Quinn Nguyen

UIN# 524002419

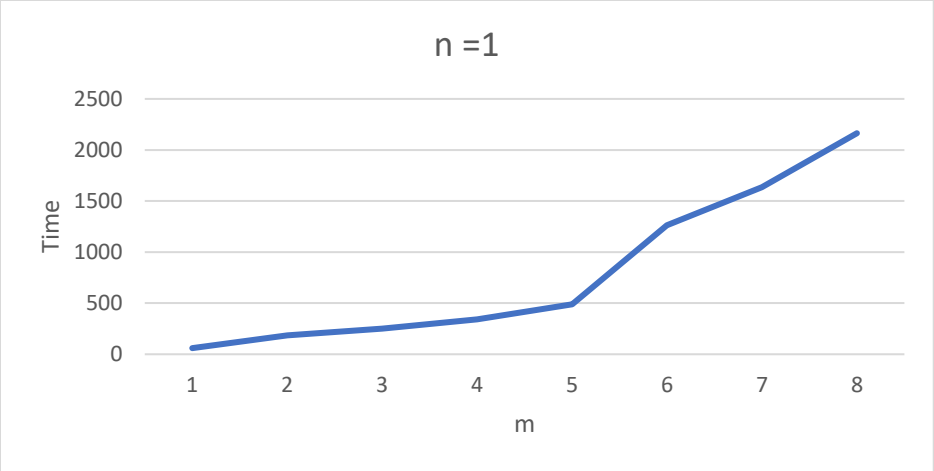
CSCE 313- 506

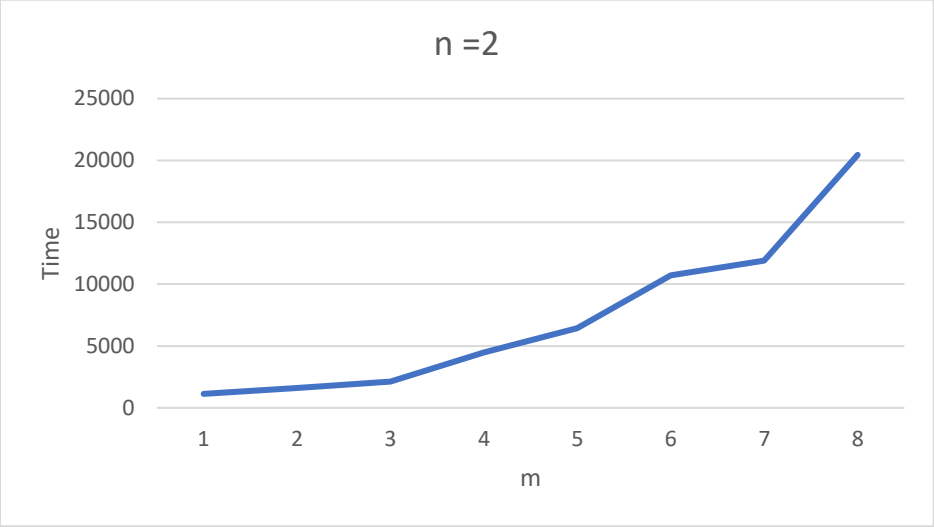
I ran experiments on my buddy allocator implementation on Ackerman function. The following is the result:

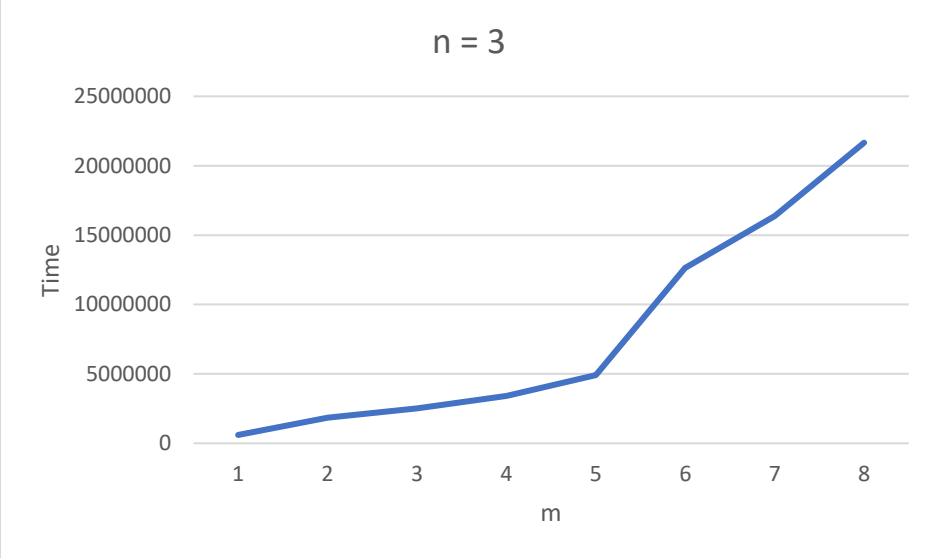
m	time(sec)	time(musec)	allocate/free cycles
1	0	60	4
2	0	185	6
3	0	251	8
4	0	342	10
5	0	490	12
6	0	1263	14
7	0	1637	16
8	0	2164	18
		n = 1	

m	time(sec)	time(musec)	allocate/free cycles
1	0	1131	14
2	0	1599	27
3	0	2120	44
4	0	4467	65
5	0	6454	90
6	0	10724	119
7	0	11893	152
8	0	20451	189
		n = 2	

m	time(sec)	time(musec)	allocate/free cycles
1	0	7039	106
2	0	50835	541
3	0	216095	2432
4	0	855777	10307
5	3	400300	42438
6	13	464393	172233
7	117	739005	693964
8	429	46899	2785999
		n = 3	







According to statistics, for any n value, the running time increases significantly when m >= 5. Also, when n =3, the running increases much more than when n = 1 and 2.

The bottleneck in the system is when we have a big block memory is available but the user only requests a small block. The system has to split all the way from big memory block to small memory block. Then, the user immediately frees that memory. The system has to merge from small block all the way to the big block again.

One possible improvement of this system is to not merge immediately when the system has a chance. The system should only merge when there is no block that is available for the requested block size. Therefore, this reduces time that the system goes back and merge the block.

How to run the system:

make all

./memtest -b (size of basic block size) -s (size of memory length)

For example: basic block size = 32 and total memory length = 1024

./memtest -b 32 -s 1024