Kim Nguyen

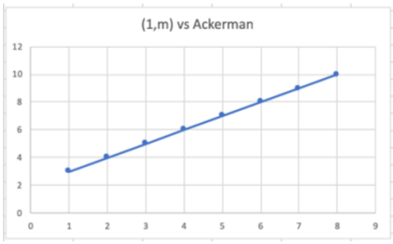
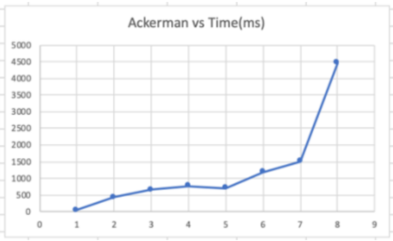
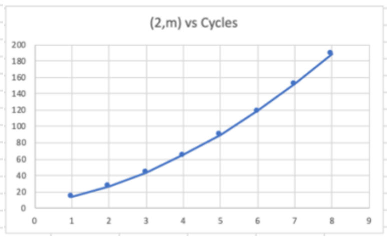
CSCE 313-502

September 20, 2019

PA1 – Report

The purpose of this assignment was to build a memory allocator that splits memory into blocks of two and allocates the smallest, larger block with a value of a power of two. In my implementation of this project, `free(char* a)` is most likely to be the bottleneck of the code because the `free(char* a)`'s while loop inside can run extraneously or infinitely, calling `merge(BlockHeader* block1, BlockHeader* block2)` every time. A way to make this function more efficient is be sure to short-circuit the prerequisite conditional before you can merge the two blocks. Since `merge(BlockHeader* block1, BlockHeader* block2)` is called in a loop, short circuiting the conditions can improve performance significantly as the program calls on `merge(BlockHeader* block1, BlockHeader* block2).`

In this report I have attached graphs: (1,m) vs Ackerman, Ackerman vs Time, and (2,m) vs Recursive Cycles in order to show that the Ackerman function is indeed recursive. I have also attached the excel files that I used to make these graphs.


(1,m) vs Ackerman

Ackerman vs Time(ms)

(2,m) vs Cycles