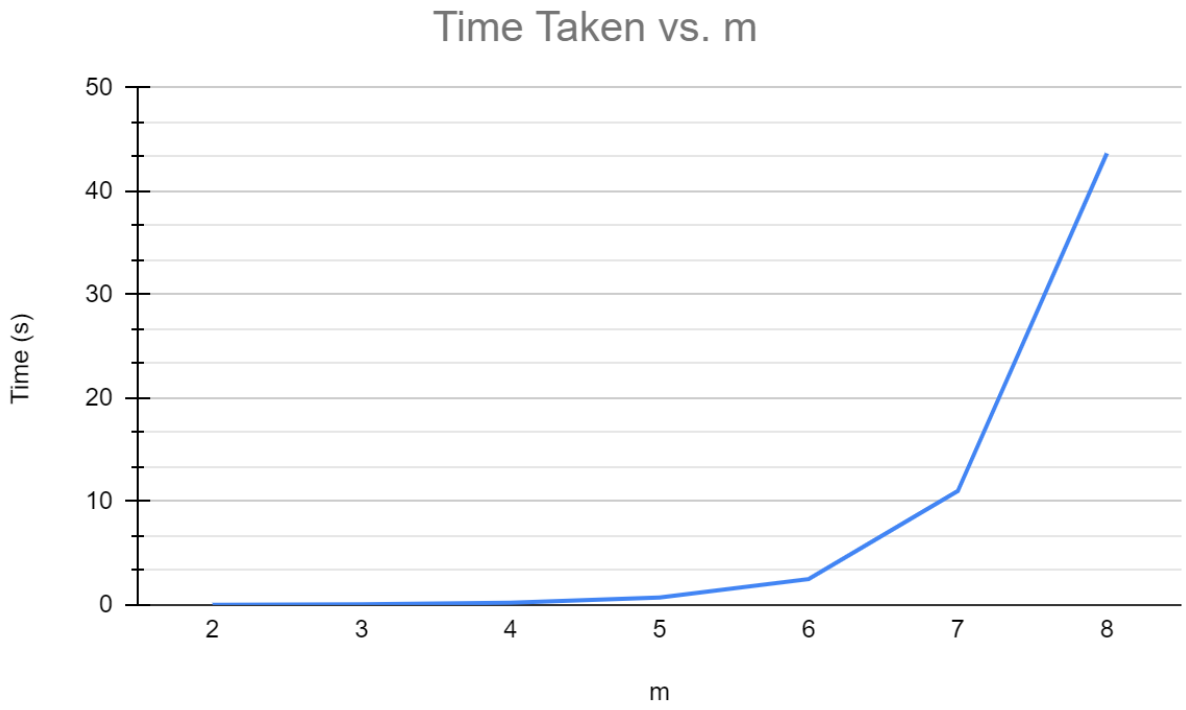

Figure 1.

Everytime you increase n or m, it will increase the number of operations that the program has to run. In this case, because there are recursive functions, in reference to the functions, it would loop through these functions a lot more as you increase the values of n and m. Because you are looping through the functions more, it increases the total time it takes for the program to finish running.

I’m not exactly sure what the bottleneck in this programming assignment is. When I compile and run the program, it seems to work just fine and doesn’t take too terribly long to run when using bigger numbers of m. The reason it takes a long time is because of the amount of iterations it must take to find a certain block, to split the blocks, and to insert them. You have to iterate through every single time to find the right index. But once you find the right index, you just “remember” it and use it in the other functions that need to start at that index. What makes the program run slower is because of the memory overhead.