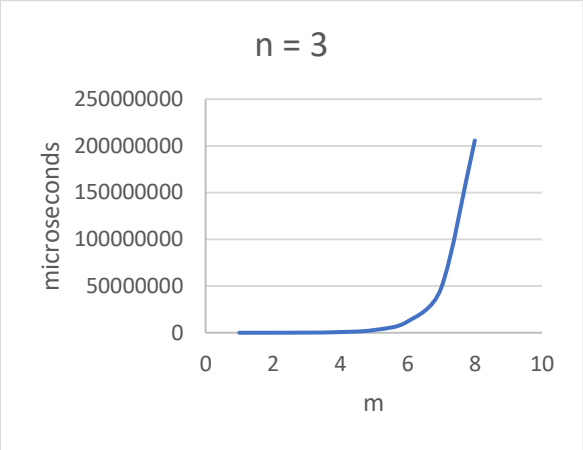
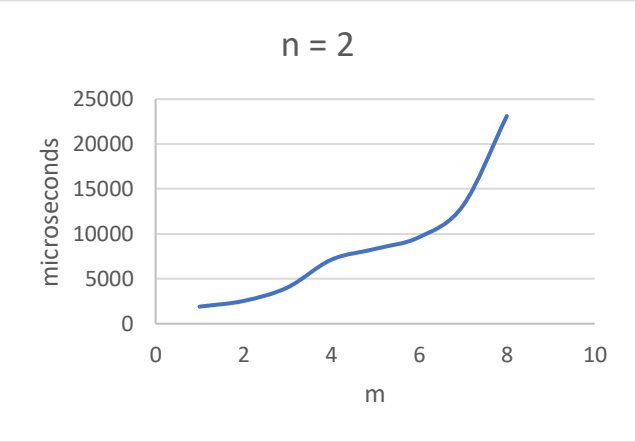
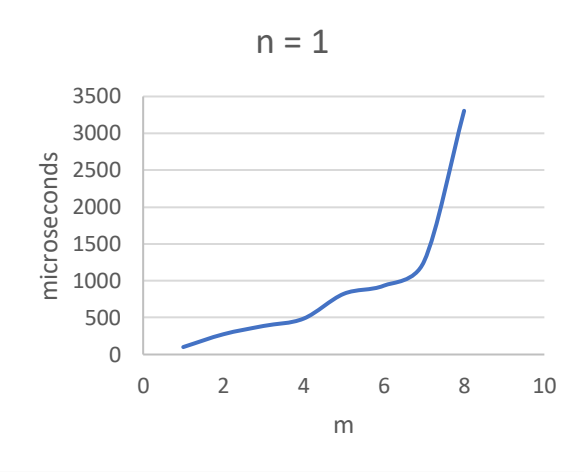
Yifei Liang
727002177

PA1 report

Graphs:




Time

| n/m | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 101 | 1899 | 7879 |
| 2 | 275 | 2531 | 48975 |
| 3 | 387 | 4023 | 164169 |
| 4 | 487 | 7119 | 729933 |
| 5 | 823 | 8327 | 2916380 |
| 6 | 934 | 9627 | 11986188 |
| 7 | 1260 | 13134 | 47803069 |
| 8 | 3306 | 23123 | 205765320 |

## Analysis:

For n = 3:

This graph has the best exponential growth run time data out of the three, the reason for this is that it required much larger memory with bigger m, so the deviation in microseconds is really hard to see.

For n = 2 and n = 1:

These graphs are not exactly exponential growth run time, because it has less increment in time with larger m inputs, and their output time vary a lot with the same m input.

**Conclusion:**

Overall my graphs are really close the **exponential runtime**. My graphis are showing that with constant n and variable m , the run time increases as the m increases. Also the time increment(**growth rate**) increases for each n increase, for example, if n = 2 and m increase by 1, the difference between the time difference between the two calculation is not significant, but if n = 2 and m increases by 1, there will be a significant increase in time and cycle. Which means that the program will run more cycle and call more alloc and free function.

**Bottlenecks:**

In my opinion, The bottlenecks of the simple buddy implementation is the internal fragmentation, for example if you request 66 kb the program will give you 128 kb, this is a lot of memory being wasted. I did a research online with this problem. One easy way to solve internal fragmentation is to introduce more block size, that is allocating the exact memory that is requested. In this new data structure, the two blocks merges as long as they are both free and are both next to each other. However, there is no way to calculate the neighbor address because the block size is no longer power of 2, thus we need to introduce a boundary tag, which is similar to the block header we are using. This tag includes two pointers, one pointing to the previous neighbor(the block that is on the left size of itself), and a next pointer pointing to the right side block. More importantly, it need to include status variable that track the status of the current block. This data structure will eliminate the internal fragmentation inside the simple buddy implementation.