Jared Jones
CSCE 313- 508
UIN: 626001380

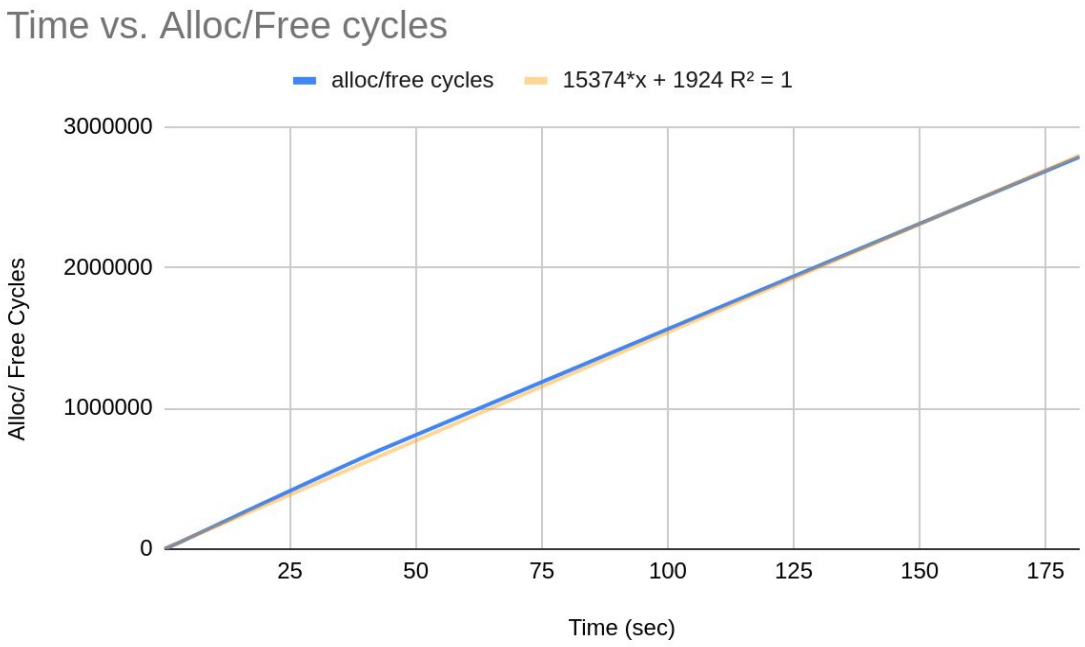
PA 1: Report

In this programming assignment a memory allocation system was created that replicates the type of memory management done by a computer by taking a large block of memory and breaking it down into smaller memory zones that would be used for each partition depending on the size and type of memory required. This program manages the blocks of memory by storing them in a vector of linked lists so that many different sizes and types of memory can be passed to the user and then returned and recombined together using the buddy system memory allocation scheme.

The below data was obtained by running multiple tests on the buddy allocator function with different values for n and m in the ackerman function to show the relation between runtime, the number of alloc/free cycles, and the amount of time each cycle takes.

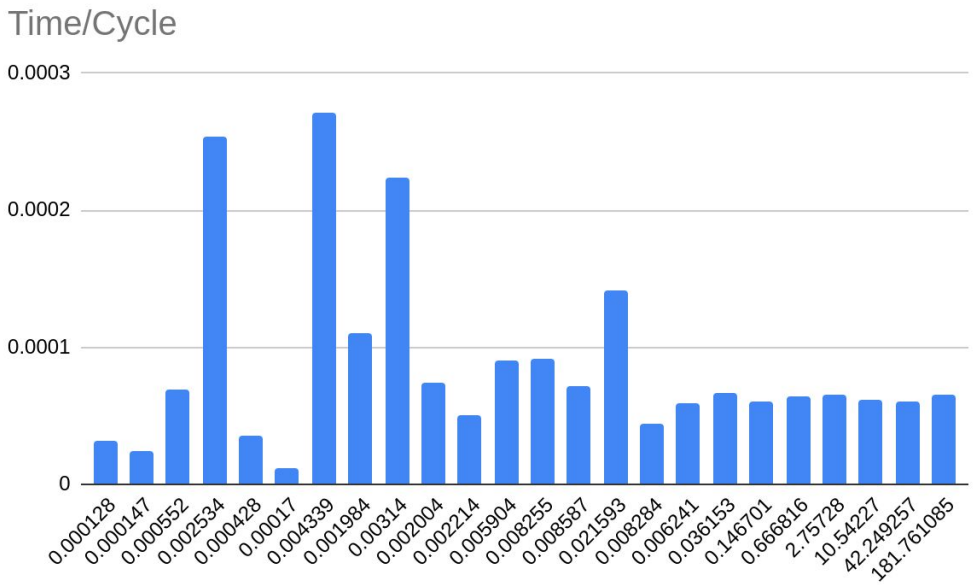
Basic Block Size	128					
Memory Length	100,000,000					
block header	16					
Ackerman		Preformance				
n	m	seconds	musec (10^-6)	total time	alloc/free cycles	time/cycle
1	1	0	128	0.000128	4	0.000032
1	2	0	147	0.000147	6	0.0000245
1	3	0	552	0.000552	8	0.000069
1	4	0	2534	0.002534	10	0.0002534
1	5	0	428	0.000428	12	0.000035666 66667
1	6	0	170	0.00017	14	0.000012142 85714
1	7	0	4339	0.004339	16	0.000271187 5
1	8	0	1984	0.001984	18	0.000110222 2222
2	1	0	3140	0.00314	14	0.000224285 7143
2	2	0	2004	0.002004	27	0.000074222 22222
2	3	0	2214	0.002214	44	0.000050318 18182

2	4	0	5904	0.005904	65	0.000090830 76923
2	5	0	8255	0.008255	90	0.000091722 22222
2	6	0	8587	0.008587	119	0.000072159 66387
2	7	0	21593	0.021593	152	0.000142059 2105
2	8	0	8284	0.008284	189	0.000043830 68783
3	1	0	6241	0.006241	106	0.000058877 35849
3	2	0	36153	0.036153	541	0.000066826 24769
3	3	0	146701	0.146701	2432	0.000060321 13487
3	4	0	666816	0.666816	10307	0.000064695 44969
3	5	2	757280	2.75728	42438	0.000064971 95909
3	6	10	542270	10.54227	172233	0.000061209 35012
3	7	42	249257	42.249257	693964	0.000060881 05003
3	8	181	761085	181.761085	2785999	0.000065240 90102



The above graph shows the linear relationship between the number of alloc / free cycles and the amount of time the program took to run. As the program tried to allocate and deallocate more memory the runtime increased directly as a result.

The amount of time per alloc/free cycle was also calculated and then the average time/cycle was found by ignoring many of the outliers produced through data with only a few cycles, and then averaging the calculations that had many alloc/free cycles to get a good approximate for the time/cycle, which was found to be 0.00006679723285. The graph below shows how the time/cycle slowly approached this value as the runtime and # of cycles increased.



This program's runtime could be improved by the use of a bitmap in order to keep track of sections of memory that are free or in use so that the program can quickly find and access the memory it is looking for as opposed to having to loop through the vector sequentially in order to find a block large enough and then if not there, split the block until it is the correct size. Bitmaps could be integrated with the current program to keep track of the different blocks so that the program could quickly and efficiently find blocks to allocate and free so it could reduce the time the program takes on each alloc/free cycle therefore reducing the runtime overall. The program could also be improved by reducing the size of the blockheaders by making the integer "blocksize" into a character that represents power of 2 size. Using an unsigned char, you can fit any value between 0 and 255 into the new log2 size. This char would only take up 1 byte as opposed to the 4 bytes an int would allowing for smaller block headers and more memory available to allocate to the user. The smaller the block header, the less allocate/free cycles the computer will have to do because the memory + block header is larger than a block. It would allow for compact memory management.