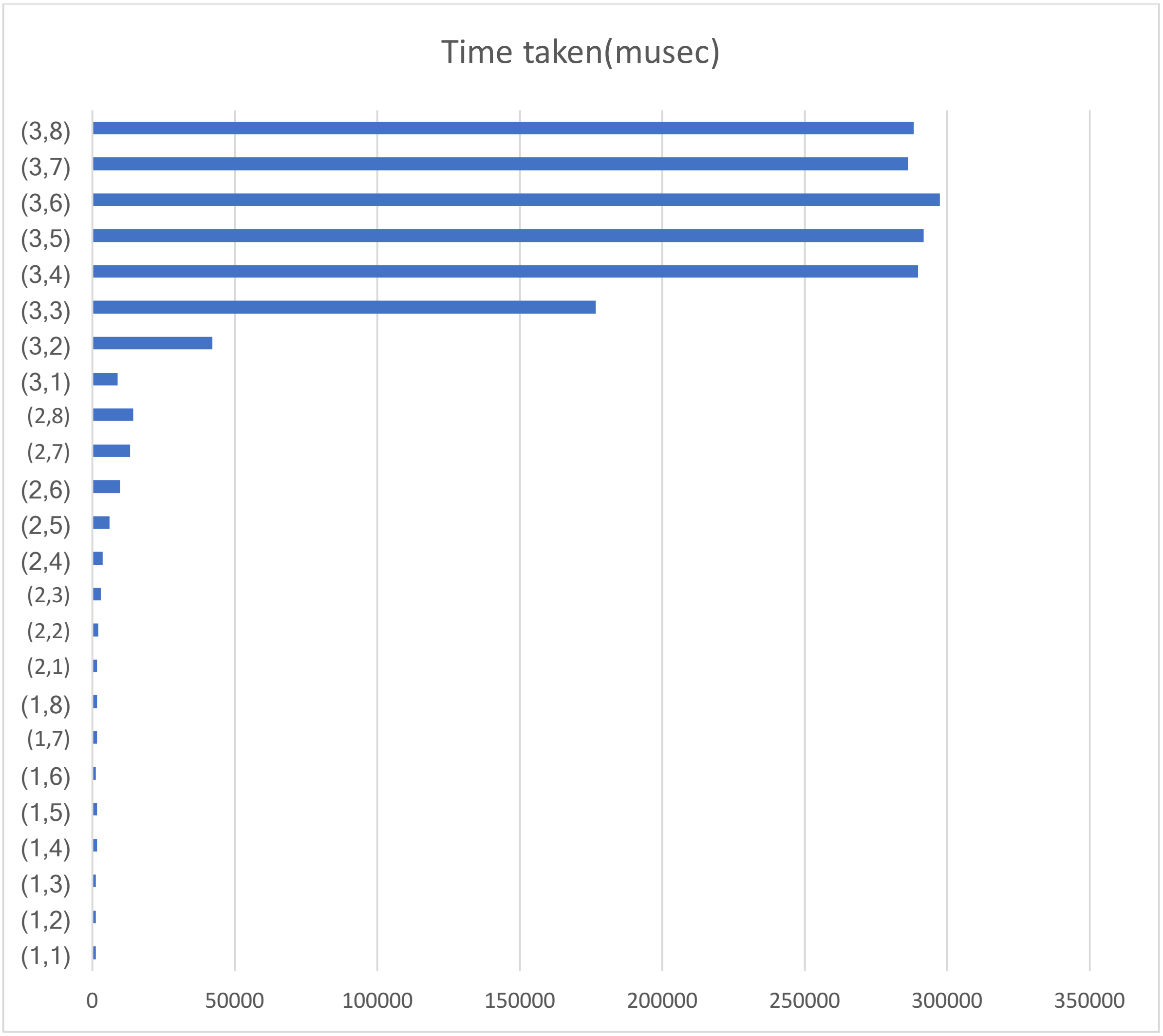
CSCE 313- Buddy System Memory Allocator

Name: Rahul Kumar

Overall, my Buddy System Memory Allocator was able to run the Ackerman test up to (3,3). My program passed for every value of M when N = 2 as well as when N = 1. When looking back and evaluating my program, I noticed some changes that I could have made in some of my functions, especially in my Alloc(), free() and split() functions. Before I created my Alloc() function, I needed to create a split function in order to get the smallest yet sufficient memory block for the user. I implemented this by first casting the char* block as a BlockHeader object and then creating a new BlockHeader that was equal to half of the original block. Next I calculate the index for removal of the original block. After that, I remove the original BlockHeader object from the freeList. I then increment the index in order to know where to insert the 2 new blocks that I create. Before inserting these blocks though, I make sure to set the size of both blocks to half of the original block.

Personally, I found writing my Alloc() whilst getting it to work properly with the rest of my functions very difficult. This is why I also believe that my Alloc() function could be improved and possibly presents a bottleneck to my program's overall robustness. After analyzing the function, I found that one of the few things I could've changed would be my index traversal. However, I found this to be the most efficient since I am able to go both up and down the freeList using one for loop. If I were to implement this any other way, I would have used 2 for loops. This would make it less complicated and the runtime would be n^2 either way.

One of the greatest bottlenecks in my code was my free() and merge() functions. After running several Ackerman tests and debugging, I found that my merge() function was keeping me from completing all Ackerman tests higher than (3,4). When I tested Ackerman values higher than (3,4), my program would slow down and the output would not make sense. I think that after attempting to execute higher Ackerman tests, my program began to have issues processing memory. Due to the time constraint of the programming assignment coupled with the amount of debugging required, I wasn't able to fully complete the program. I also believe that my free() function was inefficient and provided another bottleneck to my program, especially since it utilizes merge in order to add memory back to the freeList.


Time taken(musec)
(3,8)
(3,7)
(3,6)
(3,5)
(3,4)
(3,3)
(3,2)
(3,1)
(2,8)
(2,7)
(2,6)
(2,5)
(2,4)
(2,3)
(2,2)
(2,1)
(1,8)
(1,7)
(1,6)
(1,5)
(1,4)
(1,3)
(1,2)
(1,1)
0
50000
100000
150000
200000
250000
300000
350000

Ackerman test	Time taken(musec)	Ackerman value	Cycle count
(1,1)	1224	3	4
(1,2)	1230	4	6
(1,3)	1157	5	8
(1,4)	1499	6	10
(1,5)	1562	7	12
(1,6)	1178	8	14
(1,7)	1671	9	16
(1,8)	1552	10	18
(2,1)	1561	5	14
(2,2)	2075	7	27
(2,3)	2874	9	44
(2,4)	3598	11	65
(2,5)	5873	13	90
(2,6)	9710	15	119
(2,7)	13261	17	152
(2,8)	14312	19	189
(3,1)	8865	13	106
(3,2)	42041	29	541
(3,3)	176620	61	2432
(3,4)	289673	0	3833
(3,5)	291686	0	3835
(3,6)	297327	0	3836
(3,7)	286266	0	3838
(3,8)	288293	0	3840